FastCaplet:
An Efficient 3D Capacitance Extraction Solver Using Instantiable Basis Functions for VLSI Interconnects

by

Yu-Chung Hsiao

B.S., Electrical Engineering
National Taiwan University, 2006

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

Author .....
Department of Electrical Engineering and Computer Science
September 3, 2010

Certified by .....
Luca Daniel
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Theses

FastCaplet:
An Efficient 3D Capacitance Extraction Solver
Using Instantiable Basis Functions
for VLSI Interconnects

by

Yu-Chung Hsiao

Submitted to the Department of Electrical Engineering and Computer Science
on September 3, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

State-of-the-art capacitance extraction methods for Integrated Circuits (IC) involve scanning 2D cross-sections, and interpolating 2D capacitance values using a table lookup approach. This approach is fast and accurate for a large percentage of IC wires. It is ,however, quite inaccurate for full 3D structures, such as crossing wires in adjacent metal layers. For such cases, electrostatic field solvers are required. Unfortunately standard field solvers are inherently very time-consuming, making them completely impractical in typical IC design flows. Even fast matrix-vector product approaches (e.g., fast multipole or pre-corrected FFT) are inefficient for these structures since they have a significant computational overhead and scale linearly with the number of conductors only for very large structures with more than several hundreds of wires. In this thesis, we present, therefore, a new 3D extraction field solver that is extremely efficient in particular for the smaller scale extraction problem involving the ten to one hundred conductors in the 3D structures that cannot be handled by the 2D scanning and table look up approach.

Because of highly restrictive design rules of the recent sub-micro to nano-scale IC technologies, smooth and regular charge distributions extracted from simple model structures can be stored beforehand as "templates" and instantiated and stretched to fit practical complicated cases as basis function building blocks. This "template-instantiated" strategy largely reduces the number of unknowns and computational time without additional overhead. Given that all basis functions are obtained by the same very few stretched templates, Galerkin coefficients can be readily computed from a mixture of analytical, numerical and table lookup approaches. Furthermore, given the low accuracy (i.e., 3%–5%) required by IC extraction and the specific aspect ratios and separations of wires on ICs, we have observed in our numerical experimentations that edge and corner charge singularities do not need to be included in our templates,

hence reducing the complexity of our solver even further.

Thesis Supervisor: Luca Daniel
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

I want to express my appreciation to my thesis advisor, Prof. Luca Daniel, for his great mentoring. I would like to thank my senior labmates Dr. Tarek El-Moselhy and Dr. Bradley Bond for their advice and inspirations in this project. I would also like to thank my labmates, Zohaib Mahmood, Lei Zhang, Bo Kim, Yan Zhou, and Omar Mysore for their creating an enjoyable lab environment. Special thanks to Dr. Roberto Suaya for his constructive comments and supports. Thanks to Dawsen Huang for mathematical consulting and Derek Smith for proofreading the English text of this work. Finally, I would like to thank Grace Lee for proofreading part of this work and her seven years of companion. Wihout her, nothing can be possible.

Contents

Chapter 1

Introduction

Since the first integrated circuits were invented in 1958 [18], more and more different system integration concepts, such as System-in-a-Package (SiP), System-on-a-Chip (SoC), 3D packages and 3D ICs, have been proposed over the years, and more and more different subsystems, such as digital, analog, radio frequency (RF) front-end, micro-electro-mechanical systems (MEMS), and also optical devices, are attempted to be integrated. On the other hand, what has not changed for decades is the design methodology: design a circuit schematic, draw its layout, extract parasitics, compare the performance of the layout with the schematic, and revise it until both results are converged. Among these steps, the importance of parasitic extraction continually grows with the advance of fabrication technologies. Up to 2010, the number of transistors in commercial products has exceeded two billion in a central processing unit (CPU) and three billion in a graphics processing unit (GPU). It can be easily seen that the interconnect wires between transistors become extremely complicated as well as the electromagnetic interaction among them. Therefore, accurately extracting interconnect parasitic is the key to a functional chip.

Parasitics in an electronic interconnect system, such as resistance, capacitance, and inductance, are usually modeled as constant parameters in linear models. Although they are called parasitic, well-controlled parasitics can be treated as design components, such as a comb capacitor and a spiral inductor. Regardless of if it is an intended device or a purely unwanted effect, parasitics need to be well modeled not

only because exploiting the best performance out of a technology is beneficial, but also because bad modeling may cause serious unexpected behaviors, such as oscillation and latchups.

This thesis focuses on interconnect capacitance extraction. It is organized in the following way: in the rest of this chapter, the effects of capacitance in different parts of circuits and the survey of existing capacitance extraction techniques are introduced. Chapter 2 reviews necessary background of this work and the two performance comparison references. Chapter 3 shows some important observations that motivate the algorithm described in Chapter 4. In Chapter 5, implementation details are presented. To improve the overall performance, several accelerated integration schemes are proposed in the same chapter. Some examples are shown in Chapter 6 and their performance is compared with the two aforementioned performance references in Chapter 2.

1.1 The Role of Interconnect Capacitance

Capacitance has different roles in different types of circuits. Extraction accuracy requirement is stringent in most cases, whereas some errors can be tolerated in others. In the following, major capacitance effects and usages in different circuits are introduced.

1.1.1 Capacitance in Digital Circuits

In digital circuits, the most prevalent capacitance effect is RC time delay. The signal propagation time from one logic gate to another is called gate delay. Signals are delayed because it takes the first gate some time to charge the parasitic capacitance of the second gate input up to the point where it is activated and responds to logic transition. This effect can be modeled as a first-order RC charging circuit. Besides, all wires possess some capacitance. Changing wire voltage also suffers from this type of delay. The aggregation of this phenomenon in digital circuits limits the overall performance. Circuit designers need accurate parasitic extraction to determine the

critical path, or the most delayed logic path, and also to avoid clock skews. This effect forms a fundamental speed limit of digital circuits. A bad capacitance estimate may cause critical race condition that makes the system totally malfunctioned.

1.1.2 Capacitance in Analog Circuits

Analog signals are susceptible to noise. One of the major noise source is power wires, which connect to power supply or ground and collect all the current that flows into or out of all components and are assumed constant voltage everywhere. However, their finite resistance and large current generate a sensible voltage along the wire and the voltage noise is then coupled everywhere. A common resolution is to place large capacitors to stabilize the voltage by enlarging the RC time constant. This is a rare example where accurate capacitance modeling is not crucial. For other important analog subsystems, such as Phase-Locked Loops (PLLs), which are used to generate stable timing signals or to synthesize desirable carrier frequencies in both wired or wireless systems, accurate capacitor design is essential for both phase locking and frequency accuracy. For another important analog circuit family: Analog-to-Digital Converters (ADCs), making capacitors identical is important. Capacitance mismatch may cause signals to lose their resolution in the digital domain. It is considered as a failure if the target number of bits of resolution is not achieved.

1.1.3 Capacitance in RF Circuits

In RF, millimeter-wave, or even higher frequency circuits, capacitance is an important design parameter. Capacitors are used to achieve maximum power transfer. A badly modeled capacitor can greatly degrade the overall power output, which is one of the most important system parameters in RF communication systems. Even more serious effects can happen to the very front-end circuits, such as power amplifiers, power combiners, and antennas. Those components drain huge power from power supply. Even 1% leakage to its neighborhood through capacitive coupling may cause the system malfunction such as oscillation or damage, such as exceeding wire current

limits.

1.1.4 Capacitance in Packages and MEMS

As more different subsystems are integrated into a single system, extending design to packages is a common choice. Such capacitance extraction includes, for example, various types of wire bonding and through-silicon vias (TSVs) in 3D ICs. Since they are input and output pins of a chip and usually comparatively huge in structures, The issues discussed above, such as time delay and capacitive coupling, are also effective in the package case. In MEMS, capacitors are usually used to exert capacitive forces. The influences of inaccurate extraction depend on different usages of the force.

1.2 Contemporary Capacitance Extraction Techniques

In contemporary design flow, capacitance extraction techniques can be classified into two categories: 2D table lookup methods and 3D field solvers. The former is indeed fast, yet it is accurate only for 2D structures. Full 3D structures need the accuracy of electrostatic field solvers. They are much more accurate choices but their long computation time restricts their usage. The choice of which method to use is always a compromise between computation time and accuracy. A practically good accuracy in circuit design is within a 3% error. A 5% error is the minimum requirement and guaranteeing an error less than 1% is usually unnecessary due to the uncertainty of fabrication and measurement.

1.2.1 2D Table Lookup and Approximation Formula

One major state-of-the-art capacitance extraction approach is based on a 2D scanning approach. In this method, 3D structures are cut into 2D slides along the scanning direction. Once 2D cross-sections are formed, the 2D capacitance on each cross-section is determined by looking up and interpolating the value from a pre-computed

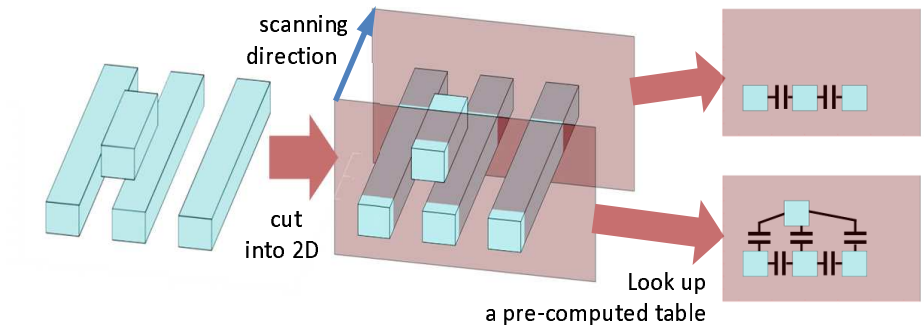


Figure 1-1: 2D scanning and table lookup extraction method. The quasi-3D capacitance is built based on the 2D capacitance on each cross-section.

table such as in [16]. Then the corresponding quasi-3D capacitance is constructed based on the 2D cross-section capacitance information. This procedure is illustrated in Figure 1-1. This method is extremely efficient, and full-chip extraction is possible, yet its accuracy is as good as a 2D approximation. Some approximation formula are proposed as alternatives to the table lookup approach, such as [25][28], or [19] for through-silicon vias. These formula are generated by curve fitting or used through interpolation. Hence, they are only accurate for some types of geometries.

1.2.2 3D Field Solvers

Field solvers are the numerical methods that provide numerical solutions to the set of governing differential equations of a physical problem. Fields can be scalar or vector fields, such as electric and magnetic fields in an electromagnetic problem, pressure and velocity fields in a fluid problem, and more. 3D field solvers are the field solvers that solve 3D differential equations. Hence, unlike the 2D scanning and table lookup method, 3D field solvers can capture 3D phenomena. These methods usually involve two steps: transforming the differential equations into a system of linear equations, and then solving the system. Such steps are called system setup and system solving, respectively, and diagrammed in Figure 1-2. Their corresponding computation time are called system setup time and system solving time, or simply setup time and solving time.

Compared with the previous class of extraction methods, field solvers are more

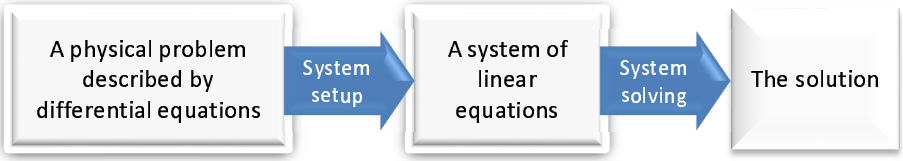


Figure 1-2: A general two-step process of field solvers.

numerically robust and accurate because they require a system solving step during extraction. Hence, minor error in calculating a coefficient in a single equation can be tolerated and "diluted" in the solution. However, they are computationally a lot more expensive, making the use of field solvers limited to partial layout extraction instead of full-chip.

Acceleration techniques were proposed, such as [21][6][24][17]. The most efficient ones are typically based on the boundary element method which will be described in more details in Chapter 2. The acceleration is usually accomplished by adopting fast matrix-vector multiplications in an iterative system solve. However, these methods have a significant overhead to initiate the acceleration and their use starts to become beneficial only when the number of conductors is larger than one hundred. At that point, the computation time begins to scale nearly linearly, O($Nlog(N)$), instead of quadratically or cubically with the number of conductors. Hence, they are ideal only for very large scale problems.

1.2.3 Practical Considerations

In practice, for quasi-3D geometries which are at least uniform in one direction, called quasi-3D structures, such as parallel wires, the 2D scanning approach is accurate. For those full-3D geometries which are not uniform in any direction, for instance, two buses crossing each other and comb capacitors, the 2D table lookup approach fails to provide accurate results within a 5% error. Designers usually identify these types of geometries by inspection, and then turn to 3D field solvers to extract them individually.

1.3 Challenges

Current 3D field solver acceleration techniques suffer from two major drawbacks: they work best for large scale problems and the most acceleration is accomplished by reducing solving time.

The full-3D structures that cannot be handled by the 2D table lookup approach are usually small but many, spread out all over a layout. This scenario is very much against the existing acceleration techniques for 3D field solvers. In small structures, the acceleration by the existing techniques is usually very limited. In a practical layout, the coverage of the full-3D structures is about a 5% area in digital design, and generally more in analog/RF/MEMS design. Due to the computation inefficiency of field solvers, extracting the 5% full-3D structures can take much more time than the remaining 95% of quasi-3D structures extracted by the 2D scanning and table lookup approach.

The second drawback of the existing acceleration methods is that their parallelization is not a trivial task. It is because most acceleration of these methods is achieved by speeding up the system solving time, which is not embarrassingly parallelizable. The use of piecewise constant basis functions (discussed in Chapter 2) in the existing acceleration methods largely limits the feasibility of parallel computing.

1.4 Our Approach: New Basis Functions for Charge Distribution

Basis functions are used by the boundary element methods to represent the solution by linear combination. Traditionally, piecewise constant basis functions are preferred, and combined with acceleration techniques in the system solving step. A huge number of unknowns and hence a giant system is generally inevitable when using this type of basis functions. However, things can be more natural and efficient by employing "specialized" basis functions which are characterized by physical properties such as sinusoidal basis functions for high frequency resonating antenna problems [15], loop-

star basis functions for diverge-free unknowns [32], conduction mode basis functions for Helmholtz current distributions inside conductors [29][7][8][23][11].

This thesis presents a new set of specialized basis functions for charge distribution, which first appeared in [12]. The key idea is to exploit the charge distribution properties under the highly restrictive design rules of the recent sub-micro to nano-scale integrated circuit technologies. In our approach, *only two* fundamental templates are needed to represent virtually every charge distribution for any valid VLSI geometries with a capacitance error less than 3%. Our strategy is to compute these fundamental shapes beforehand, and reuse them to "instantiate" the real basis functions through stretch and assembly operations on the fly when analyzing a geometry. Very few number of basis functions are needed in this approach. Hence, it is extremely efficient in solving a system, both in terms of time and memory.

Chapter 2

Background

This chapter introduces the boundary element method applied to the electrostatic problem, as well as different types of basis functions and system solving techniques. The two classical testing methods: the collocation testing and the Galerkin's testing are also derived in details. Two performance comparison are introduced as references to this work: the standard boundary element method using piecewise constant basis functions with the collocation testing and FASTCAP. The former will be termed as "the standard method" for brevity. The chapter ends with a section that derives the transformation from charge distribution to the capacitance matrix solution.

2.1 The Boundary Element Method

The standard way to extract capacitance is to integrate the surface charge distribution when conductor voltages are properly assigned. In this section, we formulate the corresponding electrostatic problem and how it can be solved given conductor voltages. The specific voltage assignment for capacitance extraction is postponed to Section 2.4.

2.1.1 The Electrostatic Formulation

In order to obtain the charge distribution in an n-conductor system embedded in a uniform dielectric material with the dielectric constant ε, it suffices to solve the integral equation

$$\int_{S'} \frac{\rho(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}-\mathbf{r}'\|}\mathrm{d}s' = \phi(\mathbf{r}) \tag{2.1}$$

for charge distribution ρ. In the equation , $\mathbf{r}$ and $\mathbf{r}' \in \mathbb{R}^3$ are position vectors in 3D space, $\phi(\mathbf{r}) : \mathbb{R}^3 \mapsto \mathbb{R}$ is the electric potential, $\rho(\mathbf{r}') : \mathbb{R}^3 \mapsto \mathbb{R}$ is the unknown charge distribution, S' is the union of conductor surfaces corresponding to the $\mathbf{r}'$ position vector, ε is the dielectric constant of the medium, and the operator $\|\cdot\|$ computes the euclidean distance of its argument. Physically, the charge distribution ρ is nonzero only on conductor surfaces. The equation says that the electric potential at position $\mathbf{r}$ is the overall potential contribution at the presence of surface charge distribution in the space. Mathematically, this equation can be also understood as the fundamental solution to the electrostatic Poisson's equation

$$\nabla^2\phi(\mathbf{r}) = -\rho(\mathbf{r})/\varepsilon$$

subject to the known boundary charge distribution on conductor surfaces, whereas in the integral equation form, the roles of charge and electric potential are interchanged.

2.1.2 System Setup

In order to solve the integral equation, the unknown charge distribution ρ can be further expressed as a linear combination of N basis functions $\psi_j(\mathbf{r}')$

$$\rho(\mathbf{r}') = \sum_{j=1}^{N} \rho_j \psi_j(\mathbf{r}'), \tag{2.2}$$

where ρ_j is the unknown coefficient corresponding to each basis function $\psi_j(\mathbf{r}') : \mathbb{R}^3 \mapsto \mathbb{R}$ supported in $s'_j \subset S'$. Substitute the above expression. The original integral

equation becomes

$$\sum_{j=1}^{N}\left[\int_{s'_j}\frac{\psi_j(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}-\mathbf{r}'\|}\mathrm{d}s'\right]\rho_j=\phi(\mathbf{r}). \tag{2.3}$$

Note that here the integration surface is replaced by s'_j , the support of the basis function $\psi_j(\mathbf{r}')$. Once these N coefficients ρ_j are determined, the charge density in the whole space is known and the self and mutual capacitance between conductors can be calculated accordingly.

In order to determine these N coefficients in the linear equation, it is necessary to at least generate a set of total N equations. There are two traditional methods to achieve this goal: the collocation testing and the Galerkin's testing. The mathematical formulations will be derived first, followed by the introduction to several different basis function selections.

2.1.3 The Collocation Testing

As the name suggests, collocation testing is performed by choosing N "proper" test points in the solution space. A common practice is to place one test point at the center of each basis function. Evaluating N test points can generates N linearly independent equations, forming a full-rank inhomogeneous system of linear equations:

$$\sum_{j=1}^{N}\left[\int_{s'_j}\frac{\psi_j(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_i-\mathbf{r}'\|}\mathrm{d}s'\right]\rho_j=\phi(\mathbf{r}_i), \tag{2.4}$$

for $i\in\{1,2,\ldots,N\}$, or equivalently in the matrix form

$$\begin{bmatrix}\int_{s'_1}\frac{\psi_1(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_1-\mathbf{r}'\|}\mathrm{d}s' & \int_{s'_2}\frac{\psi_2(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_1-\mathbf{r}'\|}\mathrm{d}s' & \cdots & \int_{s'_N}\frac{\psi_N(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_1-\mathbf{r}'\|}\mathrm{d}s'\\ \int_{s'_1}\frac{\psi_1(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_2-\mathbf{r}'\|}\mathrm{d}s' & \int_{s'_2}\frac{\psi_2(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_2-\mathbf{r}'\|}\mathrm{d}s' & & \int_{s'_N}\frac{\psi_N(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_2-\mathbf{r}'\|}\mathrm{d}s'\\ \vdots & & \ddots & \vdots\\ \int_{s'_1}\frac{\psi_1(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_N-\mathbf{r}'\|}\mathrm{d}s' & \int_{s'_2}\frac{\psi_2(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_N-\mathbf{r}'\|}\mathrm{d}s' & \cdots & \int_{s'_N}\frac{\psi_N(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_N-\mathbf{r}'\|}\mathrm{d}s'\end{bmatrix}\begin{bmatrix}\rho_1\\ \rho_2\\ \vdots\\ \rho_N\end{bmatrix}=\begin{bmatrix}\phi(\mathbf{r}_1)\\ \phi(\mathbf{r}_2)\\ \vdots\\ \phi(\mathbf{r}_N)\end{bmatrix}.$$

Hence this is a N-by-N dense linear system. To conform with the formulation of the Galerkin's testing in the next section, Eq.(2.4) can be rewritten as

$$\sum_{j=1}^{N}\left[\int_{s_i}\int_{s'_j}\frac{\delta(\mathbf{r}-\mathbf{r}_i)\psi_j(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}-\mathbf{r}'\|}\mathrm{d}s'\mathrm{d}s\right]\rho_j=\int_{s_i}\delta(\mathbf{r}-\mathbf{r}_i)\phi(\mathbf{r})\mathrm{d}s, \tag{2.5}$$

where the integration surface s_i is the support of the Dirac delta function $\delta(\mathbf{r}-\mathbf{r}_i)$, which has the sifting property:

$$\int_s f(\mathbf{r})\delta(\mathbf{r}-\mathbf{r}_i)\mathrm{d}s=f(\mathbf{r}_i).$$

In this formulation, the test points $\mathbf{r}_i$ in (2.4) are rewritten as the inner product with the test functions $\delta(\mathbf{r}-\mathbf{r}_i)$, the Dirac delta function shifted by $\mathbf{r}_i$.

2.1.4 The Galerkin's Testing

Unlike choosing proper test points in the collocation testing, the Galerkin's testing explicitly involves the notion of projection. Since mathematically the selection of finite basis functions in (2.2) cannot form a complete basis for arbitrary charge distribution in $\mathbb{R}^3$, Eq.(2.2) is a numerical approximation and the residue of using the set of basis functions $\psi_j(\mathbf{r}')$ for $j\in\{1,2,\ldots,N\}$ is defined as

$$R_\psi(\mathbf{r})=\sum_{j=1}^{N}\left[\int_{s'_j}\frac{\psi_j(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}-\mathbf{r}'\|}\mathrm{d}s'\right]\rho_j-\phi(\mathbf{r}).$$

Then enforce the projection of the residue R_ψ on the linear space spanned by the set of basis functions $\bar{\psi}_i$ for $i\in\{1,2,...,N\}$ to be zero:

$$<\bar{\psi}_i(\mathbf{r}),R_\psi(\mathbf{r})>=0,$$

where the inner product operator $<f(\mathbf{r}),g(\mathbf{r})>$ is defined as

$$\int_{s_f\cap s_g}f(\mathbf{r})g(\mathbf{r})\mathrm{d}s.$$

The collocation testing in the form of (2.5) can be interpreted as the projection on the subspace spanned by the set $\delta(\mathbf{r}-\mathbf{r}_i)$ for $i \in 1, 2, \ldots, N$. In the Galerkin's testing, the test functions are chosen as the same basis functions that expand charge distribution, i.e.,

$$\bar{\psi}_i(\mathbf{r}) = \psi_i(\mathbf{r}),$$

for $i \in \{1, 2, ..., N\}$. In parallel with (2.5), the Galerkin's testing can be formulated as

$$\sum_{j=1}^{N} \left[\int_{s_i} \int_{s'_j} \frac{\psi_i(\mathbf{r})\psi_j(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}-\mathbf{r}'\|} \mathrm{d}s'\mathrm{d}s \right] \rho_j = \int_{s_i} \psi_i(\mathbf{r})\phi(\mathbf{r})\mathrm{d}s, \tag{2.6}$$

Similarly, this is an N-by-N linear system, but it is a symmetric matrix whereas the matrix in the collocation testing is not. The corresponding charge distribution can be calculated through (2.2) once all coefficients ρ_j are solved.

2.2 Basis Functions

A set of basis functions is used to span the solution function ($\rho(\mathbf{r}) : \mathbb{R}^3 \mapsto \mathbb{R}$ in our electrostatic problem) through linear combination. Numerically, only finite basis functions are computationally feasible. The selections of basis functions are usually based on how well a solution function is approximated by the shapes of basis functions and the computation cost in an integration such as the bracketed terms in (2.6). Based on the way to the basis functions are constructed, they can be generally classified into two categories: general basis functions and specialized basis functions.

2.2.1 General Basis Functions

General basis functions are the set of basis functions that are not specifically defined as solutions for differential equations, such as piecewise constant, piecewise linear, and sinusoidal functions shown in Figure 2-1. They are usually parametrized by a number as the order of the function, and they can converge to an arbitrary target function by adding more higher order terms. The set of sine and cosine functions with integer multiples of fundamental frequency is such an example. Though the formulation of

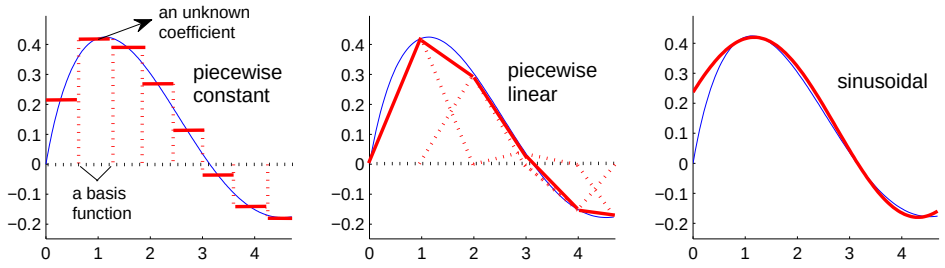


Figure 2-1: Examples of general basis functions: piecewise constant, piecewise linear, and sinusoidal basis functions. An unknown coefficient is determined during the system solving step.

these types of basis functions is independent of a specific physical problem, some choices are better than the others when fewer basis functions are needed to achieve a similar error, for instance, using sinusoid basis functions in a wave problem.

The set of piecewise constant basis functions is usually the standard choice in the boundary element method of the electrostatic problem. These basis function shapes do not favor any specific type of function. Therefore, a large number of basis functions are required for accurate capacitance extraction (within a 3% error), resulting in very long system solving time. This drawback is compensated to some extent by a relatively efficient panel integration when using the collocation testing. This fact can be observed in Eq.(2.4). The analytical expression is available as a smooth well-behaved function. Moreover, its "neutral" look also results in very small function supports because of the requirement of a fine discretization. Hence, the integration in Eq.(2.4) can be well approximated (within a 1% error) by the expression

$$\frac{\Delta(s'_j)}{(4\pi\varepsilon\|\mathbf{r}_i - \mathbf{r}\|)},$$

where the function $\Delta(\cdot)$ calculates the surface area of the argument, when the position vector $\mathbf{r}_i$ is three times of panel length from the integration surface, s'_j. When using piecewise constant basis functions, this computational shortcut generally happens to more than 90% of integrations due to their tiny supports. That feature makes piecewise constant basis functions attractive. FASTCAP [21], pre-corrected FFT [24], and IES^3 [17] are the examples adopting this type of basis functions. Hence, their

acceleration are mostly accomplished by using fast system solving methods.

Other standard basis function examples are piecewise linear basis functions in Figure 2-1 and higher degree polynomials. Piecewise linear basis functions are the standard choice in the finite element method, but in the boundary element method, these types of basis functions are less common because even though they have better shape approximation abilities, resulting in a reduction of the number of basis functions needed at a fixed error level, their drastically increase in integration computation cost almost balances the time saved in the system solving step.

The set of sinusoidal basis functions is another common choice due to its solid mathematical study. It converges well and fast in a wave problem such as time-domain transmission line analysis. In the boundary element method of the electrostatic problem, its shape does not match charge distribution and hence is not an ideal choice.

General basis functions are mathematics oriented and physics detached. They are usually elementary functions and hence the closed form Green's function integration expressions are more likely to be available. Since they are not extracted directly from a physics problem, the same type of basis functions can be used in different situations. But, generally, they are not the most efficient choice to a specific problem.

2.2.2 Specialized Basis Functions

Specialized basis functions, on the other hand, utilize the knowledge of the physical problem. They are more physically-oriented than mathematics. This type of basis functions targets a specific problem by extracting essential shapes directly from the problem and reuse them. Hence, they are more likely numerically represented. Some orthogonalization processes are usually needed after extraction in order to to increase the numerical stability.

One example of specialized basis functions is the conduction mode basis functions for current distribution at high frequency [29][7][8][23][11]. The high frequency effects, such as skin effects and proximity effects, only become prominent when wire dimension is more than two skin depth. The conduction mode basis functions are more efficient

and accurate in these cases because traditional fast algorithms are either inaccurate or inefficient in these cases. The conduction mode basis functions, on the other hand, specifically target these effects and the corresponding shape template basis functions are extracted directly from a wire geometry. Therefore, very few basis functions are needed to represent these phenomena. An orthogonalization process is performed to improve the numerical stability in [13].

In this thesis, we propose a new set of basis functions for capacitance, which first appeared in our publication [12]. Traditionally, the combination of piecewise constant basis functions and the collocation testing uses too many discretization panels for unnecessarily accurate charge distribution. That is necessary only when the detailed charge shapes are important. For instance, charge singularities, or the infinite charge density around corners and edges, are crucial when considering electrostatic discharge or other applications. In the capacitance extraction, all these charge shape details are lost because capacitance is the integration of surface charge density. The basis function extraction process focuses on capturing major charge interactions between conductors. There will be no orthogonalization process involved due to the fact that orthogonalization usually destroys the physical meaning of the shapes. A workaround and the detailed extraction procedure will be described in Chapter 4.

2.3 System Solving

Once the system of linear equations such as (2.4) or (2.6) is set up, the next step is to solve the system. Choosing a proper solving method is crucial for the overall performance, especially for piecewise constant basis functions which usually require a fine discretization. In addition to computation time, memory is also another important factor. Some approaches require much less memory than the others, making the solutions to huge systems feasible.

2.3.1 Traditional Direct Approaches

Suppose we are solving the linear system

$$Ax = b, \tag{2.7}$$

where $A \in \mathbb{R}^{N\times N}$ is a full-rank matrix without any specific structure, $x \in \mathbb{R}^N$ is the unknown vector, and $b \in \mathbb{R}^N$ is the known right-hand side. The most common choice is to perform LU decomposition

$$A = LU,$$

where $L, U \in \mathbb{R}^{N\times N}$ are lower and upper triangular matrices, respectively. Then perform two computationally cheaper back substitutions of L and U in order in (2.7) to solve for x. In solving a large and sparse matrix, the LU decomposition is an ideal choice for its complexity as low as $O(N^{1.2})$. In solving a dense matrix, it is usually not practical since the complexity becomes $O(N^3)$. However, it is still attractive in a small and dense matrix due to its low overhead. Some variations such as LU decomposition with partial pivoting

$$PA = LU,$$

where P is a permutation matrix, or LU decomposition with complete pivoting

$$PAQ = LU,$$

where in this case P and Q are both permutation matrices, are the means to improve numerical stability for the reason that numerically represented numbers have finite precision. The latter case is rarely used because of its significant overhead but marginal stability improvement [31].

In this thesis, most of the computation complexity is moved to the system setup part. The system matrix A in our case is extremely small compared with the system size when using piecewise constant basis functions. Hence, a well-implemented

standard decomposition method with twice back substitutions is an ideal choice. The selection of implementation will be discussed in Chapter 5.

Though the direct approach is used in this work, the iterative counterpart and the corresponding accelerated methods will be introduced in short paragraphs for comparison and as the technical background of the second performance comparison reference, FASTCAP.

2.3.2 Standard Iterative Approaches

When the system matrix is large and dense, as in the case of the boundary element method with piecewise constant basis functions, an iterative approach is more time efficient and hence usually preferred. A popular choice is the family of Krylov subspace methods. When solving a linear system (2.7), these methods build an iteration sequence that iterates from an initial guess x_0 through several x_n, $n \in \{1, 2, \dots\}$ such that $x_n \in x_0 + \mathcal{K}_n(A, r_0)$, where $\mathcal{K}_n(A, r_0) = \text{span}\{r_0, Ar_0, \dots, A^{n-1}r_0\}$, and $r_n = b - Ax_n$ is the residual of the n-th iteration. The orthogonal property is usually satisfied

$$r_n \perp A\mathcal{K}_n(A, r_0)$$

to ensure the minimum residual search direction in each iteration. In these methods, matrix-vector products are the dominant operations. The time complexity is $O(kN^2)$, where k is the number of iterations. One classic implementation is the GMRES algorithm by Saad and Schultz [27].

2.3.3 Accelerated Iterative Methods

The performance bottleneck of the Krylov subspace methods is the matrix-vector product in each iteration that takes $O(N^2)$. Hence, the methods can be accelerated by replacing the standard matrix vector product with an approximated or equivalent operation but with lower time complexity, such as $O(Nlog(N))$. FASTCAP is such an example by taking the advantage of multi-pole expansion [26]. It has twofold effects: reduce the matrix-vector product time to $O(Nlog(N))$ by performing

an approximated linear operation, and also the setup time of the linear operation to $O(Nlog(N))$. The multi-pole expansion, however, is restricted to the electrostatic or similar problems. Another method called pre-corrected FFT [24] relaxes this restriction and has similar performance. It can be applied to other problems such as current or inductance extraction. Its acceleration is based on the fact that the fast Fourier transform takes $O(Nlog(N))$.

The drawbacks of these accelerated iterative methods are that they require an extra start-up overhead. For instance, in the multi-pole expansion method, the expansion coefficients need to be calculated first before the iteration schemes, and in the pre-corrected FFT, the "pre-corrected" step needs extra computation to map discretization panels to uniform grids. These extra overheads are significant if the system is small. Hence they are not ideal for our approach.

2.4 Capacitance Extraction

Capacitance, by definition, is the overall accumulated charge on the surface of a conductor per unit voltage. The capacitance extracted in this thesis is the short-circuit capacitance matrix (or capacitance matrix for brevity), which can be used to generate other types of capacitance matrices such as two-terminal capacitance matrix and total capacitance [20]. In a capacitance matrix, the off-diagonal entry $C_{i,j}$ is the mutual-capacitance between two different conductors M_i and $M_j, j \neq i$, and the diagonal entry $C_{i,i}$ is called self-capacitance.

According to the definition of capacitance, the extraction problem can be transformed into a special setup of the electric potential in the electrostatic problem (2.1). In the following, we explicitly differentiate between a basis function support surface and a conductor surface by using a small letter s and a capital letter S, respectively. In order to have a general treatment for both the collocation testing and the Galerkin's testing, we first multiply both sides of (2.4) by the area of the i-th basis function support s_i, written as $A(s_i)$, to correct the inconsistent dimension in (2.5) compared with (2.6). It is caused by the singular behavior of delta functions.

$$\sum_{j=1}^{N}\left[A(s_i)\int_{s'_j}\frac{\psi_j(\mathbf{r}')}{4\pi\varepsilon\|\mathbf{r}_i-\mathbf{r}'\|}\mathrm{d}s'\right]\rho_j = A(s_i)\phi(\mathbf{r}_i). \tag{2.8}$$

In a circuit model, we assume capacitance is a constant parameter in a linear model. Hence, acoording to the principle of superposition, the unit voltage in the capacitance definition can be set up as one volt raised on the conductor of interest and zero volt on the others, i.e., when the capacitance associated with the k-th conductor is of interest,

$$\phi_k(\mathbf{r}) = \begin{cases} 1 & \text{if } \mathbf{r}\in M_k, \\ 0 & \text{otherwise.} \end{cases}$$

Substitute it in (2.8) or in (2.6) for all basis functions, forming a column vector ϕ_k on the right-hand side. Collect all such columns for each $k \in \{1, 2, \ldots, n\}$, where n is the number of conductors, to form a matrix $\Phi \in \mathbb{R}^{N\times n}$

$$\Phi = [\phi_1|\phi_2|\ldots|\phi_n].$$

Then the equation (2.8) or (2.6) becomes

$$P\rho = \Phi,$$

where $P \in \mathbb{R}^{N\times N}$ is the system matrix, each entry of which is the square bracketed term in (2.8) or (2.6), and $\rho \in \mathbb{R}^{N\times n}$ is the unknown coefficient matrix

$$\rho = \left[\rho^1|\rho^2|\ldots|\rho^n\right],$$

in which the k-th column vector is the coefficient solution under the electric potential setup ϕ_k. The capacitance matrix entry $C_{i,j}$ is the capacitance between the i-th and the j-th conductors. It is the overall accumulated charge on the i-th conductor when the j-th conductor is set to be one volt higher whereas the rest is zero. It can be formulated as

$$C_{i,j} = \int_{S_i}\rho^j(\mathbf{r})\mathrm{d}s, \tag{2.9}$$

where S_i is the surface of the i-th conductor, and $\rho^j(\mathbf{r})$ is the charge distribution solution under the electric potential setup ϕ_j. After substituting (2.2), (2.9) becomes

$$C_{i,j} = \sum_{m=1}^{N} \left[\int_{S_i} \psi_m(\mathbf{r}) \mathrm{d}s \right] \rho_m^j.$$

The integration can be rewritten as

$$\int_{S_i} \psi_m(\mathbf{r}) \mathrm{d}s = \int_{s_m} \psi_m(\mathbf{r}) \phi_i(\mathbf{r}) \mathrm{d}s$$

by treating $\phi_j(\mathbf{r})$ as an indicator function. It is identical to the right-hand side of (2.8) or (2.6) except that it is under the electric potential setup $\phi_i(\mathbf{r})$. Hence,

$$C_{i,j} = \phi_i^T \rho^j = \phi_i^T P^{-1} \phi_j.$$

Therefore, the complete capacitance matrix is

$$C = \Phi^T P^{-1} \Phi.$$

Chapter 3

Observations

The geometries in contemporary nano-scale circuit technologies have many restrictions, such as minimum wire width and separation, maximum ratios of wire size to wire thickness, and only drawn in Manhattan geometry. The charge distribution is, hence, very regular and does not change drastically. This fact brings several good properties to reduce the complexity of calculating charge distribution. In particular, in this chapter we will point out two key observations that will be the foundations for the development of the algorithm in Chapter 4.

3.1 The Effects of Charge Singularities on Capacitance

Charge singularities are the infinite charge density accumulated around a pointed geometry, such as edges, corners, or pin tips. Figure 3-1 shows a common 3D wire in a VLSI layout and the singular charge distribution on its top face. This charge distribution is obtained by using finely discretized piecewise constant basis functions combined with the collocation testing. Although the charge density is infinite, it is integrable and contributes a finite charge. This singular behavior is of great interest in several applications, such as the discharge problem in MEMS [30]. The theoretical study of the charge behaviors in the vicinity of a singular point was studied in [9].

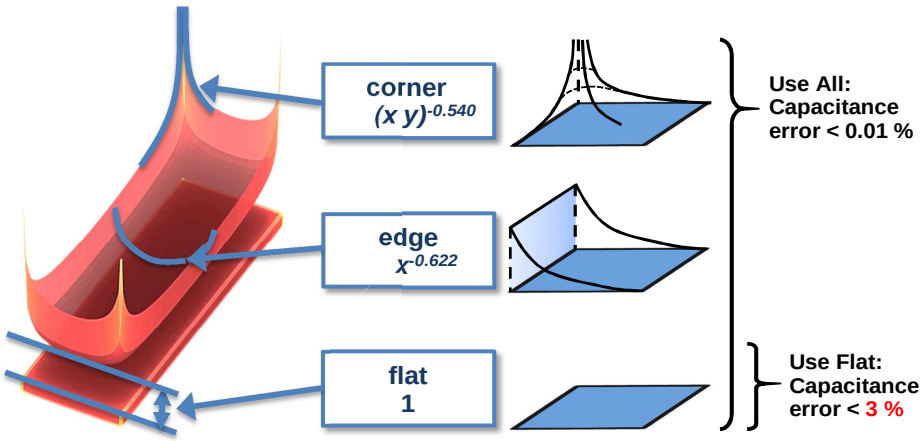


Figure 3-1: Charge singularities of a 3D interconnect wire in VLSI and the visualization of corner, edge, and flat basis functions. Using all the corner, edge, and flat basis functions in the Galerkin's testing gives a capacitance error less than 0.01%, whereas the error of using the flat basis functions alone is less than 3%.

In order to investigate the singular charge effect in the boundary element method when using the Galerkin's testing, we set up the following experiment. First, we extract the "exact" charge distribution by using the collocation testing in order to construct specialized basis functions for charge singularities. The "exact" charge distribution is extracted by setting up a straight wire with finite thickness, finely discretizing it, and solving it by using the collocation testing for both its charge distribution and its capacitance C_0. Specifically, the charge distribution on the top face of the wire is shown in Figure 3-1. By inspection, three types of basis functions can be identified: flat, edge, and corner basis functions. The flat basis function is constant all over the conductor face S. The edge basis function decays in the direction perpendicular to the edge. It can be mathematically written as $x^{-0.622}$ when $x \in S$. The corner basis function is chosen to decay in both directions at the same rate, written as $(xy)^{-0.540}$ for $x, y \in S$. All the numbers of negative fractional power are numerically selected. Their shapes are illustrated in the same figure.

The next step of this experiment is to use these three types of basis functions to calculate capacitance C. Instead of using the collocation testing, the use of the Galerkin's testing in this case is necessary for the relatively large support of each basis function. The relative capacitance error $(C - C_0)/C_0$ turns out to be less than 0.01%.

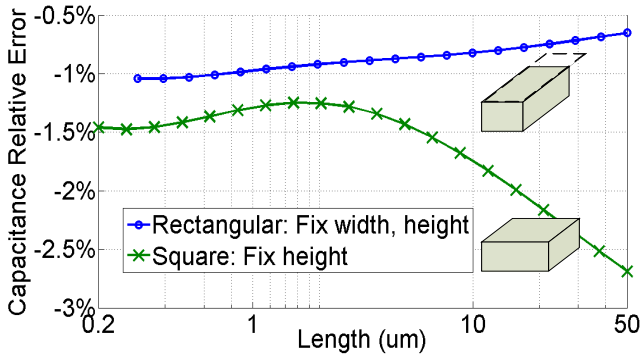


Figure 3-2: The relative capacitance errors for different wire lengths and different pad sizes with respect to very fine discretized geometries using piecewise constant basis functions with the collocation testing. The width of the wire is fixed to be 0.3 um and its thickness is 0.2 um. The thickness of the pad is also 0.2 um.

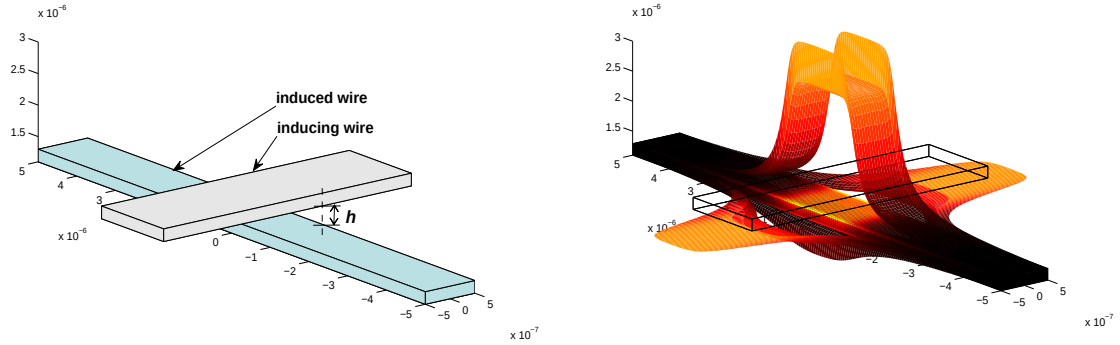
It is too accurate for practical purposes. For VLSI capacitance, we wish to relax some accuracy in trade of computation efficiency.

The last step is to pick up the necessary basis functions that make the accuracy stay at a reasonable level. We found that using flat basis functions alone can give us an error less than 3%, which is generally enough in a VLSI capacitance extraction problem, as we have discussed in Section 1.2.3. A further investigation of different wire sizes in Figure 3-2 shows similar results. In this case, we again use only a flat basis function on each face of a straight wire (the upper curve) and a square pad (the lower curve). As we can see, the resultant capacitance is no worse than 3% less than the finely discretized collocation solutions. Note that the 50 um in the figure is usually the largest dimension for a pad in a contemporary VLSI technology.

Based on these facts, it can be inferred that constructing basis functions specifically for charge singularities may not be necessary for practical capacitance problems.

Observation 3.1. Basis functions for charge singularities are negligible in capacitance extraction within a 3% error in single conductor cases in valid VLSI Manhattan geometry using the boundary element method with the Galerkin's testing.

This observation is not obvious since most previous boundary element method based algorithms, such as [21][24][17], use the collocation testing, which is a point-matching approach (see Eq.(2.4)). The Galerkin's testing is sometimes used, but their



(a) The structure.

(b) The induced charge distribution with singularities removed.

Figure 3-3: Two wires crossing each other at the centers separated by h.

implementation usually use only a few quadrature points [5][14], which is not much different from the collocation testing in the aspect of edge and corner singularities. In our experiments, however, the integration of flat basis functions is calculated analytically. Hence, the singularity effect is implicitly taken into account in an average way. A different argument was reached in [34] that a correction term is necessary when using a finite difference approach. It can be explained by considering the formulation of two methods: the boundary element method is integral equation based whereas the finite difference approach is based on a differential equation. The latter seems to be more sensitive to this singularity issue.

3.2 Stretchable Induced Charge

Induced charge, by our definition, is the separated charge distribution of a wire that mainly results from the presence of the other wire. Those wires are called the induced wire and the inducing wire, respectively. Figure 3-3(a) shows a pair of wires in different metal layers that cross each other at their centers. Its charge distribution solved by the standard method after removing the charge singuarities is shown in Figure 3-3(b). The removal is done by extracting an independent setup of the induced wire alone and subtracting the result from the crossing wire case. The charge density is rescaled during the subtraction to ensure the corner charge is zero.

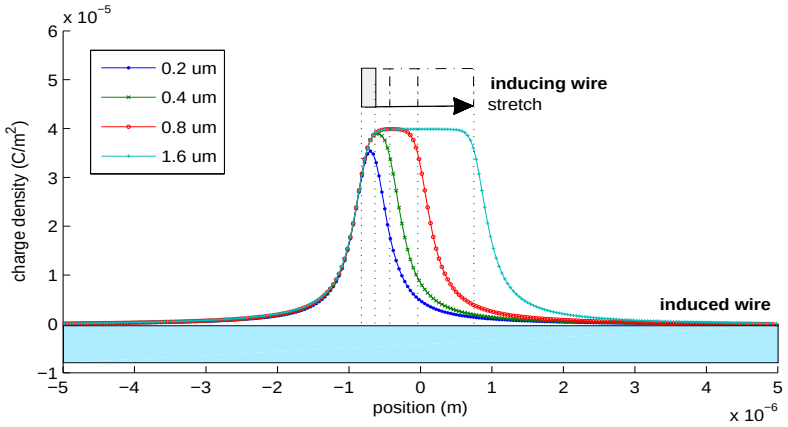


Figure 3-4: The induced charge distribution on the top face of the induced wire at the presence of the inducing wire. A series of the curves corresponding to different inducing wire widths w demonstrates the induced charge is stretchable. The center flat part is stretched with the inducing wire width, whereas the two decaying shapes on sides are only shifted outwards. The dimensions of the induced wire and of the inducing wire are $10 \times 1 \times 0.2\text{um}^3$ and $5 \times w \times 0.2\text{um}^3$, respectively. The wire geometry is not to scale in the y-axis.

In this experiment, we are interested in how the induced charge reacts to the change of the width of the inducing wire. Figure 3-4 shows a series of induced charge distribution curves that are sampled along the central line on the top face of the induced wire in Figure 3-3(b) with different widths of the inducing wire. These curves demonstrate that only the center flat part of each curve stretches or shrinks with the width of the inducing wire. The decaying shapes are only shifted without changing their slopes when the inducing wire decreases its width.

Observation 3.2. Given a pair of inducing and induced wires separated by h as in Figure 3-3(a), the shapes of the decaying charge distribution on the induced wire in the vicinity of the edges of the inducing wire are "invariant" regardless of the width of the inducing wire. When the inducing wire continuously narrows down, the flat part shrinks to zero at some point and the two decaying arch shapes start to merge from the top. In addition, the decaying arch shape only strongly depends on the wire separation h. This dependency is shown in Figure 3-5.

The last statement is made according to a separate experiment on the parameters of wire geometries that we will skip here. This shape invariant property is a great

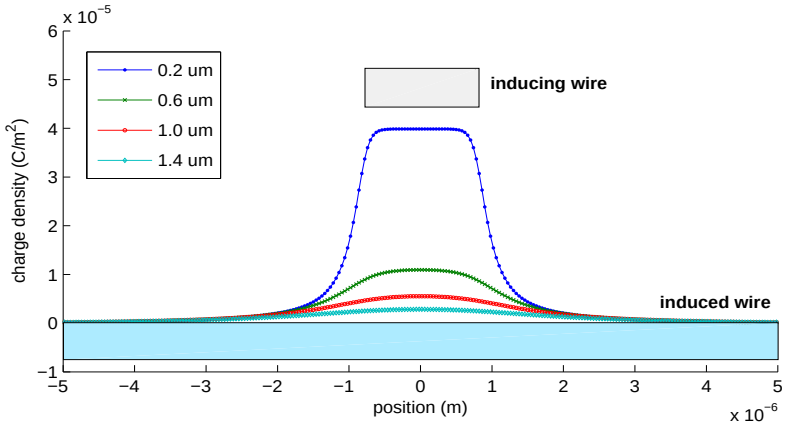


Figure 3-5: The separation dependency of the induced charge distribution. The induced charge is quickly flattened when the separation gradually increases. The dimensions of the inducing wire and the induced wire are $10 \times 1 \times 0.2$ um^3 and $5 \times 1.6 \times 0.2$ um^3, respectively. The wire geometry is not to scale in the y-axis.

implication that an induced charge distribution is decomposable. Figure 3-6 shows such a decomposition suggestion. We will describe the slope extraction procedure in Chapter 4 and reuse it everywhere in Chapter 6.

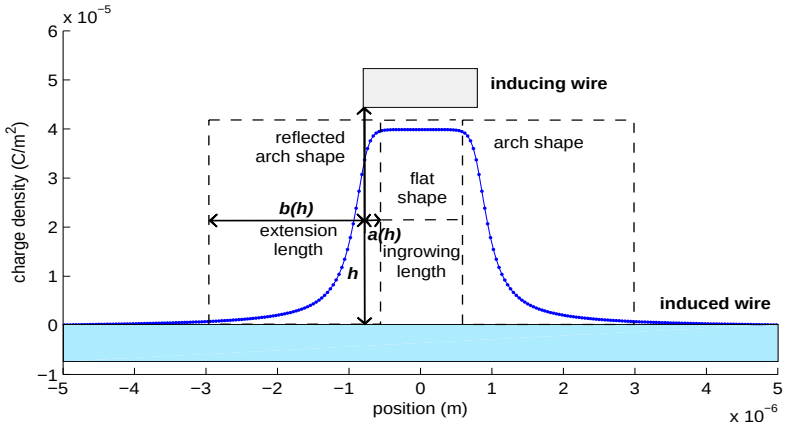


Figure 3-6: A possible induced charge decomposition, based on Observation 3.2. The parameter a denotes the extra length of arch shapes under the inducing wire, and the parameter b denotes the length of the truncated tail. They are called ingrowing length and extension length, respectively. The parameter h is the separation distance of the induced and the inducing wires. The wire geometry is not to scale in the y-axis.

Chapter 4

Instantiable Basis Functions

In this chapter, we define two templates based on the two key observations in Chapter 3 and demonstrate the instantiation process of basis functions.

4.1 The Templates

Templates are the building blocks of induced basis functions. They are extracted off line and parameterized by geometry given a fabrication technology before solving a capacitance extraction problem. Only *two* templates are needed to construct the whole algorithm in this work. They are arch templates and flat templates.

4.1.1 Arch and Flat Templates

Before having a mathematical description of the templates, several terms need to be defined first.

Definition 4.1. The **surface** S_i of the i-th conductor M_i is the union of all the interfaces between dielectric and M_i. The j-th **face** of M_i denoted as $S_{i,j} \subset S_i$ is a largest rectangle plain surface that contains a point p such that any other rectangle $X \subset S_i$ such that $p \in X \subset S_{i,j}$. They are related by $S_{i,j} \subset S_i \subset M_i$.

These notations will be useful in the algorithm description in Section 4.2.3.

Definition 4.2. A **face basis function** is a function with value 1 supported in $S_{i,j}$ for some i, j. A basis function is called an **induced basis function** if it is not a face basis function.

Face basis functions are the only basis functions used to capture self capacitance. It is based on Observation 3.1 and extended to multi-conductor problems. Using these basis functions with the Galerkin's testing, charge singularities around corners and edges are also taken into account in an average sense. Here, we need one extra definition in order to define the two templates.

Definition 4.3. An **arch shape** $A_{\mathbf{p}}(r) : \mathbb{R} \mapsto [0, 1]$ is a function parameterized by a vector of parameters $\mathbf{p}$, and supported in $[-a(h), b(h)]$, where $a(h)$ and $b(h)$ are the length under the inducing wire and the length extended from the edge of the inducing wire as shown in Figure 4-1, called the **ingrowing length** and the **extension length**, respectively.

In the definition, the parameter vector $\mathbf{p}$ contains at least the wire separation h by Observation 3.2. More parameters can be added in $\mathbf{p}$ if necessary. Some errors in the arch shapes can be tolerated. An imperfect selection of basis function shapes may only cause an additional 0.1%–1% capacitance error. It will be discussed in Section 4.1.2. Now, we are ready to define the two templates.

Definition 4.4. An **arch template** $T_{A,\mathbf{p}}(u, v) : \mathbb{R}^2 \mapsto \mathbb{R}$ is a function supported in S, a subset of some face basis function support and $T_{A,\mathbf{p}}(u, v) = A_{\mathbf{p}}(u)$, for all $(u, v) \in S$, where $A_{\mathbf{p}}(u)$ is an arch shape.

Definition 4.5. A **flat template** $T_F(u, v) : \mathbb{R}^2 \mapsto \mathbb{R}$ is a function supported in S, a subset of some face basis function support and $T_F(u, v) = 1$, for all $(u, v) \in S$.

These templates are simply formed by taking the 1D variation in Figure 3-6 and extending it to 2D by fixing its value. The choice brings many advantages and will be discussed in Chapter 5. The exact function supports of the templates have not yet been specified. They will be decided by wire geometries as described in Section 4.2.

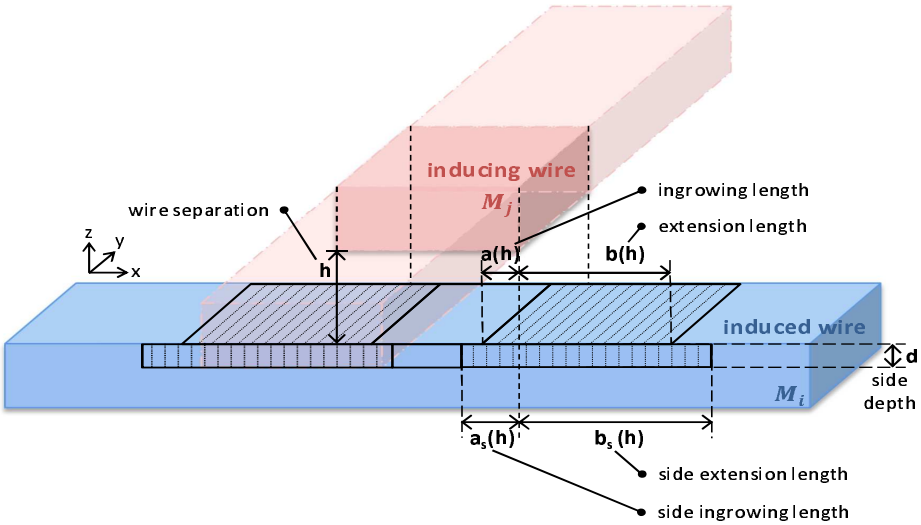


Figure 4-1: Arch extraction setup.

4.1.2 Arch Shape: Extraction

The decomposition suggested in Figure 3-6 is based on the curve extracted by the standard method. This curve can be a choice of arch shapes, but it is not the optimal one after we truncate the arch tails and neglect basis functions for the corner and edge singularities. A new arch shape extraction scheme is needed to comply with this modified setup.

In Figure 3-3(b), it can be seen that there is significant charge density induced around the top edge of the side faces. Hence, for the optimal arch shape extraction, the induced charge on the side faces also needs to be considered. This extraction setup is shown in Figure 4-1. In this figure, there are five parameters to be determined before extracting the optimal arch shapes: the side depth d, the arch extension and the side arch extension lengths $b(h)$ and $b_s(h)$, and the arch ingrowing and the side arch ingrowing lengths $a(h)$ and $a_s(h)$. Figure 4-2 shows the same setup but with the templates separated from conductor faces and placed in different layers to emphasize the fact that they are laid on the face basis functions. In these figures, each rectangle represents a piecewise constant basis function used by the Galerkin's testing. The arch templates and side arch templates are only discretized in the x-direction but not in the y-direction in order to simulate the fact that arch templates have simply

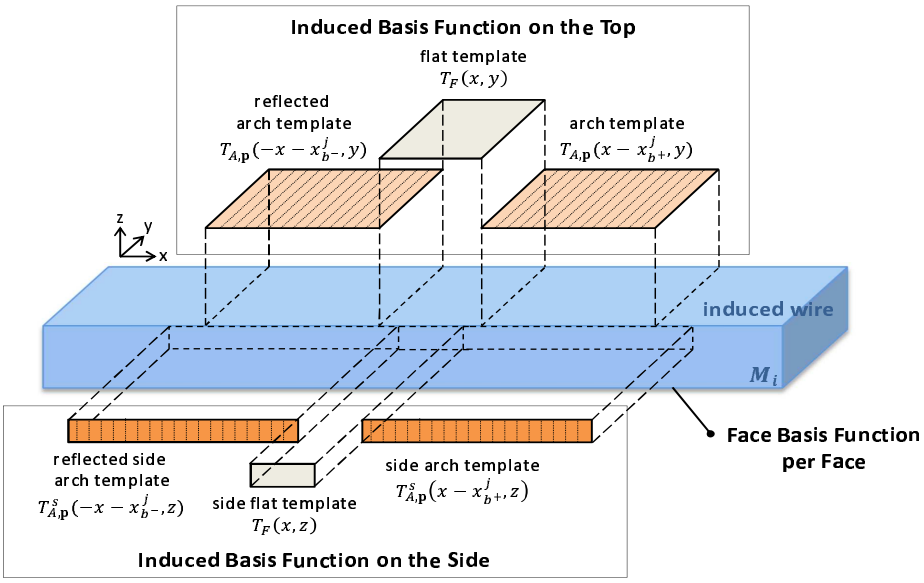


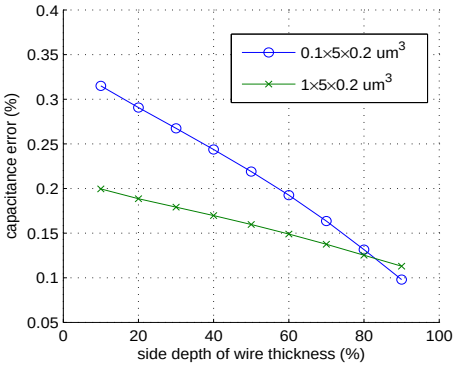
Figure 4-2: Arch extraction setup displayed in layers.

1D variation. The arch shapes, or the coefficients of the small rectangles of the arch templates, are determined in the system setup step.

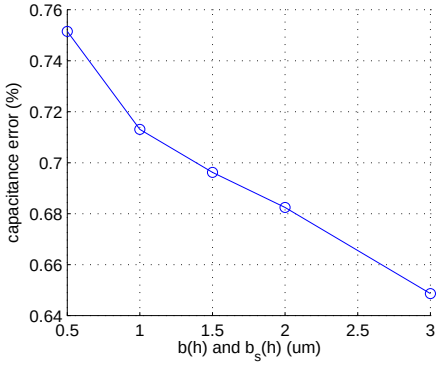
The strategy to select the five parameters in Figure 4-1 is based on capacitance errors, numerical stability, and the physical meaning of shapes. The numerical experiment in Figure 4-3(a) shows that the deepest side template produces the least capacitance error. However, the difference is limited. For the sake of numerical stability, the side depth of the induced charge induced by the closest metal layer is chosen to be the upper 50% of the wire thickness. The lower 50% is reserved for the presence of the inducing wire from the bottom. The side depth for the second closest metal layer takes 25% of the thickness. If the induction is from even farther metal layers, all the corresponding side templates are neglected.

Figure 4-4 and 4-3(b) shows the arch shapes and the side arch shapes for different extension lengths, $b(h)$ and $b_s(h)$, and their corresponding capacitance errors. It can be seen that the arch shapes and the side arch shapes overlap almost perfectly no matter how long the extension lengths are chosen. Hence, they are selected to be the distance measured from the edge of the inducing wire to where the charge density drops to zero, discarding the negative part.

The effects of different arch and side arch ingrowing lengths, or $a(h)$ and $a_s(h)$

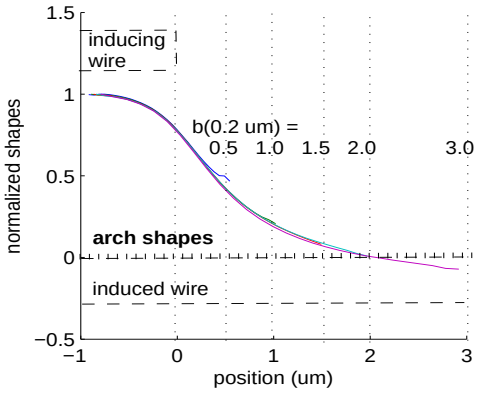


(a) Errors for different side depth when b, b_s, a, and a_s are maximized. Wire dimensions: width×length×thickness.



(b) Errors for different b and b_s when $d = 50\%$, and $a = a_s = 1$ um. Wire dimensions: $15 \times 1 \times 0.2$, $15 \times 5 \times 0.2$ um^3.

Figure 4-3: Capacitance errors for different arch extraction setup parameters.



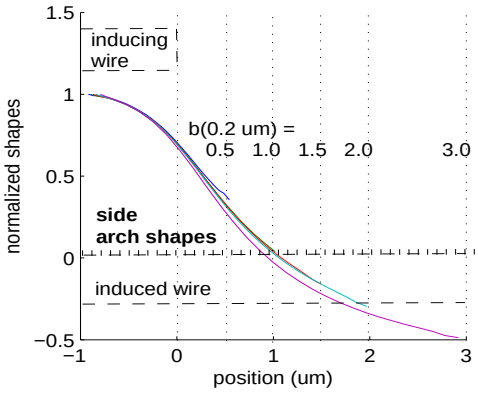


Figure 4-4: Arch shapes and side arch shapes for different extension lengths b and b_s when $d = 50\%$, and $a = a_s = 1$ um. Wire dimensions: $15 \times 1 \times 0.2$, $15 \times 5 \times 0.2$ um^3.

are similar: the largest $a(h)$ and $a_s(h)$ gives the most accurate capacitance. It is reasonable since the solver has more degree of freedom to find the most suitable shapes. It is, however, not practical because the integration involving arch templates is more computationally expensive and less accurate. Hence, the best strategy is to minimize the support of arch templates without much loss of accuracy, while at the same time keeping the physical meaning of arch shapes. Once all these parameters are determined, the optimal shapes for this template setup can be extracted by using the boundary element method with the Galerkin's testing.

The family of arch shapes $A_{\mathbf{p}}(r)$ is parametrized by the parameter vector $\mathbf{p}$. In addition to the most dominant parameter h, the separation between inducing and induced wires, we add one extra parameter w, the width of the induced wire, or more precisely, the width of the arch template itself, to account for the effect that edge singularities are averaged into the extracted shapes by the Galerkin's testing. Several extractions for different $\mathbf{p}$ are necessary to fully characterize the shapes. Fortunately, they are very smooth functions of $\mathbf{p}$ and thus both storing them and retrieving them are very efficient.

4.1.3 Arch Shape: Storing and Retrieval

The arch shapes $A_{(h,w)}(r)$ extracted in the previous section are decaying functions of r and smooth in all h, w, and r. Although three-dimensional, the functions are so smooth that storing them in a table with several coarse discrete values and reusing them via linear interpolation is possible and memory efficient. They can also be fit by polynomial functions. However, due to the decaying nature of these functions, a polynomial representation is not the most efficient choice, especially when higher degree polynomials are computed. A more attractive representation is a multivariable rational function of degree (n, m)

$$f(\mathbf{p}, r) = \frac{f_N(\mathbf{p}, r)}{f_D(\mathbf{p}, r)}, \tag{4.1}$$

where $\mathbf{p}$ is a vector of k parameters and the numerator and the denominator are both polynomial functions defined as

$$f_N(\mathbf{p}, r) = \sum_{|\alpha|+\gamma \le n, \forall \alpha, \gamma} \beta^N_{\alpha,\gamma} \mathbf{p}^\alpha r^\gamma,$$

and

$$f_D(\mathbf{p}, r) = \sum_{|\alpha'|+\gamma' \le m, \forall \alpha', \gamma'} \beta^D_{\alpha',\gamma'} \mathbf{p}^{\alpha'} r^{\gamma'},$$

in which the non-negative integer multi-indices α and α' are k-tuples, their absolute values are the sum of all elements, i.e., $|\alpha| = \sum_{i=1}^{k} \alpha_i$, and γ is a non-negative integer as well. In order to fit the rational function to the arch shapes $A_{\mathbf{p}}(r)$, Eq.(4.1) is rewritten as

$$f(\mathbf{p}, r) f_D(\mathbf{p}, r) - f_N(\mathbf{p}, r) = 0.$$

Substitute M data points $A_{\mathbf{p}_i}(r_i)$, for $i = 1, 2, \ldots, N$, and formulate the problem into a constrained optimization problem. Then we have

$$\begin{aligned} \underset{\beta^N_{\alpha,\gamma}, \beta^D_{\alpha',\gamma'}}{\text{minimize}} \quad & \sum_{i=1}^{M} |A_{\mathbf{p}_i}(r_i) f_D(\mathbf{p}_i, r_i) - f_N(\mathbf{p}_i, r_i)|^\nu \\ \text{subject to} \quad & \sum_{|\alpha'|+\gamma' \le m, \forall \alpha', \gamma'} \beta^D_{\alpha',\gamma'} = 1. \end{aligned} \tag{4.2}$$

When $\nu = 1$, it is linear programming. When $\nu = 2$, it forms a sum of squares programming problem. The latter can be solved by using a MATLAB-based software, STINS [3][4]. Both norms provide similar results. The constraint in (4.2) is important. Otherwise, the minimum happens when all coefficients $\beta^N_{\alpha,\gamma}, \beta^D_{\alpha',\gamma'}$ are equal to 0. In this rational function representation, only the degrees n, m and the coefficients $\beta^N_{\alpha,\gamma}$ and $\beta^D_{\alpha',\gamma'}$ need to be stored. The polynomial degrees for the denominator and the numerator are usually less than cubic. In the case $\mathbf{p} = (h, w)$, $n = 2$ and $m = 3$ are enough for a 0.3% relative shape error. Computing arch shapes through this way is also efficient even compared with the table lookup approach, which needs to search for indices for six values in the table and perform 3D linear intepolation.

4.2 Instantiable Basis Functions

The two templates defined in Section 4.1.1 are the *only two* building blocks we need to generate all basis functions for induced charge. This instantiation process will be demonstrated first by an example, then followed by a general algorithm.

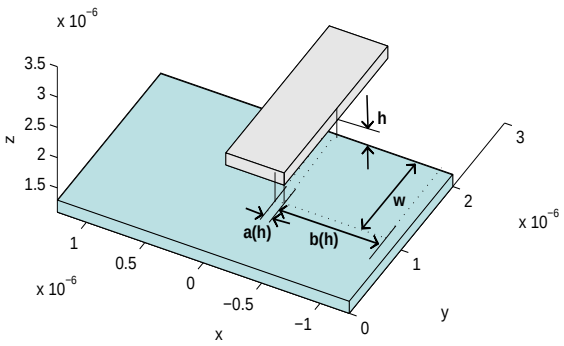
4.2.1 An Example: Partially Overlapping Wires

Figure 4-5(a) shows a pair of partially overlapping wires in adjacent metal layers. The bottom and the top wires are called the induced wire and the inducing wires, respectively. We are going to instantiate the face-induced basis function and then the side-induced basis function.

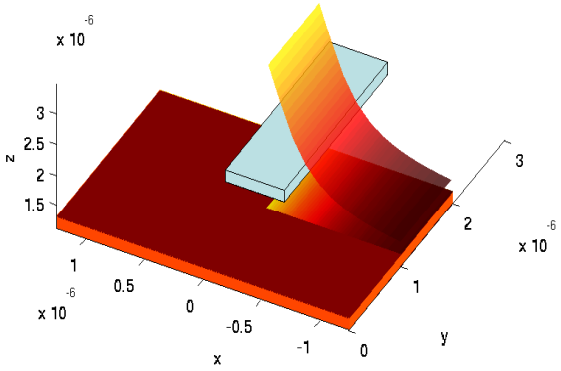
Face-Induced Basis Functions

Face-induced basis functions occur when an induced basis function is induced by a parallel face. Their instantiation procedure in this example is described below. First, we scan the z-direction, the direction that includes different metal layers, of the induced wire M_i within its x and y boundaries, denoted by $x^i_{b^\pm}$ and $y^i_{b^\pm}$. The top wire M_j is found, which is separated from M_i by a distance h in the z-direction. Its x and y boundaries are denoted by $x^j_{b^\pm}$ and $y^j_{b^\pm}$, respectively. Since they are overlapped, at least a flat template will be placed on the top face $S_{i,\text{top}}$ of M_i but only until all arch templates are determined. Next, we examine each edge of the top face $S_{j,\text{top}}$ of M_j to check whether it is within the range of $x^i_{b^\pm}$ and $y^i_{b^\pm}$. The right-front edge of $S_{j,\text{top}}$, oriented in the y-direction, is first examined and its overlapping segment $[y^j_{b^-}, y^i_{b^+}]$ of length w is determined. Then we place an arch template $T_{A,(h,w)}(x - x^j_{b^+}, y)$ supported in $\mathcal{P}_{x\text{-}y}([x^j_{b^+} - a(h), x^j_{b^+} + b(h)] \times [y^j_{b^-}, y^i_{b^+}] \times z^i_{b^+} \cap S_{i,\text{top}})$, where $z^i_{b^+}$ is the upper z boundary of M_i, and $\mathcal{P}_{x\text{-}y}(\cdot)$ projects its argument set onto the x-y plane. The result is illustrated in Figure 4-5(b).

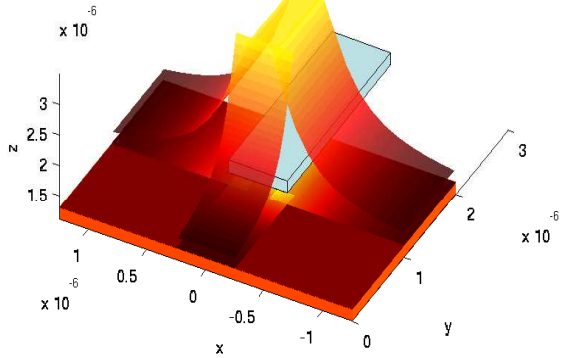
The same procedure can be followed for the left-front edge and the left-back edge. The right-back edge has no effect for the induced basis function on the top face since it is not within the induced wire boundaries. Finally, a flat template supported



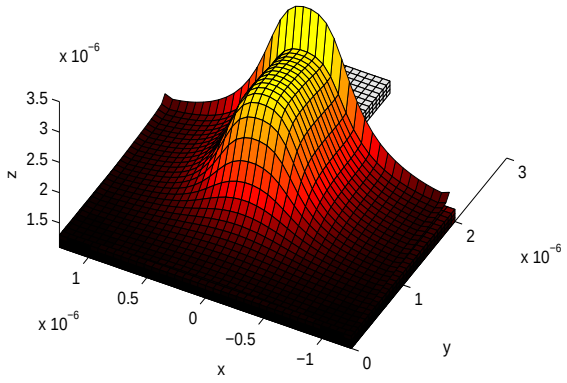
(a) The wire geometry.



(b) Instantiate an arch template.



(c) The complete face-induced basis fucntion.



(d) Charge distribution solved by the standarrd method after removing singularities

Figure 4-5: Partially overlapping wires and face-induced basis function instantiation

in $\mathcal{P}_{x\text{-}y}([x^j_{b^-} + a(h), x^j_{b^+} - a(h)] \times [y^j_{b^-} + a(h), y^i_{b^+}] \times z^i_{b^+})$ is placed to connect the three arch templates as shown in Figure 4-5(c). A single induced basis function is formed now to represent all the induced charge on the top face induced by M_j. The charge distribution of the same structure solved by the standard method after removing singularities is shown in Figure 4-5(d) for comparison. Even though the charge distribution outside the corner of the incuding wire is not represented by a specific template or basis function, this approach is very accurate and only results in a 1.6% capacitance error. Note that these four pieces, though integrated separately in Chapter 5, only contributes a single unknown to the linear system. Hence, the system size can be expected to be very small. If there is more than one inducing wires present, an additional induced basis function is needed for each inducing wire. The additional induced basis functions can be instantiated in the same way as if all the other inducing wire are not present, and then their best coefficients will be determined when the linear system is solved.

Side-Induced Basis Functions

Since the induced charge distribution on the side face of the induce wire is considered when we extract the optimal arch shapes in Section 4.1.2, side arch and side flat templates are used in an actual extraction problem. The instantiation procedure is similar to the one we had for the face-induced basis functions, but the side-induced basis functions are necessary only when the inducing wire protrudes from the boundary of the induced wire. The wire setup in Figure 4-5(a) is such an example. When we examine each edge of the top face of M_j as we did for the induced basis function on the top face, the right-back edge is found outside the range of $x^i_{b^\pm}$ and $y^i_{b^\pm}$. Although it has no effect for the basis function on the top, its non-overlapping feature indicates that the protrusion of M_j from M_i happens in its direction. The placement of the induced basis function on the side $S_{i,\text{side}}$ of M_i is as the following: first, a pair of side arch templates $T^s_{A,(h,w)}(x - x_{b^+}, z)$ and $T^s_{A,(h,w)}(-x - x_{b^-}, z)$ are placed with the supports $(x, z) \in \mathcal{P}_{x\text{-}z}([x^j_{b^+} - a_s(h), x^j_{b^+} + b_s(h)] \times [z^i_c, z^i_{b^+}] \times y^i_{b^+} \cap S_{i,\text{side}})$ and $\mathcal{P}_{x\text{-}z}([x^j_{b^-} - b_s(h), x^j_{b^-} + a_s(h)] \times [z^i_c, z^i_{b^+}] \times y^i_{b^+} \cap S_{i,\text{side}})$, respectively, where z^i_c and

$z^i_{b^+}$ denote the center point of M_i and the upper boundary in z-direction, respectively. Second, we connect them by placing a flat template supported in $\mathcal{P}_{x\text{-}z}([x^j_{b^-} + a_s(h), x^j_{b^+} - a_s(h)] \times [z^i_c, z^i_{b^+}] \times y^i_{b^+})$. This step completes the instantiation of the induced basis function on the side face. The placement looks the same as the one on the side face when we extract the arch templates in Figure 4-1.

4.2.2 The Merge Condition

The merge condition happens when the inducing wire is so narrow that the function supports of the two arch templates are merged and the center flat template disappears. The charge distribution induced by the narrowest wire in Figure 3-4 is an example. When the condition happens, the procedure described above needs a simple modification: instead of placing each arch template one after another, a pair of arch templates varying in the same direction is considered together. If the width of the inducing wire is narrower than $2a(h)$, then the function support of each arch template only extends inwards to the center line of the inducing wire, and the corresponding center flat template is discarded. This modified procedure is listed in Algorithm 1.

4.2.3 The Complete Algorithm

Suppose the two conductors M_i and M_j are both simple straight wires in different metal layers, as the case in Figure 4-5(a). At most five induced basis functions need to be instantiated: one face-induced basis function on the top and four side-induced basis functions on the sides. The algorithm of placing a face-induced basis function and four side-induced basis functions are listed in Algorithm 2 and 3, respectively. They are the general form of the procedures we demonstrated in the example of the partially overlapping wires in Figure 4-5(a). Note that a template function supported in an empty set in the algorithm description is automatically considered discarded.

When M_i and M_j are in the same metal layer, we can easily extend the above algorithm in the following way: a face-induced basis function is instantiated on the side face of M_i when M_i and M_j overlap "horizontally," i.e., in their projection onto

the x-z plane or y-z plane. The side-induced basis functions are generally not included because the metal thickness in the same layer is approximately equal and no wire significantly protrudes from another in the z-direction.

For general wiring of a layout in Manhattan geometry, connected conductors can be decomposed into several straight wire segments. We apply the above algorithm to each pair of wire segments from different conductors. Note that in this algorithm, only several geometry operations are used, such as union and intersection of rectangle shapes, which can be implemented as simple coordinate comparisons and hence computed efficiently. Even when extracting a wire geometry that involves multiple conductors, each of which contains several segments, the computation complexity of instantiating basis functions is at the level of number of segments, usually fewer than one hundred in our target problem. Compared to the overheads of FASTCAP and pre-corrected FFT, which are at the level of discretization, the overhead in our approach is extremely small.

Algorithm 1: Instantiate arch templates in the u-direction

input : A pair of edges $E^{\pm}$ of $S_{j,\text{top}}$ in the u-direction, separation distance h, and $u^i_{b^\pm}, v^i_{b^\pm}, z^i_{b^\pm}$, $u^j_{b^\pm}, v^j_{b^\pm}, z^j_{b^\pm}$, the boundaries of M_i, M_j

output: A pair of arch templates $T^{\pm}_u$ with the u-direction variation

define : T^+ and T^- are an arch and a reflected arch template, respectively.

begin

$[v_1, v_2] \longleftarrow [v^i_{b^-}, v^i_{b^+}] \cap [v^j_{b^-}, v^j_{b^+}]$

if `layer(`M_j`) - layer(`M_i`)` > 0 **then** $s \leftarrow +$ **else** $s \leftarrow -$

if $u^j_{b^+} - u^j_{b^-} < 2a(h)$ **then**

$u^j_c \longleftarrow (u^j_{b^+} + u^j_{b^-})/2$

$S^- \longleftarrow [u^j_{b^-} - b(h), u^j_c] \times [v_1, v_2] \times z^i_{b^s} \cap S_{i,\text{top}}$

$S^+ \longleftarrow [u^j_c, u^j_{b^+} + b(h)] \times [v_1, v_2] \times z^i_{b^s} \cap S_{i,\text{top}}$

else

$S^- \longleftarrow [u^j_{b^-} - b(h), u^j_{b^-} + a(h)] \times [v_1, v_2] \times z^i_{b^s} \cap S_{i,\text{top}}$

$S^+ \longleftarrow [u^j_{b^+} - a(h), u^j_{b^+} + b(h)] \times [v_1, v_2] \times z^i_{b^s} \cap S_{i,\text{top}}$

$T^-_u \longleftarrow T_{A,\mathbf{p}}(-u - u^j_{b^-}, v)$ supported in $\mathcal{P}_{u\text{-}v}(S^-)$

$T^+_u \longleftarrow T_{A,\mathbf{p}}(u - u^j_{b^+}, v)$ supported in $\mathcal{P}_{u\text{-}v}(S^+)$

end

Algorithm 2: Instantiate a face-Induced Basis Function

input : $S_{i,\text{top}}$, $S_{j,\text{top}}$ of M_i and M_j, respectively

output: A face-induced basis function $\psi(\mathbf{r})$

define : The support of a template, $\text{supp}(T)$, is bounded by $x_{b^\pm}(T), y_{b^\pm}(T)$, where $x_{b^\pm}(T), y_{b^\pm}(T) \equiv \pm\infty$ if $\text{supp}(T) = \emptyset$

begin

Instantiate the arch templates $T^{\pm}_x$ and $T^{\pm}_y$ on $S_{i,\text{top}}$ induced by $S_{j,\text{top}}$

$S \longleftarrow [x_{b^+}(T^-_x), x_{b^-}(T^+_x)] \times [y_{b^+}(T^-_y), y_{b^-}(T^+_y)] \times z^i_{b^\pm} \cap S_{i,\text{top}}$

$T_0 \longleftarrow T_F(x, y)$ supported in $\mathcal{P}_{x\text{-}y}(S)$

$\psi(\mathbf{r}) \longleftarrow T_0 + T^-_x + T^+_x + T^-_y + T^+_y$

end

Algorithm 3: Instantiate Side-Induced Basis Functions

input : $E_{\pm x}, E_{\pm y}$ of $S_{j,\text{top}}$, $S_{i,\pm x}, S_{i,\pm y}$ of M_i, separation distance h, and $u^i_{b^\pm}, v^i_{b^\pm}, z^i_{b^\pm}$, $u^j_{b^\pm}, v^j_{b^\pm}, z^j_{b^\pm}$, the boundaries of M_i, M_j

output: A set of side-induced basis functions Ψ

define : $E_{s(v)}$ is an edge of $S_{j,\text{top}}$ on the $s(v)$-direction side.
$S_{i,s(v)}$ is the face of M_i facing the $s(v)$-direction.
$|S_{i,\text{top}}| \equiv [u^i_{b^-}, u^i_{b^+}] \times [v^i_{b^-}, v^i_{b^+}] \times \mathbb{R}$

begin

- $\Psi \longleftarrow \emptyset$
- **switch** `layer(`M_j`)-layer(`M_i`)` **do**
  - **case** *1,-1*
    - $[z_1, z_2] \longleftarrow$ the closest first half segment of $[z^i_{b^-}, z^i_{b^+}]$
  - **case** *2,-2*
    - $[z_1, z_2] \longleftarrow$ the closest first quarter segment of $[z^i_{b^-}, z^i_{b^+}]$
  - **otherwise**
    - $[z_1, z_2] \longleftarrow \emptyset$
- **foreach** $v \in \{x, y\}$, $s \in \{+, -\}$ **do**
  - **if** $E_{s(v)} \cap |S_{j,top}| = \emptyset$ **then**
    - **if** $u^j_{b^+} - u^j_{b^-} < 2a(h)$ **then**
      - $u^j_c \longleftarrow (u^j_{b^+} + u^j_{b^-})/2$
      - $S^- \longleftarrow [u^j_{b^-} - b(h), u^j_c] \times [z_1, z_2] \times v^i_{b^s} \cap S_{i,s(v)}$
      - $S^+ \longleftarrow [u^j_c, u^j_{b^+} + b(h)] \times [z_1, z_2] \times v^i_{b^s} \cap S_{i,s(v)}$
    - **else**
      - $S^- \longleftarrow [u^j_{b^-} - b(h), u^j_{b^-} + a(h)] \times [z_1, z_2] \times v^i_{b^s} \cap S_{i,s(v)}$
      - $S^+ \longleftarrow [u^j_{b^+} - a(j), u^j_{b^+} + b(h)] \times [z_1, z_2] \times v^i_{b^s} \cap S_{i,s(v)}$
    - $T^- \longleftarrow T^s_{A,\mathbf{p}}(-u - u^j_{b^-}, z)$ supported in $\mathcal{P}_{u\text{-}z}(S^-)$
    - $T^+ \longleftarrow T^s_{A,\mathbf{p}}(u - u^j_{b^+}, z)$ supported in $\mathcal{P}_{u\text{-}z}(S^+)$
    - $S \longleftarrow [u_{b^+}(T^-), u_{b^-}(T^+)] \times [z_1, z_2] \times v^i_{b^s} \cap S_{i,s(v)}$
    - $T_0 \longleftarrow T_F(u, z)$ supported in $\mathcal{P}_{u\text{-}z}(S)$
    - $\psi(\mathbf{r}) \longleftarrow T_0 + T^+ + T^-$
    - $\Psi \longleftarrow \Psi \cup \{\psi(\mathbf{r})\}$

end

Chapter 5

Implementation

In this chapter, we discuss the implementation of FastCaplet from the system setup to the system solving step. In particular, in the system setup step, we propose four efficient panel-to-panel integration strategies for computing the Galerkin's integration.

5.1 System Setup

In Chapter 4, all the basis functions are instantiated by performing Algorithm 2 and 3. Next, we construct the system of linear equations $Pq = \Phi$.

5.1.1 Integration Schemes

The bracketed integration of the i, j-th entry of P in Eq.(2.6) is reproduced here. We assume the basis functions $\psi_i(\mathbf{r})$ and $\psi_j(\mathbf{r}')$ are functions of x and y with the position vector $\mathbf{r}$ and $\mathbf{r}'$ substituted by (x, y, z) and (x', y', z'), respectively.

$$p_{i,j} = \int_{s_i}\int_{s_{j'}} \frac{\psi_i(x,y)\psi_j(x',y')}{4\pi\varepsilon\sqrt{(x-x')^2+(y-y')^2+(z-z')^2}}\mathrm{d}x'\mathrm{d}y'\mathrm{d}x\mathrm{d}y. \tag{5.1}$$

Since a basis function $\psi_i(x, y)$ is the summation of several templates, and each template $T_{i,l}(x, y)$ is supported in a rectangle region $[x_{l,1}, x_{l,2}] \times [y_{l,1}, y_{l,2}]$, Eq.(5.1) can be

decomposed into

$$p_{i,j} = \sum_l \sum_m \left[\int_{y_{l,1}}^{y_{l,2}} \int_{x_{l,1}}^{x_{l,2}} \int_{y'_{m,1}}^{y'_{m,2}} \int_{x'_{m,1}}^{x'_{m,2}} \frac{T_{i,l}(x,y) T_{j,m}(x',y')}{4\pi\varepsilon\sqrt{(x-x')^2+(y-y')^2+(z-z')^2}} \mathrm{d}x'\mathrm{d}y'\mathrm{d}x\mathrm{d}y \right]. \tag{5.2}$$

When $T_{i,l}$ and $T_{j,m}$ only vary in the x-direction and the y-direction, respectively, we can rearrange the bracketed integration in Eq.(5.2) into

$$\int_{x_{l,1}}^{x_{l,2}} T_{i,l}(x) \int_{y'_{m,1}}^{y'_{m,2}} T_{j,m}(y') \left[\int_{y_{l,1}}^{y_{l,2}} \int_{x'_{m,1}}^{x'_{m,2}} \frac{1}{4\pi\varepsilon\sqrt{(x-x')^2+(y-y')^2+(z-z')^2}} \mathrm{d}x'\mathrm{d}y \right] \mathrm{d}y'\mathrm{d}x. \tag{5.3}$$

The inner 2D integration is identical to the integration of the collocation testing in Eq.(2.4), which is analytically integrable and well-behaved. The rest of the outer 2D integration can be integrated numerically by Gaussian quadrature. Since the integrations involve two panels, which may orient in different directions, three other similar schemes are shown in the category of Arch-Arch integration of Type 1 in Figure 5-1. If the inner 2D integration happens to integrate the same x-direction or y-direction twice, referred as Arch-Arch Integration of Type 2, the analytical expression is available but it is not well-behaved when the two templates overlap. Therefore, Gaussian quadrature is not accurate. In this case, we subdivide the overlapping region into several small pieces and integrate each piece as a flat template.

The other two categories, Arch-Flat integration and Flat-Flat integration, are used when only one template is flat and both templates are flat, respectively. The corresponding 3D and 4D analytical integrations are employed in each case. All these schemes are summarized in Figure 5-1.

5.1.2 Integration Strategies

The integration schemes mentioned above are all based on the analytical expressions of 2D, 3D or 4D integrations. As a result, we can speed up the overall system setup process if we accelerate the computation of these integrations. Here, we propose four integration strategies.

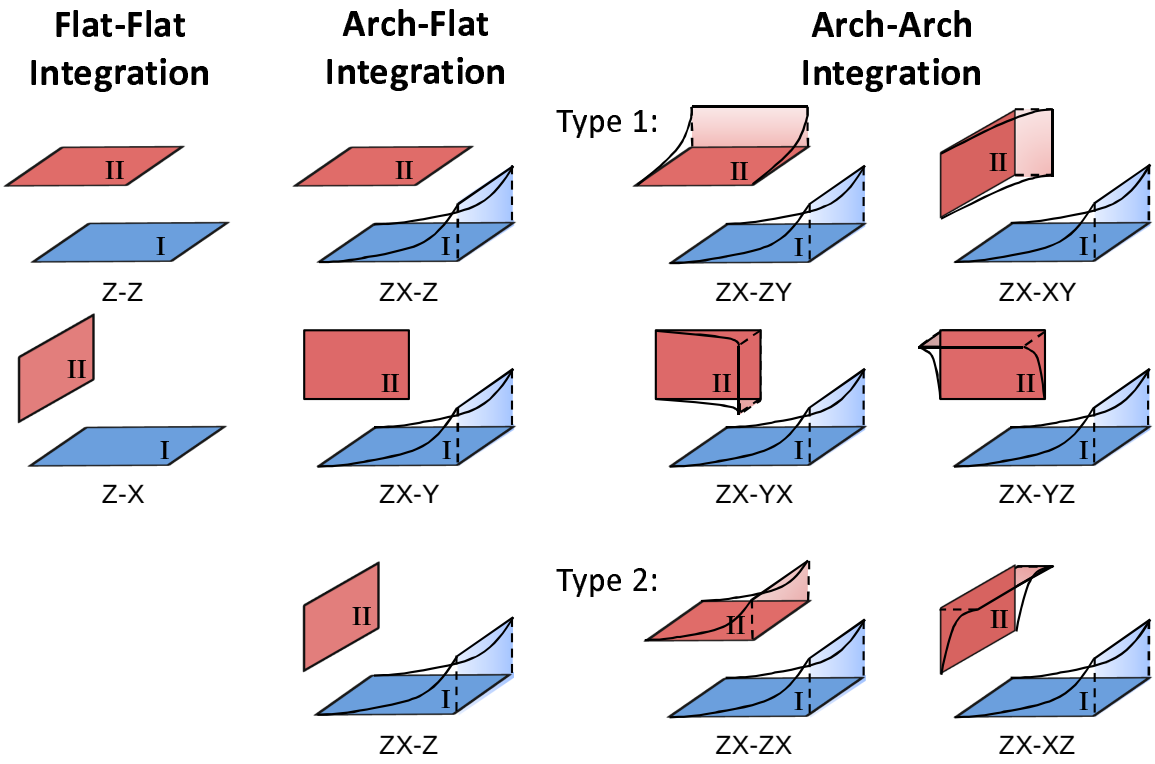


Figure 5-1: A list of integration schemes: Flat-Flat, Arch-Flat, Arch-Arch integrations. Each template is denoted by either a single letter if it is flat, or two letters if it is an arch template. The first letter denotes the direction of the panel, and the second letter denotes the direction of function variation.

Tabulation of Definite Integration

In this method, we tabulate the analytical expression, for example, in the 4D case,

$$F_{\text{definite}} = \int_{y_1}^{y_2}\int_{x_1}^{x_2}\int_{y_1'}^{y_2'}\int_{x_1'}^{x_2'} \frac{1}{4\pi\varepsilon\|\mathbf{r}-\mathbf{r}'\|}\mathrm{d}x'\mathrm{d}y'\mathrm{d}x\mathrm{d}y. \tag{5.4}$$

After being properly normalized, it is a six-parameter table. The error of this method can be easily controlled, but it involves a huge table which is feasible only when sampled carefully. Its acceleration is also limited due to the expensive six-dimensional linear interpolation.

Tabulation of Indefinite Integration

Instead of tabulating the whole definite integration expression, we can reduce the tabulation parameters from six to three by tabulating the integration without substituting the upper and lower limits

$$F_{\text{indefinite}}(x-x',y-y',z)\ |_{x_1'}^{x_2'}|_{y_1'}^{y_2'}|_{x_1}^{x_2}|_{y_1}^{y_2} = \int_{y_1}^{y_2}\int_{x_1}^{x_2}\int_{y_1'}^{y_2'}\int_{x_1'}^{x_2'} \frac{1}{4\pi\varepsilon\|\mathbf{r}-\mathbf{r}'\|}\mathrm{d}x'\mathrm{d}y'\mathrm{d}x\mathrm{d}y. \tag{5.5}$$

In this case, though the number of parameters is reduced, the table size is still large because the overall integration requires very accurate indefinite integration results. Also, the error of the overall integration is indirectly related to the table. The error control is not as easy as the previous method.

Tabulation of Expensive Elementary Functions

In the 2D, 3D, and 4D analytical expressions, more than half of the computation time is spent on computing elementary functions, such as arctangent and logarithmic functions. They are expensive, taking roughly 30 times more computation time than a multiplication of two single-precision floating points. Tabulating these single parameter functions is memory efficient even when the accuracy requirement is stringent. The IEEE-754 representation of floating-point numbers is also useful for the

tabulation of the logarithmic function

$$\log_2(\{\text{mantissa}\} \times 2^{\{\text{exponent}\}}) = \{\text{exponent}\} + \log_2(\{\text{mantissa}\}).$$

Therefore, only $\log_2(\{\text{mantissa}\})$ needs to be tabulated[22], and in our experiments, tabulating the first 14 bits of mantissa is enough for a 1% overall integration error in a 4D integration. This approach is faster than the built-in logarithmic function and arctangent function by a factor of five and four, respectively, on a Xeon 3.2 GHz machine.

System Identification of Integrations

It can be seen from Eq.(5.5) that an analytical expression of an integration is an ill-conditioned representation. Several significant digits of the indefinite integration are canceled out when substituting the upper and the lower limits. A more efficient and numerically stable expression, such as rational functions in Eq.(4.1), can be adopted. In this case, we treat the equation (5.4) as a six-input-single-output system without feedback, and use STINS[3] to identify the "model" of the integration. In order to fully characterize the integration, a large number of sampling points are required. Hence, it is necessary to have an appropriate selection of sampling points to make the identification process feasible.

Considering the compatibility with the original analytical expressions and the memory efficiency, we will use the method of tabulating elementary functions in the examples in Chapter 6. Although it is a partial tabulation of the whole integration, it is, in fact, slightly faster than the direct tabulation of the whole integration in our implementation of the 2D integration.

5.2 System Solving

In the system solving part, we use a standard LU decomposition method. Among many optimized linear algebra libraries, we choose a GotoBLAS-based LAPACK li-

brary in our implementation [10][2]. The reason of this selection is based on performance and portability. As we will see in Chapter 6, the selection of the linear system solver is not critical because most of the computation is moved to the system setup part by using the instantiable basis functions.

Chapter 6

Examples

In this chapter, we first describe the methodology of performance comparison and, then, present several extraction examples of common interconnect structures in a VLSI layout.

6.1 Performance Comparison Methodology

In the following examples, we first use FASTCAP to extract the reference capacitance matrix C^0 of a structure by adaptively refining the discretization until the capacitance matrix is converged. In the adaptive process, we increase the number of panels by 10% in each step. We stop the process and consider the result converged when each entry of the matrix in the last step is less than 0.1% different from the same entry in the previous step. The convergence criterion used here requires a very large number of panels that the reference capacitance solution is not feasible when using the standard method because of its $O(N^2)$ memory requirement. As a result, we choose FASTCAP instead of the standard method to extract the reference capacitance.

Second, we extract the capacitance matrix C^{Caplet} of the same structure by using our implementation, FastCaplet, and calculate the relative error of each entry with respect to the diagonal entry

$$e_{ij} = \frac{C_{ij}^{\text{Caplet}} - C_{ij}^0}{C_{ii}^0}.$$

Table 6.1: Comparison of extraction methods in the comb capacitor example

Method	FastCaplet	FASTCAP	Impr.	PWC Collocation	Impr.
Total time (ms)	55.8	1040	18.6×	2310	41.2×
Setup time (ms)	54.8	670	12.2×	133	23.3×
Solving time (ms)	1.00	370	370×	980	980×
Number of unknowns	212	5640	26.6×	5640	26.6×
Memory	180 KB	74 MB	410 ×	255 MB	1410×
Error	2.08%	2.24%		1.76%	

The capacitance error of the structure extracted by FastCaplet is the largest absolute value of e_{ij}

$$e^{\text{Caplet}} = \max|e_{ij}|.$$

Finally, we adjust the discretization in order to make the error of the capacitance extracted by FASTCAP with respect to C^0 is equal to e^{Caplet}. We then make the performance comparison between this work and FASTCAP in this discretization. A similar procedure is followed for the standard method.

6.2 Examples

All the results of the examples demonstrated below are computed on an Intel Xeon W3570 3.2GHz machine.

6.2.1 A Comb Capacitor

Figure 6-1 shows a comb capacitor. The comparison of different extraction methods is listed in Table 6.1. We compare this work with FASTCAP and the standard method, which is called the piecewise constant (PWC) collocation method in the table. The improvement factor is calculated and listed to the right of the column of each method.

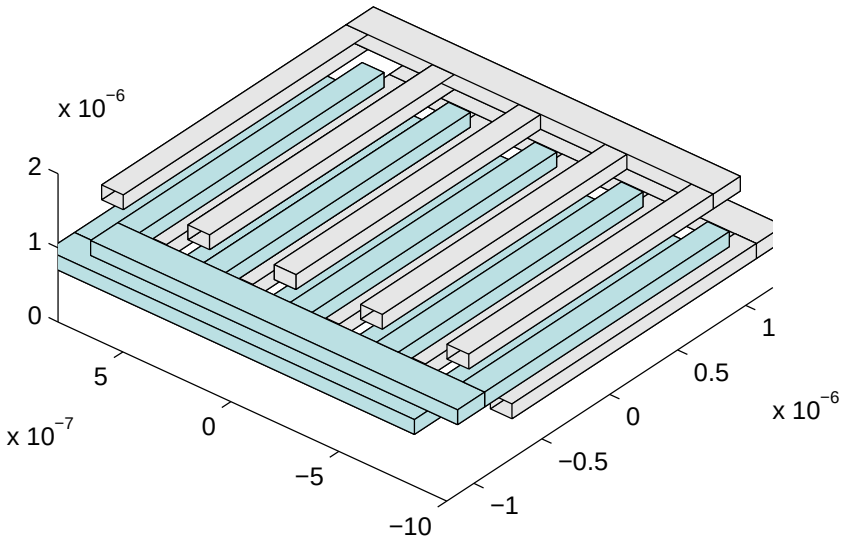


Figure 6-1: A comb capacitor.

6.2.2 Three-by-Three Buses

Figure 6-2 shows an example of 3-by-3 buses. It is a very common structure in a circuit layout and known to be inaccurate when using 2D scanning and table lookup approach. The widths of the wires in this example are intentionally set to be different in order to demonstrate the versatility of the instantiable basis functions. The performance comparison is summarized in Table 6.2.

Table 6.2: Comparison of extraction methods in the 3x3-bus example

Method	FastCaplet	FASTCAP	Impr.	PWC Collocation	Impr.
Total time (ms)	30.2	380	12.6×	3900	129×
Setup time (ms)	29.5	180	6.1×	2840	96.3×
Solving time (ms)	0.70	200	285×	1060	1510×
Number of unknowns	90	2692	30×	2692	30×
Memory	37.4 KB	16.4 MB	439×	58.2 MB	1560×
Error	2.38%	2.94%		3.06%	

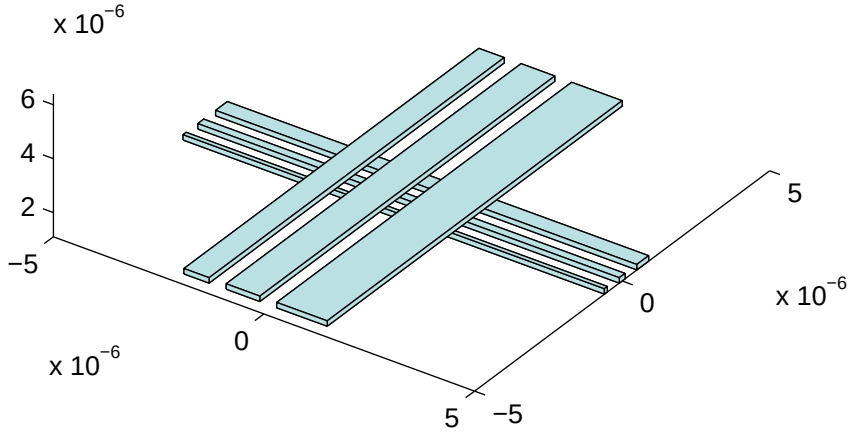


Figure 6-2: Three-by-three buses. The length and thickness of the wires are fixed to be 10 and 0.2 um. The width of each wire is 0.1, 0.2, 0.4, 0.6, 0.8, and 1.2 um, respectively. The separation between wires is 0.2 um. The two metal layers are separated by 0.2 um.

6.2.3 A Spiral Inductor

Figure 6-3 shows a spiral inductor. Spiral inductors are common structures in a RF circuit layout. In addition to the inductance, the parasitic capacitance is also of great interest because, for example, it directly affects the operating frequency of a LC-tank-based oscillator. The metal ground plane is patterned in order to reduce the induced eddy current but increase the capacitance between the inductor and the ground plane. The performance comparison is summarized in Table 6.3.

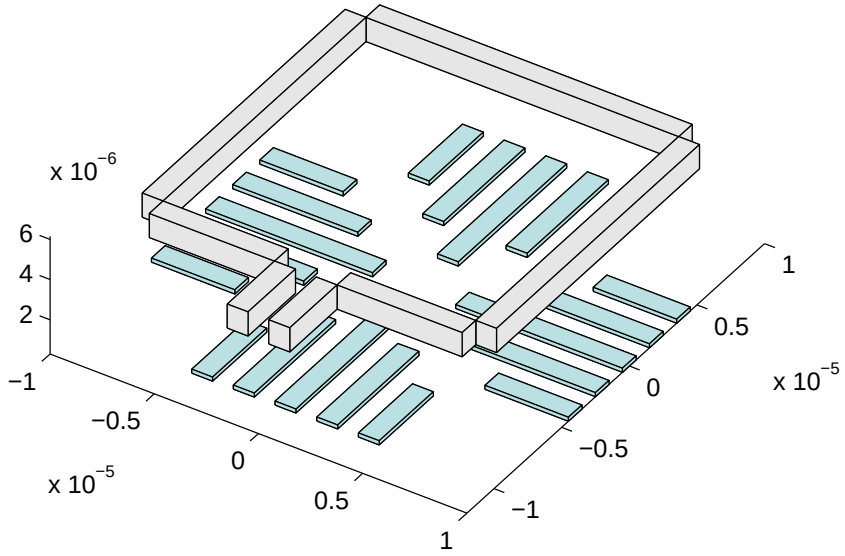


Figure 6-3: A spiral inductor.

Table 6.3: Comparison of extraction methods in the spiral inductor example

Method	FastCaplet	FASTCAP	Impr.	PWC Collocation	Impr.
Total time (ms)	41.6	650	15.6×	7250	90.1×
Setup time (ms)	40.6	390	9.6×	1734	42.7×
Solving time (ms)	1.00	260	260×	2010	2010×
Number of unknowns	197	7250	36.8×	7250	36.8×
Memory	150 KB	41.0 MB	270 ×	420 MB	2800×
Error	1.46%	1.82%		1.80%	

Chapter 7

Conclusions

We propose a new set of basis functions for charge distribution, which is very effective to capture dominant induced charge. These basis functions are instantiated on the fly by analyzing wire geometries and assembling template components. We use only two important templates: the arch template and the flat template. The arch template is defined with 1D variation, which is beneficial both for storing the shapes and for computing the Galerkin's integrations efficiently. Our implementation, FastCaplet, is based on these basis functions. In our examples, it is more than 10 times faster than FASTCAP, and roughly 100 times faster than the boundary element method using piecewise constant basis functions with the collocation testing. The memory usage of our method is very efficient: only less than 1/100 storage of the FASTCAP is needed. It is also very accurate, generating only 3% capacitance errors demonstrated in our examples.

Future Work

Using our new template instantiable basis functions is efficient because the resultant field solver is faster than the existing accelerated field solvers in small structures. Even more, our method moves most of the computational efforts from system solving to system setup. Widely accepted methods, such as [21] and [24], spend more than 50% of computational time in solving system in small structures and more than 90% in large structures. Parallelization of these specialized system solving processes is not

trivial, and the acceleration is very limited [33][1]. In our approach, on the other hand, more than 90% of computation time is spent on the system setup step, which is embarrassingly parallelizable. As a result, an efficient parallelization of our current method will surely have an even bigger improvement in computational time.

In addition, we assume all conductors are embedded in a uniform dielectric material. However, in a common fabrication technology, there are several very thin sub-layers with different dielectric constants between metal layers. Traditionally, extracting capacitance including the effects of sub-layers is computationally expensive. It can be useful to extend the use of the instantiable basis functions to layered dielectric structures.

Bibliography

[1] N. R. Aluru, V. B. Nadkarni, and J. White. A parallel precorrected fft based capacitance extraction program for signal integrity analysis. In *Proceedings of the 33rd annual Design Automation Conference*, DAC '96, pages 363–366, New York, NY, USA, 1996. ACM.

[2] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammerling, J. Demmel, C. Bischof, and D. Sorensen. Lapack: a portable linear algebra library for high-performance computers. In *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, Supercomputing '90, pages 2–11, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[3] B. Bond. *STINS: A MATLAB tool for stable identification of nonlinear systems via semidefinite programming.* [Online]. Available: http://www.rle.mit.edu/cpg/codes/stins.

[4] B.N. Bond, Z. Mahmood, Yan Li, R. Sredojevic, A. Megretski, V. Stojanovi, Y. Avniel, and L. Daniel. Compact modeling of nonlinear analog circuits using system identification via semidefinite programming and incremental stability certification. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(8):1149 –1162, aug. 2010.

[5] X. Cai, K. Nabors, and J. White. Efficient galerkin techniques for multipole-accelerated capacitance extraction of 3-d structures with multiple dielectrics. pages 200 –211, mar. 1995.

[6] Y.L. Le Coz and R.B. Iverson. A stochastic algorithm for high speed capacitance extraction in integrated circuits. *Solid-State Electronics*, 35(7):1005 – 1012, 1992.

[7] L. Daniel, A. Sangiovanni-Vincentelli, and J. White. Interconnect electromagnetic modeling using conduction modes as global basis functions. In *Electrical Performance of Electronic Packaging, 2000, IEEE Conference on.*, pages 203–206, 2000.

[8] L. Daniel, A. Sangiovanni-Vincentelli, and J. White. Proximity templates for modeling of skin and proximity effects on packages and high frequency interconnect. In *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, pages 326–333, Nov. 2002.

[9] R. De Smedt and J. Van Bladel. Field singularities at the tip of a metallic cone of arbitrary cross section. *Antennas and Propagation, IEEE Transactions on*, 34(7):865–870, Jul 1986.

[10] Kazushige Goto and Robert A. van de Geijn. Anatomy of high-performance matrix multiplication. *ACM Trans. Math. Softw.*, 34:12:1–12:25, May 2008.

[11] Ki Jin Han, E. Engin, and M. Swaminathan. Cylindrical conduction mode basis functions for modeling of inductive couplings in system-in-package (sip). In *Electrical Performance of Electronic Packaging, 2007 IEEE*, pages 361–364, Oct. 2007.

[12] Yu-Chung Hsiao, T. El-Moselhy, and L. Daniel. Efficient capacitance solver for 3d interconnect based on template-instantiated basis functions. In *Electrical Performance of Electronic Packaging and Systems, 2009. EPEPS '09. IEEE 18th Conference on*, pages 179–182, Oct. 2009.

[13] Xin Hu, Tarek Moselhy, Jacob White, and Luca Daniel. Optimization-based wideband basis functions for efficient interconnect extraction. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '07, pages 1200–1205, San Jose, CA, USA, 2007. EDA Consortium.

[14] Ching-Chao Huang. Galerkin's method with triangular patchesin three-dimensional capacitance calculation. pages 70 –72, apr. 1992.

[15] W. Imbriale and P. Ingerson. On numerical convergence of moment solutions of moderately thick wire antennas using sinusoidal basis functions. *Antennas and Propagation, IEEE Transactions on*, 21(3):363–366, May 1973.

[16] Won-Young Jung, Ghun-Up Cha, Young-Bae Kim, Jun-Ho Baek, and Choon-Kyung Kim. Integrated interconnect circuit modeling for vlsi design. pages 165 –169, aug. 1995.

[17] S. Kapur and D.E. Long. Ies3: efficient electrostatic and electromagnetic simulation. *Computational Science & Engineering, IEEE*, 5(4):60–67, Oct-Dec 1998.

[18] J. S. C. Kilby. Turning potential into realities: The invention of the integrated circuit (nobel lecture). *ChemPhysChem*, 2:482–489, 2001.

[19] Y. P. R. Lamy, K. B. Jinesh, F. Roozeboom, D. J. Gravesteijn, and W. F. A. Besling. Rf characterization and analytical modelling of through silicon vias and coplanar waveguides for 3d integration. *Advanced Packaging, IEEE Transactions on*, PP(99):1 –1, 2010.

[20] David D. Ling and Albert E. Ruehli. Interconnection modeling. *Circuit Analysis, Simulation, and Design — Advances in CAD for VLSI*, Vol. 3, Part II:Chapter 11.

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

[22] N. Mirghafori O. Vinyals, G. Friedland. Revisiting a basic function on current cpus: A fast logarithm implementation with adjustable accuracy. *International Computer Science Institute*, 2007.

[23] S. Ortiz and R. Suaya. Fullwave volumetric maxwell solver using conduction modes. In *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, pages 13–18, Nov. 2006.

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

[25] A. Ringhandt and H.G. Wagemann. An exact calculation of the two-dimensional capacitance of a wire and a new approximation formula. *Electron Devices, IEEE Transactions on*, 40(5):1028 –1032, may. 1993.

[26] V Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60(2):187 – 207, 1985.

[27] Youcef Saad and Martin H Schultz. Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[28] Xiaomeng Shi, Jian-Guo Ma, Kiat Seng Yeo, Manh Anh Do, and Erping Li. Equivalent circuit model of on-wafer cmos interconnects for rfics. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(9):1060 – 1071, sep. 2005.

[29] P. Silvester. Modal network theory of skin effect in flat conductors. *Proceedings of the IEEE*, 54(9):1147–1151, Sept. 1966.

[30] Y Su, E T Ong, and K H Lee. Automatic classification of singular elements for the electrostatic analysis of microelectromechanical systems. *Journal of Micromechanics and Microengineering*, 12(3):307–315, 2002.

[31] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Math, 1997.

[32] G. Vecchi. Loop-star decomposition of basis functions in the discretization of the efie. *Antennas and Propagation, IEEE Transactions on*, 47(2):339–346, Feb 1999.

[33] Yanhong Yuan and Prith Banerjee. A parallel implementation of a fast multipole-based 3-d capacitance extraction program on distributed memory multicomputers. *Journal of Parallel and Distributed Computing*, 61(12):1751–1774, 2001.

[34] Yimin Zhang and A.H. Zemanian. Contributions of corner singularities of the capacitances of interconnections wires. In *Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on*, volume 2, pages 1420–1423 vol.2, Apr-3 May 1995.