Implementation of Algorithms to Determine the Capacitance Sensitivity of Interconnect Parasitics in the Magic VLSI Layout Tool

by

Nick Kuan-Hsiang Huang
B.A.Sc., The University of British Columbia, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The College of Graduate Studies

(Electrical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

September 2009

Abstract

VLSI interconnect capacitance is becoming more significant and also increasingly subject to process variation in the deep submicron regime. A new set of capacitance models is implemented in the Magic VLSI layout tool to improve the capacitance accuracy based on 2.5D capacitance models. This involves a new technology file, equations, and search algorithms. In addition, a simple technique to extract from layout the sensitivity of interconnect parasitic capacitance to linewidth process variation is proposed based on the new capacitance models and implemented in Magic. The derivative of each extracted capacitance with respect to linewidth variation in every level is obtained. Coincident edges in layout result in distinct "shrinking" and "bloating" derivatives. The derivatives therefore form a gradient that may be multiplied by a vector of the variations on each level to give the total expected deviation from nominal capacitance. The gradient allows the process sensitivity of each capacitance to be determined by simply inspecting the netlist. In the end, the impact of process variation is simulated in a crosstalk application to emphasize the necessity of process variation awareness.

Table of Contents

List of Tables

List of Figures

List of Acronyms

3D	**T**hree-**D**imensional
AMS	**A**nalog/**M**ixed-**S**ignal
ASIC	**A**pplication-**s**pecific **I**ntegration **C**ircuit
BEM	**B**oundary-**E**lement **M**ethod
CAD	**C**omputer-**a**ided **D**esign
CMP	**C**hemical **M**echanical **P**olishing
CPLD	**C**omplex **P**rogrammable **L**ogic **D**evice
CPU	**C**entral **P**rocessing **U**nit
DRC	**D**esign-**R**ule **C**heck
DSM	**D**eep **S**ub**m**icron
EDA	**E**lectronic **D**esign **A**utomation
FD	**F**inite-**D**ifference
FE	**F**inite-**E**lement
FFT	**F**ast **F**ourier **T**ansform
FMM	**F**ast **M**ultipole **M**ethod
FHM	**F**ast **H**ierarchical **M**ethod
HILEX	**HI**erarchical **L**ayout **EX**traction
IC	**I**ntegrated **C**ircuit
LER	line **e**dge **r**oughness
LFD	**L**itho-**F**riendly **D**esign
LM	**L**evenberg-**M**arquardt
LSI	**L**arge-**S**cale **I**ntegration

MOS	**M**etal-**O**xide-**S**emiconductor
MSI	**M**edium-**S**cale **I**ntegration
nm	**n**ano-**m**eter
OPC	**O**ptical **P**roximity **C**orrection
QMM	**Q**uasi-**M**ultiple **M**edium
RF	**R**adio-**F**requency
RWM	**R**andom **W**alk **M**ethod
SI	**S**ignal **I**ntegrity
SoC	**S**ystem-**o**n-**C**hip
SSI	**S**mall-**S**cale **I**ntegration
SVD	**S**ingular **V**alue **D**ecomposition
UDSM	**U**ltra-**D**eep **S**ub**m**icron
VLSI	**V**ery-**L**arge-**S**cale **I**ntegration

Acknowledgements

I would first like to thank my academic supervisor, Dr. Andrew Labun, for the guidance, technical advice, support, dedication, and patience. This work would not have been possible without his help.

I would then like to express my thanks to Dr. Roberto Musedere from the University of Windsor for his willingness to serve as my external examiner and Dr. Peter Hallschmid and Dr. Jonathan Holzman for serving on the committee. I really appreciate their valuable time and constructive comments on my thesis.

I would also like to thank my friends and colleagues in School of Engineering for making my graduate life enjoyable at UBC Okanagan and providing a friendly work environment in the office. In particular, thank you to Xian Jin, Chiun-Shen Liao, Xuegui Song, Ning Wang, Junfeng Zhao, James Sun, Yeyuan Xiao, Mingbo Niu, and a short-time colleague, Carl Wong.

Special thanks are owed to my family, especially my parents, for their unwavering love and continual support throughout my years of education.

Chapter 1

Introduction

1.1 Motivation

Nowadays, as integrated circuit (IC) manufacturing technology improves into the deep submicron (DSM) regime, interconnect has been identified as one of the most critical challenges for IC designers [1]. In the past, chip performance was determined by the transistor performance. However, as the technology continues to scale down, interconnect behavior becomes indispensable in the DSM domain and will dominate the overall chip performance, reliability, and cost eventually. Accuracy in predicting the performance of a design becomes the first step in this interconnect-dominated technology [2–4].

The 2007 International Technology Roadmap for Semiconductor addresses "three dimensional (3D) control of interconnect features" among the five "interconnect difficult challenges" both for technology generation greater or equal and less than 22 nano-meters (nm),

and it is the only challenge in near term (2007-2013) and distinct future (2014-2022). Contributions to dimensional variation include "line edge roughness (LER), trench depth and profile, via shape, etch bias, thinning due to cleaning, and chemical mechanical polishing (CMP) effects" [5, 6].

As the shrinking of geometric dimension and change of physical properties affect the sensitivity and increase the variability of resistance and capacitance, the accuracy of estimates of interconnect parasitics due to resistances and capacitances variability can have significant impacts on the performance and reliability in very-large-scale integration (VLSI) circuits of semiconductor technology development [7–9].

1.2 Research Goals

There are several key challenging problems associated with the scaling technology. One such challenge is the increasing effects of process variation. The typical variations in the interconnect geometries take account of linewidth, thickness, height, and inter-layer dielectric thickness, and these variations may lead to significant disagreement between the conceptual design and real fabricated chip [10]. These discrepancies are collectively referred to as *process variations* [11]. The management of variation is playing an important role in shrinking fundamental dimensions. The continuing increase in operating frequency is also another challenge in chip power density and within-die temperature fluctuations [12, 13].

Both devices and interconnects have become increasingly susceptible to variations during manufacturing processes with the scaled technology. In the past, designers were principally concerned with device variations, which can be captured by worst/best case corner points

techniques [14, 15]. In today's VLSI products, however, process variations are not restricted to device variations. Interconnect variations become more pronounced and dominate over device variations in future technologies [16]. As a result, designers need more accurate parasitic capacitance and resistance values to minimize design margins and cost. Accuracy of capacitance sensitivity extraction is increasingly essential to account for process effects and is a challenging issue [7, 12, 17, 18].

Magic is an open source layout editor. It not only provides user friendly operations, but it is also flexible for users to modify its internal structures. As a result, the idea to implement capacitance process variation into Magic was brought up. Magic contains approximately 9,000 lines of C languages in extraction part [19], which caused the initial understanding of Magic's internal structure to be more difficult than expected. As Magic has been widely in use for lots of universities in education purpose since it was first published, it is certainly a good idea to implement more capabilities to make Magic a more sophisticated layout tool without much cost, and it would be a benefit for education purposes in the future.

In this thesis, the focus is on the improvement of capacitance extraction in Magic and the capacitance sensitivity on the linewidth process variation in interconnect. The demonstration of this method for the first time is done by extracting capacitance sensitivity to linewidth variation from layout.

The goals for this thesis are two-fold: First, CUP capacitance models, a capacitance extraction, are presented and implemented in the Magic VLSI layout tool. Second the derivative of capacitance models are computed for process variation effects on linewidth. The approach to implement CUP models is to understand Magic's internal structures and

apply better formulae to calculate capacitances. To compute the derivatives, it is first necessary to find the real edge of a conductor, which leads to an invented type of edge, the "coincident edge" explained in Chapter 4. New metal search algorithms are created along with a new extraction output format.

The contributions of this thesis are summarized as follows:

1. Implementation of CUP models to improve Magic's capacitance extraction accuracy.
2. Implementation of capacitance derivatives to account for process variation sensitivity.
3. Comparison and evaluation of proposed algorithms.

1.3 Thesis Organization

This thesis is composed of six chapters. Chapter Two starts with the background of interconnect design problems in the DSM regime, and then discusses relevant work done on capacitance extraction and process variation in industrial tools. In Chapter Three, a detailed description of how Magic extracts capacitances is presented that includes a description of Magic's internal structure and behavior. Chapter Four then discusses the implementation of CUP models for capacitance and interconnect capacitance sensitivity into Magic. Real and coincident edges and pixel-based search algorithms are introduced and described. Evaluation is performed in Chapter Five, including simulations with spice tool and comparisons with a commercial capacitance extraction tool. In the end, Chapter Six concludes with the summary, suggestions, and areas for future work.

Chapter 2

Background and Relevant Work

In this chapter, background information and relevant work for this thesis are presented. The first part of this chapter emphasizes interconnect design theory. Before investigating further, a brief historical perspective is introduced. The evolution of ICs began with Small-Scale Integration (SSI) containing only a few transistors, Medium-Scale Integration (MSI) containing hundreds of transistors, Large-Scale Integration (LSI) containing tens of thousands of transistors. In the final stage, VLSI was introduced containing hundreds of thousands to several billions transistors nowadays in a single chip. VLSI technology is divided into several regimes from micron, submicron, deep submicron (DSM), and ultra-deep submicron (UDSM). This section focuses on the extraction of interconnect parasitics for high aspect ratio interconnect, typical of the DSM or UDSM regime.

2.1 Interconnect Design Theory

It is well known that interconnect design is becoming a more and more significant concern in DSM process technologies, and the influence of shrinking interconnect line width and height size fluctuations becomes more severe in the fabrication processes beyond 45 nm [20]. Therefore, a detailed discussion of interconnect background and design is presented in this section.

2.1.1 Deep-Submicron Interconnect

In DSM technology, interconnect has numerous issues that would affect the performance of circuits. IC designers are facing more challenges, such as coupling parasitics, IR drop, and electromigration. However, one of the most widespread challenges for interconnect design is interconnect delay in a critical path [21]. Reference [22] showed an example of delay components for the 65 nm technology, in which the transistor delay component is 43%, the intra-cell resistance (coming from contact, via, diffusion, poly, and metal resistances) and the capacitance (coming from contact-to-gate, poly, metal, and via capacitances) contributes 22% of the total delay, and the inter-cell (between cells) metal and via resistance and capacitance account for 35% of the total delay. It is observed that RC components are a significant fraction and contribute approximate 57% of overall circuit delay.

In addition, optimal buffer insertion methods were used to reduce delay from quadratic to linear. However, this method becomes difficult as the number of buffers increases at a fast rate in DSM technologies [23]. As size dimensions become smaller, the primary contributor to interconnect delay is wire parasitic. The parasitics of resistance (R) and capacitance

(C) are the physics properties of the wire, and the amount of resistance and capacitance is dependent on the physical dimensions of the interconnect and materials used. In next few sections, parasitic resistance and capacitance are introduced.

2.1.2 Parasitic Resistance

The resistance is determined from the cross-sectional area of the interconnect. With the resistance equation given by

$$R = \rho \frac{l}{w \times t} \tag{2.1}$$

a larger cross-sectional area (width (w) by thickness (t)) means a lower resistance. The conductor resistivity ρ is in $\Omega \cdot m$. As the technology shrinks down to DSM, wires tend to be much thinner than before (see Figure 2.1). This fact leads to an increase in parasitic resistance for a minimum width wire. For instance, interconnect cross-sectional area is reduced by 50% from the 90 nm to the 65 nm technology node, but it results in 100% increase in resistance for a line of the same length.

Improvements have been made by replacing the materials in the interconnect system (from Al and SiO_2 to Cu and so-called low-κ dielectric materials) to reduce interconnect resistances. Table 2.1 lists the resistivity for Al and Cu. This approach, however, causes serious delay as integrating new low-κ materials is not a straightforward process [24, 25]. Moreover, this is only a one-time improvement; the resistance continues to increase dramatically in further smaller geometries. In the meantime, the simplest solution is to increase the wire width to reduce resistance. Unfortunately, this is not always possible because an increase in wiring density is needed to keep up with the increase in logic density as transistors

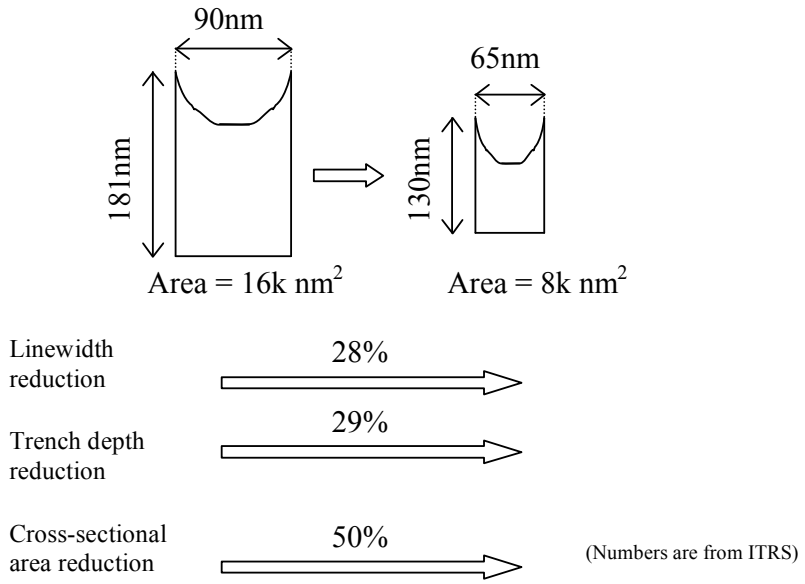

FIGURE 2.1: Resistance cross-sectional area reduction from 90 nm to 65 nm technology nodes [5].

are scaled.

2.1.3 Parasitic Capacitance

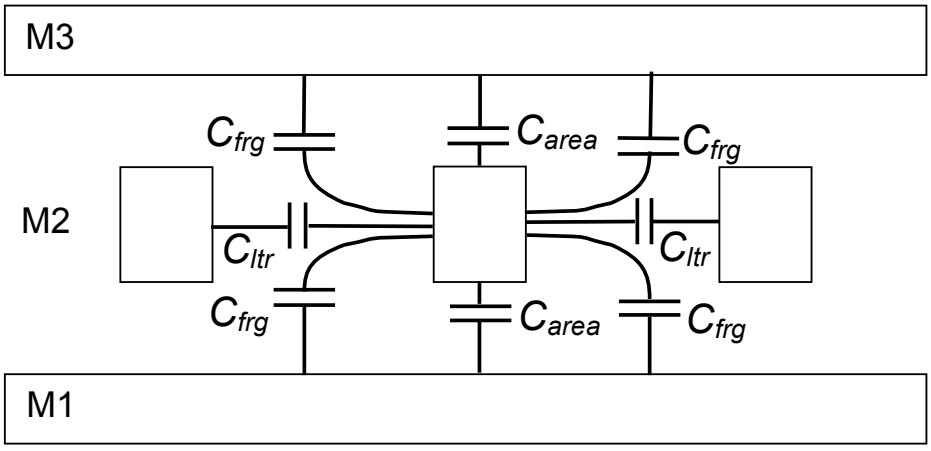

FIGURE 2.2: Capacitance configuration in a typical DSM technology with 1.5 aspect ratio.

Coupling between neighboring conductors gives rise to parasitic capacitance, and to first order the amount of capacitance is proportional to the ratio of the conductive areas facing each other but inversely proportional to the separation of two conductors. Given the charges on both conductors' plates are $+Q$ and $-Q$ with a V voltage between them, the equation for the capacitance is shown with integrals:

$$C = \frac{Q}{V} \; ; \quad Q = \int_0^V dV \cdot C(V) \tag{2.2}$$

A typical structure of a DSM interconnect with the most dominant parasitic capacitances labeled is shown in Figure 2.2. Metal 2 conductors have lateral capacitance with the adjacent conductors and also have area and fringe capacitances with metal 1 and 3 conductors on the above and below. Area capacitance is caused by two different conductors overlapped, and it was the dominant element in earlier technologies. However, lateral and fringe capacitances have grown dramatically to be the major contribution to the coupling capacitance due to the narrower wires and spacing in DSM technology.

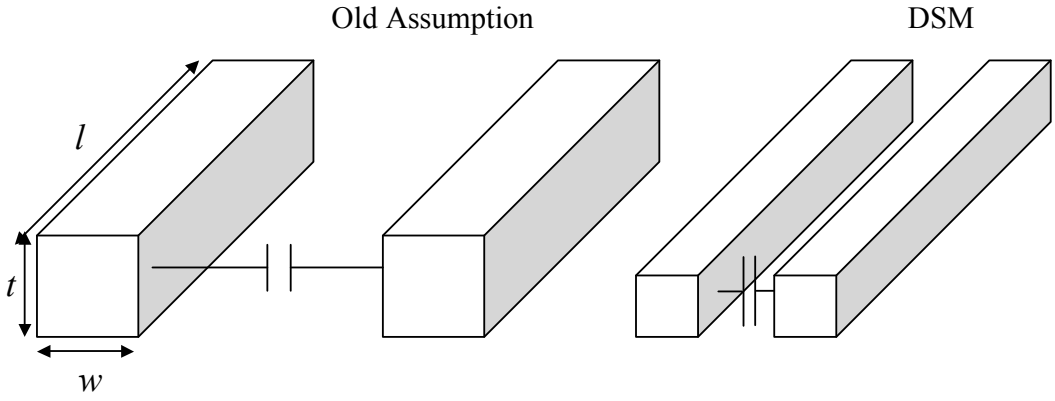

FIGURE 2.3: Capacitance in old assumption and DSM.

To improve performance, advanced methods, such as low-κ dielectric insulators and air gaps, are strongly required to reduce κ to around 1.9 [26], but it is no longer useful in further scaling of technology down to 90 nm and below (see Figure 2.4 and Table 2.1). A comparison between interconnects and device delays by using two different materials, Al and Cu with SiO_2 and low-κ from the National Technology Roadmap for Semiconductors SEMATECH [27] is made. By observing the interconnect delay for Al with SiO_2 and Cu with low-κ, it is clear that the total delay for a gate with Cu and low-κ is definitely lower than Al and SiO_2, and it solves the delay problem for the 250 nm and 180 nm technology generations. However, the delay started to increase exponentially with further scaling down of the technology. Challenges develop as the delay time starts to increase parabolically beyond 130 nm.

TABLE 2.1: Information of Figure 2.4 (adapted from [27]).

Aluminum (Al)	3.0 $\mu\Omega/cm$
Copper (Cu)	1.7 $\mu\Omega/cm$
SiO_2	κ= 4.0
Low-κ	κ= 2.0
Al & Cu	0.8 μ Thick
Al & Cu Line	43 μ Long

The easiest solution for circuit designers to obtain a lower capacitance is to increase the spacing between wires, but at the cost of more area in the design. The benefits of increasing the wire spacing are limited by diminishing returns and can only be used at the cost of losing interconnect density.

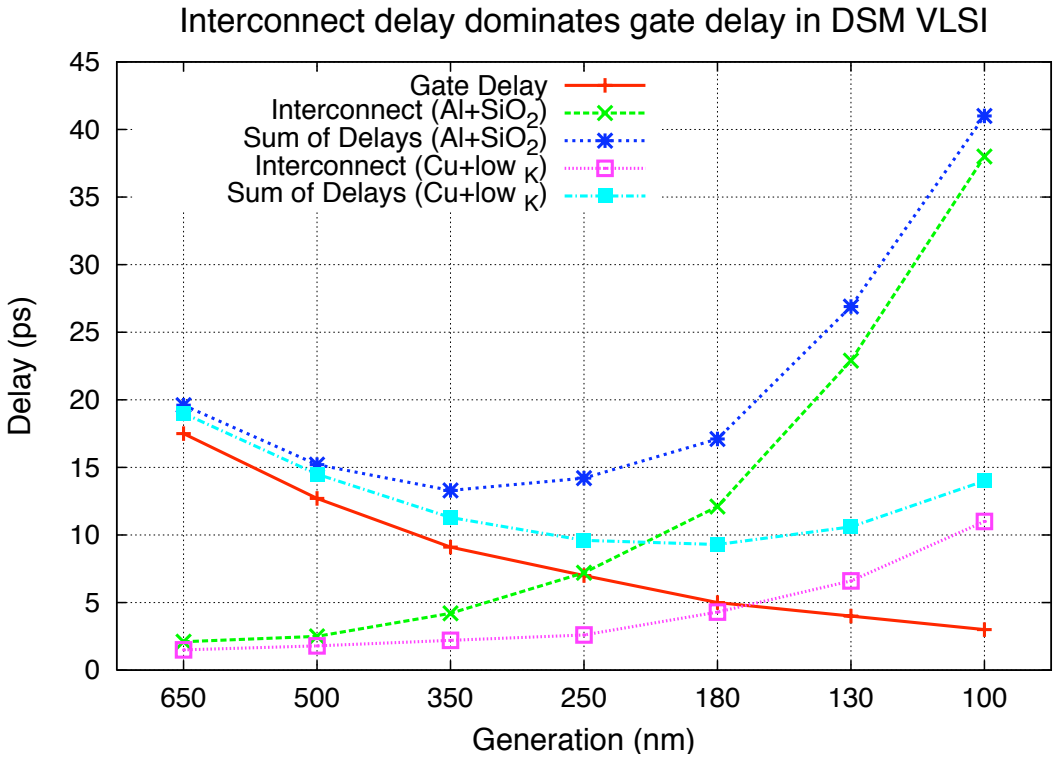

FIGURE 2.4: Interconnect vs gate delay (adapted from [27]).

2.1.4 Inductance

Inductance is another critical parasitic effect. A severe inductive effect includes overshoots, undershoots and oscillation in signal waveform, and aggravates crosstalk and power grid noises [28, 29]. These signal integrity faults can potentially manifest themselves as glitches or worse, as false transitions at the end of wires. Inductance is the ratio of the magnetic flux Φ generated by the electric current i and passing through a closed loop with N turns, and the equation is shown:

$$L = \frac{N\Phi}{i} \tag{2.3}$$

In the past, inductance was not a critical concern because RC effects dominated at low frequencies. However, the increasing inductance effect arises not only from the result of higher switching frequency, but also from other factors: resistance and capacitance reductions by Cu and low-κ dielectric respectively, denser and smaller geometries, and growing complexity of interconnect design [30–32]. The promotional literature for one of the leading industrial tools, Synopsys Star RCXT, claims that modeling global interconnects of RC circuits is no longer adequate, and the inductance effect must be included to avoid underestimating the signal integrity problem [33].

In this thesis, inductance was neglected for a few reasons. The first reason is that it is very difficult to predict and capture on-chip inductance effect because it depends strongly on the overall construction of the integrated circuit [34]. The uncertainty of the current return path, prior to parasitic extraction and circuit model simulation, has challenged inductance modeling and analysis in the DSM era [35, 36]. Unlike capacitance, which is based strongly on neighboring features, inductive effects have a much larger spatial range. The second reason is that inductance does not cause severe signal or propagation delay of interconnects problems. It is shown that the worst-case inductance would cause an 8% reduction in the delay [28]. The third, and most significant, reason is that Magic does not include an inductance extraction system. Such a modification would have required more human resources, which is beyond the scope of the work intended for this thesis. The scope of this Master's thesis was mainly about implementing the sensitivity analysis in capacitance extraction.

2.2 Capacitance Extraction

In the DSM regime, interconnect behavior dominates the overall chip performance. Therefore, accurate estimates of interconnect due to resistance-capacitance are significant for predicting performance and reliability in VLSI circuits [4, 7]. There are many different approaches to capacitance extraction. In this section, a number of previous approaches are presented, and one of them is chosen for investigation and implementation in the Magic layout tool.

2.2.1 Analytical Formulae

Capacitance values are estimated by analytical formulae in many cases because there is a compelling need for simple and fast approximation formulae, and these formulae are especially useful for hand calculations.

The basic capacitance calculation is the parallel plate formula given by

$$C = \epsilon \frac{A}{d} \tag{2.4}$$

where A is the area of two parallel planar conductors separated by distance d and ϵ is the dielectric constant. As the IC technology advances, the interconnect widths become narrower than their vertical thickness, which means fringe capacitance is comparable to overlap area capacitance, and the need to model the capacitance more accurately becomes important as shown in Figure 2.2.

Two accurate formulae were proposed using approximate conformal mapping techniques in [37], and both formulae are more accurate, within 1% error, for a metal line of width greater than the dielectric thickness.

Later a simple analytical formula was presented to include a direct physical interpretation in [38]. The replacement of rectangular wire cross-section with an "oval" one is made, and the final cross-section is composed of a rectangle and two half-cylinders as shown in Figure 2.5. The formula is shown:

$$C = \epsilon \left[\frac{w - t/2}{h} + \frac{2\pi}{ln\left(1 + \frac{2h}{t} + \sqrt{\frac{2h}{t}\left(\frac{2h}{t} + 2\right)}\right)} \right] \tag{2.5}$$

It is shown that this formula is simpler and more accurate than the proposed approximation

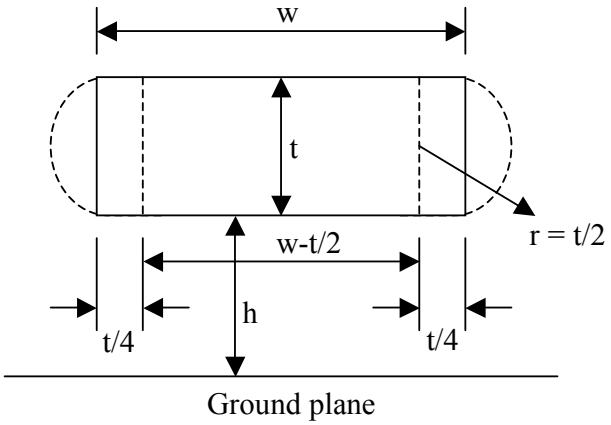

FIGURE 2.5: Conductor with a rectangular cross-section (adapted from [38]).

in [37].

A purely empirical formula based on evaluating numerical solutions was proposed in [39], and it was later improved by extending the empirical expression and simultaneously

reducing the range of validity in [40]. The formula is shown:

$$C = \epsilon \left[\frac{w}{h} + 0.77 + 1.06 \left(\frac{w}{h} \right)^{0.25} + 1.06 \left(\frac{t}{h} \right)^{0.5} \right] \tag{2.6}$$

where the first term describes the parallel plate capacitor and the other three terms represent side effects.

Reference [41] includes a complete comparison of these analytical formulae, and it is shown that Meijs and Fokkema [40] is the best choice in every respect, both most accurate and fastest in runtime.

2.2.2 Field Solver

Electric field solvers produce accurate results for multi-conductor systems in capacitance extraction. In [4], several reasons to use field solver for parasitic extraction were discussed: field solvers deliver accurate distributed self- and coupling capcaitances, and it handles general, non-Manhattan geometries and scales to DSM domain. Three major field solver approaches are discussed below.

2.2.2.1 Random Walk Method

The random walk method (RWM), also known as a stochastic algorithm, used for high-speed capacitance extraction in multi-level VLSI interconnects, was introduced by LeCoz and Iverson [42, 43]. The basic idea is to present boundary-integral solutions to compute electric potential and electric field at a cube center. The capacitance C_{AB} between two

electrical conductors A and B can be shown by Monte Carlo integration:

$$C_{AB} = \oint \oint \frac{d^2 r_1 \varepsilon(r_1) E(r_1)}{V_B} \tag{2.7}$$

Each random walk consists of a series of steps onto cube boundaries. Figure 2.6 shows the random walk starting on an integration surface around conductor A and terminating on a conductor B, and the estimated capacitance between both electrodes is associated with Eq. 2.7. Reference [44] addresses that the random-walk method performs with excellent computational efficiency because it requires no numerical mesh. Random walk methods have been successfully implemented in some commercial tools, such as Magma QuickCap [45] and Synopsys Raphael NXT [46].

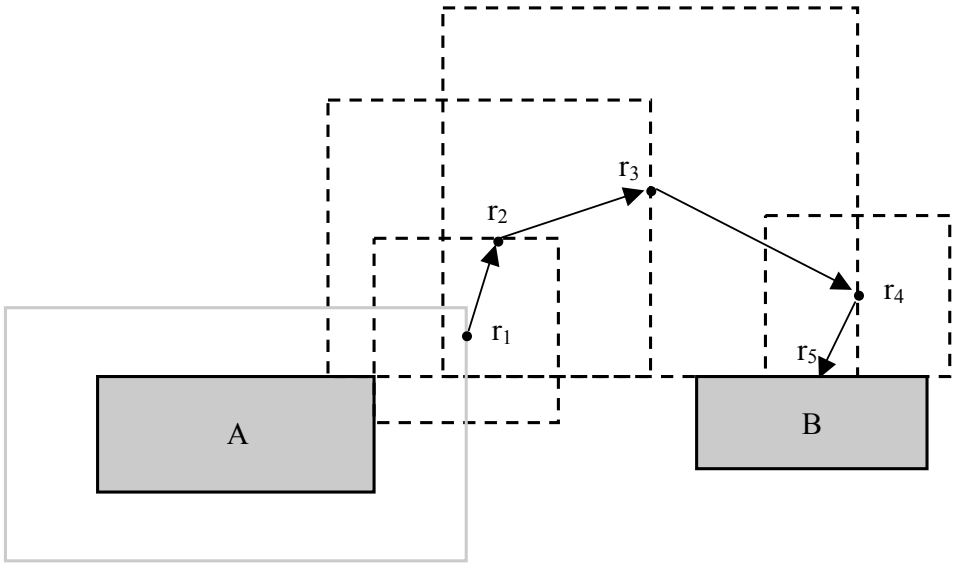

FIGURE 2.6: Sample random walk method (adapted from [44]).

2.2.2.2 Volume-Based Method

In volume-based analysis methods, Laplace's equation is solved outside the conductors (see Eq. 2.8).

$$\nabla \cdot \epsilon \nabla \Phi = 0 \tag{2.8}$$

One common way to solve 3D Laplace's equation numerically is to use Finite-Difference (FD) [47, 48] or Finite-Element (FE) [49–51] Methods, in which FD's main idea is to approximate the derivatives in the equations. The basic idea in FE method is to find an approximate solution to simplifying a complicated problem. For instance, suppose a structure (beam) is given, with FE method. It is then divided into several elements in a suitable solution in each element (see Figure 2.7). Volume-based methods are computationally expensive with very

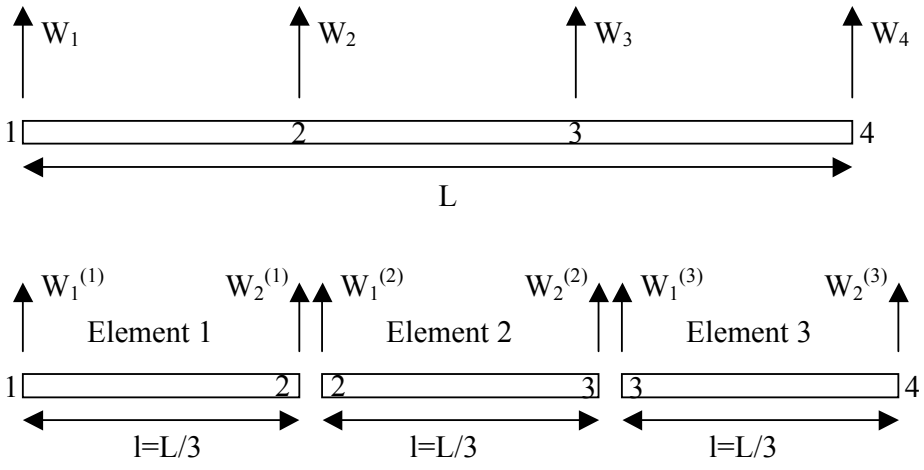

FIGURE 2.7: Sample finite element method (idealization of a beam).

large sparse matrices and require numerical techniques to improve the run time efficiency and memory storage. FD and FE methods have been used in some commercial tools, such as Ansoft Maxwell [52] and Synopsys Raphael [53].

2.2.2.3 Surface-Based Method

The surface-based method, also known as Integral Equation Formulation Method, is a well-known method in VLSI capacitance extraction, and many proposed algorithms are based on it. The following describes a set of different techniques for parasitic extraction over the past two decades, and they are all based and extended on the integral equation formulation method.

BEM. Boundary-Element Methods (BEM) are commonly used to perform electrostatic analysis [54], and a technique is shown in [55]. Many of the following techniques are based on BEM.

The purpose of BEM is to find the short-circuit capacitance matrix in a domain V containing M conductors (see Eq. 2.9).

$$Q = C_s \ \Phi \tag{2.9}$$

where Q is conductor charge $Q^T = [Q_1, Q_2, \cdots Q_M]$ and conductor potentials

$\Phi^T = [\Phi_1, \Phi_2, \cdots \Phi_M]$

$$\phi(p) = \int_V G(p,q) \ \rho(q) \ dq, \tag{2.10}$$

where $\rho(q)$ is the charge distribution and $G(p,q)$ is the Greens function for V. BEM subdivides the conductors' surfaces and approximates inverse matrix to eventually compute capacitance matrix C_s. One of the commercial tools, Space 3D, uses this method in capacitance extraction [56].

FMM. In 1991, a 3-D capacitance extraction program based on the multipole expansions was published: FastCap [57]. The algorithms, based on fast multipole method (FMM), accelerates the boundary-element solution with conjugate residual method for reducing order of computations and complexity. It is noticed that for problems with as few as 12 conductors the multiple accelerated boundary element method can be nearly 500 times faster than Gaussian elimination based algorithms, and five to ten times faster than the iterative method alone, depending on the required accuracy [57]. FastCap has become very practical, popular, widely used in industry and academia, and has initiated an extensive amount of research in the area of the development of fast integral equation solution techniques [58, 59].

FFT. Several years later, a new algorithm, "precorrected-fast Fourier transform (FFT)", was presented for accelerating the potential calculation and solving electromagnetic boundary integral equations arising in the extraction of coupling capacitances in 3-D geometries [54]. The comparison shows that for a wide variety of geometries in IC packages, the new algorithm is superior to the fast multipole algorithm used in FastCap in terms of execution time and memory use by more than an order of magnitude [54].

SVD. Another integral equation 3-D solver is IES^3 ("ice cube"), which uses singular value decomposition (SVD) algorithm, an extremely effective tool for compression of rank-deficient matrices. The result shows that IES^3 is dramatically faster than the multipole-based approaches [60].

Hierarchical. Shi and *et al.* presented a fast hierarchical algorithm for computing 3-D capacitance extraction, and it is significantly faster with less memory used than the

previous famous algorithm used in FastCap. It is noticed that the this hierarchy-based algorithm is 60 times faster than FastCap and uses 1/80 of the memory used by FastCap with accuracy of 2.5%. Also, this algorithm is 5 to 150 times faster than commercial software QuickCap with the same accuracy [61, 62].

FMM+Hierarchical. Beattie and Pileggi combined FMM with the hierarchical refinement methods to propose solutions that use both windowing and shift-truncate bounds to get an error value for the extracted capacitances to improve the speed further, called adaptive window sizing methodology. This methodology is of use in identifying crosstalk problem zones for interconnect optimization and noise reduction [63].

Nebula. Nebula was presented for accurate large-scale capacitance calculations with a charge distribution that decouples charge variation from conductor geometry. This separation significantly reduces the discretization size, the time and memory requirements compared to the previous approaches, and Nebula, based on FMM, is efficient enough to compute full capacitance matrix of typical interconnect problems [64].

QMM. The quasi-multiple medium (QMM) method greatly reduces the central processing unit (CPU) time and memory usage of large-scale direct BEM computations by transferring the coefficient matrix into a highly sparse block matrix and iterative equation solver [58]. Later, an enhanced QMM-BEM solver for 3-D multiple-dielectric capacitance extraction was proposed to achieve much higher speed and adaptability. With two enhancements, an automatic determination of QMM cutting pair number and preconditioning technique, the results showed over 10X speed-up and memory saving over the multipole approach [65].

FHM. Fast hierarchical method (FHM) is an efficient acceleration algorithm for 3-D capacitance extraction. A preconditioning technique is proposed by virtue of inherent properties of hierarchical data structures, and experiments show a remarkable improvement on coverage of the iterative procedure [66].

2.2.2.4 Field Solver Summary

The random-walk method is practically beneficial for suitability to rectilinear geometries, statistical-error cancellation, selective integration over Gaussian surfaces, and direct capacitance matrix evaluation [43]. The surface-based method is robust and has advantages over volume based method, including a better conditioning, smaller dimensionality, and the ability to treat arbitrary regions [60].

In general, the random walk method is the best for calculating self-capacitance for a complicated net; the surface based method is most suitable for calculating small coupling capacitances, and the volume-based method is good at dealing with multiple dielectrics.

2.2.3 Library Look-Up Based Method

Library look-up based method is a practical approach to estimate capacitance efficiently in layout. The analytical or the table-look-up models are automatically generated based on numerical simulations, and this method is often used to extract parasitics directly from the given mask layout in a chip and fabrication processes. A couple of techniques have been proposed in [67–69], and [67] has been chosen to be implemented in the project because Magic is a VLSI layout tool, parasitics values are obtained in the extraction. Figure 2.8

shows the block diagram of the library model procedure as implemented in this project. The details will be discussed in a later section of this chapter.

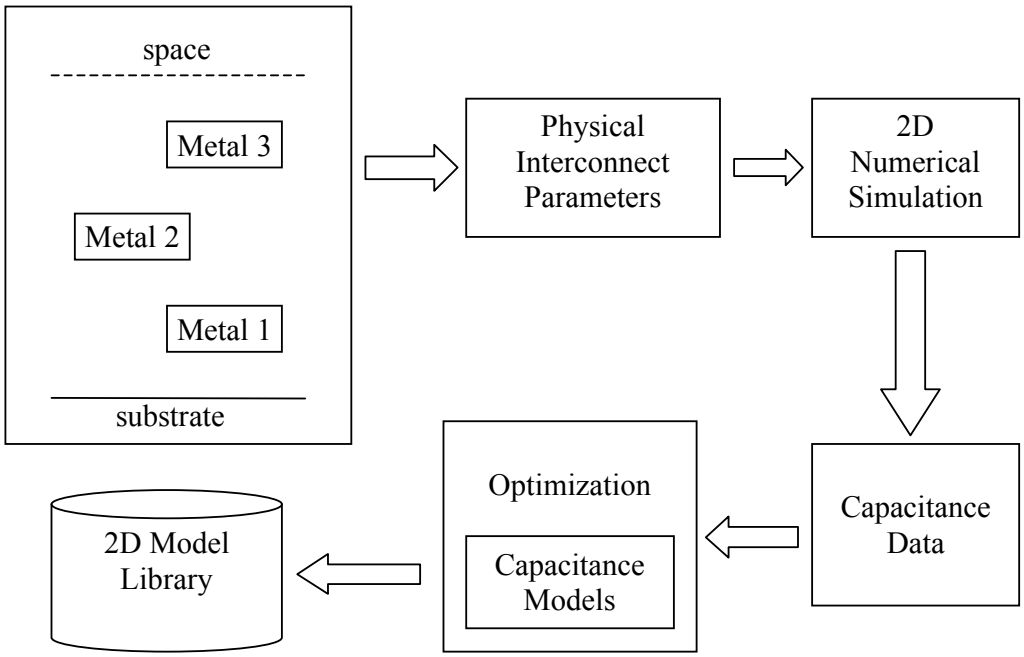

FIGURE 2.8: Overall flow of interconnect library model (adapted from [67]).

2.3 Commercial Parasitic Extraction

This section describes several industry parasitic extraction tools from some of the largest electronic design automation (EDA) companies, such as Synopsys, Cadence Design Systems, Mentor Graphics, Zuken, and Magma Design Automation, to demonstrate that this thesis research is important.

2.3.1 Synopsys - Star-RCXT

Synopsys Star-RCXT is a standard for parasitic extraction in EDA industry. It provides a parasitic extraction solution for application-specific integrated circuit (ASIC), System-on-Chip (SoC), custom digital, analog/mixed-signal (AMS), radio-frequency (RF) and memory design. Synopsys claims that "the offered capabilities include variation-aware parasitic extraction, chemical-mechanical polishing (CMP) based and litho-aware extraction, inductance extraction, and analog mixed signal design flow" [33].

In addition, Star-RCXT could be integrated into industry standard design flows, layout verification and simulation tools easily, and its incorporated flow with Raphael NXT, a 3D field solver capacitance extractor providing silicon-accurate and coupling capacitances for circuit design, makes it an accurate extraction [33].

Star-RCXT uses a statistical technique to model interconnect process variation effects accurately at advanced process nodes. This solution eliminates the need to run traditional multi-corner extractions and provides multiple netlists and a single netlist with sensitivity (see Figure 2.9) to enhance the productivity and performance. The variation of each process parameter, including conductor, dielectric thickness, is available and used based on the process variations [33]. The focus is on the linewidth parameter variation in this thesis.

2.3.2 Cadence QRC Extraction

As advanced process technologies become more critical, parasitic extraction becomes a key factor not only during the validation phase but also during design implementation. Cadence claims that "QRC Extraction is a 3D full-chip parasitic extractor in industry, and

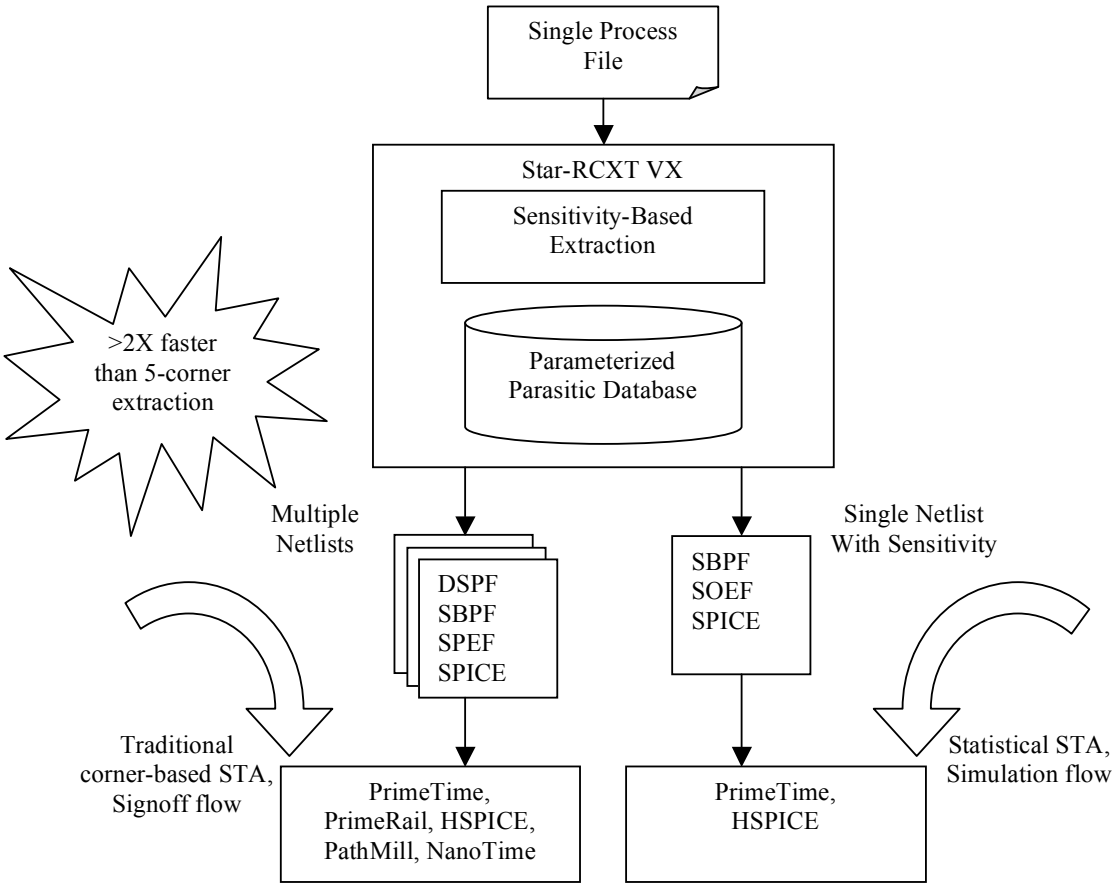

FIGURE 2.9: Star-RCXT's sensitivity-based extraction solution (adapted from [33]).

the fast and accurate RLCK extraction make it an integrated extraction solution for design implementation and validation at 90 nm and below" [70].

Figure 2.10 shows the key components of Cadence QRC Extraction, and multi-corner and statistical extraction are briefly discussed here. Multi-corner extraction is required to predict parasitic effects, and the number of corners and the process geometry grow inversely proportionally to each other. It extracts multi-corners at once to efficiently reduce extraction runtimes at advanced process nodes. QRC also takes random parameter variations into account to provide accurate results in statistical extraction [70].

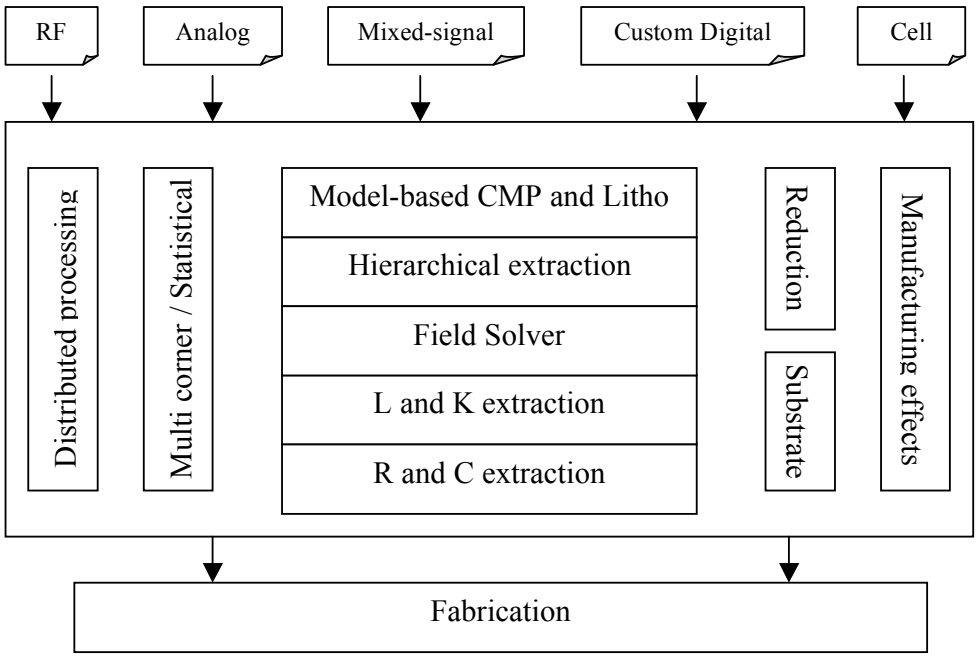

FIGURE 2.10: Cadence QRC extraction's advanced capabilities (adapted from [70]).

2.3.3 Mentor Graphics - Calibre LFD

The key benefit of Mentor Graphics Litho-Friendly Design (LFD) is "the capability to capture process variations in the design flow." It captures information on process window effects in order for designers to improve the layout and make a design more robust and less sensitive to process window variations [71].

2.3.4 Magma - QuickCap

QuickCap NX is a 3D parasitic extractor for critical circuit analysis. As with Star-RCXT, Magma claims that "QuickCap NX is for parasitic extraction and in production use in the top 10 semiconductor companies" [72].

Magma's QuickCap NX is an accurate 3D extractor that precisely models advanced process effects. It is used in process studies with performance on accurate noise and timing analysis, proved to deliver capacitance value that are within 1 percent silicon measurements. Magma claims that "the average difference between QuickCap NX capacitance values and actual silicon measurements have been reduced from 9.79 percent to 0.11 percent by taking process effects into account" [72]. In addition, QuickCap NX's multiple techniques reduce runtime and makes it memory efficient. Figure 2.11 shows how QuickCap NX accurately models advanced process effects. This thesis concentrated on the width variation.

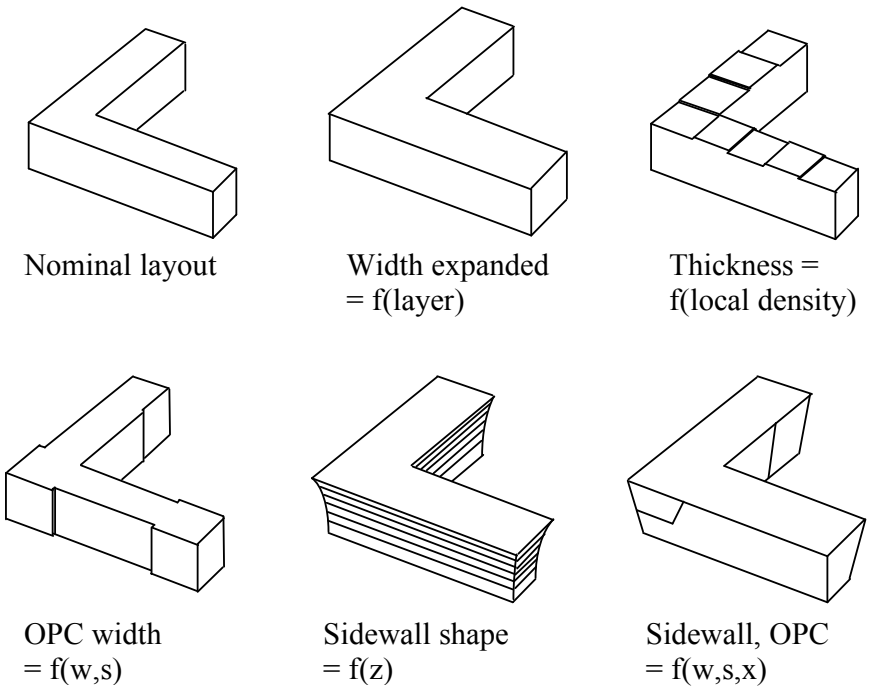

FIGURE 2.11: QuickCap NX's process variation solution (adapted from [72]).

2.3.5 Space 3D

Space 3D is a layout-to-circuit extractor from OptEM Corporation, and it uses BEM to accurately and efficiently compute 3D interconnect capacitances of ICs based upon the layout description, which implies a lower number of discretization elements is used for 3D situations. Moreover, the implementation of a new matrix inversion technique that computes only coupling effect between "nearby" elements makes itself capable of quick extraction and memory efficiency [56].

2.3.6 Industry Parasitic Extraction Summary

It is noticed that parasitic extraction for process variation is the key challenge for these industry extraction tools as the real fabrication is not exactly the same as the layout in the design. Table 2.2 shows a summary of commercials tools with different types of extraction methods. FEM and RWM extractors are broadly used in industries for parasitic extraction. As a result, the research of process variation is a necessity in modern technologies.

Table 2.2: Commercial extraction methods.

Commercial Tools	**Extraction Methods**			
	BEM	FEM	RWM	other
Synopsys-Star-RCXT				multi-corner/statistical
Synopsys-Raphael	√	√		
Synopsys-Raphael NXT			√	
Cadence-QRC Extraction				multi-corner/statistical
Mentor-Calibre				design for manufacturing
Magma-QuickCap NX			√	
Ansoft-Maxwell		√		
OptEM-Space 3D	√			

2.4 Magic Technology Files for Look Up-Based Method

In this section, a capacitance library model procedure for Magic is presented similar to that shown in Figure 2.8. This extractor consists of a file reader, a stack file, conductor looping, 2D field solver from [73], and Techgen to read in an input data, which includes anisotropic dielectric layers and multiple types of material properties, and generate a technology file with specified sets of capacitance coefficients. The procedure is used to compute the coefficients for CUP models, which will be introduced in Chapter 4, and a comparison will be made between Magic using CUP models and Space 3D, a commercial capacitance extraction tool from OptEM Engineering Inc., in Chapter 5.

2.4.1 Stack File

A stack file is composed of name, conductor, dielectric and some other sections, but only the important sections are discussed in this thesis. The name identifies the name of the stack. The conductor identifies that data about the conducting layers as follows. In each layer, layer order, layer name, minimum spacing, maximum spacing, width, thickness, and max_dist are listed. The dielectric identifies that data about the dielectric layers, and each layer contains the information of layer order, dielectric name, corresponding conductor layer, thickness, and dielectric constant. The stack file used in this thesis and a diagram illustrating its metal configuration is provided and presented in Appendix A.

2.4.2 Techgen

Techgen is the program that implements the capacitance library model procedure that reads in a Magic template technology file and generates a new one with the new model parameters. Techgen then reads in a standard Magic technology file line by line, echoes each line, and adds or modifies the capacitance model information that needs to be changed. Below is a more in-depth description of each function in techgen.

Checkdimensions: This function checks for the first word "width" or "spacing" in the technology file, and it changes the distance to agree with the stack file. A deeper explanation on width and spacing dimensions can be found in the Technology File section of Chapter 3.

Area and Overlap: Magic's area model is specified in lines beginning with "areacap" or "overlap" in the technology file. This function checks for the first word "areacap" or "overlap". Instead of replacing the Magic capacitance model value, it *adds* a similar line that specifies CUP models.

Sidewall and Sideoverlap: Magic's lateral model is specified by in lines beginning with "sidewall" and its fringe model is specified in lines beginning with "sideoverlap". These functions are similar to Area and Overlap, and they just check the first word "sidewall" and "sideoverlap" and add another similar line that includes CUP models with parameters.

When techgen is completed, a new technology file is ready to be used in Magic, with the original Magic capacitance models and new CUP models, both of which are read by Magic. Techgen and the revisions to Magic to read the enhanced technology file were written by undergraduate students Josh Schlenker and Travis Skippon. Some minor changes and corrections have been made.

Chapter 3

Existing Capacitance Extraction Model of Magic

This chapter begins with a brief overview of Magic VLSI internal structures and then follows the technology file section and explanations of its extraction algorithms, which includes overlap, sidewall, and sidewall overlap coupling capacitances. After capacitance extraction, a brief explanation of extraction output is presented. Finally, a summary of the Magic VLSI layout tool is presented in the end of the chapter.

Magic is an interactive layout editing system for creating and modifying VLSI circuit layouts. Magic is not only a color painting tool, but it also provides the user with powerful operations. These include the feature of design-rule checking to avoid layout violation at any time to enhance performance [74]. In addition, Magic contains approximately 90,000 lines of code written in the C language. Magic's extractor occupies one-tenth of Magic's C code [19, 75].

3.1 Magic Database

As mentioned to in Chapter 1, Magic is an interactive layout editing system for large-scale MOS custom ICs. As in most layout editors, Magic layout consists of cells, and each cell contains two types of things: geometrical shapes and subcells. In this section, the principle database in Magic is illustrated.

3.1.1 Geometry

Geometry is the basic database of a software, and a number of structures apart from the basic cell, plane, and tile definitions, are used frequently throughout the Magic source. Most of these basic structures have to do with geometry and can be found in the geometry header file under the Magic directory.

3.1.1.1 Point

A point in 2-dimensional integer space is referred to coordinates `p_x` and `p_y`. Figure 3.1 shows the basic point structure.

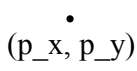

FIGURE 3.1: Point structure.

3.1.1.2 Rectangle

A rectangle, with area specified by lower-left and upper-right corners (points), is referred to point structures `r_ll` and `r_ur` for the lower left-hand corner and upper right-hand corner,

respectively. Note that while the four coordinates of a rectangle are, as defined, `r_ll.p_x`, `r_ll.p_y`, `r_ur.p_x`, and `r_ur.p_y`, the magic source code almost always refers to these points by macro shorthand, which are `r_xbot`, `r_ybot`, `r_xtop`, and `r_ytop` respectively. Figure 3.2 illustrates the basic rectangle structure.

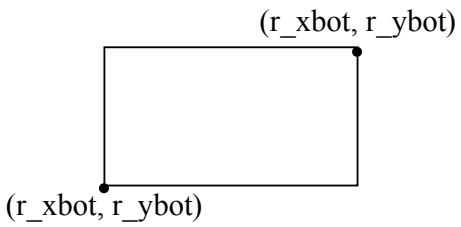

FIGURE 3.2: Rectangle structure.

3.1.2 Corner Stitched Tile

Magic uses a corner-stitched structure to present the substances of cells, that is a geometrical data structure for representing rectangular two-dimensional objects [76], or called Manhattan shapes (those whose boundaries contains only horizontal and vertical segments, see Figure 3.3) Moreover, corner-stitching structure is the reason for fast extraction speed in Magic.

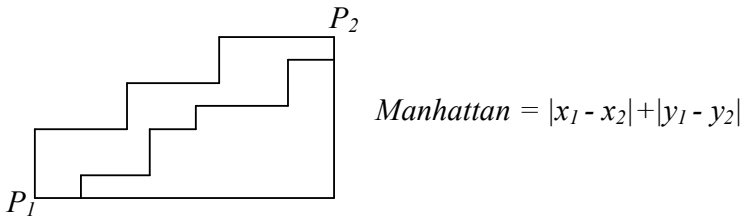

FIGURE 3.3: Corner-stitching (Manhattan shape).

A tile is a basic unit in a simple rectangular shape that corresponds to a length of interconnects. Each tile has the following structure

```
typedef struct tile

{

    ClientData   ti_body; /* Body of tile */

    struct tile *ti_lb;          /* Left bottom corner stitch */

    struct tile *ti_bl;          /* Bottom left corner stitch */

    struct tile *ti_tr;          /* Top right corner stitch */

    struct tile *ti_rt;          /* Right top corner stitch */

    Point        ti_ll;          /* Lower left coordinate */

    ClientData   ti_client;      /* This space for hire. Warning: the default

                                  * value for this field, to which all users

                                  * should return it when done, is MINFINITY

                                  * instead of NULL.

                                  */

} Tile;
```

This is the fundamental data structure in Magic. ti_body describes which layer the tile is, and *ti_lb, *ti_bl, *ti_tr, and *ti_rt represent different corner stitched tiles respectively. Ti_ll is the lower left coordinate, so the position of the any specific tile is recognized. The following figure illustrates the tile structures in details. The corner-stitched image connects each tile to four sides of its neighbors.

The heavily used macros on (Tile *)tiles in Magic are listed below.

- TOP(tile)
- BOTTOM(tile)
- LEFT(tile)

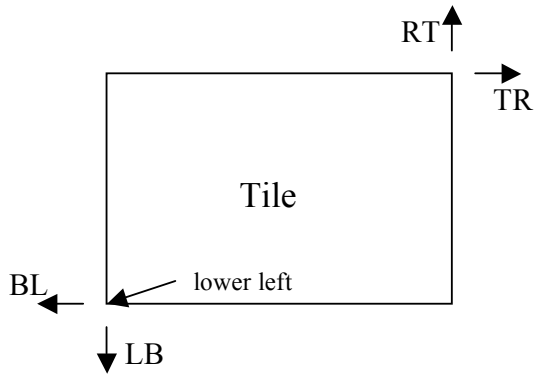

FIGURE 3.4: Basic tile structure.

- RIGHT(tile)
- RT(tile)
- LB(tile)
- BL(tile)
- TR(tile)

The first four macros return a position in Magic's internal coordinates, and the last four macros return a pointer to the appropriate neighboring tile (see Figure 3.4). This use has made the programmer easier to implement and reduce the confusion of tile definition.

3.1.3 Plane

In Magic, all geometries are split into partially independent planes. In general, these planes correspond roughly to the mask layers from which an IC is built. Typically, one defines a new plane for each set of layer types that are largely non-interacting with other types. This works especially well for metal routing layers, since the metal layers are completely non-interacting (excluding capacitance extraction, which is separate from the database representation), and only interact where cuts (vias) are defined. This works much less well for layers close to the

original wafer surface. Polysilicon, diffusion regions, and wells are all heavily interacting, and therefore tend to work best when they are defined all on a single plane.

Even though it would be possible to define everything in the layout on one plane, the corner-stitched tile structures are broken up into pieces for the types overlap in the same plane. For instance, if type1 and type2 are overlapped in a single plane, Magic requires separate types type1, type2, and an additional type type1-plus-type2, and the corner-stitched structures will be subdivided until each tile can be represented by one of these three types. This is highly effective in some cases when, for example, type1 is n-diffusion, type2 is polysilicon, and thus type1-plus-type2 works out "naturally" to be layer type n-transistor.

The plane structure holds a corner-stitched tile representation and looks like:

```
typedef struct
{
    Tile        *pl_left;        /* Left pseudo-tile */
    Tile        *pl_top;         /* Top pseudo-tile */
    Tile        *pl_right;       /* Right pseudo-tile */
    Tile        *pl_bottom;      /* Bottom pseudo-tile */
    Tile        *pl_hint;        /* Pointer to a "hint" at which to
                                  * begin searching.
                                  */
} Plane;
```

Each cell contains several corner-stitched planed to represent the cell's geometries and subcells, and each plane consists of rectangular tiles of different types. There are three important

properties in a corner-stitched plane: coverage, strips, and stitches.

3.1.3.1 Coverage

Every point in a corner-stitched plane is contained in exactly one tile. Empty space is represented, as well as the area covered with material.

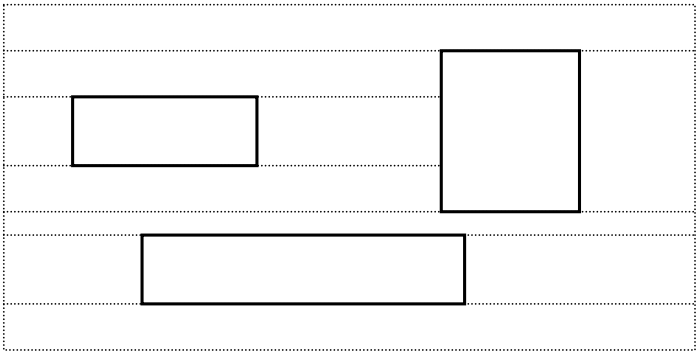

FIGURE 3.5: Coverage tile.

In the case shown in Figure 3.5, there are three solid tiles and space tiles (dotted lines) cover the rest of the plane. The space tiles on the sides extend to infinity. In general, a plane can contain many different types of tiles.

3.1.3.2 Strips

Horizontal strips represent material of the same type. The strip structure provides a canonical form for the database and prevents it from fracturing into a number of small tiles.

Areas of the same type of material are represented with horizontal strips that are as wide as possible. In each of the Figure 3.6, the tile structure on the left is illegal and is converted into the tile structure on the right. In Figure 3.6 (a), all tiles must be in rectangle shapes. In Figure 3.6 (b), it is illegal for two tiles of the same type to share a vertical edge.

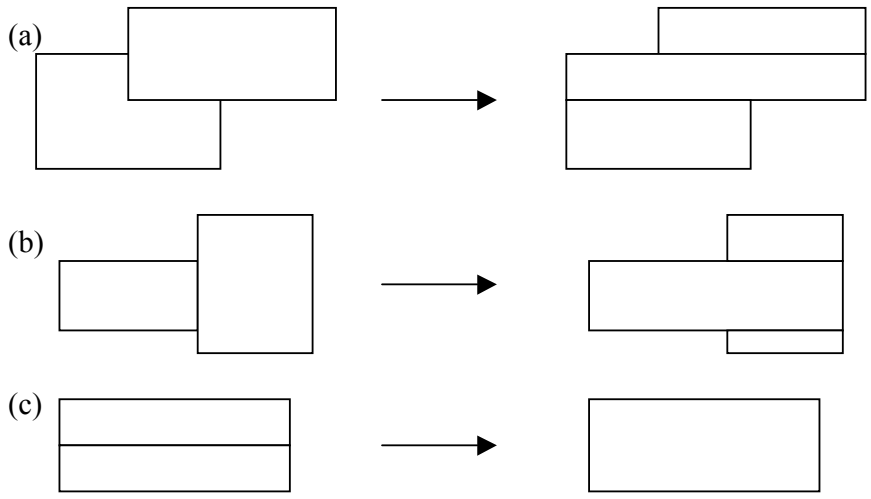

FIGURE 3.6: Strips tile (adapted from [19]).

In Figure 3.6 (c), the two tiles must be merged as one because they share exactly the same horizontal span.

3.1.3.3 Stitches

The records describing the tile structure are linked in the database through four links per tile, called stitches. The links point to neighboring tiles at two of the tile's four corners.

A illustration in Figure 3.7 shows the detailed stitch tiles in a plane with labels pointing to each other tiles, where space tiles are defined when there is no conductor presented. The boundary of the plane is positive and negative infinity defined in the corner (see Figure 3.7).

Magic searches tiles in the "tile search" function. It first finds the left top tile in a rectangle and then walks through rightward. A pseudo code is shown in Figure 3.8, and this function is heavily used throughout Magic codes. Appendix B shows an example of how Magic searches tiles, step by step.

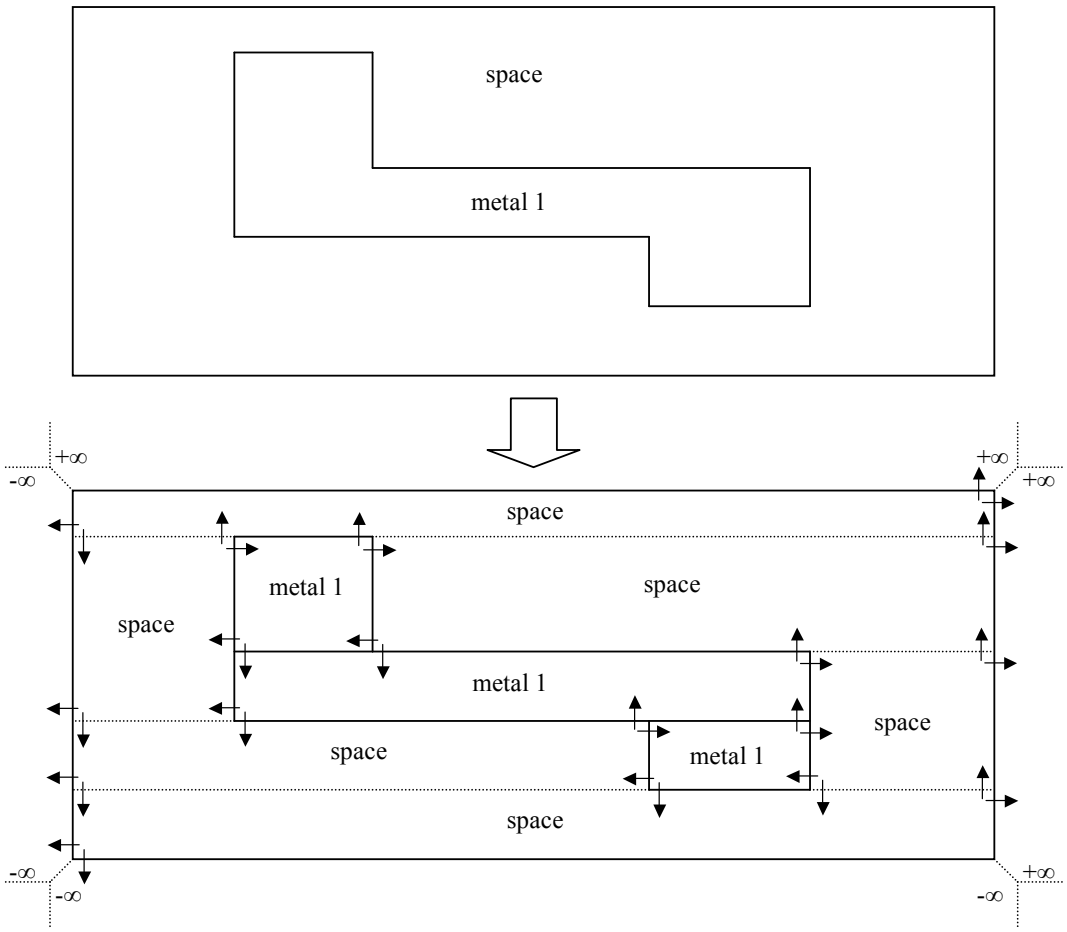

FIGURE 3.7: Stitch tile.

3.2 Magic Technology File

As mentioned earlier, Magic reads in its own technology files. Even though Magic contains an extensive amount of knowledge about integrated circuits, it is a technology-independent layout tool [74]. In other words, the information is not embedded directly in code, and it is contained in a technology file that Magic reads when launched. This file defines the abstract layers and the corner-stitched planes; it also describes design-rule checking in detail, such as minimum width and spacing between objects on the same layer. In addition, it provides

```
tile is defined to be the left top tile of rectangle that is to be searched
tp = tile; tpnew = new_tile;
while (TOP(tile) > BOTTOM(rectangle))
{
enumerate:
      if (tile is in the region) go to function;

      tpnew = TR(tp);
      if (LEFT(tpnew) < RIGHT(rect)) /* */
      {
            while (BOTTOM(tpnew) >= BOTTOM(rect)) tpnew = LB(tpnew);
            if (BOTTOM(tpnew) >= BOTTOM(tp) || BOTTOM(tp) <= BOTTOM(rect))
            {
                  tp = tpnew;
                  goto enumerate;
            }
      }
      while (LEFT(tp) > LEFT(rect))  /* Each iteration returns one tile further to the left */
      {
            if (BOTTOM(tp) <= BOTTOM(rect)) return (0);
            tpnew = LB(tp);
            tp = BL(tp);
            if (BOTTOM(tpnew) >= BOTTOM(tp) || BOTTOM(tp) <= BOTTOM(rect))
            {
                  tp = tpnew;
                  goto enumerate;
            }
      }
      for (tp = LB(tile); RIGHT(tp) <= LEFT(rect); tp = TR(tp)) /* Walk down to next tile */
}
```

FIGURE 3.8: Tile search algorithm pseudo code.

the model coefficients for coupling capacitances, substrate capacitance and resistance for different layers in the extraction section.

3.2.1 Planes

The **planes** section of the technology file specifies how many planes will be used to store tiles in a given technology, and each plane has its own name. Each line in Table 3.1 defines a plane by giving a comma-separated list of the names [74].

TABLE 3.1: Planes section.

planes
well, w
implant, i
active, diffusion, polysilicon, a
metal1, m1
metal2, m2
metal3, m3
oxide, ox
end

3.2.2 Contact

The **contact** section describes which types are contacts and the planes and component types to which they are connected in Magic. Each contact line begins with a tile type, *base*, to be a contact. This type is then referred to as a contact's *base type*. The remainder of each line is a list of non-contact tile types that are connected by the contact, referred to as the *residues* of the contact. In Table 3.2, for instance, the type **m2contact** is the base type of a contact connecting the residue layers on the planes **metal1** and **metal2** [74].

TABLE 3.2: Contact section.

contact			
pcontact	poly	metal1	
ndcontact	ndiff	metal1	
pdcontact	pdiff	metal1	
ppcontact	ppdiff	metal1	
nncontact	nndiff	metal1	
m2contact	metal2	metal1	
pad	metal1	metal2	glass
end			

3.2.3 DRC

The design rules checked in Magic are completely specified in the technology file. Two simple rules are introduced in this section, **width** and **spacing**. Table 3.3 shows a sample width and spacing rules in the drc section.

TABLE 3.3: DRC section.

drc					
width	poly,pc/a	2	*error message*		
spacing	poly,pc/a	poly	2	**touching_ok**	*error message*
width	m1	3	*error message*		
spacing	m1	m1	3	**touching_ok**	*error message*
width	m2	4	*error message*		
spacing	m2	m2	3	**touching_ok**	*error message*
width	m3	4	*error message*		
spacing	m3	m3	4	**touching_ok**	*error message*
end					

3.2.3.1 Width rules

The minimum width of a collection of types is expressed in the following format:

width *type-list width error*

where type-list is a set of tile types, and width is an integer, and error is a string, which can be printed to the console to let the user know if the rule is violated.

3.2.3.2 Spacing rules

Another simple kind of design rule is the spacing rule, and it has two flavors: **touching_ok** and **touching_illegal**, both with the following format:

spacing *types1 types2 distance flavor error*

The **touching_ok** allows *types1* and *types2* to be adjacent to each other, and it requires that any type in the set *types1* must be separated by a "Manhattan" distance of at least *"distance"* units from any type in the set *types2* that is not immediately adjacent to the *types1*. An example of Manhattan distance is illustrated in Figure 3.3.

The other flavor of spacing rule, **touching_illegal**, prohibits *types1* and *types2* from being immediately adjacent. This rule separates *types1* and *types2* from each other. **touching_illegal** is less frequently used because it is less significant than **touching_ok**, thus only a brief explanation is provided here.

3.2.4 Extract

The **extract** section of a technology file provides the parameters used by Magic's circuit extractor. A keyword in the beginning of each line specifies the remainder of the line. Table 3.4 is a sample **extract** section in a Magic technology file.

The keywords **areacap** and **perimcap** define the capacitance to substrate, and **resist** identifies the sheet resistivity of each Magic layers. Their formats are followed by a list of types and a value respect to substrate, as follows:

areacap *types C*

perimc *intypes outtypes C*

resist *types R*

where the C is in attofarads per square lambda and R is in milliohms per square. Once an extraction is performed, substrate capacitance and sheet resistance are output in the output `'ext'` file in the node section (see Table 3.5).

Table 3.4: Extract section.

extract			
style	lambda=1.0		
lambda	100		
step	100		
sidehalo	6		
areacap	poly	33	
areacap	metal1	17	
areacap	metal2	11	
perimc	poly	80	
perimc	metal1	41	
overlap	metal1	poly	30
overlap	metal2	metal1	45
overlap	metal2	poly	19
sidewall	poly	15	
sidewall	metal1	27	
sidewall	metal2	33	
sideoverlap	metal1	poly	41
sideoverlap	metal2	metal1	42
sideoverlap	metal2	poly	22
resist	poly	25	
resist	metal1	60	
resist	metal2	40	
end			

Magic also extracts internodal coupling capacitances, which involve keywords **overlap**, **sidewall**, **sideoverlap**, and **sidehalo** to provide the parameters to perform the task.

Overlap capacitance exists between two pairs of tile types when they overlap, and it is described as the following:

overlap *toptypes bottomtypes cap*

Magic's extractor searches for tiles in *toptypes* that overlap tiles in *bottomtypes*. When an

overlap area is searched, the capacitance to substrate in types *toptypes* is deducted and then replaced by a capacitance to types in *bottomtypes*.

Sidewall, or parallel wire capacitance, is between pairs of edges of the same type in a close separation defined by *'sidehalo'* parameter as follows:

sidehalo *distance*

sidewall *intypes outtypes neartypes fartypes cap*

Sidehalo *distance* defines the threshold distance beyond which the effects of sidewall capacitance are neglected, and this method reduces the amount of searching time in Magic. The **sidewall** keyword is illustrated in Figure 3.9, where capacitance applies between tiles *tinside* and *tfar*.

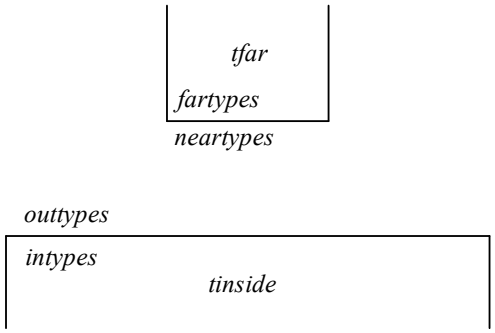

FIGURE 3.9: Sidewall technology line description (adapted from [19]).

Sidewall overlap capacitance arises between material on the inside of an edge and overlapping material of a different type, described as follows:

sideoverlap *intypes outytpes ovtypes cap*

where *intypes* are on the inside edge and *outtypes* are on the outside surface, and that overlaps a tile in *ovtypes*. The *cap* described in this section are all in attofarads per lambda square.

3.2.4.1 Resistance

Magic extracts a lumped resistance for each node, rather than a point-to-point resistance between each pair of devices connected to that node. The result is that all such point-to-point resistances are approximated by the worst-case resistance between any two points in that node [74].

Node resistances are approximated rather than computed exactly by default. For a node comprised entirely of a single type of material, the node's total perimeter and area are computed by Magic [74]. Magic's resistance extraction can be done with the command "`extresist`".

3.3 Magic Capacitance Extraction

The database in Magic makes circuit extraction almost trivial. Magic uses the traditional pattern-matching method for capacitance extraction. The extractor does not need to register layers or infer the structure and type of transistors and contacts; it only needs to traverse the tile structure and record the connection information. This leads Magic to become a fast circuit extractor.

Magic extracts four kinds of capacitances, and three of them are major coupling capacitances: overlap, sidewall, and sidewall overlap. A coupling capacitance is formed when two

different layers overlap each other on the surface or edge, or when two of the same materials overlap the edge within a certain distance defined in the technology file. In addition, the other capacitance is substrate capacitance. Each node has a corresponding substrate capacitance respect to substrate. These values are computed in `'ext'` file after a Magic layout circuit is extracted.

3.3.1 Overlap Capacitance Extraction

Overlap capacitance is the most straightforward of the three coupling capacitance components. It is formed when one conductor overlaps another conductor on a different plane, and they are not linked together in the same net. The following figure shows a simple example of overlap capacitance with dashed lines indicating the minimum design of one λ unit.

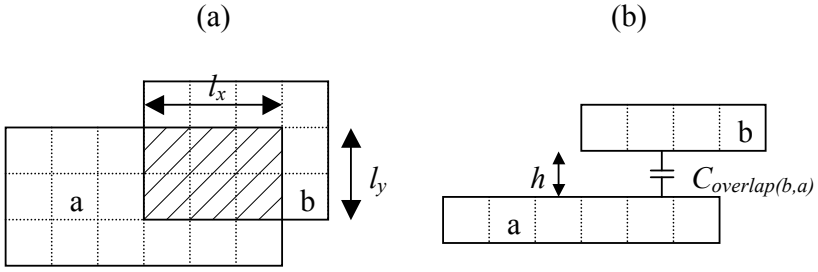

FIGURE 3.10: Overlap capacitance.

Figure 3.10 (a) shows the top view of the layout. The shaded area is the overlapping area composed of l_x and l_y. Figure 3.10 (b) is the side view of the layout. The overlap capacitance is between conductor a and b. The formula is well known as shown:

$$C_{a,b} = \epsilon_o \epsilon_r \frac{A}{h} = \epsilon \frac{l_x l_y}{h} \tag{3.1}$$

where ϵ is the appropriate dielectric coefficient of the volume separating conductors a and b, l_x and l_y are the horizontal and vertical rectangular dimensions of the overlap area, and h is the dielectric thickness. In Magic, the formula could be simplified as

$$C_{a,b}^{overlap} = coefficient \times A \tag{3.2}$$

where *coefficient* is defined and retrieved in technology file, and A is the overlap area. The value of overlap capacitance is accumulated to the capacitance between the nets to which a and b belong and is output in the `'ext'` file.

3.3.2 Sidewall Capacitance Extraction

Sidewall capacitance is formed when two adjacent conductors on the same level overlap on the vertical edges. The following figure illustrates a simple example.

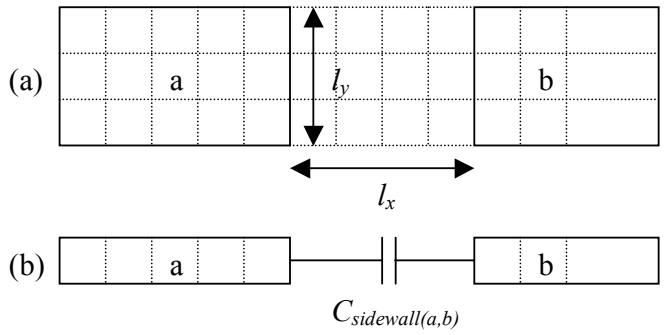

FIGURE 3.11: Sidewall capacitance.

Figure 3.11 (a) shows the top view of sidewall capacitance with l_x and l_y determination. Figure 3.11 (b) is the side view of sidewall capacitance between conductor a and b. Magic

uses a short-range formula as shown:

$$C_{a,b}^{sidewall} = \epsilon \frac{l_y h}{l_x} \tag{3.3}$$

and the formula could be simplified as

$$C_{a,b}^{sidewall} = coefficient \times \frac{L}{D} \tag{3.4}$$

where *coefficient* is defined in the technology file, L is the length of the edge overlap, and D is the distance between two conductors.

In addition, the sidewall capacitance coefficient value in Magic's technology file is halved, since Magic computes sidewall capacitance twice in its algorithms. In every plane, tiles are found by tile searching algorithm. For each tile, Magic looks for sidewall capacitance on four edges of the rectangle. As a result, if a sidewall capacitance is found, it will be found for the second time in its corresponding tile. This leads to two calculations of the same sidewall capacitance, and the coefficient parameters in the technology file need to be adjusted.

In the Magic technology file, there is a parameter named *"sidehalo"* to determine the furthest distance D to have sidewall capacitance between conductors. This parameter is set to be a small number such as 6 by default. Any distance beyond that point will be considered zero capacitance, and sidewall capacitance will not be extracted. Magic searches a short range on the outside of each edge for nearby parallel edges. The search is limited to the sidehalo of a few units away from the original edge because parallel edge capacitance decreases rapidly with distance.

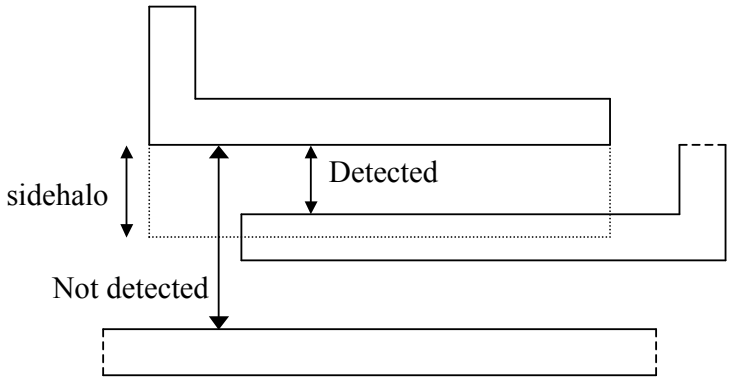

FIGURE 3.12: Sidewall capacitance detection (adapted from [19]).

3.3.3 Sidewall Overlap Capacitance Extraction

Sidewall overlap capacitance arises when one vertical edge of a conductor overlaps the horizontal surface of another conductor on a different plane. Magic's sidewall overlap capacitance estimate is unsophisticated. It does not accurately calculate fringing to finite-size horizontal surface or account for shielding influence of nearby conductors. Moreover, if conductors do not overlap or coincide with each other, Magic does not calculate the sidewall overlap capacitance. The following figure shows three different examples of sidewall overlap capacitance model in Magic.

Figure 3.13 shows three different cases to account for sidewall overlap capacitance in Magic. In Figure 3.13 (a), conductor b over conductor a with edges overlapping surface on each other. Conductor b's edge overlapping conductor a's surface is shown in Figure 3.13 (b), and Figure 3.13 (c) illustrates conductor b over conductor a without edges overlapping. The formula used in Magic could be simplified as following:

$$C_{a,b}^{sidewall-overlap} = coefficient \times edge \tag{3.5}$$

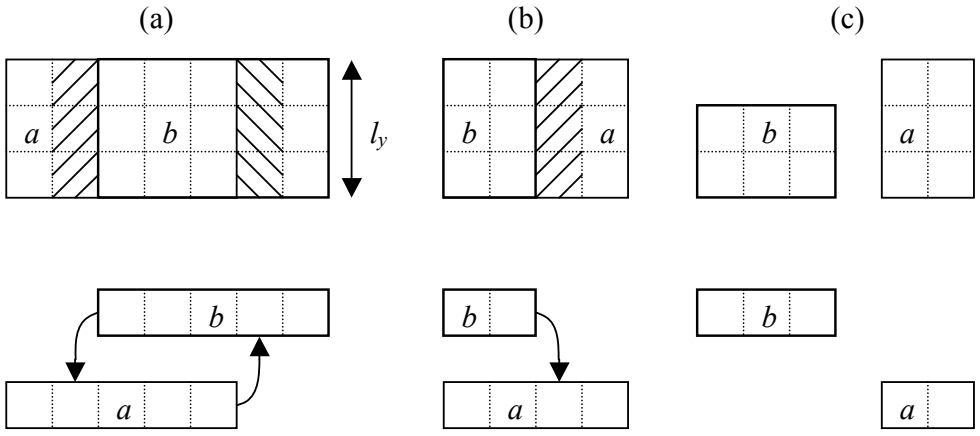

FIGURE 3.13: Sidewall overlap capacitance.

It is clearly observed that Magic does not account for fringing to finite-size horizontal surface; it only considers one lambda-size in estimating sidewall overlap capacitance, which is sent to be completely inaccurate when compared to CUP models and OptEM in chapter 4. In addition, when two conductors do not overlap or coincide with each other, Magic does not calculate sidewall overlap capacitance (see Figure 3.13 (c)). The sidewall overlap capacitance is proportional to the length of the overlap edge.

3.3.4 Substrate Capacitance Extraction

Every node has a corresponding capacitance respect to substrate, and its value is computed in the `'ext'` output file. The substrate capacitance computed in Magic is dependent on two factors: area and perimeter. In figure 3.14 (a), the substrate capacitance of conductor a is just its layout with respect to substrate as there is no shielding layer beneath it. The substrate capacitance of conductor b is the shaded layout because conductor a is shielding

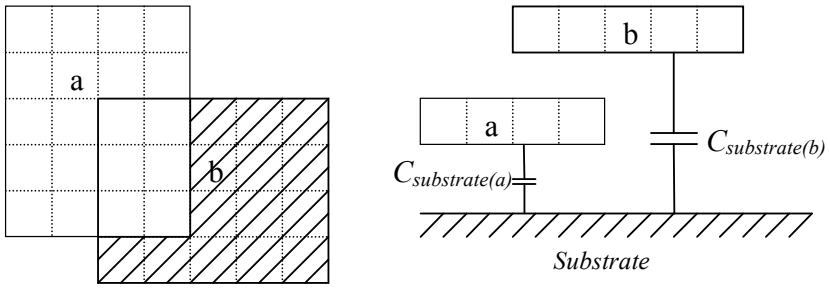

FIGURE 3.14: Substrate capacitance.

part of the layout of conductor b with respect to substrate.

$$C_b^{substrate} = coefficient \times area + coefficient \times perimeter \tag{3.6}$$

3.4 Magic Extraction Output

As mentioned in chapter 1, Magic is an interactive layout editing system for large-scale MOS custom integrated circuits, and it creates a layout file with extension `'mag'`. With the `'extract all'` command, `'magic'` files are extracted into `'extract'` files. The circuit netlist is then described in a text format with extension `'ext'`, and this file can further be converted to simulation file formats, such as sim and spice formats. The open source IRSIM simulator can read in sim files and simulate the models of circuits. Spice files can also be simulated with similar open source software tools, such as ngspice and hspice.

3.4.1 Extraction File

In every `'ext'` file, some standard information is listed, such as version, technology, style, scale and resistclasses. After standard information, each set of node and coupling capacitance information is shown in the following format.

Table 3.5: Node output.

definition	example	unit
Node name	node	
"bottom layer"	"*a_n*8_28#"	
Resistance	156	milliohms
Substrate capacitance	21264	attoFarad
Left down corner (x)	-8	λ
Left down corner (y)	28	λ
Layer name	pc	
Nwell area	0	λ^2
Nwell perimeter	0	λ
Pwell area	0	λ^2
Pwell perimeter	0	λ
Poly area	104	λ^2
Poly perimeter	60	λ
Nwell area	0	λ^2
Nwell perimeter	0	λ
Pwell area	0	λ^2
Pwell perimeter	0	λ
Poly area	0	λ^2
Poly perimeter	0	λ
Metal1 area	328	λ^2
Metal1 perimeter	180	λ
Metal2 area	0	λ^2
Metal2 perimeter	0	λ
Metal3 area	0	λ^2
Metal3 perimeter	0	λ

And the following table shows the coupling capacitance format formed between any two circuit nodes.

TABLE 3.6: Coupling capacitance output.

Cap	'node one'	'node two'	Capacitance value (aF)
cap	"$a\_1\_20\#''$	"$a\_n8\_28\#''$	3156

3.4.2 Simulation File

There are plenty of simulation tools in the world to model the circuit. Simulations and analysis do not perform what the circuit does, but it demonstrates what the model of the circuit does.

Two types of simulation files can be generated from an extracted file by Magic: `sim` and `spice`. `'Sim'` files work with the IRSIM simulator. SPICE is the most widely use circuit simulator for detailed analysis of transistor level designs, and SPICE files work with very well recognized simulators, such as Hspice, ngspice, and so on.

Some applications with examples simulated in ngspice are presented in Chapter 5.

3.5 Summary

This chapter provides a brief exploration of the Magic VLSI layout tool, an open source tool that was originally developed at the University of California, Berkeley in 1983. It is shown that Magic is an efficient hierarchical circuit extractor because it uses corner-stitching, pattern-matching, and Manhattan distance. Magic's extractor demonstrates corner-stitching is well-suited for use in a circuit extractor[19].

Regarding Magic's capacitance extraction, it is found that Magic calculates sidewall overlap capacitance by a simple method; it does not calculate the fringing to finite-size

horizontal surface or account for shielding influence of nearby conductor. In addition, sidewall capacitance is computed in a short-range formula in Magic. As a result, to rectify the deficiencies in Magic, CUP models were applied to Magic. The next chapter assesses the implementation of CUP models and derivatives in Magic to improve the performance, that is to increase accuracy, and compute process variation sensitivities.

Chapter 4

Integration of New Capacitive Model into Magic

This chapter describes the implementation of HILEX (HIerarchical Layout EXtraction tool)/CUP models in Magic with process variation sensitivity in capacitance extraction.

4.1 CUP models

Digital Equipment Corp. developed proprietary CAD software for layout extraction called HILEX/CUP, consisting of HILEX (roughly equivalent to Magic) and CUP (roughly equivalent to Magic's parasitic capacitance and resistance extraction algorithms). These codes were licensed to Simplex Solutions Inc. in 1995, where they formed the basis of the Fire and Ice interconnect physical verification software, eventually acquired by Cadence [67].

HILEX reduced the layout geometry into base elements or so called trapezoids, and then the transformations were applied to them to compute the capacitance. The transformation

was necessary as capacitance models were extracted using 2-D simulations whereas the actual layout geometry was 3-D in nature. There were three components to estimate the capacitance of each trapezoid: area, lateral, and fringe capacitances.

A description of the capacitance extraction procedure in CUP is presented:

- The layout obtained from HILEX is flattened and passed to CUP.
- The layout geometry is divided into stripes.
- Each stripe is fractured into elemental areas that consist of rectangles or triangles.
- Starting from the lowest level, capacitance for each elemental area is calculated.

As mentioned earlier, the capacitance of interest at any node consists of area, lateral and fringe capacitance (see Figure 4.1). In general, the interconnect capacitance at any node is given by the sum of all the capacitance components.

$$C_{total} = \Sigma C_{area} + \Sigma C_{lateral} + \Sigma C_{fringe} \tag{4.1}$$

4.1.1 Area Capacitance

Area, or overlap in Magic, capacitance is formed by the surface overlap of two conductors in different planes. Figure 4.2 shows the area capacitance model in cross-sectional view between conductors metal 2 and metal 1 and between metal 1 and polysilicon, and the dimensions width (l_x), thickness (t), and height (h) are labeled for reference. Length l_y is into the page. Magic and CUP models use the same equation to evaluate area capacitance, the well-known

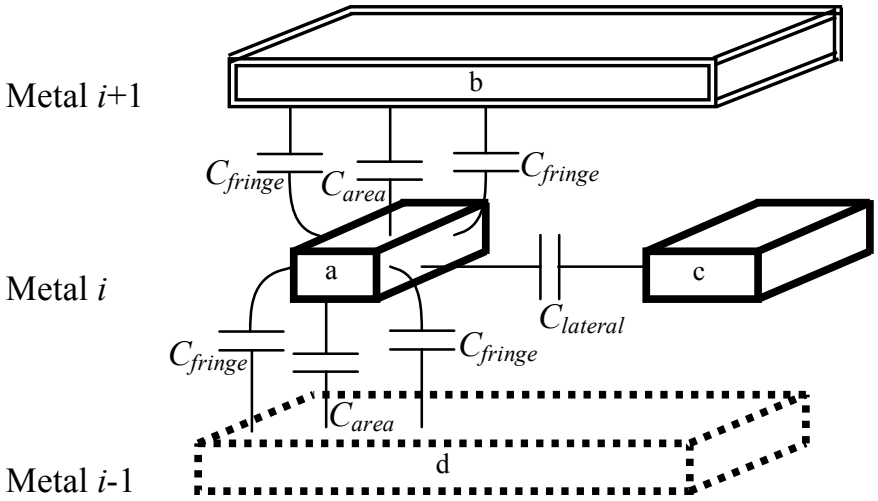

FIGURE 4.1: Capacitance configuration.

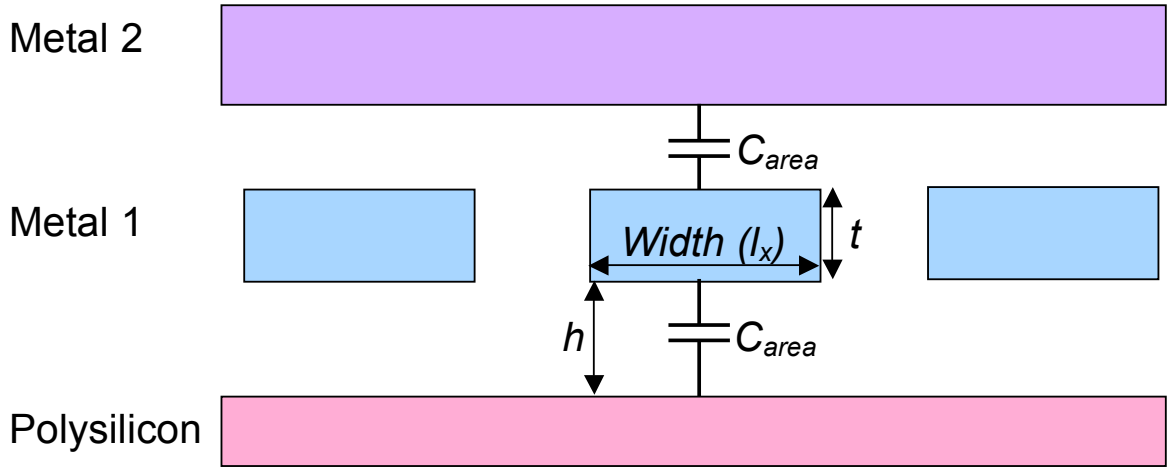

FIGURE 4.2: Area capacitance.

formula 3.1:

$$C_{area} = \epsilon_o \epsilon_r \frac{A}{h} = \epsilon \frac{l_x l_y}{h} \tag{4.2}$$

where ϵ is the appropriate dielectric constant. The interesting part is to get the sensitivity of capacitance extraction, so the derivatives of capacitance formula are computed with the following formulae:

$$\frac{\partial C_{area}}{\partial x} = \epsilon \frac{l_y}{h} \; ; \quad \frac{\partial C_{area}}{\partial l_y} = \epsilon \frac{l_x}{h} \tag{4.3}$$

The partial derivative with respect to x is just l_y times the coefficient ($\frac{\epsilon}{h}$) in the Magic technology file. Therefore, the derivative of area capacitance is very straightforward.

4.1.2 Lateral Capacitance

The lateral capacitance, or sidewall in Magic, is the coupling between two of the same conductors when they are adjacent. An example of the lateral capacitance model is shown with side view in Figure 4.3.

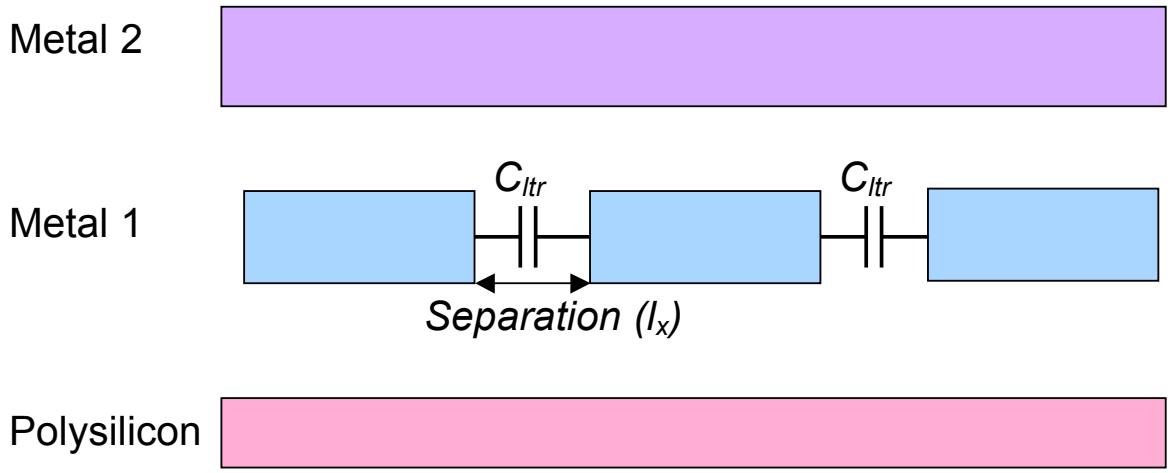

FIGURE 4.3: Lateral capacitance.

The lateral capacitance depends on the length of the edge overlap, and the formula presented in the CUP model is:

$$C_{lateral} = \frac{\sum_{i=1}^{n} A_i \cdot F^i_{l_1 l_1}(l_x) \cdot l_y}{l_x \cdot l_y} \tag{4.4}$$

where

$$F^i_{l_1 l_1}(l_x) = a_0 + \frac{a_1}{l_x} + \frac{a_2}{{l_x}^2} + \frac{a_3}{{l_x}^3} + \frac{a_4}{{l_x}^4} \tag{4.5}$$

where A_i is the area of a region within the lateral capacitance rectangle with a particular combination of upper and lower shading layers, i is the region number, n is the number of shading regions, l_y is the length of edge parallel to the other, and l_x is the separation. The coefficients are fitted to 2D numerical simulations.

Notice that it is possible to do process variation in Magic without using CUP formulae. However, the CUP formulae should be more accurate. The derivative of Magic's formula for sidewall capacitance is shown below but not implemented:

$$C^{Magic}_{sidewall} = \epsilon \frac{l_y h}{l_x} \Rightarrow \frac{\partial C^{Magic}_{sidewall}}{\partial l_x} = -\epsilon \frac{l_y h}{{l_x}^2} \tag{4.6}$$

It is important to realize that lateral capacitance also depends on the presence of shading layers above and/or below the level of the two conductors, so separate sets of different coefficients were derived. Partially-shielded cases are handled by an area-weighted average. A comprehensive figure illustrates this procedure (see Figure 4.4).

It is observed that the lateral capacitance rectangle between two metal 1 conductors is divided into separate regions in Figure 4.4 (b). The area A_i of each region i is given by its length l_i and width d_i, and the sum of all the areas is

$$l_x l_y = \Sigma^n_{i=1}\ A_i = \Sigma^n_{i=1}\ l_i d_i \tag{4.7}$$

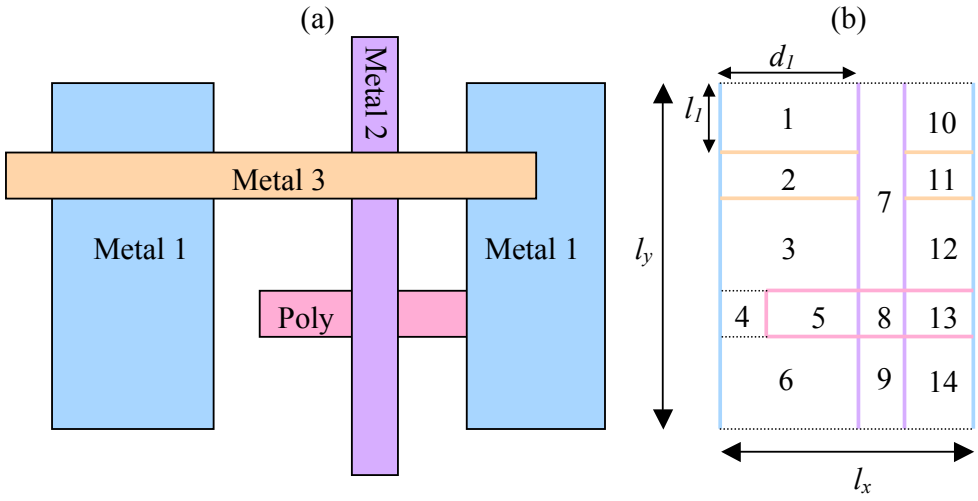

FIGURE 4.4: Lateral capacitance schematic.

The colors indicate the real edges of interesting section to account for lateral capacitance, and each division with numbers labeled have a different set of shading layers above and below. The linewidth derivative of lateral capacitances is more complicated than area models.

The lateral capacitance in every region can now be expressed as:

$$C^{lateral}_{region\ i} = \frac{l_i d_i}{l_x}\left(a_0 + \frac{a_1}{l_x} + \frac{a_2}{{l_x}^2} + \frac{a_3}{{l_x}^3} + \frac{a_4}{{l_x}^4}\right) \tag{4.8}$$

with $l_i \cdot d_i$ equal to the area in Eq. 4.4. As a result, the linewidth derivative is computed as:

$$\frac{\partial C^{lateral}_{region\ i}}{\partial t_{metal\ j}} = \frac{\partial C_i}{\partial d_i}\frac{\partial d_i}{\partial t_j} + \frac{\partial C_i}{\partial l_i}\frac{\partial l_i}{\partial t_j} + \frac{\partial C_i}{\partial l_x}\frac{\partial l_x}{\partial t_j} \tag{4.9}$$

where j indicates the metal conductor, and the first term in each product are shown:

$$\frac{\partial C^{lateral}_{region\ i}}{\partial d_i} = \frac{l_i}{l_x}\left(a_0 + \frac{a_1}{l_x} + \frac{a_2}{{l_x}^2} + \frac{a_3}{{l_x}^3} + \frac{a_4}{{l_x}^4}\right) \tag{4.10}$$

$$\frac{\partial C^{lateral}_{region\ i}}{\partial l_i} = \frac{d_i}{l_x}\left(a_0 + \frac{a_1}{l_x} + \frac{a_2}{{l_x}^2} + \frac{a_3}{{l_x}^3} + \frac{a_4}{{l_x}^4}\right) \tag{4.11}$$

$$\frac{\partial C^{lateral}_{region\ i}}{\partial l_x} = l_i \cdot d_i\left(\frac{-a_0}{{l_x}^2} + \frac{-2a_1}{{l_x}^3} + \frac{-3a_2}{{l_x}^4} + \frac{-4a_3}{{l_x}^5} + \frac{-5a_4}{{l_x}^6}\right) \tag{4.12}$$

The second factor in each product in Eq. 4.9 is either a 0 or ± 1, depending on metal layer and shape. Each division has a different set of shading layers above and below. A pixel-based search algorithm was invented to perform the lateral and fringe capacitance computations accurately and is discussed in next section.

4.1.3 Fringe Capacitance

In earlier technologies prior to DSM technology, fringe capacitances could be neglected because they are very small in terms of total capacitance. As the technology scales down to DSM regime, fringe capacitance is becoming increasingly important since the interconnects are even closer than ever. Fringing becomes an important component, so it is necessary to adjust the accuracy in order to compute the total capacitance correctly. As described previously, fringe capacitance is formed between the vertical edge of one conductor and the horizontal surface of a second conductor above or below. Figure 4.5 illustrates the fringe capacitance for the metal1 conductor in the middle with metal2 and polysilicon conductors in cross-sectional view.

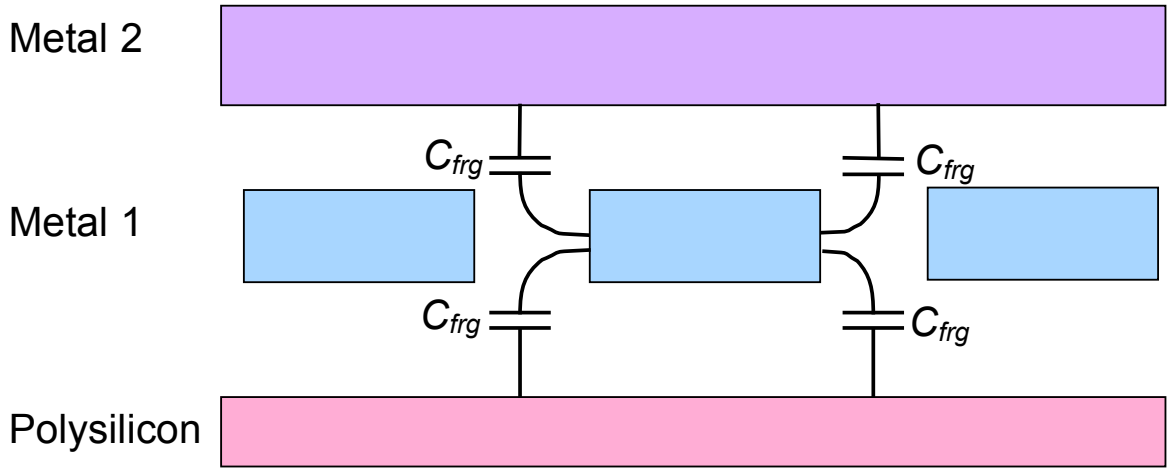

FIGURE 4.5: Fringe capacitance.

The original CUP formula for fringe capacitance rectified Magics deficiencies mentioned in Chapter 3 by considering fringing to finite areas:

$$C_{fringe} = l_y b_0 (e^{-b_1 x_1} - e^{-b_1 x_2}) \tag{4.13}$$

with coefficients b_0, where x_1 and x_2 are the distances from the vertical edge of conductor a to the near and far edges of conductor d (see Figure 4.6). The same approach to handling the presence of shielding layers is applied as in lateral capacitance. Separate sets of coefficients were fitted to account for shielding layers on above or below. With conductors a and d overlapped, x_1 is defined to be 0. The dotted line shows the rectangular area which are taken into account for fringe capacitance computation.

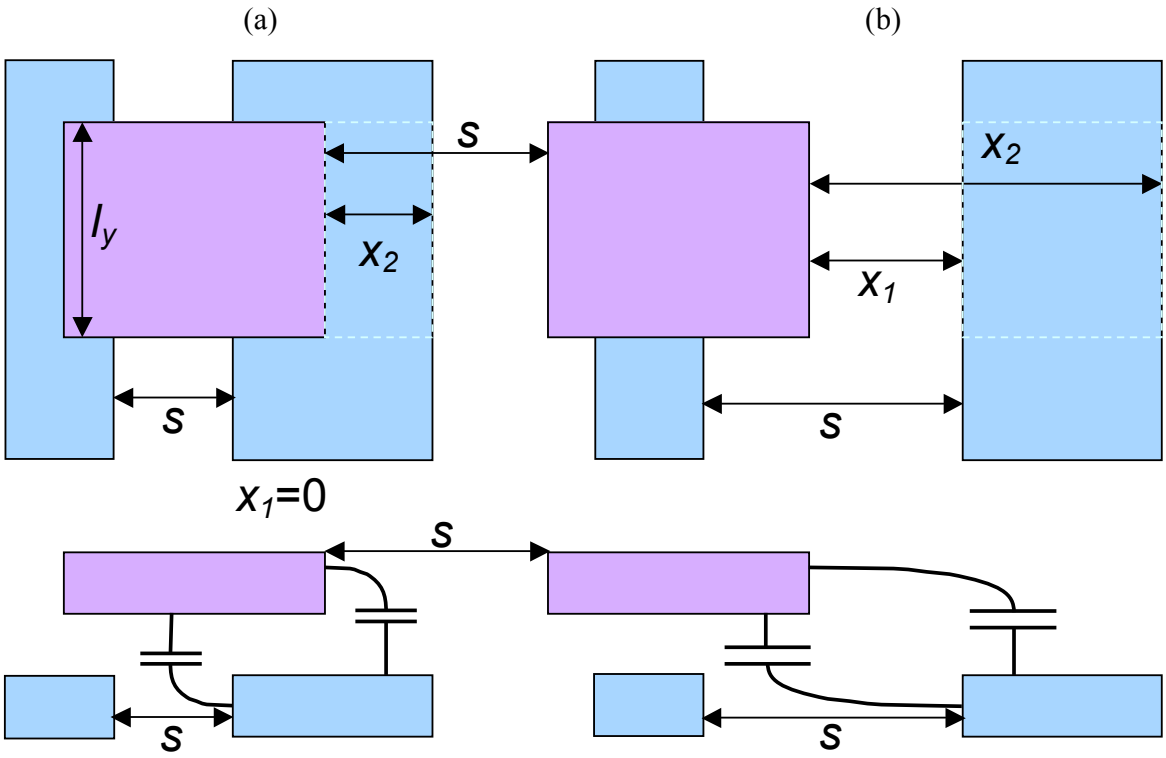

FIGURE 4.6: Lateral capacitance between two layers.

The CUP formula implemented in the present work also considers the effect of a neighbouring conductor at distance s to further improve accuracy in the DSM regime.

$$C_{fringe} = l_y \left[b_0 + \frac{b_1}{\left(1 + \frac{b_2}{s}\right)^{b_3}} \left(1 - e^{-\left(\frac{b_4 + x}{b_5 + s \cdot b_6}\right)} \right) \right] \tag{4.14}$$

In the limit $x \to +\infty$, for $(b_5 + s \cdot b_6) > 0$,

$$C_{fringe} = l_y \left[b_0 + \frac{b_1}{\left(1 + \frac{b_2}{s}\right)^{b_3}} \right] \tag{4.15}$$

In the limit $s \to +\infty$, $x \to +\infty$, for $b_6 > 0$ and $b_3 > -1$,

$$C_{fringe} = l_y \cdot b_0 \tag{4.16}$$

The equation could be simplified if x or s approaches $+\infty$. Eq. 4.15 is used to compute fringing to substrate capacitance as the distance is infinity. Eq. 4.16 is equivalent to the Magic model in having a single parameter and not taking into account the width of the horizontal surface.

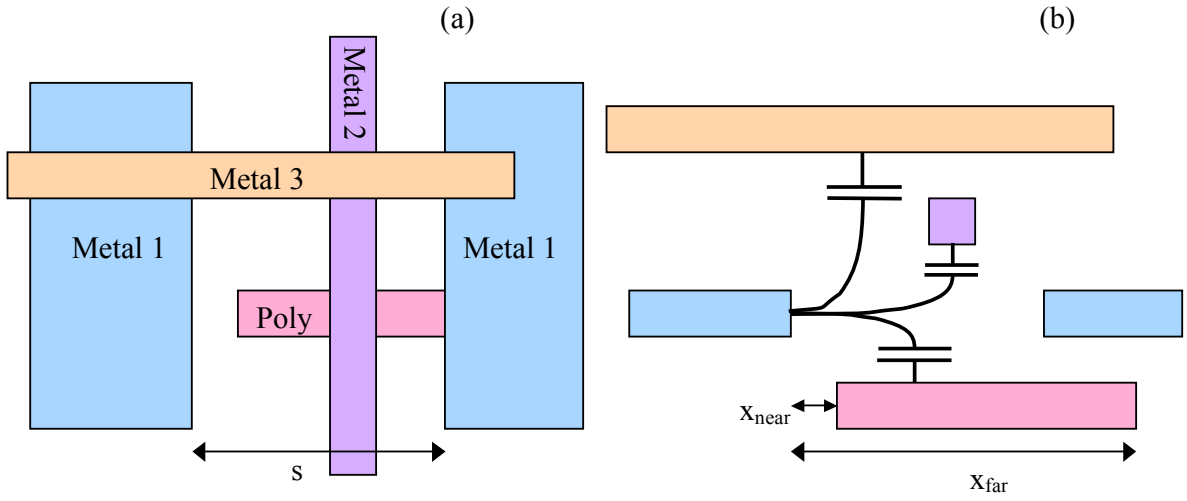

FIGURE 4.7: Fringe capacitance schematic.

Figure 4.7 shows the top and side views of fringe capacitance models, and the separation of the same conductors on the relevant edge is computed. If a conductor on the same metal layer is not found in the layout, $s = +\infty$. Magic fringe model is thus improved to the much more sophisticated CUP fringe model. The linewidth derivative of the CUP fringe model is

given by

$$\frac{\partial C_{fringe}}{\partial l_{metal\ j}} = \frac{\partial C_{fringe}}{\partial s}\frac{\partial s}{\partial l_j} + \frac{\partial C_{fringe}}{\partial x}\frac{\partial x}{\partial l_j} \tag{4.17}$$

with the first factor in each product

$$\frac{\partial C_{fringe}}{\partial s} = \frac{-b_1 b_6 \left(b_4 + x\right)}{\left(b_5 + s \cdot b_6\right)^2 \left(1 + \frac{b_2}{s}\right)^{b_3}} \left(e^{-\frac{b_4+x}{b_5+s\cdot b_6}}\right) + \frac{b_1 b_2 b_3}{s^2 \left(1 + \frac{b_2}{s}\right)^{b_3+1}} \left(1 - e^{-\frac{b_4+x}{b_5+s\cdot b_6}}\right) \tag{4.18}$$

$$\frac{\partial C_{fringe}}{\partial x} = \frac{b_1 e^{-\frac{b_4+x}{b_5+s\cdot b_6}}}{\left(b_5 + s \cdot b_6\right) \left(1 + \frac{b_2}{s}\right)^{b_3}} \tag{4.19}$$

In the same way as for lateral capacitance, the second factor in each product in Eq. 4.17 is either a 0 or 1, and the pixel-based search method is applied to compute fringing capacitance as well to minimize the cost.

4.1.4 Model Fit

As discussed in chapter 2, CUP model coefficients are fitted to Field Solver results and generated in Techgen, which uses the Levenberg-Marquardt (LM) method to fit the model to the data. The curve fits to the field solver results figures are plotted below for lateral (see Figure F.1 to Figure F.3) and fringe capacitances (see Figure F.4 to Figure F.11) since area capacitance is straight-forward.

Lateral capacitances have a very accurate model fitting curve as the lateral capacitance formula is not complicated. Lateral capacitance between three different levels on metal 1, 2, and 3 is plotted separately with different cases of shading layers.

Fringe capacitance curve fits provide a fairly good result for each case. There are more cases in fringe capacitance because two parameters are specified in the formula, s and x. For each case, a variation of s, separation, is applied and plotted. As separation of conductors are further apart, the capacitance value is larger due to the less shielding influence.

4.2 Changes to Magic File Formats

The Techgen program was written to generate CUP models fits to BEM field solver results. It includes features such as anisotropic dielectric layers and multiple sets of material properties. For example, both dielectric constants and thermal conductivity are included. Techgen's results are used to modify existing Magic Technology files as explained here.

4.2.1 Technology File Changes

Since new CUP models are implemented in Magic, some new parameters must be included in the technology file. Table 4.1 lists a set of new parameters, and they are loaded in when Magic reads the technology file at startup. The extracted section of the technology file used in this thesis is reproduced in Appendix D.

4.2.2 New Extraction Output

Once a layout is finished, its parasitic capacitance is extracted to an output file for simulations. Since derivatives are implemented in Magic, a new format extract output file needs to be generated to write the derivative information. Table 4.2 shows the capacitance derivative array implemented in Magic. $\partial/\partial L_1$ and $\partial/\partial L_2$ are linewidth derivatives. Two gradients

TABLE 4.1: New technology parameters.

parameter	No. of coefficients	new parameter	No. of coefficients
areacap	1	CUPareacap	1
sidewall	1	CUPsidewall	5
sideoverlap	1	CUPsideoverlap	8
resist	1	CUPresist	1
cscale	1	CUPscale	1
Example			
Standard Magic model statement: sideoverlap $(m1, ndc, pdc, nwc, pwc, nbdc, capc, ec, clc, emc, pbc, pc, via)/m1$ $\sim (m1, ndc, pdc, nwc, pwc, nbdc, capc, ec, clc, emc, pbc, pc, via)/m1$ $(m2, m2c, m3c, pad)/m2$ 13.27			
Magic CUP model statement: CUPsideoverlap $(m1, ndc, pdc, nwc, pwc, nbdc, capc, ec, clc, emc, pbc, pc, via)/m1$ $\sim (m1, ndc, pdc, nwc, pwc, nbdc, capc, ec, clc, emc, pbc, pc, via)/m1$ $(m2, m2c, m3c, pad)/m2$ 3.498741 30.068360 0.074953 40.124750 0.101676 2.915537 0.112488 14.857140			

are presented for each layer, to handle coincident edges, as discussed in the next section. $\partial/\partial t$ and $\partial/\partial h$ are thickness and height derivatives, and they may be implemented in future projects.

The capacitance derivative array is added to the end of every node and coupling capacitance in `'ext'` files. Simulators `'ext2sim'` and `'ext2spice'` are also modified to read the derivatives after each capacitance. They output the derivatives as comments in the sim and spice files. Typical examples are shown in Appendix E.

4.3 Linewidth Variation

There are many different types of process variations, and the focus in this research is on the linewidth variation. A figure is shown to explain the width expansion and reduction for a given a layout (see Figure 4.8). Suppose an isometric layout is drawn with each grid showing one λ unit. A width bloat by one λ would expand one λ in all widths. On the other hand,

TABLE 4.2: Capacitance derivative array.

layer	definition	Magic/CUP
	cap	[0]
pwell	$\partial/\partial L_1$	[1]
pwell	$\partial/\partial L_2$	[2]
nwell	$\partial/\partial L_1$	[3]
nwell	$\partial/\partial L_2$	[4]
poly	$\partial/\partial L_1$	[5]
poly	$\partial/\partial L_2$	[6]
poly	$\partial/\partial t$	[7]
poly	$\partial/\partial h$	[8]
metal 1	$\partial/\partial L_1$	[9]
metal 1	$\partial/\partial L_2$	[10]
metal 1	$\partial/\partial t$	[11]
metal 1	$\partial/\partial h$	[12]
metal 2	$\partial/\partial L_1$	[13]
metal 2	$\partial/\partial L_2$	[14]
metal 2	$\partial/\partial t$	[15]
metal 2	$\partial/\partial h$	[16]
metal 3	$\partial/\partial L_1$	[17]
metal 3	$\partial/\partial L_2$	[18]
metal 3	$\partial/\partial t$	[19]
metal 3	$\partial/\partial h$	[20]

the shrinking of one λ would reduce the layout by one λ. Of course, the variation in real fabrication is unpredictable, generally only a fraction of a λ unit, but the minimum design unit is one λ in Magic. Demonstrations of bloating and shrinking in this thesis will therefore all be shown in terms of whole λ units.

Linewidth variation on level i, t_i, are considered to be either *shrinking* ($t_i < 0$), that is all real edges shift in the direction of the metal, or *bloating* ($t_i > 0$), where all real edges shift in the direction of dielectric. The set of linewidth variations on all levels is given as $\vec{t} = (t_1, t_2, \ldots)$.

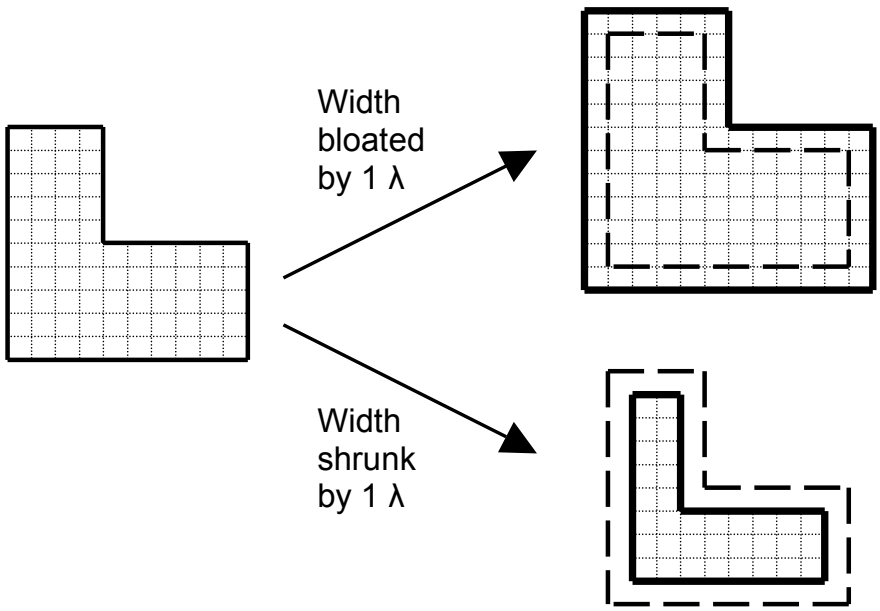

FIGURE 4.8: Linewidth variation.

4.4 Algorithms Implementation

To implement new extraction formulae (HILEX/CUP models), new metal search algorithms are invented and designed in Magic tool. That is, to include new data structures for new derivative information and other physics, a significant modification to Magic is necessary.

To account for derivatives of capacitance sensitivity to variations, the information about the layout planes that correspond to metal levels, and the types of edges around the rectangular areas on which capacitance is evaluated. The first change is the data structure of hash table which stores capacitance throughout Magic. Originally Magic only stores a real scalar capacitance value for each node-pair that has a coupling capacitance, but now the derivatives with respect to each metal layer need to be stored as well. Therefore, a 2-D vector is implemented to store the capacitance and two linewidth derivatives for every metal level.

Once the hash entry/table is capable of storing the 2-D matrix, the derivative information is ready to be implemented. The following sections describe the detailed discussion of real and coincident edges and pixel-based search algorithms.

4.4.1 Real Edges

In particular, edges may be *real* or *unreal*, and they may also be *coincident*, which will be discussed in next subsection. A real edge is an edge that corresponds to a vertical conductor face. That is, the interesting edge in a given layout. For instance, Figure 4.9 (a) illustrates two overlapping conductors X and Y on metal levels i and *i-1*, and the perimeter of the area capacitance rectangle between X and Y composed by real edges is shown in (b).

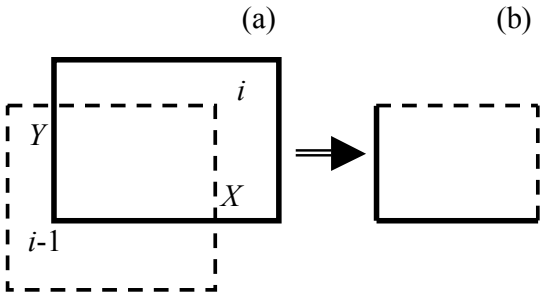

FIGURE 4.9: A sample layout illustrating real edges. (a) Conductors X and Y on levels i and *i-1*. (b) The real edges of overlapping area between X and Y.

4.4.2 Coincident Edges

A coincident edge occurs when two or more real edges on different levels coincide with each other. In such a scenario, real edges may be turned into two or more different cases regarding the treatment of bloating and shrinking on metal layers. The inclusion of coincident edge is invented and presented to comprise a more accurate estimate in process variations.

In the illustration in Figure 4.10 (a), there are three overlapping conductors U, V, and T on levels *i+1*, *i*, and *i-1* respectively. Consider conductor V having a coincident edge with T on one edge. In this case, a distinction must be made between the handling of bloating and shrinking. In Figure 4.10 (b), conductors T's edge is both real and coincident with V's edge, and this case may be treated as default real edges for this layout. Figure 4.10 (c) demonstrates when conductor V bloats, the area is reduced according to V, and real edges remain the same as the default. Another case is shown in Figure 4.10 (d), where either conductor V shrinks or conductor T shrinks. If V shrinks, the area is unaffected, where as the area is reduced along the entire rectangular length if T *shrinks*. However, both cases turn conductor T's coincident edge, on the right side, to real edge. Consequently,

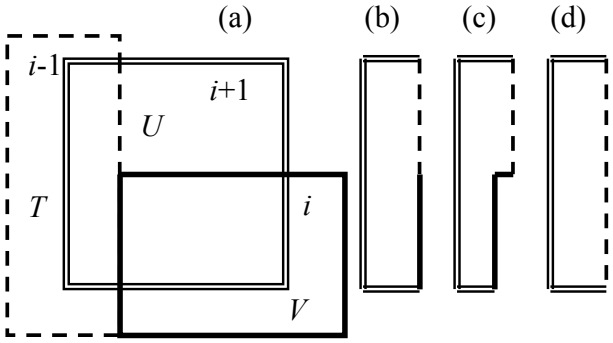

FIGURE 4.10: A sample layout illustrating coincident edges. (a) Three overlapping conductors U, V, and T on metal levels *i+1*, *i*, and i-1, respectively. (b) The perimeter of the area capacitance rectangle between U and T consists of real edges. Conductor T's edge is both real and coincident with V's edge. (c) When level i bloats, the area is reduced according to the length of the level i real edges. (d) When level i shrinks, the area capacitance is unaffected.

separate bloating and shrinking derivatives for each layer are computed and stored with the assumption that the other layer's dimensions remain constant.

A new function is written to compute the real and coincident edges, given two tiles along with coincident edges. A pseudo code is provided in Figure 4.11, and the C code can be obtained in Appendix G.

To begin with, given two tiles and the area to search with, this `extRealEdge()` function determines the real edges of the tiles in the given search area. First, it determines if the area is a vertical or horizontal segment. If the area is a vertical segment, it is easier to implement the real edges due to the way tiles are defined. (A vertical segment only involves one tile). On the other hand, if the area is a horizontal segment, it is harder to find the real edges for the tiles because there may be some tiles on the above or below attached to the horizontal segment. A top or bottom walk is performed to take care of the unreal edges in the algorithms.

If the area is a normal rectangle, four sides: top, right, left, and bottom walks are performed step by step. Once again, right and left side walks are easier to implement because they are vertical segments.

An example of area capacitance real edge implementation is provided in Appendix C. It shows all cases for metal 3 over metal 2, metal 1 and polysilicon, and step by step solution is illustrated.

4.4.3 Pixel-based Search Algorithms

In order to calculate lateral and fringe capacitances accurately, a pixel-based search algorithm is implemented to evaluate the capacitance model. A pixel corresponds to a square of the minimum design unit λ. The presence of a conductor above or below is noted in

```
if (the search area is a vertical segment)
{
   if ( RIGHT/LEFT(tile_above/below) == segment )
      compute the real edge for tile_above and tile_below;
}
else if (the search area is a horizontal segment)
{
   if (TOP(tile_below) == segment
      walk thru TOP of tile_below and compute real/coincident edges
   if (TOP(tile_above) == BOTTOM(area)
      walk thru TOP of tile_above and compute real/coincident edges
   if (BOTTOM(tile_below) == TOP(area)
      walk thru BOTTOM of tile_below and compute real/coincident edges
   if (BOTTOM(tile_above) == TOP(area)
      walk thru BOTTOM of tile_above and compute real/coincident edges
}
else /* the search area is a rectangle */
{
/* TOP */
   if ( TOP(tabove)>TOP(tbelow) && TOP(tbelow) == BOTTOM(area) )
   {
      walk thru TOP of tile_below and compute real edge for tile_below;
   }
   else if ( TOP(tile_above) == TOP(area) )
   {
      compute real edges for tile_above;
      if TOP(tile_above) == TOP(tile_below)
      {
         walk thru TOP of tile_below to compute coincident edges for tile_above/below;
      }
      walk thru TOP of tile_above;
      subtract real edge of tile_above;
      if ( TOP(tile_above) != TOP(tile_below) )
         subtract coincident edge of tile_above;
      if ( TOP(tile_above) == TOP(tile_below) )
      {
         walk thru TOP of tile_below and subtract real/coincident edges;
         compute real_edge for tile_below;
      }
  }

/* RIGHT */
   if ( RIGHT(tile_above)>RIGHT(tile_below) && RIGHT(tile_below) == RIGHT(area) )
   {
      walk thru RIGHT of tile_below and compute real/coincident edges for tile_below;
   }
   else if ( RIGHT(tile_above) == RIGHT(area) )
   {
      if ( RIGHT(tile_above) == RIGHT(tile_below) )
         compute/subtract real/coincident edges for tile_above/below;
         walk thru RIGHT of tile_above;
   }

/* BOTTOM */
   follow the /*TOP*/ algorithms with opposite direction

/* LEFT */
   follow the /*RIGHT*/ algorithms with opposite direction
}
```

FIGURE 4.11: extRealEdge() function pseudocode.

every pixel of the rectangle. The derivatives are also calculated with real edges accumulated in the gradient data structure. Figure 4.12 shows a lateral capacitance sample layout with sensitivity extraction algorithm.

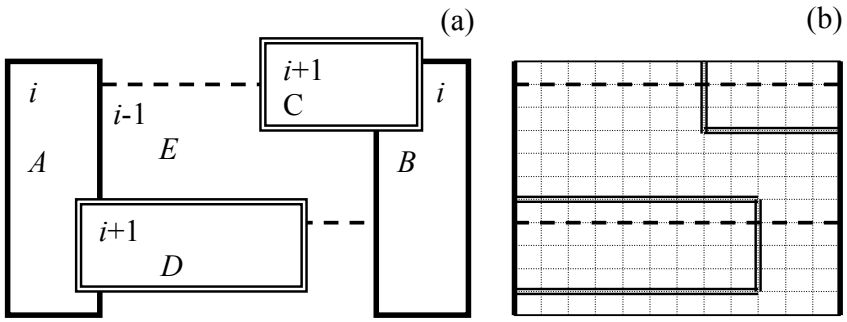

FIGURE 4.12: Pixel-based search algorithm.

Figure 4.12 (a) illustrates a sample layout for lateral capacitance between conductors A and B on level i, with shielding conductors C and D on level $i + 1$ and E on level $i - 1$, and Figure 4.12 (b) shows the rectangle on which lateral capacitance is calculated, showing λ-sized pixels.

To account for fringe capacitance, a similar approach is applied to calculations. One difference is that each fringe capacitance rectangle has a separate capacitance with respect to the conductor above or below in each pixel. Moreover, there is a maximum side distance to compute fringe capacitance in each conductor, and the distance to the edge of a neighboring conductor on the same level is also considered in every capacitance rectangle. The C code is available in Appendix G.

4.5 Summary

This chapter first presents CUP models and then three main coupling capacitances: area, lateral, and fringe.

CUP models have been implemented to improve Magic's sidewall overlap capacitance. Magic is now able to calculate the fringing to finite-size horizontal surfaces and account for nearby shielding conductors. Moreover, the CUP lateral capacitance model has improved Magic's short-range sidewall capacitance extraction.

Linewidth variations are implemented into Magic, and a new function is written to calculate real and coincident edges for derivative computation when evaluating capacitances. In the next chapter, simulations are performed to demonstrate and validate the CUP models and linewidth process variation sensitivity.

Chapter 5

Simulation and Evaluation

Before the evaluation of process variation sensitivity is discussed further, a figure presenting the abstract view of computer-aided design (CAD) flow in Magic before and after the proposed process is shown.

Figure 5.1 (a) is the original CAD flow in Magic, which creates tile trees from the layout. Magic computes the nominal capacitances and then lumps C to node-pairs and stores them in `'ext'` files for simulations in SPICE or IRSIM. Figure 5.1 (b) shows the new implemented capacitances, CUP models, with gradients, ∇C. Some user-specified process variation vector $\vec{t}$ is applied to obtain varied C'. A shell script is performed to merge information to SPICE wirelists prior to simulations.

This chapter is divided into three parts: a capacitance extraction comparison is made between Magic and a commercial tool, Space 3D from OptEM, to validate CUP model capacitances values. Then linewidth process variation sensitivities are evaluated to determine the accuracy of the derivative-based method. Finally the cost of implemented method,

runtime, is discussed.

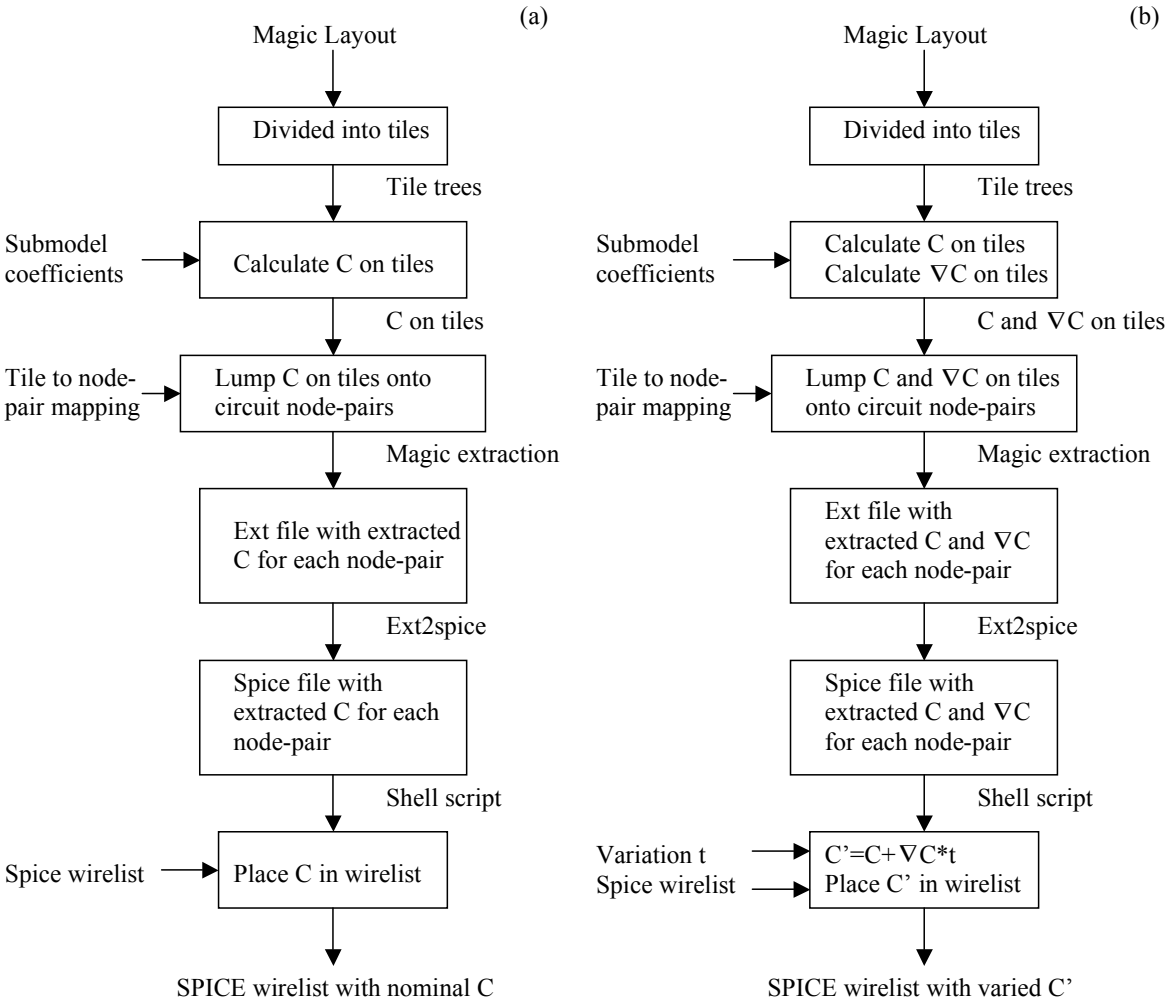

FIGURE 5.1: CAD flow.

5.1 Capacitance Comparison

Capacitance accuracy is very important in the DSM technology. This makes the evaluation of CUP models very attractive. In this section, comparisons are made between Magic using

CUP models and the standard Magic models, and the Space 3D extraction tool from OptEM for the same technology.

5.1.1 Space 3D - OptEM

A circuit layout is designed and created in Space 3D, and then laid out again in Magic to perform an extraction to generate both CUP and Magic netlists. The layout drawn in Space 3D is shown in Figure 5.2.

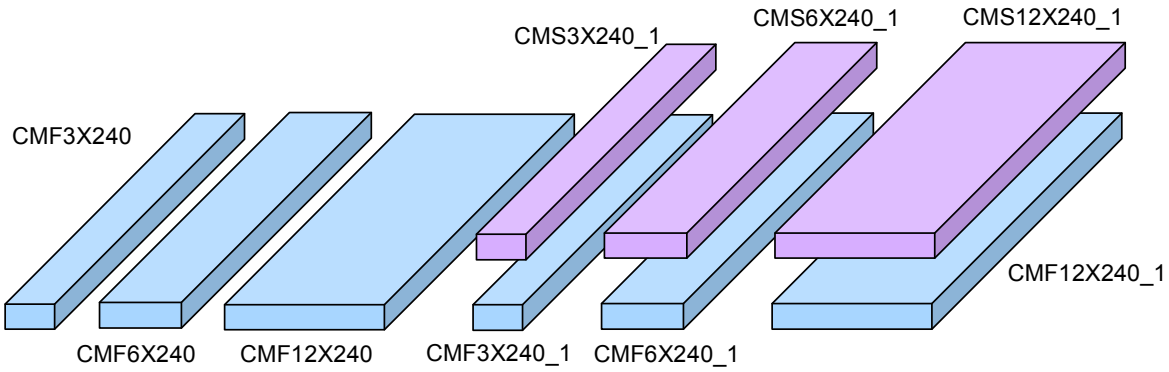

FIGURE 5.2: Space 3D layout.

This layout is composed of nine pieces of metal layers, CMF3X240 is a metal 1 conductor with dimension 3x240 λ^2 (width by length). CMF6X240 and CMF12X240 are 6 and 12 λ wide respectively. A 3 λ distance separates these conductors. CMF3X240_1 and CMS3X240_1 are metal 1 and metal 2 layers overlapped exactly in superposition, and the remaining conductors are also the same size and location. A 6 λ distance separates these conductors from each other and CMF12X240. This circuit layout resulted in 27 separate capacitances in Space 3D, and the results are shown in the next section.

5.1.2 Capacitance Comparison Results

The capacitance values with node names and % error are tabulated in Table 5.1. Under the 'Capacitance Label' column, the '*C*' section is the type of capacitance extracted in CUP models, which only includes area, lateral, and fringe capacitances. Since Space 3D is a BEM field solver, it does not use these separate model types.

TABLE 5.1: Capacitance comparison.

Capacitance Label (*C*)	**Node1**	**Node2**	**Space 3D** C(fF)	**CUP** C(fF) [% error]	**Magic** C(fF) [% error]
C1 (N/A)	CMF3X240	CMF12X240	1.02	0 [-100%]	0 [-100%]
C2 (N/A)	CMS12X240_1	CMS3X240_1	1.02	0 [-100%]	0 [-100%]
C3 (L)	CMF12X240	CMF3X240_1	8.28	9.4 [+13.57%]	6.7 [-19.05%]
C4 (L)	CMF3X240	CMF6X240	18.50	18.9 [+2.18%]	13.5 [-27.02%]
C5 (L)	CMF3X240_1	CMF6X240_1	7.54	9.4 [+24.73%]	6.7 [-11.10%]
C6 (L)	CMF6X240	CMF12X240	19.44	18.9 [-2.76%]	13.5 [-30.54%]
C7 (L)	CMF6X240_1	CMF12X240_1	7.65	9.4 [+22.91%]	6.7 [-12.39%]
C8 (L)	CMS12X240_1	CMS6X240_1	13.64	12.6 [-7.59%]	8.1 [-40.6%]
C9 (L)	CMS3X240_1	CMS6X240_1	12.77	12.6 [-1.34%]	8.1 [-36.58%]
C10 (A)	CMS12X240_1	CMF12X240_1	20.48	16.1 [-21.4%]	16.1 [-21.4%]
C11 (A)	CMF3X240_1	CMS3X240_1	6.89	4 [-41.91%]	4 [-41.91%]
C12 (A)	CMF6X240_1	CMS6X240_1	10.29	8.1 [-21.3%]	8.1 [-21.3%]
C13 (F)	CMS6X240_1	CMF12X240_1	1.87	1.5 [-19.59%]	0 [-100%]
C14 (F)	CMS3X240_1	CMF6X240_1	1.76	1.1 [-37.62%]	0 [-100%]
C15 (F)	CMF6X240_1	CMS12X240_1	1.84	1.4 [-23.92%]	0 [-100%]
C16 (F)	CMF12X240	CMS3X240_1	5.86	1.3 [-77.82%]	0 [-100%]
C17 (F)	CMF12X240	CMS6X240_1	0.66	0.2 [-69.85%]	0 [-100%]
C18 (F)	CMF3X240_1	CMS6X240_1	1.76	1 [-43.31%]	0 [-100%]
C19 (A+F)	CMF12X240	GND	28.41	27.6 [-2.84%]	21.5 [-24.31%]
C20 (A+F)	CMF12X240_1	GND	33.17	38.1 [14.88%]	21.5 [-35.18%]
C21 (A+F)	CMF3X240	GND	19.33	17.5 [-9.49%]	5.4 [-72.07%]
C22 (A+F)	CMF3X240_1	GND	13.28	14.2 [6.90%]	5.4 [-59.35%]
C23 (A+F)	CMF6X240	GND	17.98	12.7 [-29.35%]	10.8 [-39.92%]
C24 (A+F)	CMF6X240_1	GND	18.22	19.9 [9.23%]	10.8 [-40.72%]
C25 (A+F)	CMS12X240_1	GND	10.26	8.9 [-13.30%]	0 [-100%]
C26 (A+F)	CMS3X240_1	GND	4.24	2.4 [-43.44%]	0 [-100%]
C27 (A+F)	CMS6X240_1	GND	2.65	2.5 [-5.83%]	0 [-100%]
		Total	**288.81**	**269.7 [-6.62%]**	**166.9 [-42.21%]**

*A=area, L=lateral, F=fringe, N/A=not available

The capacitance values are plotted in Figure 5.3 with different Magic capacitance sections on lateral, area, fringe, and area+fringe. For each point on the plot, CUP values have a

closer point to Space 3D than Magic except for area and those 'not available' capacitances as they are one of limitations discussed in Chapter 6.

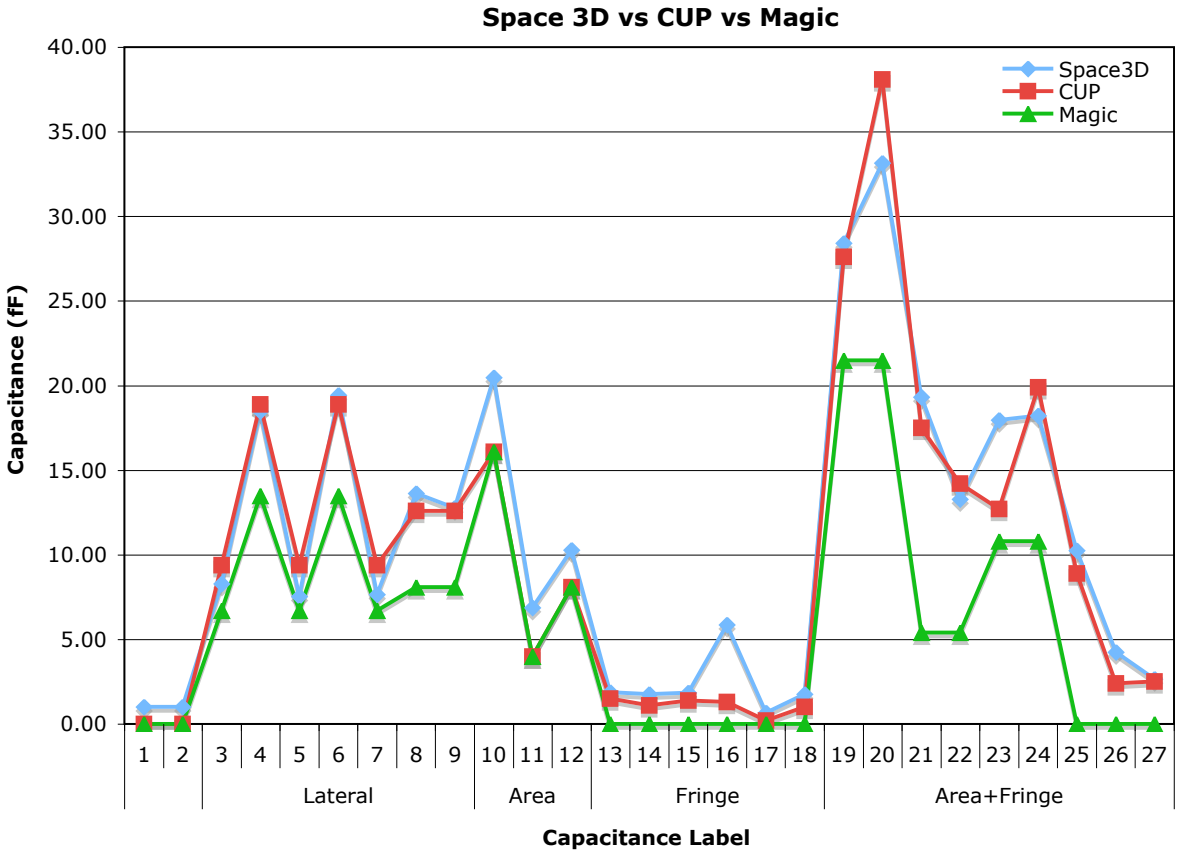

FIGURE 5.3: Capacitance comparison between Space 3D, CUP, and Magic.

It is observed that lateral capacitance has the most accurate results in CUP models. In area capacitance, CUP and Magic have the same values since they use the same coefficients and computation method. The values are all slightly less than Space 3D because Magic does not evaluate the vertical edge capacitance, or co-facial capacitance, between the overlapping areas. Fringe capacitances extracted in CUP models are smaller than Space 3D models. The reason for this is that, in Techgen, the fringe capacitance is computed by subtracting the area capacitance from the field-solvers total capacitance for the primary conductor. However,

the overestimate of area capacitance leads to an underestimate of fringe capacitance. Also, Magic is incapable of evaluating fringe effects between the horizontal surfaces of conductors, which leads to the underestimation. However, the fringe capacitance only accounts for a small portion of overall capacitance and does not affect the overall capacitance much. In addition, it has been improved by a lot from Magic models, which does not compute the fringe capacitance in this case.

Magic models have been replaced and improved by CUP models, and the accuracy of the total capacitance value has been improved from 42.21 % to 6.62 % error in this example layout as compared to Space 3D, which is taken as the true capacitance.

5.2 Linewidth Process Variation Evaluation

To validate the estimation of interconnect capacitance sensitivity to linewidth variation, a simple victim-aggressor crosstalk configuration is presented. The focus of this section is on the accuracy and effect of process variations sensitivity, and an application to the crosstalk problem is implemented in Magic and simulated in ngspice.

5.2.1 Crosstalk

As a signal propagates in a conductor, it may encounter corners, vias, tees, and connectors as well as changes in coupling to the environment. These instantaneous changes in impedance seen by the signal cause undesirable signal effects in another circuit. In addition, neighboring conductors may induce unwanted signals on signal-carrying conductors. Such phenomena are generally referred to as *crosstalk* [77].

With the shrinking technology, higher aspect ratio and reduced wiring pitches have made it progressively more difficult to analyze crosstalk of coupled interconnect considering process variations. Thus, crosstalk has been recognized as the most critical signal integrity (SI) concern, and likely predetermined to play a more important role in future scaled technologies [78, 79].

A crosstalk configuration has two signals: aggressor and victim (see Figure 5.4).

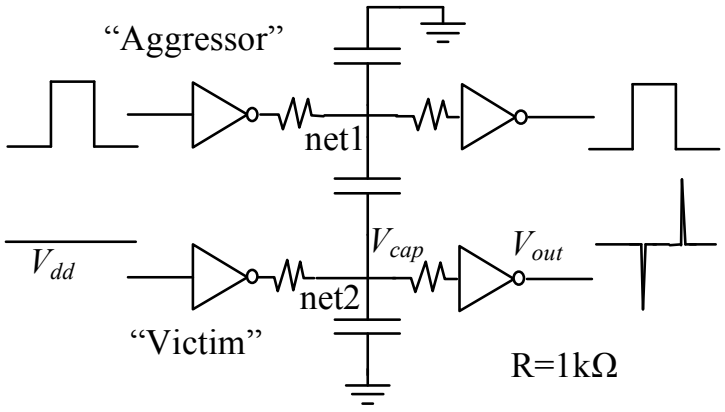

FIGURE 5.4: Crosstalk configuration.

5.2.1.1 Interconnect

A layout with two serpentine, highly-coupled nets was created along with two straight coupling metal lines aside in Magic, and it is replicated with a 1 λ "bloat" on metal 1 (nets A and C), metal 2 (net B), and metal 3 (net D) to verify the accuracy of the linewidth variation sensitivity extraction (see Figure 5.5).

Table 5.2 shows the comparison between the extracted C (e.g. 46.88 fF for nets A-gnd) and the estimated C' (e.g. 27.06 fF for nets B-gnd) capacitances, using the extracted gradients. Linewidth variation gradients ∇ are formatted as $([t_1 > 0, t_1 < 0], [t_2 > 0, t_2 < 0])$

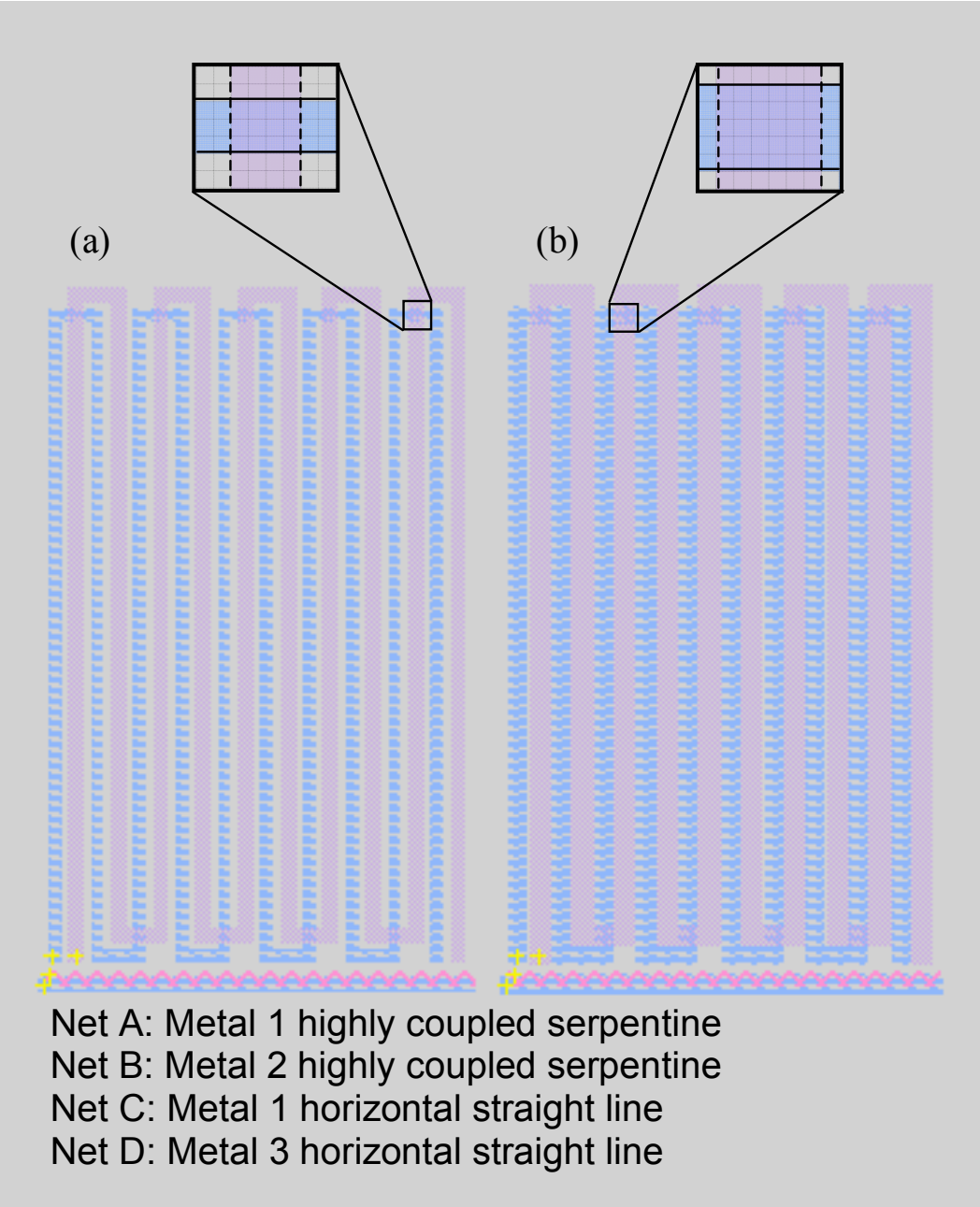

FIGURE 5.5: Magic interconnect layouts with nets A, C on metal 1, B on metal 2, and D on metal 3. (a) Original, nominal linewidth layout. (b) Layout with $\vec{t}$=(1,1) bloat relative to nominal layout.

for both nominal and bloated layouts. C extracted from bloated layout are to be compared to estimated $C'(varied) = C(nominal) + \nabla \cdot \vec{t}$, with $\vec{t} = (a, b)$ for different process variation $\vec{t}$. In this case, $\vec{t}$ is set to (1,0) (ie. metal 1 bloats, metal 2 is nominal width) or (0,1) for nominal layout. Capacitances extracted from the bloated layout are likewise compared back to nominal capacitances with a shrinking $\vec{t}$.

The gradients involving nets C-D have different "bloating" and "shrinking" derivative values because of the existence of coincident edges between nets C and D. For instance, a bloat of metal 1 will *reduce* the D-gnd capacitance by 0.16 fF/λ because it increases the shielding area, and it will *increase* nets C-D (area) and A-C (lateral) as the overlap area enlarges and the separation distance shortens. However, a shrink in metal 1 will *increase* D-gnd capacitance by 0.162 fF/λ and *reduce* C-D and A-C capacitances by 0.30 fF/λ and 1.09 fF/λ respectively. On the other hand, nets A-gnd and C-gnd capacitances are not shielded by any conductor and therefore lack coincident edges, hence the bloat and shrink gradients are equal.

The differences between extracted and estimated capacitance values are evident in four cases: area and lateral (Figure 5.6), substrate with fringing effect (Figure 5.7), fringe (Figure 5.8), and area with fringe (Figure 5.9). Extracted and estimated capacitances are presented in patterned and solid columns, respectively. The odd-number columns for each pair of nets demonstrates the nominal-width capacitance comparison for capacitance extracted from nominal layout and estimated from bloated layout, and the capacitance comparison for extracted bloated layout capacitances are compared with estimated bloated-linwidth capacitances estimated from nominal layout in even-number columns. An error % is computed

TABLE 5.2: Extracted vs estimated capacitances.

Nets	**Nominal Layout**				**Bloated Layout**			
(C)	$C(fF)$	$\nabla C(fF/\lambda)$	$\vec{t}$	$C'(fF)$	$C(fF)$	$\nabla C(fF/\lambda)$	$\vec{t}$	$C'(fF)$
	Space3D				Space3D			
A-gnd	46.88	{[31.30, 31.30],	1	78.18	78.21	{[31.36, 31.36],	-1	46.85
(A)	n/a	[0, 0]}	0		n/a	[0, 0]}	0	
A-gnd	185	{[41.2, 41.2],	1	226.2	195	{[44.0, 44.0],	-1	151
(A+F)	140	[0, 0]}	0		159	[0, 0]}	0	
B-gnd	18.03	{[-0.17, -0.17],	0	27.06	27.06	{[-0.23, -0.23],	0	18.01
(A)	n/a	[9.03, 9.03]}	1		n/a	[9.05, 9.05]}	-1	
B-gnd	22.9	{[-0.16, -0.16],	0	31.9	28.9	{[-0.23, -0.23],	0	19.8
(A+F)	38.5	[9.03, 9.03]}	1		39.3	[9.05, 9.05]}	-1	
C-gnd	3.98	{[2.05, 2.05],	1	6.03	6.06	{[2.11, 2.11],	-1	3.95
(A)	n/a	[0, 0]}	0		n/a	[0, 0]}	0	
C-gnd	12.4	{[2.81, 2.81],	1	15.2	13.4	{[2.94, 2.94],	-1	10.5
(A+F)	12.3	[0, 0]}	0		14.0	[0, 0]}	0	
D-gnd	0.321	{[-0.16, -0.162],	0	0.486	0.497	{[-0.165, -0.165],	0	0.318
(A)	n/a	[0.165, 0.167]}	1		n/a	[0.174, 0.174]}	-1	
D-gnd	1.07	{[-0.16, -0.162],	0	1.234	1.24	{[-0.165, -0.165],	0	1.07
(A+F)	4.96	[0.165, 0.167]}	1		5.39	[0.174, 0.174]}	-1	
A-C	3.34	{[3.27, 1.09],	1	6.61	6.45	{[9.40, 3.13],	-1	3.32
(L)	4.65	[0, 0]}	0		8.26	[0.026, 0.008]}	0	
A-D	0.303	{[0.075, 0.075],	0	0.369	0.386	{[0.096, 0.096],	0	0.312
(F)	0.675	[0.066, 0.066]}	1		0.741	[0.074, 0.074]}	-1	
B-C	0.124	{[0.092, 0.092],	0	0.195	0.213	{[0.145, 0.145],	0	0.105
(F)	1.34	[0.071, 0.071]}	1		1.37	[0.108, 0.108]}	-1	
B-D	0.257	{[0.153, 0.153],	1	0.410	0.440	{[0.233, 0.233],	-1	0.207
(F)	3.94	[0.152, 0.152]}	0		4.55	[0.236, 0.236]}	0	
C-D	0.819	{[0.296, 0.300],	1	1.115	1.018	{[0.277, 0.277],	-1	0.841
(**A**+F)	1.61	[0.295, 0.291]}	0		1.78	[0.304, 0.304]}	0	
A-B	38.32	{[21.24, 21.24],	0	55.62	55.51	{[28.00, 28.00]	0	34.69
(A+**F**)	94.8	[17.30, 17.30]}	1		101	[20.82, 20.82]}	1	
A-B	55.51	{[28.00, 28.00],	1	83.51	79.06	{[34.17, 34.17]	-1	44.89
(A+**F**)	101	[20.82, 20.82]}	0		114	[27.95, 27.95]}	0	

*A=area, L=lateral, F=fringe

on the top of each column.

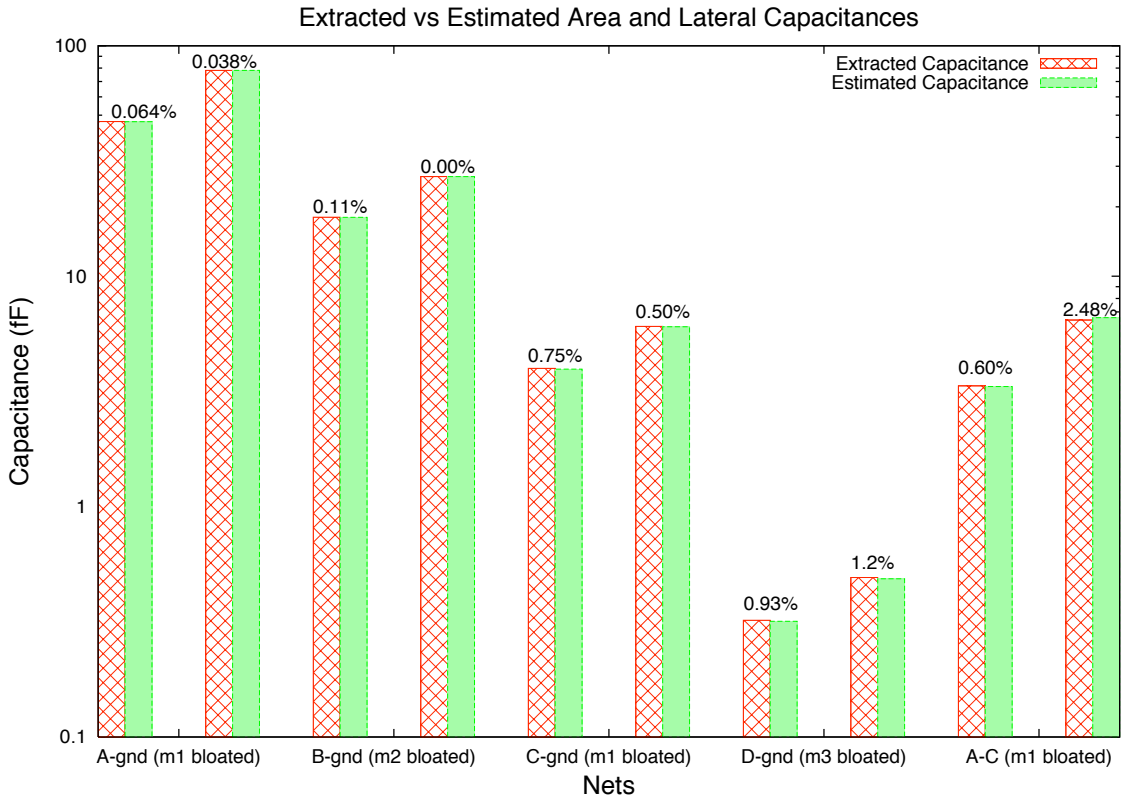

FIGURE 5.6: Extracted vs estimated area and lateral capacitances.

From Figure 5.6, it is observed that the average error for pure area capacitances is very small ($\sim$ 0.45%). In this case, the fringing to ground substrate capacitance is disabled because only pure area capacitance is evaluated. Since there is no shading layer between nets A,B,C,D-gnd, a very accurate result is computed. Lateral capacitance between nets A-C also shows a very accurate result for both *bloating* and *shrinking*.

In another case, the fringing to substrate is enabled. Figure 5.7 shows the average error % for area substrate capacitances with fringing effect. It is shown that the estimated capacitances are not as accurate as pure area capacitance in Figure 5.6, except for D-gnd,

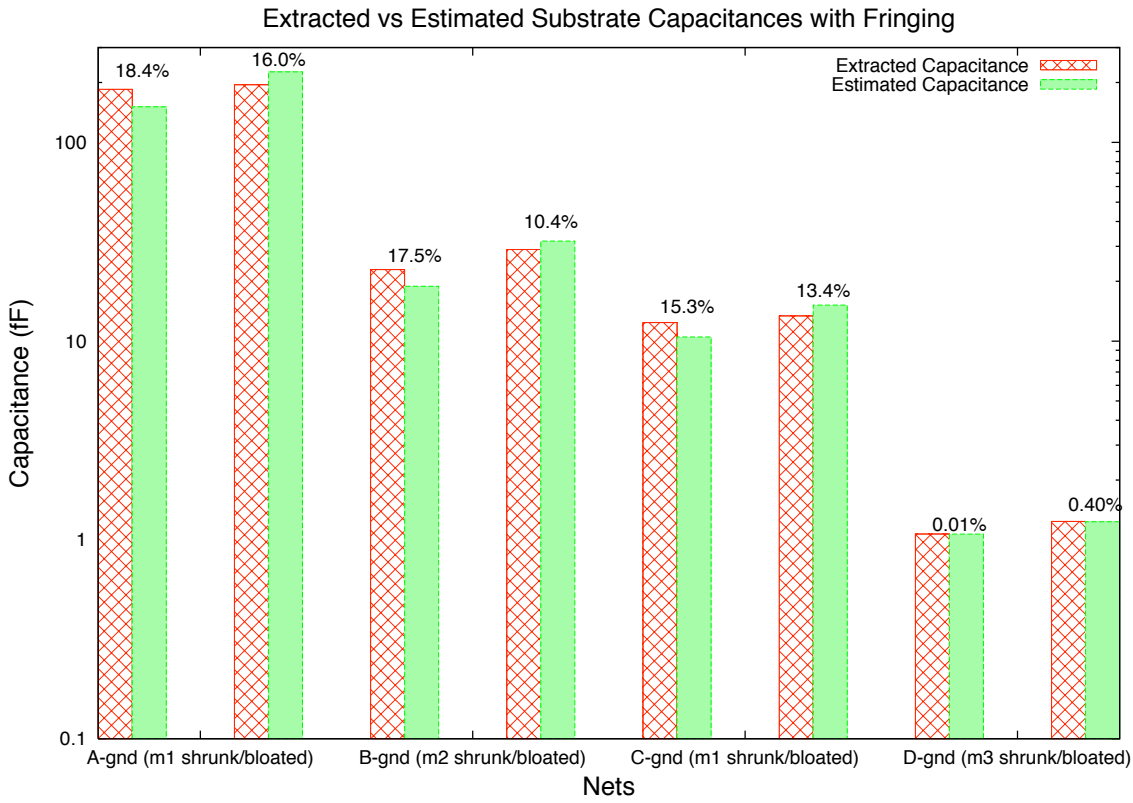

FIGURE 5.7: Extracted vs estimated substrate capacitances with fringing.

in which the value is too small. This implies the derivative of fringing capacitance decreases the overall performance for the substrate capacitance.

Pure fringe capacitance does not reveal a very accurate result (see Figure 5.8), but the average error % is still around 10%. The result varies depending on the layout dimension itself. It is hard to predict the fringe capacitance with a very good result.

For the nets that contain both area and fringe capacitances, results vary like fringe capacitance since fringe capacitance dominates area capacitance in these examples. However, the average error % is less than pure fringe capacitance's ($\sim 7.8\%$). It gives a slightly better result because the area capacitance plays an insignificant role between these nets to balance the error %.

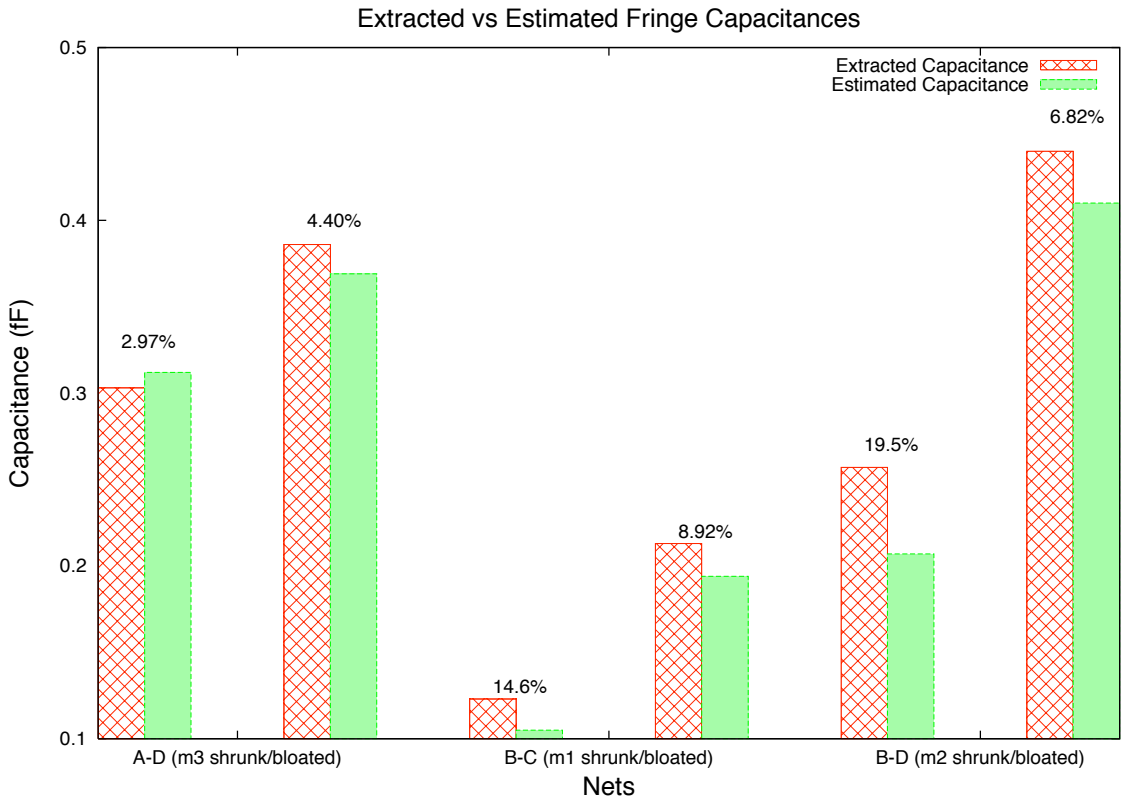

FIGURE 5.8: Extracted vs estimated fringe capacitances.

The accuracy of linewidth process variation sensitivity on three coupling capacitances can be listed from highest to lowest: area, lateral, then fringe capacitance.

5.2.2 Crosstalk Simulation Results

In real fabrication, a bloat or shrink of λ is rather large. Since the derivatives for the nonlinear lateral and fringe models are first order, they are only strictly accurate at the nominal layout dimension. The extracted capacitance is only approximated by the models, and the actual capacitances and its variations will behave differently in real circuits.

The graph of signals in Figure 5.10 illustrates crosstalk sensitivity to linewidth variation. The output with nominal linewidth only had a small glitch, but a 1 λ bloat on metal 1 and

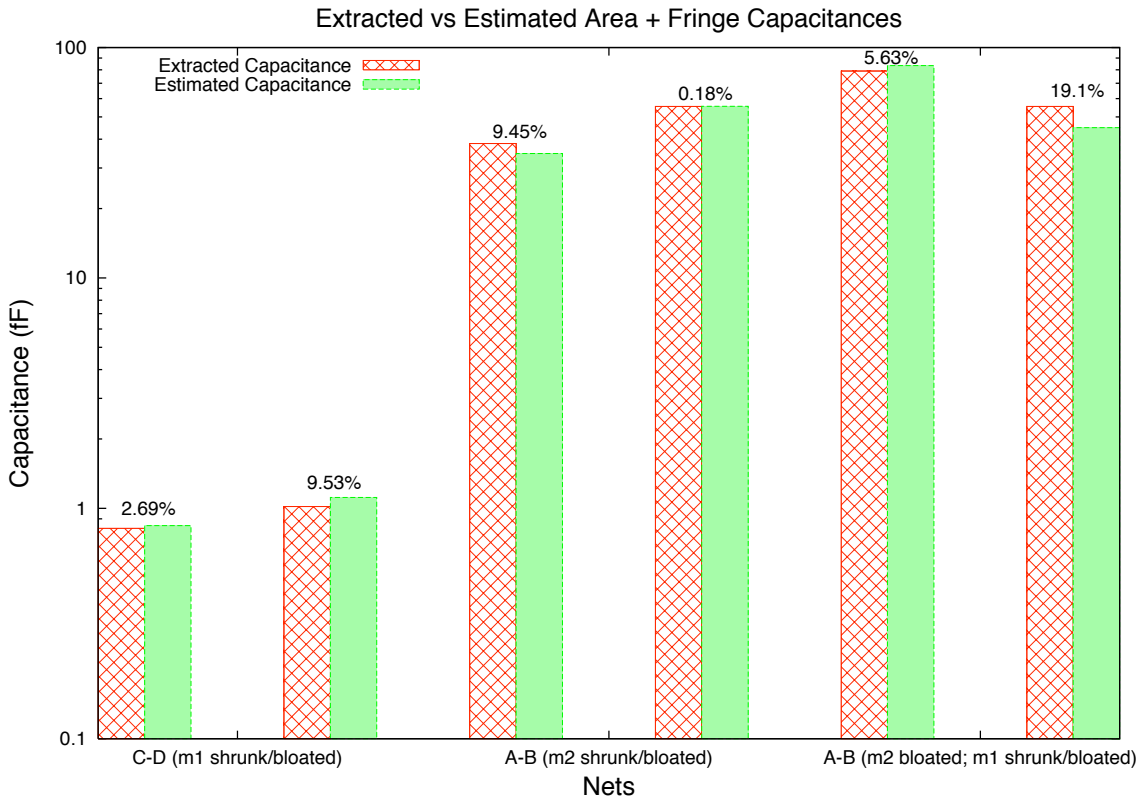

FIGURE 5.9: Extracted vs estimated area + fringe capacitances.

metal 2 caused a glitch that exceeds 0.5Vdd. The sensitivity information is helpful for layout designers to determine the amount of variation necessary to trigger this fault, or calculate the functional yield based on the foundry variability information.

From the simulation, it is noticed that crosstalk becomes a serious difficulty in SI. Failure to meet targeted frequencies in products has happened to reduce product yield due to flawed circuit behavior in crosstalk aggravation. Since process variations have an enormous impact on the amount of crosstalk, circuit designers must optimize designs to account for crosstalk noise [80].

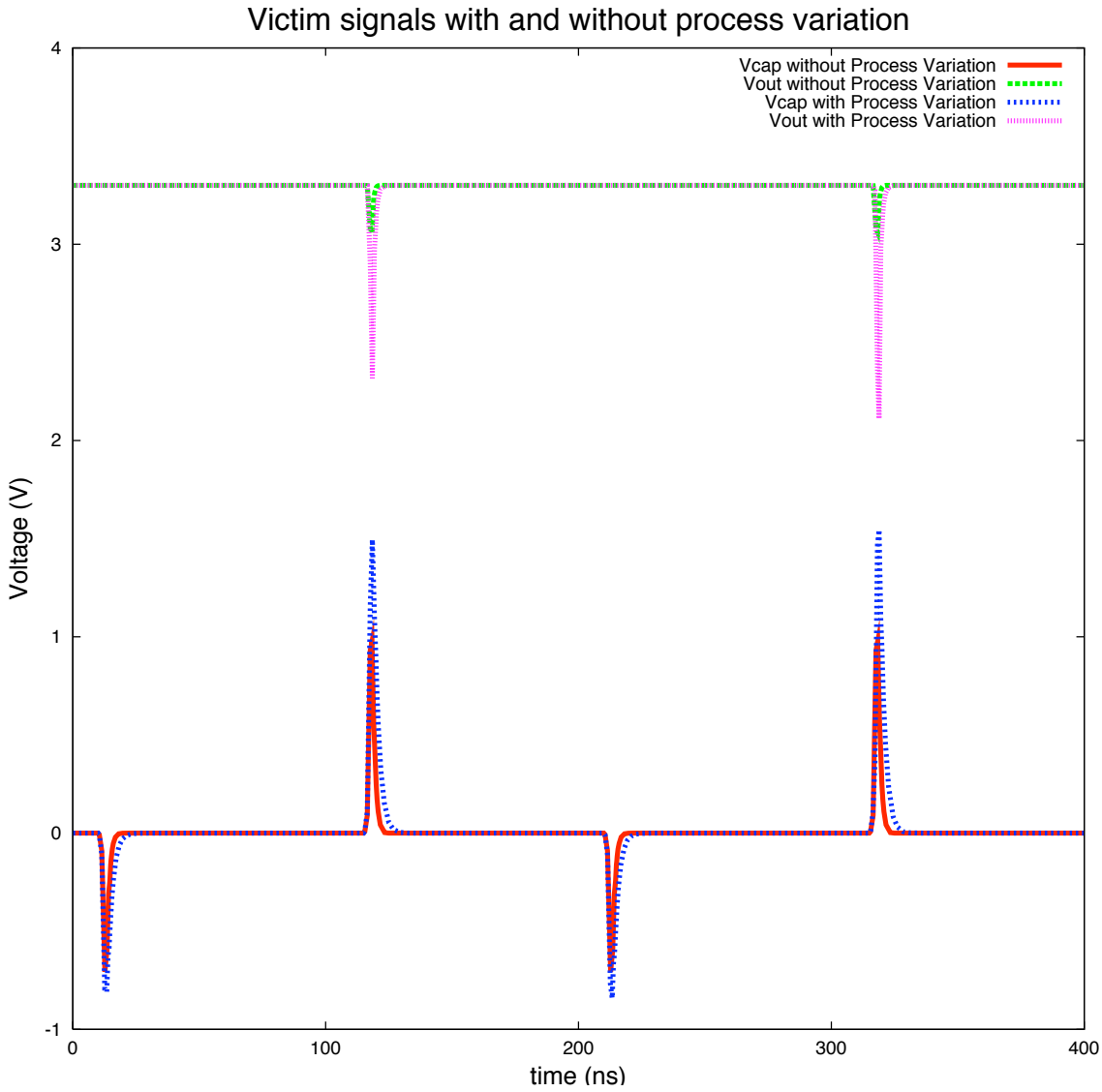

FIGURE 5.10: Victim signals with and without process variation.

5.3 Runtime Analysis

An analysis on runtime is simulated in this section. A Complex Programmable Logic Device (CPLD), product-term based, layout was used for analysis with layout size 2224 x 2671 λ^2 (see Figure 5.11).

The runtime analysis is performed under different cases. Since the standard Magic

capacitance model is already implemented in Magic tool, it would be run for analysis every time. For CUP models, the area capacitance, lateral capacitance, and fringe capacitance are run in two combinations: with and without derivatives. The results are tabulated in Table 5.3 with a check mark indicating which analysis cases were enabled. It is found that fringe capacitance occupies most of the runtime (73%) because it involves the pixel search algorithm on each of four sides in every tile with a certain distance to look for, and it also searches for conductors on the planes above and below, which takes more runtime in the implementation. However, no additional memory allocation is required in the pixel-based search algorithm.

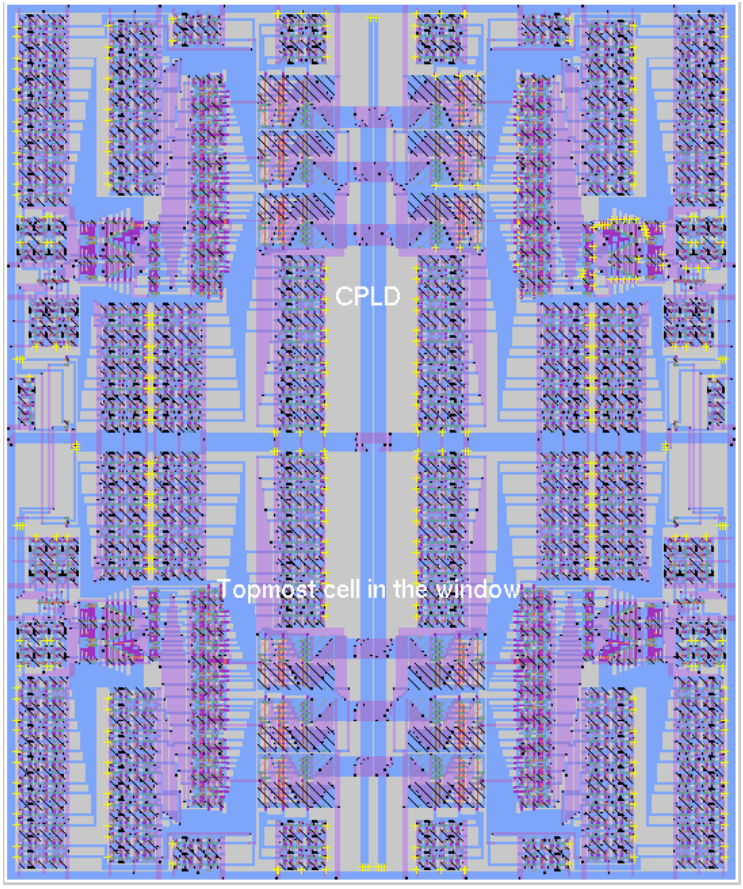

FIGURE 5.11: CPLD layout.

TABLE 5.3: Runtime record.

CPLD (2224 x 2671 λ^2)						
Magic	**CUP**				**Space 3D**	**Runtime** (seconds)
	∇C	area	lateral	fringe		
√						1.85
√	√	√				17.2
√		√				16.9
√	√		√			15.7
√			√			11.1
√	√			√		72.3
√				√		54.2
√	√	√	√	√		100.6
					√	118768
System: SUSE Linux Enterprise Desktop 10 **Processor:** AMD Athlon(tm) 64x2 Dual Core Processor 3800+ **Memory:** 2GB						

The results are plotted in Figure 5.12 to show the percentage occupation of each capacitance model. Among all three capacitance models, lateral capacitance accounts for the least and fringe capacitance accounts for the most runtime.

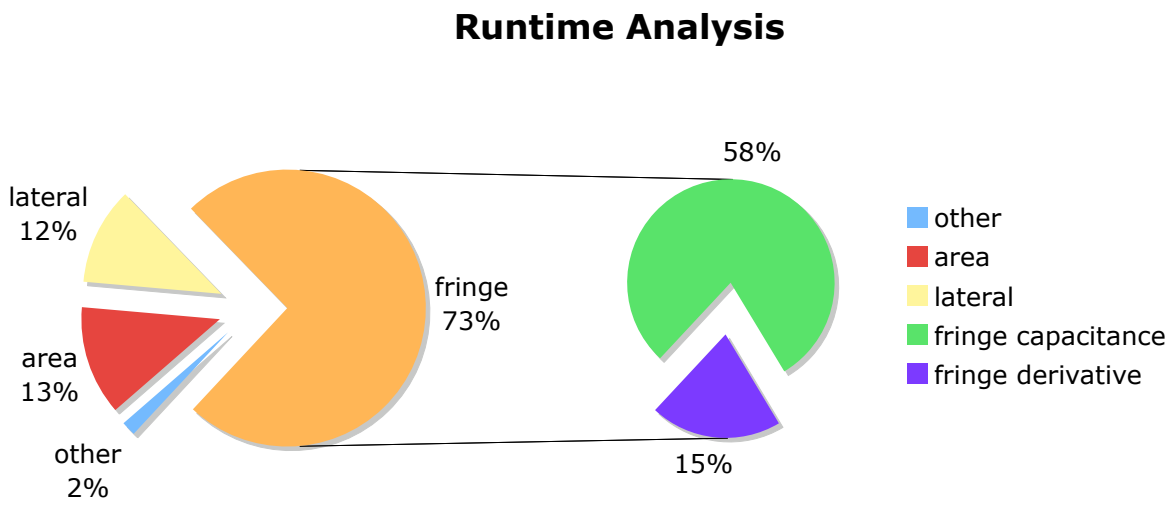

FIGURE 5.12: Runtime analysis.

The runtime analysis was first performed in the layout shown in Figure 5.2, and it took exactly four minutes to extract the circuit in Space 3D while it only took less than a second

for CUP models in Magic. From this example, Magic with CUP models and process variation extracted the small layout shown in Figure 5.2 ∼250X faster than Space 3D in overall.

Space 3D takes much longer time to extract the same circuit. According to Table 5.3, the total runtime for Magic circuit extraction of CPLD with standard models was 1.85 seconds, and with the CUP models including the derivative was 100.6 seconds. The CPLD circuit extraction example was also run in Space 3D, and the runtime of the complete extraction was 118768 seconds. This result concludes that the execution had proceeded for ∼1180X longer than Magic with CUP models. If accurate results can be obtained using CUP models, Magic could become a standard VLSI layout tool.

5.4 Summary

This chapter first compares CUP models with the Space 3D extraction tool from OptEM for capacitance evaluation and then performs linewidth process variation sensitivity in a crosstalk application. Magic has been improved to include CUP models with a certain better performance in terms of capacitance accuracy. Moreover, it is noted that area capacitance linewidth process variation estimation has the best result out of the three types of coupling capacitances while lateral also gives a good result, and crosstalk must be reduced for optimization to minimize the possibility of false circuit behavior in a product. In addition, the runtime analysis is presented to discuss the cost of methods implemented into Magic, in which fringe capacitances occupies the most runtime. In next chapter, conclusion and future work are discussed to summarize this thesis.

Chapter 6

Conclusion and Future Work

6.1 Contributions

In this thesis, Magic was described with its internal structures, and its capacitance extraction was detailed discussed with equations. CUP models were implemented and found to be much more accurate than the standard Magic models. Taking advantage of CUP capacitance models' accuracy, the sensitivity of linewidth process variation on layout was examined, and applied to an example of Magic layout with a crosstalk application. Results are discussed in next section.

6.2 Interpretation of Results

Based on the assumption that Space 3D results are the true solution, Magic has been improved by the addition of CUP models. In the example of Figure 5.2, total capacitance

estimated by the standard Magic models had an error of 42.21 % with respect to the Space 3D field solver results, but the CUP models had an error of 6.62 % in total capacitance.

In linewidth process variation sensitivity, the pure area model provides a very accurate result (within 1% error), and lateral also provides a good result (within 5% error) while fringe capacitance has a various accuracy (∼10-15% error generally). The reason for various fringe and lateral capacitance is the s variable. The derivative formula for separation variable is not accurate in terms of variation estimation in lateral and fringe capacitance equations. On the other hand, the derivative formula for area capacitance is straightforward, thus its linewidth process variation sensitivity has a very accurate result.

There is always a cost for a better performance. In this project, as accuracy is improved and process variation sensitivity is implemented, the cost is the runtime in circuit extraction. In example of Figure 5.11, the runtime of CUP models is ∼55X slower than original Magic model. And it will grow exponentially as the layout is larger. However, accuracy is more important than runtime. If the accuracy could be improved within 5% error in overall, this Magic tool will become a better and trustful VLSI circuit layout tool as its runtime is much faster than commercial field solvers.

6.3 Limitations

Magic only considers 2-D effects while Space 3D is a true 3-D capacitance extractor, thus Magic could never be as accurate as Space 3D.

Table 5.2 shows that there are two sets of capacitances that could not be found in Magic: C1 and C2. These are the capacitance models between two of the same kind of

metal layers with a shielding same kind of metal layer found in Space 3D. In Magic's metal search algorithm, this kind of capacitance could not be found and therefore is computed with a zero capacitance, resulting in a 100 % error. In addition, The area capacitance is always underestimated in both the standard Magic and CUP capacitance models because the vertical edge capacitance is neglected between the overlapping areas in Magic's 2D method. These capacitance models could be included for better performance in future work as discussed below.

6.4 Future Work

Several areas for future work are discovered and described throughout the thesis, and they are discussed below.

6.4.1 Process Variations

The process variation implementation in this thesis is only for linewidth. However, there are more process variation parameters, as discussed in previous chapter, such as height, thickness, and inter-layer dielectric thickness. These variations could be implemented once the information is all given, and it will involve more complicated formulae and data structures.

6.4.2 Capacitance Models

The capacitance models introduced in this thesis include area (overlap in Magic terminology), lateral (sidewall in Magic), and fringe (sidewall overlap in Magic). However, these

techniques may be extended beyond these models to include the capacitance of more configurations of metal surface relevant to DSM interconnect. In [7], parallel, cofacial and cross capacitance models are introduced and thus their sensitivities to process variations could also be applied in Magic.

6.4.3 Accuracy

Lateral and fringe capacitances have been improved dramatically in CUP models over Magic's unsophisticated models. However, the accuracy for fringe capacitance is not as good as lateral model, and it is underestimated. A possible reason is that there might be a systematic underestimation of the magnitude of the fringe capacitance in the Techgen field solver. Although the trends of capacitance with distance are correct, the scale may be wrong.

6.4.4 Runtime

There are several reasons for the increase of runtime in extraction, which includes derivative implementation for all three coupling capacitance models, and the pixel search algorithms. Since the pixel search algorithm is used to evaluate all coupling capacitance models, a solution to reduce the runtime for the pixel search algorithm is proposed here: For each side of edges to perform pixel algorithm, instead of searching pixel by pixel, the required area is searched first along with conductors and real edges. This method will require more complicated algorithms than the pixel algorithm, but it will definitely reduce runtime significantly.

Bibliography

[1] R. H. J. M. Otten, "Global wires: harmful?," *International Symposium on Physical Design*, pp. 104–109, 1998.

[2] M. Walker, "Modeling the Wiring of Deep Submicron ICs," *IEEE Spectrum*, vol. 37, pp. 65–71, March 2000.

[3] J. Cong and D. Pan, "Interconnect Estimation and Planning for Deep Submicron Designs," *Proceedings of 36th Design Automation Conference*, pp. 507–510, 1999.

[4] M. Bachtold, M. Spasojevic, C. Lage, and P. Ljung, "A System for Full-Chip and Critical Net Parasitic Extraction for ULSI Interconnects using a Fast 3-D Field Solver," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, pp. 325–338, March 2000.

[5] International Technology Roadmaps for Semiconductors Interconnect 2007 http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[6] T. El-Moselhy and L. Daniel, "Stochastic Integral Equation Solver for Efficient Variation-Aware Interconnect Extraction," *45th ACM/IEEE Design Automation Conference (DAC), 2008.*, pp. 415–420, June 2008.

[7] A. Labun, "Rapid Method to Account for Process Variation in Full-Chip Capacitance Extraction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, pp. 941–951, June 2004.

[8] Z. J. Lin, C. Spanos, L. Millor, and Y. T. Lin, "Study of Circuit Sensitivity to Interconnect Variation," *2nd International Workshop on Statistical Metrology*, pp. 28–31, June 1997.

[9] N. Lu, "Statistical and Corner Modeling of Interconnect Resistance and Capacitance," *IEEE 2006 Custom Intergrated Circuits Conference (CICC)*, pp. 853–856, September 2006.

[10] T. El-Moselhy, I. Elfadel, and D. Widiger, "Efficient Algorithm for the Computation of On-chip Capacitance Sensitivities with respect to a Large Set of Parameters," *45th ACM/IEEE Design Automation Conference (DAC), 2008.*, pp. 906–911, June 2008.

[11] R. Ananthakrishna and S. Batterywala, "MoM - A Process Variation aware Statistical Capacitance Extractor," *19th International Conference on VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design.*, January 2006.

[12] B. Li, L.-S. Peh, and P. Patra, "Impact of Process and Temperature Variations on Network-on-Chip Design Exploration," *IEEE International Symposium on Networks-on-Chip*, pp. 117–126, April 2008.

[13] K. J. Kuhn, "Reducing Variation in Advanced Logic Technologies: Approaches to Process and Design for Manufacturability of Nanoscale CMOS," *IEEE Internationl Electronic Devices Meeting*, pp. 471–474, December 2007.

[14] F. Huebbers, A. Dasdan, and Y. Ismail, "Computation of accurate interconnect process parameter values for performance corners under process variations," *43rd ACM/IEEE Design Automation Conference*, pp. 797–800, July 2006.

[15] E. Acar, S. Nassif, Y. Liu, and L. Pileggi, "Assessment of True Worst Case Circuit Performance under Interconnect Parameter Variations," *2001 International Symposium on Quality Electronic Design*, pp. 431–436, 2001.

[16] G. Lopez, R. Murali, R. Sarvari, K. Bowman, J. Davis, and J. Meindl, "The Impact of Size Effects and Copper International Process Variations on the Maximum Critical Path Delay of Single and Multi-Core Microprocessors," *IEEE International Interconnect Technology Conference*, pp. 40–42, June 2007.

[17] A. Bansal, B. C. Paul, and K. Roy, "An Analytical Fringe Capacitance Model for Interconnects Using Conformal Mapping," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 2765–2774, December 2006.

[18] A. Mutlu, J. Le, R. Molina, and M. Celik, "Parametric Analysis to Determine Accurate Interconnect Extraction Corners for Design Performance," *10th International Symposium on Quality Electronic Design.*, vol. 0, pp. 419–423, 2009.

[19] W. S. Scott and J. K. Ousterhout, "Magic's Circuit Extractor," *IEEE Design and Test of Computers*, vol. 3, pp. 24–34, February 1986.

[20] H. Kitada, T. Suzuki, T. Kimura, H. Kudo, H. Ochimizu, S. Okano, A. Tsukune, S. Suda, S. Sakai, N. Ohtsuka, T. Tabira, T. Shirasu, M. Sakamoto, A. Matsuura, Y. Asada, and T. Nakamura, "The Influence of the Size Effect of Copper Interconnects on RC Delay Variability beyond 45nm Technology," *IEEE 2007 International Interconnect Technology Conference*, pp. 10–12, June 2007.

[21] N. Lu and J. H. McCullen, "Enablement of Variation-Aware Timing: Treatment of Parasitic Resistance and Capacitance," *8th International Symposium on Quality Electronic Design, ISQED '07.*, pp. 743–748, March 2007.

[22] N. Nagaraj, W. Hunter, P. Chidambaram, T. Garibay, U. Narasimha, A. Hill, and H. Shichijo, "Impact of Interconnect Technology Scaling on SOC Design Methodologies," *Proceedings of the IEEE 2005 International Interconnect Technology Conference, 2005.*, pp. 71–73, June 2005.

[23] V. Venkatraman and W. Burleson, "Impact of Process Variations on Multi-level Signaling for On-Chip Interconnects," *18th International Conference on VLSI Design*, pp. 362–367, January 2005.

[24] V. Hoang, G. Doornbos, J. Michelon, A. Kumar, A. Nackaerts, and P. Christie, "Balancing Resistance and Capacitance of Signal Interconnects for Power Saving," *IEEE 2007 International Interconnect Technology Conference*, pp. 126–128, June 2007.

[25] R. Hoofman, J. Michelon, P. Bancken, R. Daamen, G. Verheijden, V. Arnal, O. Hinsinger, L. Gosset, A. Humbert, W. Besling, C. Goldberg, R. Fox, L. Michaelson, C. Guedj, J. Guillaumond, V. Jousseaume, L. Arnaud, D. Gravesteijn, J. Torres, and G. Passemard, "Reliability Challenges Accompanied with Interconnect Downscaling and Ultra Low-K Dielectrics," *Proceedings of the IEEE 2005 International Interconnect Technology Conference, 2005.*, pp. 85–87, June 2005.

[26] T. Harada, A. Ueki, K. Tomita, K. Hashimoto, J. Shibata, H. Okamura, K. Yoshikawa, T. Iseki, M. Higashi, S. Maejima, K. Nomura, K. Goto, T. Shono, S. Muranaka, N. Torazawa, S. Hirao, M. Matsumoto, T. Sasaki, S. Matsumoto, S. Ogawa, M. Fujisawa, A. Ishii, M. Matsuura, and T. Ueda, "Extremely Low Keff (1.9) Cu Interconnects with Air Gap Formed Using SiOC," *IEEE 2007 International Interconnect Technology Conference*, pp. 141–143, June 2007.

[27] The National Technology Roadmaps for Semiconductors (SIA) Technology Needs 1997 Edition.

[28] K. Banerjee and A. Mehrotra, "Analysis of On-Chip Inductance Effects for Distributed RLC Interconnects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, pp. 904–915, August 2002.

[29] T.-H. Chen, C. Luk, H. Kim, and C. Chung-Ping Chen, "INDUCTWISE: Inductance-Wise Interconnect Simulator and Extractor," *2002 IEEE/ACM International Conference on Computer Aided Design. ICCAD 2002.*, pp. 215–220, November 2002.

[30] C. Luk, T.-H. Chen, and C.-P. Chen, "Frequency-Dependent Reluctance Extraction," *Proceedings of the ASP-DAC 2004. Asia and South Pacific Design Automation Conference, 2004.*, pp. 793–798, January 2004.

[31] K. Gala, V. Zolotov, R. Panda, B. Young, J. Wang, and D. Blaauw, "On-Chip Inductance Modeling and Analysis," *Proceedings of 37th Design Automation Conference, 2000.*, pp. 63–68, 2000.

[32] M. Beattie and L. Pileggi, "Inductance 101: Modeling and Extraction," *Proceedings Design Automation Conference, 2001.*, pp. 323–328, 2001.

[33] "Synopsys - Star-RCXT Datasheet," 2009. Available from http://www.synopsys.com.

[34] Y. Du and W. Dai, "Partial Reluctance Based Circuit Simulation Is Efficient and Stable," *Proceedings of the 2005 conference on Asia South Pacific design automation*, pp. 483–488, 2005.

[35] R. Jiang and C. Chung-Ping Chen, "SCORE: SPICE COmpatible Reluctance Extraction," *Proceedings Design Automation and Test in Europe Conference and Exhibition, 2004.*, vol. 2, pp. 948–953, February 2004.

[36] A. Devgan, H. Ji, and W. Dai, "How to Efficiently Capture On-Chip Inductance Effects: Introducing a New Circuit Element K," *2000 IEEE/ACM International Conference on Computer Aided Design. ICCAD-2000.*, pp. 150–155, 2000.

[37] W. Chang, "Analytical IC Metal-Line Capacitance Formulas," *IEEE Transactions on Microwave Theory and Techniques*, vol. 24, pp. 608–611, September 1976.

[38] C. Yuan and T. Trick, "A Simple Formula for the Estimation of the Capacitance of Two-Dimensional Interconnects in VLSI Circuits," *IEEE Electron Device Letters*, vol. 3, pp. 391–393, December 1982.

[39] T. Sakurai and K. Tamaru, "Simple Formulas for Two- and Three-Dimensional Capacitances," *IEEE Transactions on Electron Devices*, vol. 30, pp. 183–185, February 1983.

[40] N. V. Meijs and J. T. Fokkema, "VLSI Circuit Reconstruction from Mask Topology," *Integration - The VLSI Journal*, vol. 2, no. 2, pp. 85–119, 1984.

[41] E. Barke, "Line-to-Ground Capacitance Calculation for VLSI: A Comparison," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, pp. 295–298, February 1988.

[42] Y. Le Coz and R. Iverson, "A Stochastic Algorithm for High Speed Capacitance Extraction in Integrated Circuits," *Solid-State Electronics*, vol. 35, no. 7, pp. 513–520, 1992.

[43] Y. Le Coz and R. Iverson, "A High-Speed Capacitance Extraction Algorithm for Multi-Level VLSI Interconnects," *Proceedings of Eighth International IEEE VLSI Multilevel Interconnection Conference, 1991.*, pp. 364–366, June 1991.

[44] R. Iverson and Y. L. Coz, "A Floating Random-Walk Method for Efficient RC Extraction of Complex IC-Interconnect Structures," *Technical Proceedings of the 1998 International Conference on Modeling and Simulation of Microsystems*, pp. 117–121, 1998.

[45] "Magma - QuickCap Datasheet," 2006. Available from http://www.magma-da.com.

[46] "Synopsys - Raphael NXT Datasheet," 2009. Available from http://www.synopsys.com.

[47] A. Zemanian, R. Tewarson, C. Ju, and J. Jen, "Three-Dimensional Capacitance Computations for VLSI/ULSI Interconnections," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, pp. 1319–1326, December 1989.

[48] A. Seidl, H. Klose, M. Svoboda, J. Oberndorfer, and W. Rosner, "CAPCAL-A 3-D Capacitance Solver for Support of CAD Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, pp. 549–556, May 1988.

[49] M. Mukai, T. Tatsumi, N. Nakauchi, T. Tobayashi, K. Koyama, Y. Komatsu, R. Bauer, G. Rieger, and S. Selberherr, "The Simulation System for Three-Dimensional Capacitance and Current Density Calculation with a User Friendly GUI," *Simulation of Semiconductor Devices and Processes*, vol. 6, pp. 151–154, September 1995.

[50] A. van Genderen and N. van der Meiis, "Using Articulation Nodes to Improve the Efficiency of Finite-Element Based Resistance Extraction," *33rd Design Automation Conference Proceedings 1996*, pp. 758–763, June 1996.

[51] R. Martins, W. Pyka, R. Sabelka, and S. Selberherr, "High-Precision Interconnect Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 1148–1159, November 1998.

[52] "Ansoft - Maxwell Datasheet," 2009. Available from http://www.ansoft.com.

[53] "Synopsys - Raphael Datasheet," 2006. Available from http://www.synopsys.com.

[54] J. Phillips and J. White, "A Precorrected-FFT Method for Electrostatic Analysis of Complicated 3-D Structures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 1059–1072, October 1997.

[55] Z.-Q. Ning, P. M. Dewilde, and F. L. Neerhoff, "Capacitance Coefficients for VLSI Multilevel Metallization Lines," *IEEE Transactions on Electron Devices*, vol. 34, pp. 644–649, March 1987.

[56] "OptEM - Inspector User Manual, Space 3D," 2008.

[57] K. Nabors and J. White, "FastCap: A Multipole Cccelerated 3-D Capacitance Extraction Program," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, pp. 1447–1459, November 1991.

[58] W. Yu, Z. Wang, and J. Gu, "Fast Capacitance Extraction of Actual 3-D VLSI Interconnects using Quasi-Multiple Medium Accelerated BEM," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, pp. 109–119, January 2003.

[59] N. Shahbazi and H. Sarbazi-Azad, "Accelerating 3-D Capacitance Extraction in Deep Sub-Micron VLSI Design using Vector/Parallel Computing," *International Conference on Parallel and Distributed Systems, 2007*, vol. 2, pp. 1–8, December 2007.

[60] S. Kapur and D. Long, "IES3: A Fast Integral Equation Solver for Efficient 3-Dimensional Extraction," *IEEE/ACM International Conference on Computer-Aided Design, 1997.*, pp. 448–455, November 1997.

[61] W. Shi, J. Liu, N. Kakani, and T. Yu, "A Fast Hierarchical Algorithm for Three-Dimensional Capacitance Extraction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, pp. 330–336, March 2002.

[62] W. Shi, J. Liu, N. Kakani, and T. Yu, "A Fast Hierarchical Algorithm for 3-D Capacitance Extraction," *Proceedings of Design Automation Conference, 1998.*, pp. 212–217, June 1998.

[63] M. Beattie and L. Pileggi, "Error Bounds for Capacitance Extraction via Window Techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 311–321, March 1999.

[64] S. Kapur and D. Long, "Large-Scale Capacitance Calculation," *Proceedings of 37th Design Automation Conference, 2000.*, pp. 744–749, 2000.

[65] W. Yu and Z. Wang, "Enhanced QMM-BEM Solver for Three-Dimensional Multiple-Dielectric Capacitance Extraction within the Finite Domain," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, pp. 560–566, February 2004.

[66] W. Ding and G. Wang, "An Efficient Preconditioning Scheme for Fast Hierarchical Method in 3-D Capacitance Extraction of IC Interconnect," *7th International Conference on ASIC, 2007. ASICON '07.*, pp. 1190–1192, October 2007.

[67] N. D. Arora, K. V. Raol, R. Schumann, and L. M. Richardson, "Modeling and Extraction of Interconnect Capacitances for Multiplayer VLSI Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 58–67, January 1996.

[68] U. Choudhury and A. Sangiovanni-Vincentelli, "Automatic Generation of Analytical Models for Interconnect Capacitances," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, pp. 470–480, April 1995.

[69] S.-Y. Oh, W.-Y. Jung, J.-T. Kong, and K.-H. Lee, "Interconnect Modeling in Deep Submicron Design," *6th International Conference on VLSI and CAD, 1999. ICVC '99.*, pp. 73–80, 1999.

[70] "Cadence QRC Extraction Datasheet," 2008. Available from http://www.cadence.com.

[71] "Mentor - Calibre LFD Datasheet," 2005. Available from http://www.mentor.com.

[72] "Magma - QuickCap NX Datasheet," 2008. Available from http://www.magma-da.com.

[73] C. Wei, R. Barrington, J. Mautz, and T. Sarkar, "Multiconductor Transmission Lines In Multilayered Dielectric Media," *IEEE Transactions on Microwave Theory and Techniques*, vol. 32, pp. 439–450, April 1984.

[74] T. Edwards, "Magic VLSI Layout Tool," 2008. Available from http://www.opencircuitdesign.com/magic/.

[75] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Talor, "The Magic VLSI Layout System," *IEEE Design and Test of Computers*, vol. 2, pp. 19–30, January 1985.

[76] J. K. Ousterhout, "Corner Stitching: A Data Structuring Technique for VLSI Layout Tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 3, pp. 87–100, January 1984.

[77] A. Kuo, A. Labun, N. Swart, and A. Ivanov, "Crosstalk Timing Model for High-Speed Interconnects with Impedance Discontinuity," *IEEE Workshop on Signal Propagation on Interconnects, 2007. SPI 2007.*, pp. 194–197, May 2007.

[78] M. Martina and G. Masera, "A Statistical Model for Estimating the Effect of Process Variations on Crosstalk Noise," *Proceedings of the 2004 internaional workshop on System level interconnect prediction*, pp. 115–120, February 2004.

[79] X. Li, J. Wang, W. Tang, and H. Wu, "Stochastic Analysis for Crosstalk Noise of Coupled Interconnects with Process Variations," *IEEE International Conference on Integrated Circuit Design and Technology and Tutorial, 2008. ICICDT 2008.*, pp. 289–292, June 2008.

[80] A. Sinha, S. Nazarian, and T. Mak, "Simulating the Effects of Process Variations on Capacitive Crosstalk," *13th IEEE International Conference on Electronics, Circuits and Systems, 2006. ICECS '06.*, pp. 604–607, December 2006.

Appendices

Appendix A

Stack file

A.1 Create a Stack File

A stack file is created to run in Techgen.

```
-------------------------------------------------------------------------------
.NAME AR=1.5SIO2
!  For Lambda=1um (AR=1.5 Eps=3.8 process)
.CONDUCTOR
!number of conductors
3
!order    layer    min_spacing   max_spacing    width    thickness    max_dist
    1      PMT1          3             8           3         5            50
    2      PMT2          3             8           4         6            30
    3      PMT3          4             9           4         6            50
.DIELECTRIC
!num
7
!order    NAME    layer    thickness    e_x    e_y
1          Di1      1          5        3.8    3.8
2          Di2      1          4.5      3.8    3.8
3          Di3      2          6        3.8    3.8
4          Di4      2          6        3.8    3.8
5          Di5      3          6        3.8    3.8
6          Di6      3          6        3.8    3.8
7          Di7      4          4        3.8    3.8
.PARAMS
!number of parameters
1
Thermal
12   12
12   12
12   12
12   12
12   12
12   12
12   12
-------------------------------------------------------------------------------
```

A.2 Stack Diagram

Figure A.1 shows the metal configuration from the stack file created above.

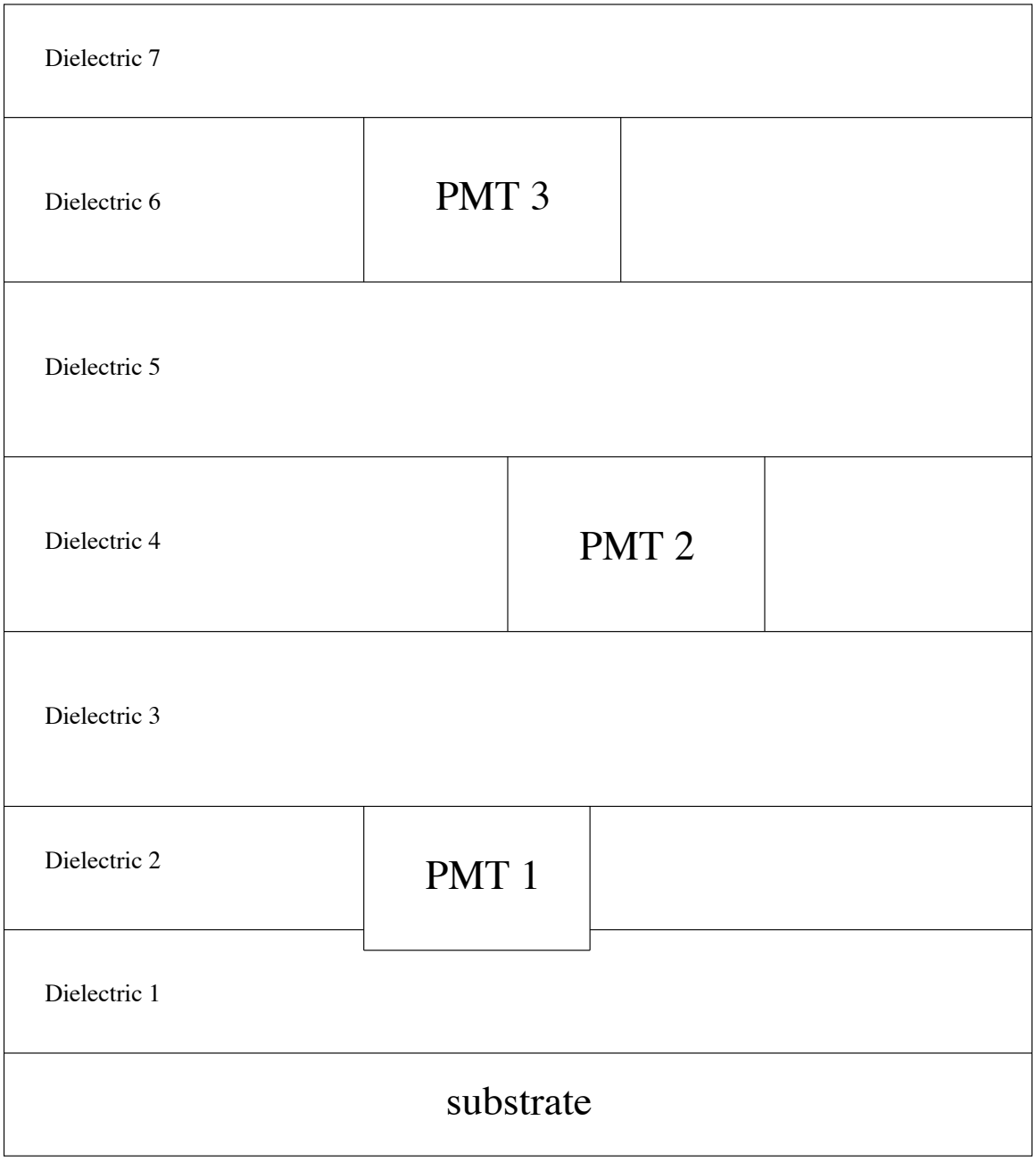

FIGURE A.1: Metal configuration from the stack file.

Appendix B

Magic Tile Search Algorithm

The following figure shows a sample of Magic tile search algorithms and is based on the pseudo code in figure 3.8. Tiles in alphabetical order indicating the tile search transition, and each arrow indicates an action after a certain condition is found.

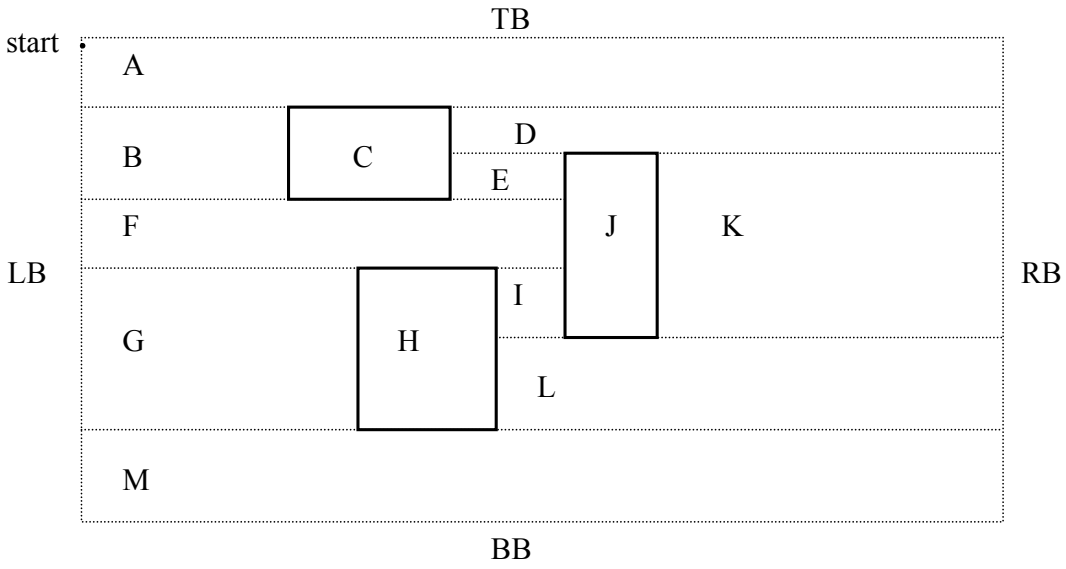

FIGURE B.1: Magic tile search algorithm.

Given a rectangular area, Magic starts to search tiles one by one. This tile search algorithm finishes running completely in five while loops, and each loop is described in details below:

- Loop One: tp = ‘A’, tpnew = ‘RB’ → tpnew = LB(tp) = ‘B’, tp = tpnew = ‘B’.

- Loop Two: tp = ‘B’, tpnew = ‘C’, tp = tpnew = ‘C’ → a metal conductor is found, go to function.

  tpnew = ‘D’, tp = tpnew =‘D’, tpnew = ‘RB’ → tpnew = LB(tp) = ‘E’, tp = tpnew = ‘E’, tpnew = ‘J’ → tpnew = LB(tp) = ‘F’, tp = BL(tp) = ‘C’, tpnew = LB(tp) = ‘F’, tp = BL(tp) = ‘B’, tp = LB(tp) = ‘F’.

- Loop Three: tp = ‘F’, tpnew = ‘J’ → tpnew = LB(tp) = ‘G’, tp = LB(tp) = ‘G’.

- Loop Four: tp = ‘G’, tpnew = ‘H’, tp = tpnew = ‘H’ → a metal conductor is found, go to function.

  tpnew = ‘I’, tp = tpnew = ‘I’, tpnew = ‘J’, tp = tpnew = ‘J’ → a metal conductor is found, go to function.

  tpnew = ‘K’, tp = tpnew = ‘K’, tpnew = ‘RB’ → tpnew = LB(tp) = ‘L’, tp = LB(tp) = ‘J’, tpnew = LB(tp) = ‘L’, tp = BL(tp) = ‘I’, tpnew = LB(tp) = ‘L’, tp = BL(tp) = ‘H’, tp = tpnew = ‘L’, tpnew = TR(tp) = ‘RB’ → tpnew = LB(tp) = ‘M’, tp = BL(tp) = ‘H’, tpnew = LB(tp) = ‘M’, tp = BL(tp) = ‘G’, tp = LB(tp) = ‘M’.

- Loop Five: tp = ‘M’, tpnew = ‘RB’, → tpnew = LB(tp) = ‘BB’, tp = tpnew = ‘BB’.

Appendix C

Area Capacitance Edge Implementation

Figure C.1 shows the real edges in an area capacitance. Cases are shown for metal 3 over metal 2, metal 3 over metal 1, and metal 3 over poly. The '+' and '-' signs are defined to correspond to its own color edges.

A step-by-step description is provided in the case of metal 3 overlapping metal 1.

- Metal 1 is found to be overlapping with metal 3.
- Compute the real edges for the overlapping area between metal 1 and metal 3.
- A shielding metal 2 layer is found in the overlapping area.
- Compute the real edges for the overlapping area between metal 1 and metal 2.
- Add the real edge of the shielding layer (metal 2).
- Subtract the real edge for the bottom layer (metal 1) shielded by metal 2.

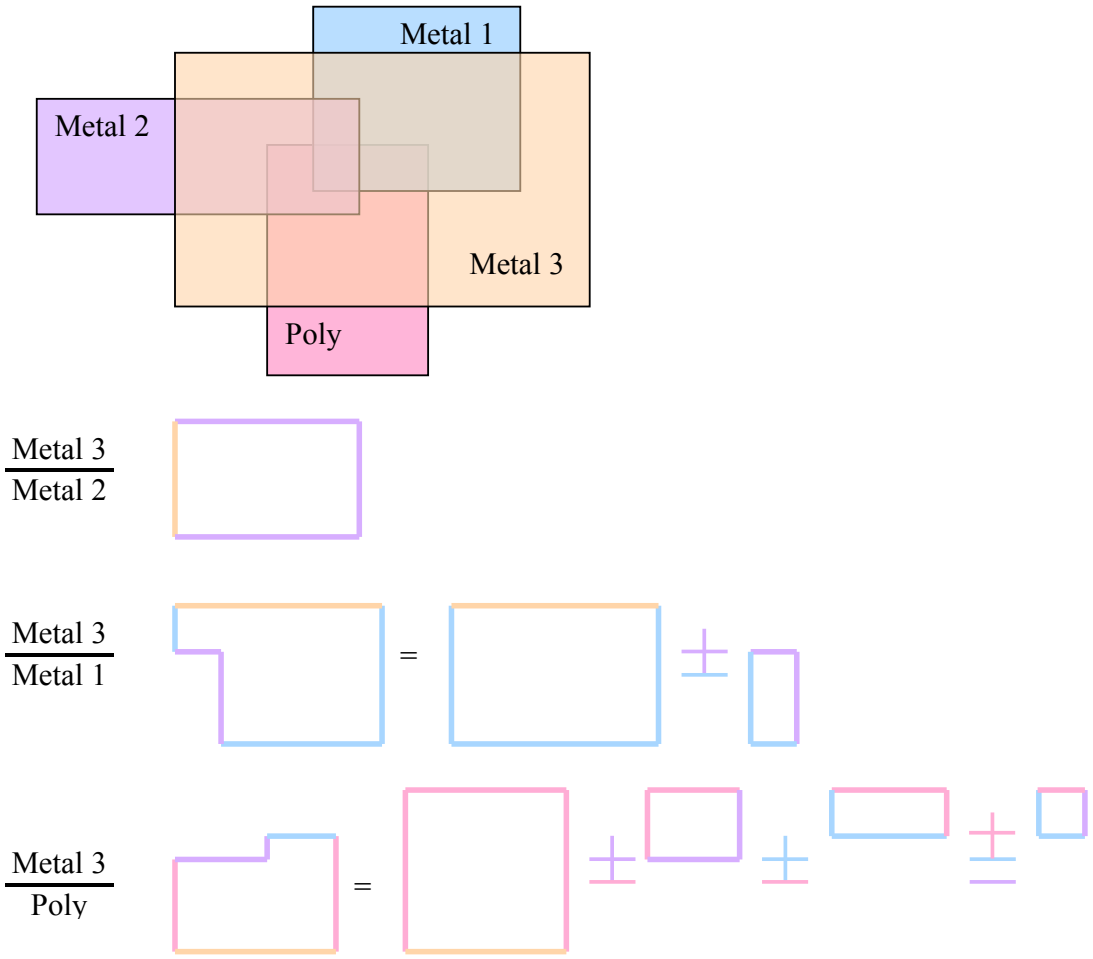

FIGURE C.1: Area capacitance edge computation.

Appendix D

Magic Technology File on Extraction

The following contains the partial extraction section in the Magic technology file used in this thesis.

```
----------------------------------------------------------------------------------------
extract

# Interconnect based on the lambda=0.5 style from the Magic scmos tech file
 style AR=1.5

 CUPcscale 1
 Thermalcscale 1
 ThermalCUPcscale 1
 cscale 1
 lambda 100
 step 100
 sidehalo 20


 areacap (ndiff,nsd,ndc,nsc)/a 0
 perimc (ndiff,nsd,ndc,nsc)/a space,pwell 0

 areacap (pdiff,psd,pdc,psc)/a 0
 perimc (pdiff,psd,pdc,psc)/a space,nwell 0
 areacap (poly,pc)/a 11.21
 areacap cc/a,cap 11.21
 overlap (poly,pc)/a nwell,pwell 33.63
 areacap poly2,ec/a 11.21

 perimc (poly,pc)/a ~(poly,pc)/a 16.815
 sideoverlap (poly,pc)/a ~(poly,pc)/a nwell,pwell 50.44

 CUPareacap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 7.476870
 ThermalCUPareacap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 2.666667
```

```
Thermalareacap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 2.666667
areacap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 7.476870

overlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 nwell,pwell 0.1053
(poly,pc)/a,(ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a
overlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
(ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a 0.1053
overlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (poly,pc)/a 0.2106
overlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 poly2,cap 0.243

sideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 nwell,pwell 2.97
((poly,pc)/a,(ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a)
sideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
(ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a 2.97
sideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (poly,pc)/a 2.97
CUPsideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (m2,m2c,m3c,pad)/m2
-3.507350 30.106000 0.074497 40.455530 0.103507 2.906032 0.113898 14.857140
ThermalCUPsideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (m2,m2c,m3c,pad)/m2
 -1.250622 10.743480 0.081591 37.003110 0.103311 2.904532 0.114126 14.857140
Thermalsideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (m2,m2c,m3c,pad)/m2 0.976629
sideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (m2,m2c,m3c,pad)/m2 13.267799
CUPsideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (m3,m3c,pad)/m3
 -2.106622 12.042360 0.076343 32.146360 0.134950 11.066430 -0.275945 50.000000
ThermalCUPsideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (m3,m3c,pad)/m3
 -0.751709 4.300820 0.096063 25.656080 0.136128 11.058780 -0.274670 50.000000
Thermalsideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (m3,m3c,pad)/m3 1.193190
sideoverlap (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 (m3,m3c,pad)/m3 22.447109
CUPsidewall (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
-14.011380 394.547900 -601.013900 988.395000 -698.710000
ThermalCUPsidewall (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
  (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
  -4.997234 140.717600 -214.354800 352.516300 -249.198600
Thermalsidewall (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
  (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 30.000000
sidewall (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
```

```
 ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
  ~(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
  (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 84.114805

CUPareacap (m2,m2c,m3c,pad)/m2 2.170705
ThermalCUPareacap (m2,m2c,m3c,pad)/m2 0.774193
Thermalareacap (m2,m2c,m3c,pad)/m2 0.774193
areacap (m2,m2c,m3c,pad)/m2 2.170705

overlap (m2,m2c,m3c,pad)/m2 nwell,pwell 0.03005 (poly,pc)/a,
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1,
(ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a
overlap (m2,m2c,m3c,pad)/m2
(ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a 0.03005
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
overlap (m2,m2c,m3c,pad)/m2 (poly,pc)/a 0.035073
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
overlap (m2,m2c,m3c,pad)/m2 poly2 0.0324
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
CUPoverlap (m2,m2c,m3c,pad)/m2
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 5.607653
ThermalCUPoverlap (m2,m2c,m3c,pad)/m2
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 5.607653
Thermaloverlap (m2,m2c,m3c,pad)/m2
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 5.607653
overlap (m2,m2c,m3c,pad)/m2
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 5.607653

sideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 nwell,pwell 1.98
((m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1,(poly,pc)/a,
(ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a)
sideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
 (ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a 1.8
  (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
sideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 (poly,pc)/a 1.98
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
CUPsideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 -6.003871 19.457710 0.069604 43.480850 -0.042962 3.750783 0.023243 30.000000
ThermalCUPsideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 -1.566505 9.652053 0.086266 34.510350 0.042902 3.684667 0.031321 30.000000
Thermalsideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 0.976629
sideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1 13.267799
CUPsideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 (m3,m3c,pad)/m3
-3.376266 30.750130 0.076322 41.179970 0.007847 2.966524 0.084409 30.000000
ThermalCUPsideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 (m3,m3c,pad)/m3
-1.663323 11.153050 0.060601 46.574010 0.219921 2.877683 0.091172 30.000000
Thermalsideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 (m3,m3c,pad)/m3 0.976629
sideoverlap (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 (m3,m3c,pad)/m3 13.267799
CUPsidewall (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
```

```
(m2,m2c,m3c,pad)/m2 -5.301961 427.496300 -652.724900 1189.587000 -928.858800
ThermalCUPsidewall (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
 (m2,m2c,m3c,pad)/m2 -2.280283 153.747500 -235.337400 425.043000 -328.899500
Thermalsidewall (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
 (m2,m2c,m3c,pad)/m2 36.000000
sidewall (m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2 ~(m2,m2c,m3c,pad)/m2
(m2,m2c,m3c,pad)/m2 100.937766

CUPareacap (m3,m3c,pad)/m3 1.223488
ThermalCUPareacap (m3,m3c,pad)/m3 0.436364
Thermalareacap (m3,m3c,pad)/m3 0.436364
areacap (m3,m3c,pad)/m3 1.223488

overlap (m3,m3c,pad)/m3 nwell,pwell 0.014013 (poly,pc)/a,
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1,(m2,m2c,m3c,pad)/m2,
(ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a
overlap (m3,m3c,pad)/m3
 (ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a
 0.014013 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1,(m2,m2c,m3c,pad)/m2
overlap (m3,m3c,pad)/m3 (poly,pc)/a 0.015066
(m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1,(m2,m2c,m3c,pad)/m2
CUPoverlap (m3,m3c,pad)/m3 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
1.869218 (m2,m2c,m3c,pad)/m2
ThermalCUPoverlap (m3,m3c,pad)/m3 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
1.869218 (m2,m2c,m3c,pad)/m2
Thermaloverlap (m3,m3c,pad)/m3 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
1.869218 (m2,m2c,m3c,pad)/m2
overlap (m3,m3c,pad)/m3 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
1.869218 (m2,m2c,m3c,pad)/m2

CUPoverlap (m3,m3c,pad)/m3 (m2,m2c,m3c,pad)/m2 5.607653
ThermalCUPoverlap (m3,m3c,pad)/m3 (m2,m2c,m3c,pad)/m2 5.607653
Thermaloverlap (m3,m3c,pad)/m3 (m2,m2c,m3c,pad)/m2 5.607653
overlap (m3,m3c,pad)/m3 (m2,m2c,m3c,pad)/m2 5.607653

sideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 nwell,pwell 1.53
 ((ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a,(poly,pc)/a,
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1,(m2,m2c,m3c,pad)/m2)
sideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3
 (ndiff,pdiff,em,col,ppd,nnd,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc)/a 1.53
  ((m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1,(m2,m2c,m3c,pad)/m2)
sideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (poly,pc)/a 1.53
 ((m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1,(m2,m2c,m3c,pad)/m2)
CUPsideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 -3.049528 9.633470 0.064389 42.184760 0.402869 13.609610 -0.364724 50.000000
  ((m2,m2c,m3c,pad)/m2)
ThermalCUPsideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 -0.979124 4.835686 0.061433 39.960010 0.442047 16.097220 -0.509733 50.000000
  ((m2,m2c,m3c,pad)/m2)
Thermalsideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
```

```
 1.193190 ((m2,m2c,m3c,pad)/m2)
sideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3
 (m1,ndc,pdc,nwc,pwc,nbdc,capc,ec,clc,emc,pbc,pc,via)/m1
 22.447109 ((m2,m2c,m3c,pad)/m2)
CUPsideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (m2,m2c,m3c,pad)/m2
-4.670860 21.080550 0.074197 58.957620 -0.511804 3.204557 0.089685 50.000000
ThermalCUPsideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (m2,m2c,m3c,pad)/m2
-1.379702 10.856200 0.074391 50.353130 -0.027483 3.639437 0.086175 50.000000
Thermalsideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (m2,m2c,m3c,pad)/m2 0.976629
sideoverlap (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (m2,m2c,m3c,pad)/m2 13.267799

CUPsidewall (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (m3,m3c,pad)/m3
-3.591101 527.719200 -1355.391000 3383.379000 -3528.000000
ThermalCUPsidewall (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (m3,m3c,pad)/m3
 -0.354863 180.661800 -455.155600 1150.623000 -1212.000000
Thermalsidewall (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (m3,m3c,pad)/m3
36.000000
sidewall (m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 ~(m3,m3c,pad)/m3 (m3,m3c,pad)/m3
100.937766

device mosfet pfet pfet pdiff,pdc nwell Vdd! 204 450
device mosfet nfet nfet ndiff,ndc pwell Gnd! 150 450
device mosfet epfet epfet pdiff,pdc nwell Vdd! 204 450
device mosfet enfet enfet ndiff,ndc pwell Gnd! 150 450

device capacitor None cap,capc/a poly,pc 120 745

device bjt npn pbase,pbc/a emit,emc/a nwell

# resistances based on the non-specific 0.18 micron process
# sample6m.tech at http://www.opencircuitdesign.com/magic/index.html

fetresis nfet linear 8000
fetresis pfet linear 8000
fetresis nfet saturation 8000
fetresis pfet saturation 8000

fetresis enfet linear  8000
fetresis epfet linear 8000
fetresis enfet saturation 8000
fetresis epfet saturation 8000

CUPresist (ndiff,nsd,ndc/act,nsc/act) 8000
Thermalresist (ndiff,nsd,ndc/act,nsc/act) 8000
ThermalCUPresist (ndiff,nsd,ndc/act,nsc/act) 8000
resist (ndiff,nsd,ndc/act,nsc/act) 8000
CUPresist (pdiff,psd,pdc/act,psc/act) 8000
Thermalresist (pdiff,psd,pdc/act,psc/act) 8000
ThermalCUPresist (pdiff,psd,pdc/act,psc/act) 8000
resist (pdiff,psd,pdc/act,psc/act) 8000
CUPresist (poly,pc/act,pfet,nfet) 7400
Thermalresist (poly,pc/act,pfet,nfet) 7400
ThermalCUPresist (poly,pc/act,pfet,nfet) 7400
```

```
resist (poly,pc/act,pfet,nfet) 7400
CUPresist poly2,ec/a,enfet,epfet,cap,capc/a,nffet,pffet,hnfet,hpfet 7400
Thermalresist poly2,ec/a,enfet,epfet,cap,capc/a,nffet,pffet,hnfet,hpfet 7400
ThermalCUPresist poly2,ec/a,enfet,epfet,cap,capc/a,nffet,pffet,hnfet,hpfet 7400
resist poly2,ec/a,enfet,epfet,cap,capc/a,nffet,pffet,hnfet,hpfet 7400
CUPresist em,emc/a 7400
Thermalresist em,emc/a 7400
ThermalCUPresist em,emc/a 7400
resist em,emc/a 7400
CUPresist pbase,pbc/a 900000
Thermalresist pbase,pbc/a 900000
ThermalCUPresist pbase,pbc/a 900000
resist pbase,pbc/a 900000
CUPresist (metal1,m2c/metal1) 110
Thermalresist (metal1,m2c/metal1) 110
ThermalCUPresist (metal1,m2c/metal1) 110
resist (metal1,m2c/metal1) 110
CUPresist (metal2,via/m2,pad) 80
Thermalresist (metal2,via/m2,pad) 80
ThermalCUPresist (metal2,via/m2,pad) 80
resist (metal2,via/m2,pad) 80
CUPresist (metal3) 80
Thermalresist (metal3) 80
ThermalCUPresist (metal3) 80
resist (metal3) 80
CUPresist nwell 900000
Thermalresist nwell 900000
ThermalCUPresist nwell 900000
resist nwell 900000

contact pc 4 10000
contact ec/a,capc/a 4 10000
contact ndc,pdc,nsc,psc 4 10000
contact pdc/a,psc/a 4 10000
contact m2c,m3c 4 6800

planeorder implant 0
planeorder well 1
planeorder active 2
planeorder metal1 3
planeorder metal2 4
planeorder metal3 5
planeorder oxide 6

end
```

Appendix E

Magic Output Files

The following contains the output files (‘ext’ and ‘spice’) extracted from the Magic VLSI layout tool in layout Figure 5.5 (a). Gradients information is shown in the end of every capacitance node and is shown in the next line in this Appendix.

E.1 Ext File

The following is the ‘ext’ file extracted directly from a layout in the Magic VLSI layout tool.

```
--------------------------------------------------------------------------------
timestamp 1248734708
version 7.5
tech scmos_m3_L1.0AR1.5CUP
style AR=1.5
scale 1000 1 100
resistclasses 8000 8000 7400 7400 7400 900000 110 80 80 900000
#Node contains substrate gradient cap info in the end
node "D" 3 1070.75 -1 -7 m3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 524 270 0 0
0 0 0 0 0 0 0 0 -160 -162 0 0 0 0 0 0 167 165 0 0
node "B" 42 22944.5 6 0 m2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8412 4214 0 0 0 0
0 0 0 0 0 0 0 0 -156 -156 0 0 9030 9030 0 0 0 0 0 0
node "C" 4 12405.2 -3 -9 m1 0 0 0 0 0 0 0 0 0 0 0 0 0 532 274 0 0 0 0 0 0
0 0 0 0 0 0 0 0 2810 2810 0 0 0 0 0 0 0 0 0 0
node "A" 77 184628 0 0 m1 0 0 0 0 0 0 0 0 0 0 0 0 0 6270 4186 0 0 0 0 0 0
0 0 0 0 0 0 0 0 41184 41184 0 0 0 0 0 0 0 0 0 0
cap "A" "D" 302.897
0 0 0 0 0 0 0 0 75.1041 75.1041 0 0 0 0 0 0 66.0343 66.0343 0 0
cap "D" "B" 256.933
0 0 0 0 0 0 0 0 0 0 0 0 152.97 152.97 0 0 152.055 152.055 0 0
cap "C" "A" 3335.28
0 0 0 0 0 0 0 0 3274.03 1091.34 0 0 0 0 0 0 0 0 0 0
cap "C" "B" 123.927
0 0 0 0 0 0 0 0 92.2553 92.2553 0 0 71.3081 71.3081 0 0 0 0 0 0
cap "A" "B" 38317
```

```
0 0 0 0 0 0 0 0 21248.6 21248.6 0 0 17301.4 17301.4 0 0 0 0 0 0
cap "C" "D" 818.553
0 0 0 0 0 0 0 0 296.284 300.022 0 0 0 0 0 0 295.03 291.292 0 0
-------------------------------------------------------------------------------
```

E.2 Spice File

The following is the 'spice' file converted from ext file in Magic.

```
-------------------------------------------------------------------------------
* SPICE3 file created from xtalk.ext - technology: scmos_m3_L1.0AR1.5CUP

.option scale=1u

C0 A D 0.3fF * 0 0 0 0 0 0 0 0 0.075104 0.075104 0 0 0 0 0 0 0.066034 0.066034 0 0
C1 C B 0.1fF * 0 0 0 0 0 0 0 0 0.092255 0.092255 0 0 0.071308 0.071308 0 0 0 0 0 0
C2 B D 0.3fF * 0 0 0 0 0 0 0 0 0 0 0 0 0.15297 0.15297 0 0 0.152055 0.152055 0 0
C3 A B 38.3fF * 0 0 0 0 0 0 0 0 21.2486 21.2486 0 0 17.3014 17.3014 0 0 0 0 0 0
C4 C D 0.8fF * 0 0 0 0 0 0 0 0 0.296284 0.300022 0 0 0 0 0 0 0.29503 0.291292 0 0
C5 A C 3.3fF * 0 0 0 0 0 0 0 0 3.27403 1.09134 0 0 0 0 0 0 0 0 0 0
C6 D GND 1.1fF **FLOATING 0 0 0 0 0 0 0 0 -0.16 -0.162 0 0 0 0 0 0 0.167 0.165 0 0
C7 B GND 22.9fF **FLOATING 0 0 0 0 0 0 0 0 -0.156 -0.156 0 0 9.03 9.03 0 0 0 0 0 0
C8 C GND 12.4fF **FLOATING 0 0 0 0 0 0 0 0 2.81 2.81 0 0 0 0 0 0 0 0 0 0
C9 A GND 184.6fF **FLOATING 0 0 0 0 0 0 0 0 41.184 41.184 0 0 0 0 0 0 0 0 0 0
-------------------------------------------------------------------------------
```

E.3 Final Spice File

The following is the final 'spice' file after running the shell script.

```
-------------------------------------------------------------------------------
*Process Variation t-vector: pw=0.0 nw=0.0 py=0.0 m1=1.0 m2=0.0 m3=0.0
* SPICE3 file created from xtalk.ext - technology: scmos_m3_L1.0AR1.5CUP

.option scale=1u

C0 A D 0.38fF
C1 C B 0.19fF
C2 B D 0.30fF
C3 A B 59.55fF
C4 C D 1.10fF
C5 A C 6.57fF
C6 D GND 0.94fF
C7 B GND 22.74fF
C8 C GND 15.21fF
C9 A GND 225.78fF
-------------------------------------------------------------------------------
```

Appendix F

Model Fit to Field Solver

The following figures show the curve fits to field solver results for lateral (Figure F.1 to Figure F.3) and fringe capacitances (Figure F.4 to Figure F.11).

In the field solver, the first thing to deal with fringe capacitance is to subtract the parallel plate value for the area capacitance. However, since the parallel plate formula assumes uniform electrical flux concentration for the entire plate, it overestimates the true area capacitance, which is reduced near the edges. The negative fringe flux for very small fringe areas is thus made to be negative to compensate for the overestimation of the area capacitance. It is an artificial and computational effect (see Figure F.4 to Figure F.11 at small λ).

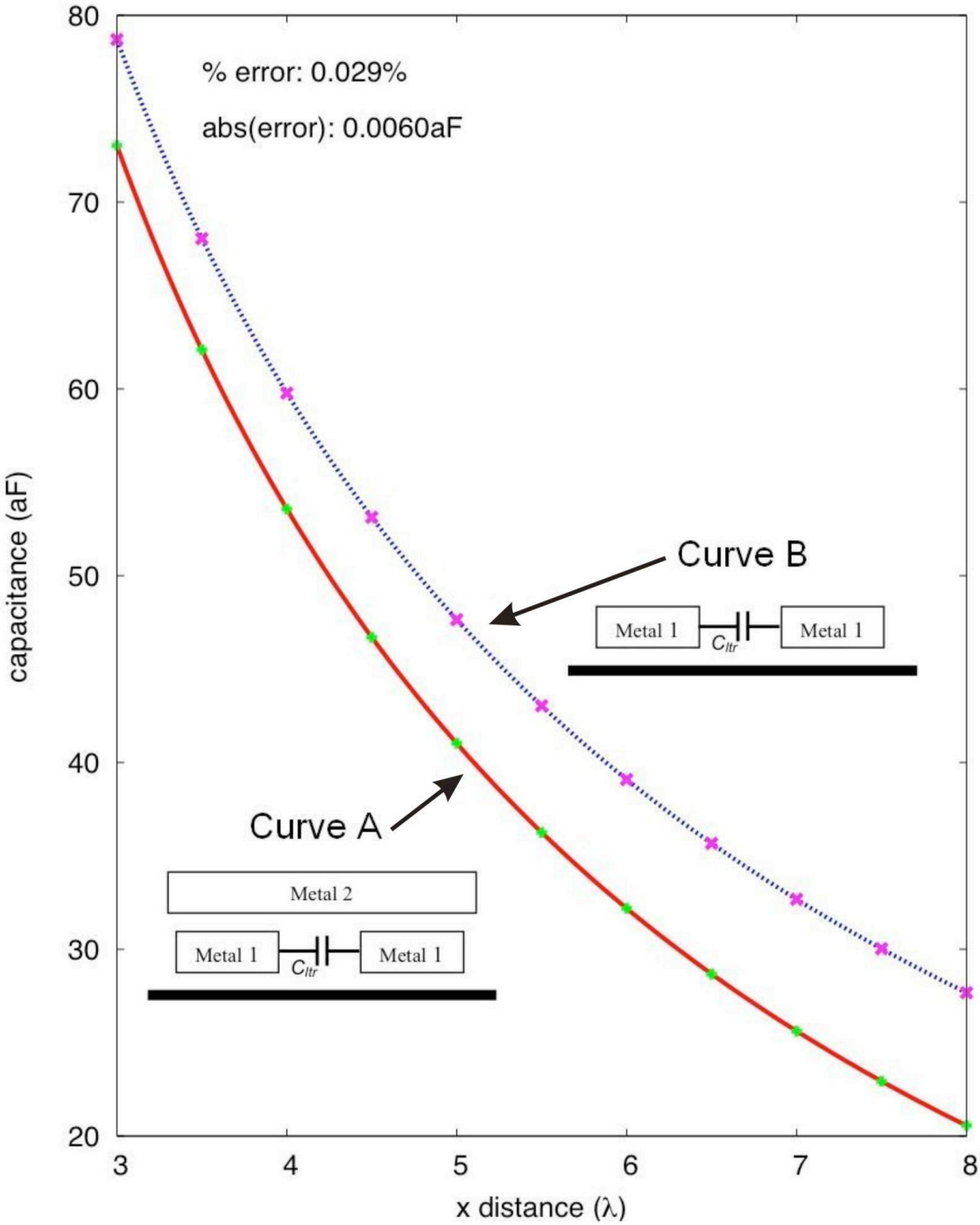

FIGURE F.1: Lateral capacitance on metal 1 per λ.

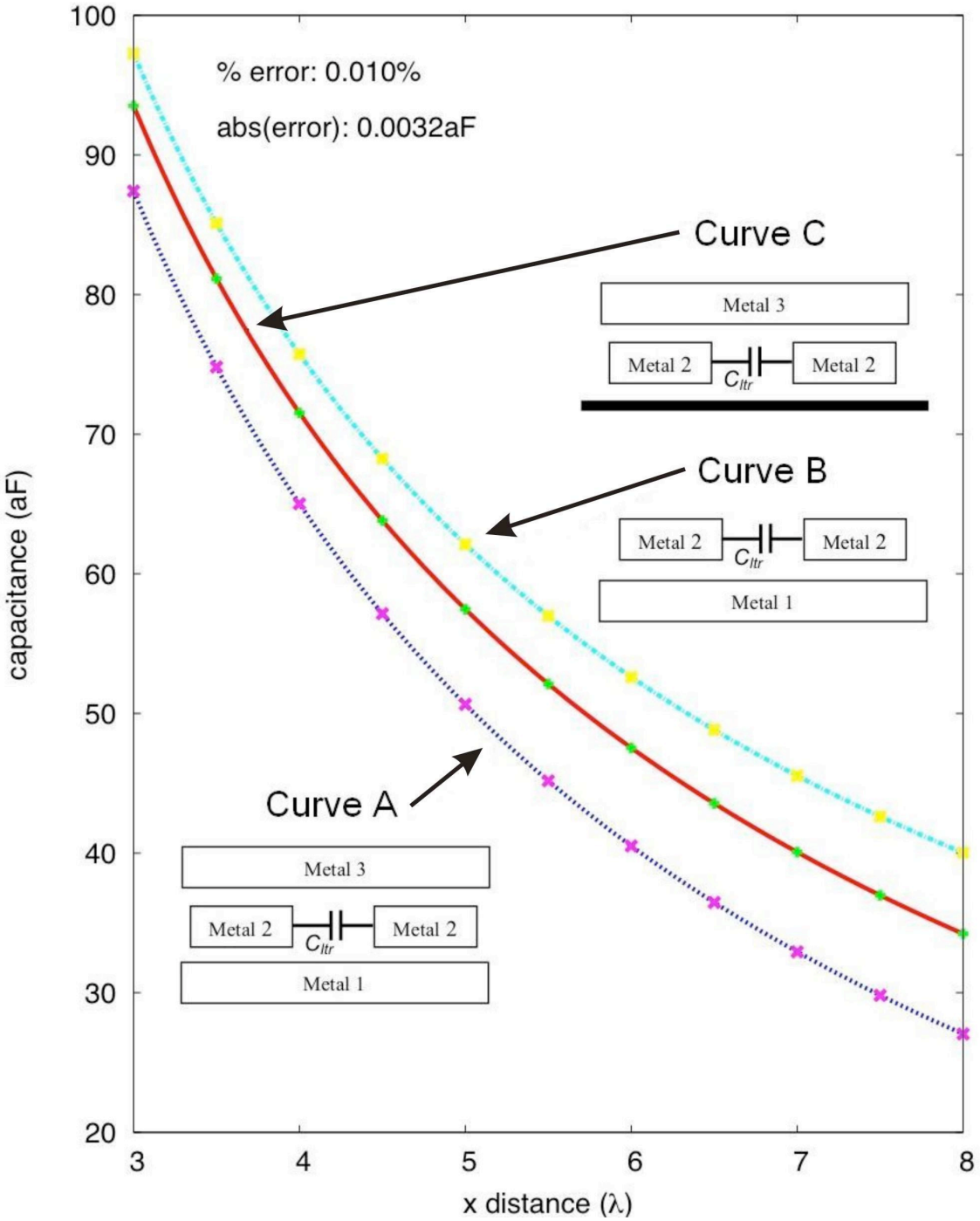

FIGURE F.2: Lateral capacitance on metal 2 per λ.

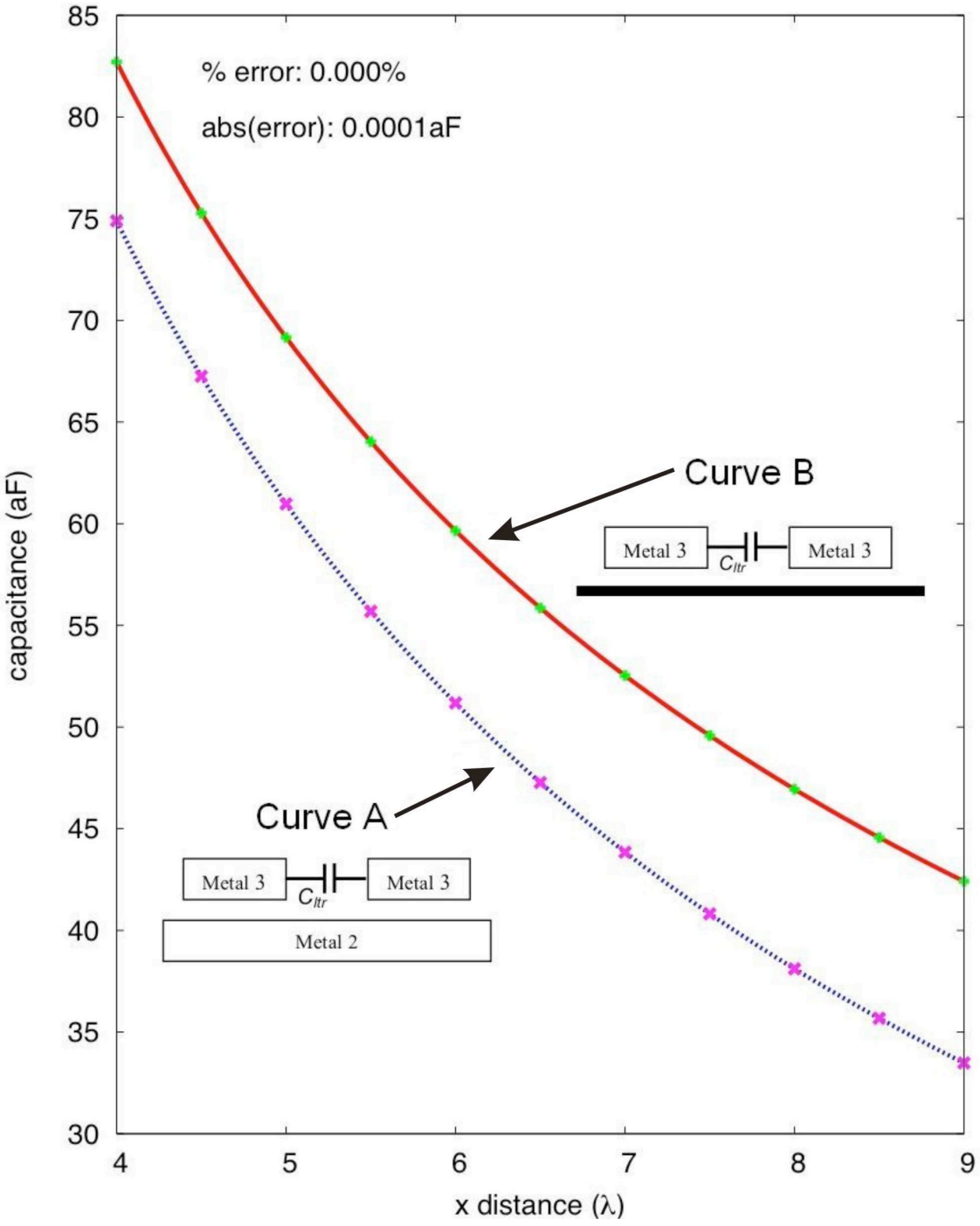

FIGURE F.3: Lateral capacitance on metal 3 per λ.

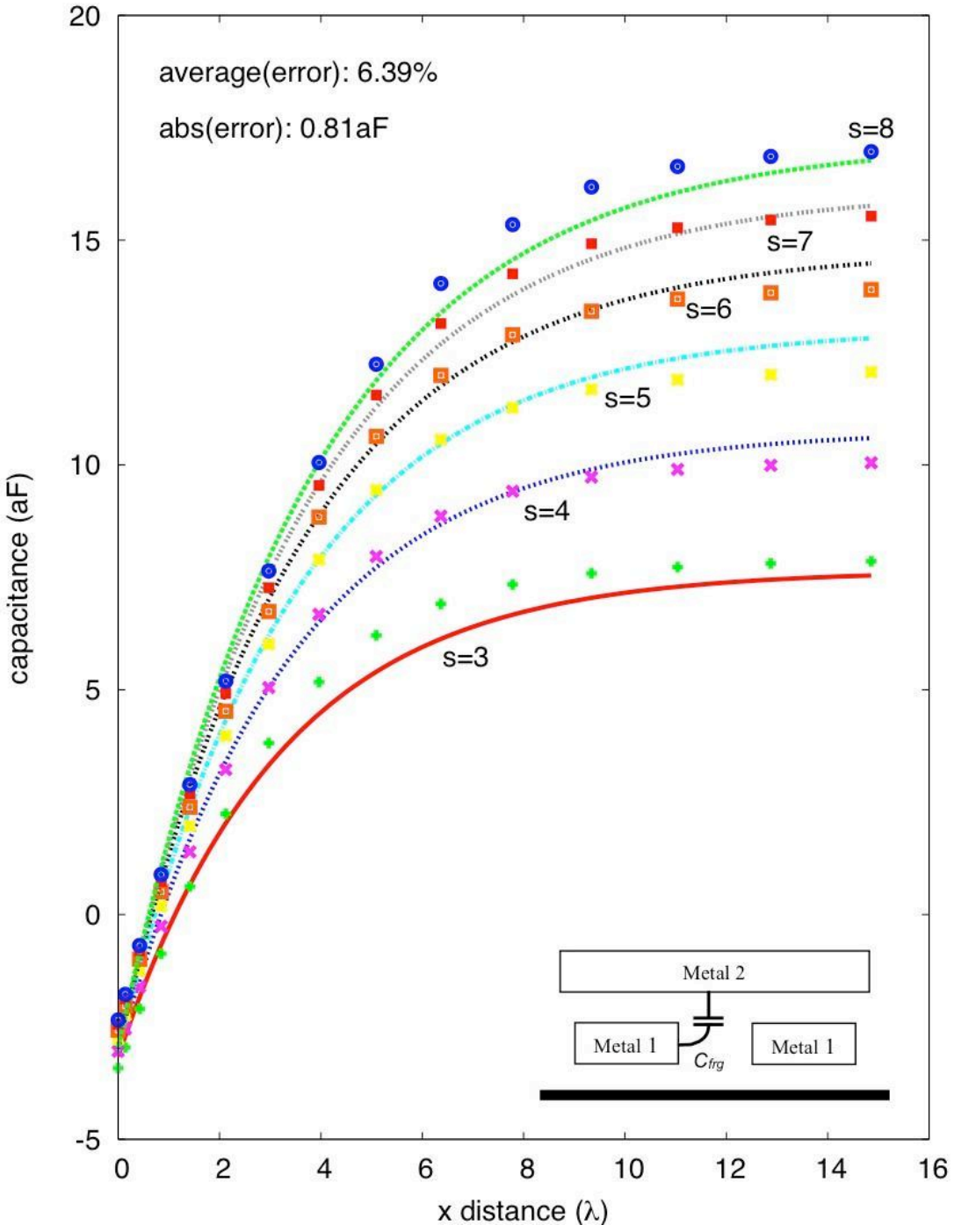

FIGURE F.4: Fringe capacitance metal 1 to metal 2 with substrate per λ.

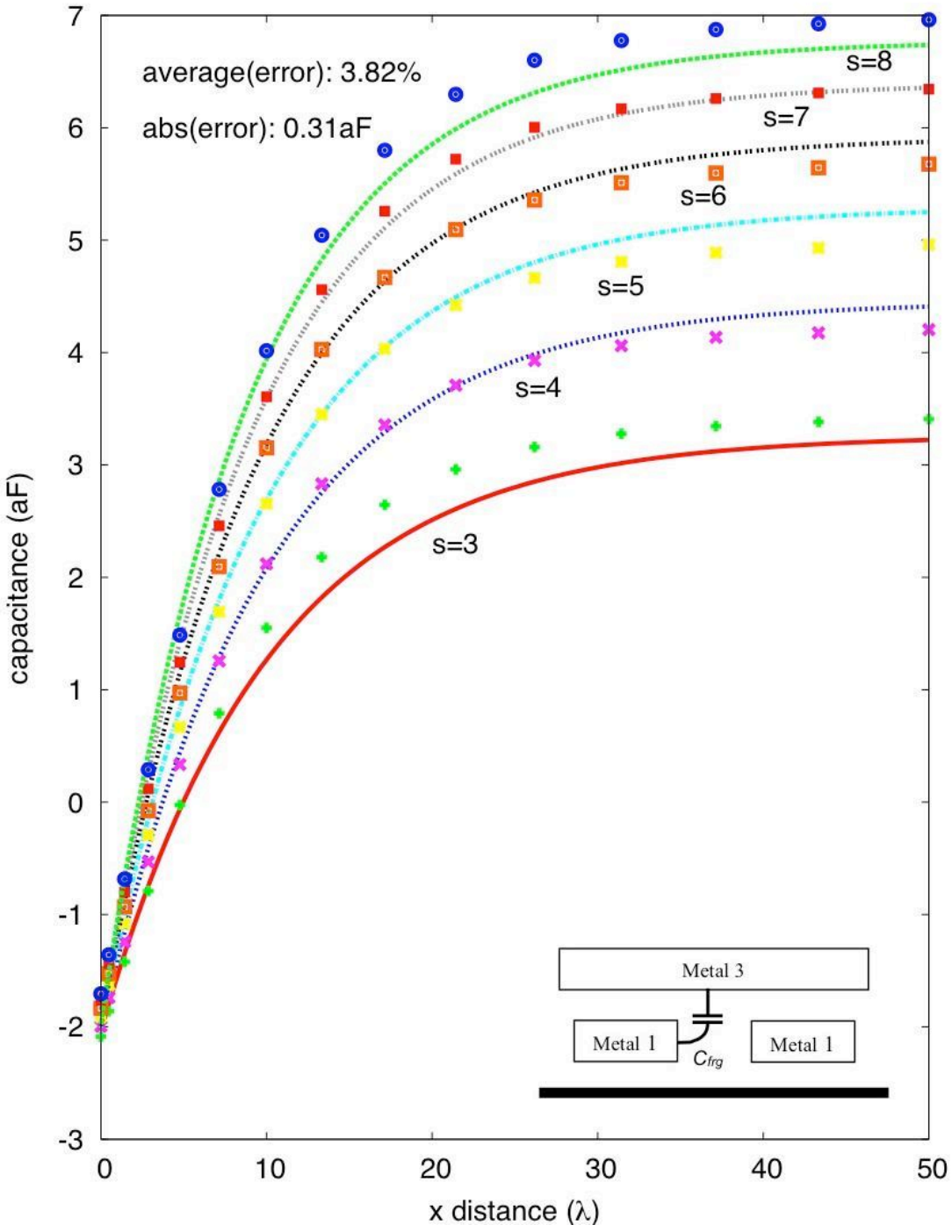

FIGURE F.5: Fringe capacitance metal 1 to metal 3 with substrate per λ.

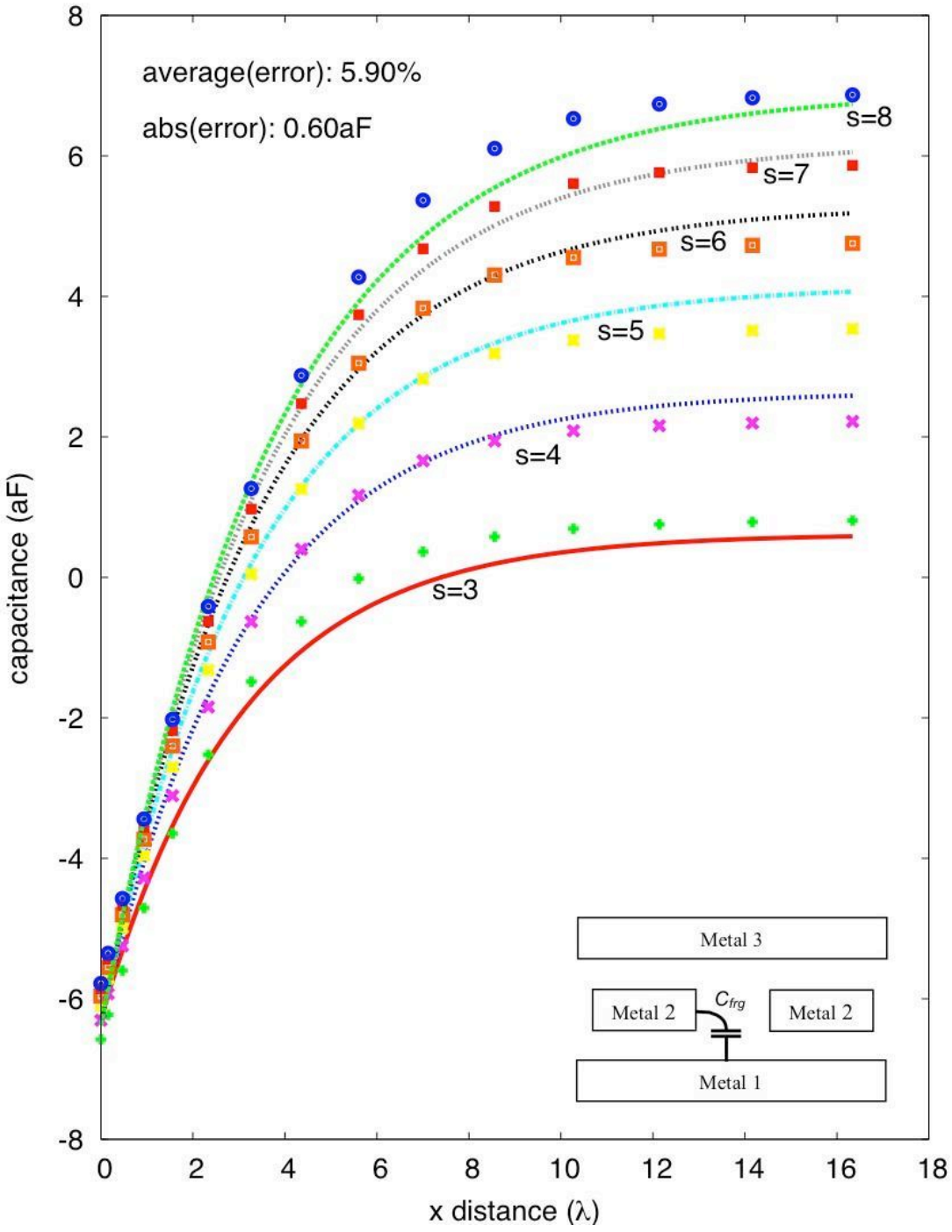

FIGURE F.6: Fringe capacitance metal 2 to metal 1 with metal 3 above per λ.

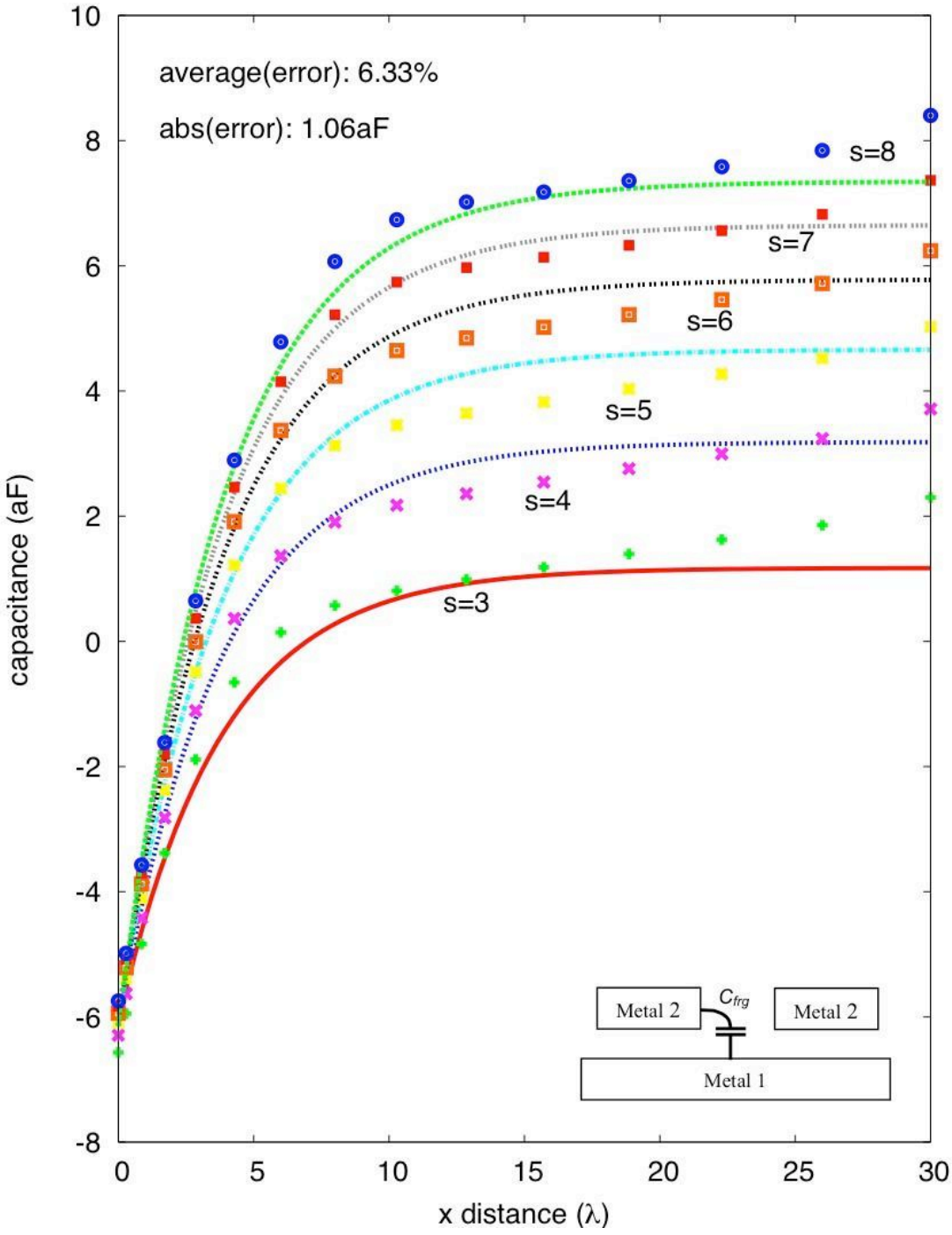

FIGURE F.7: Fringe capacitance metal 2 to metal 1 with space above per λ.

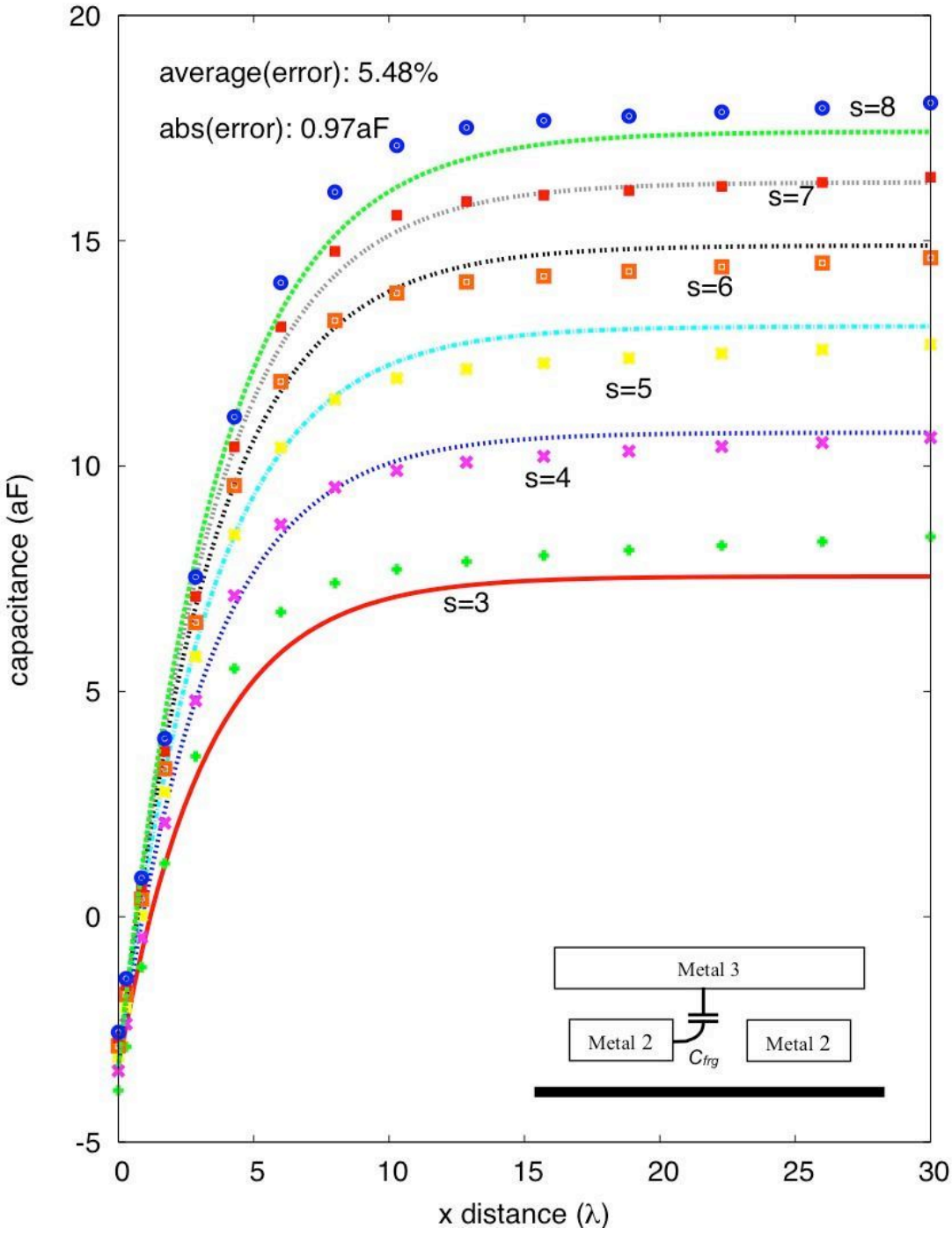

FIGURE F.8: Fringe capacitance metal 2 to metal 3 with substrate per λ.

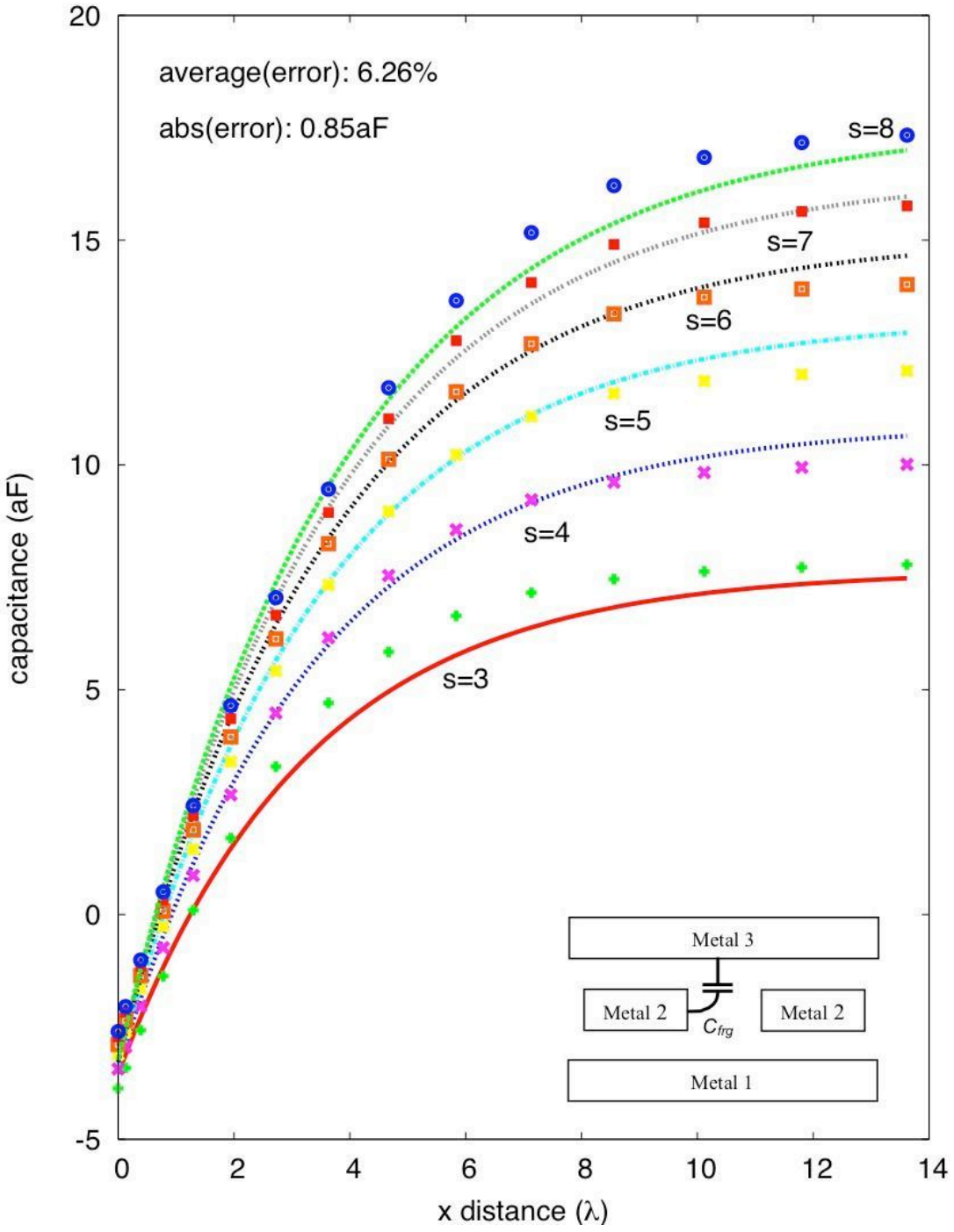

FIGURE F.9: Fringe capacitance metal 2 to metal 3 with metal 1 below per λ.

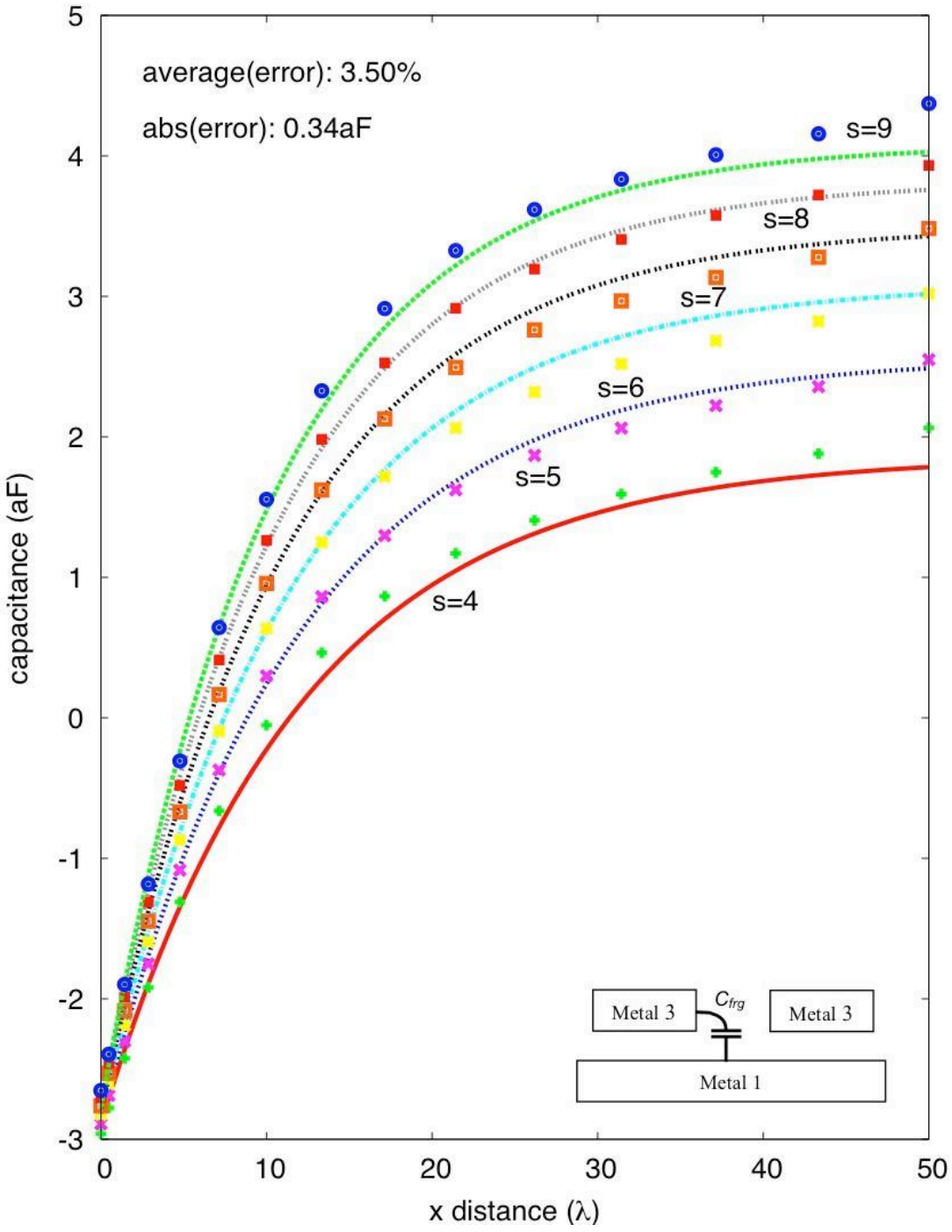

FIGURE F.10: Fringe capacitance metal 3 to metal 1 with space above per λ.

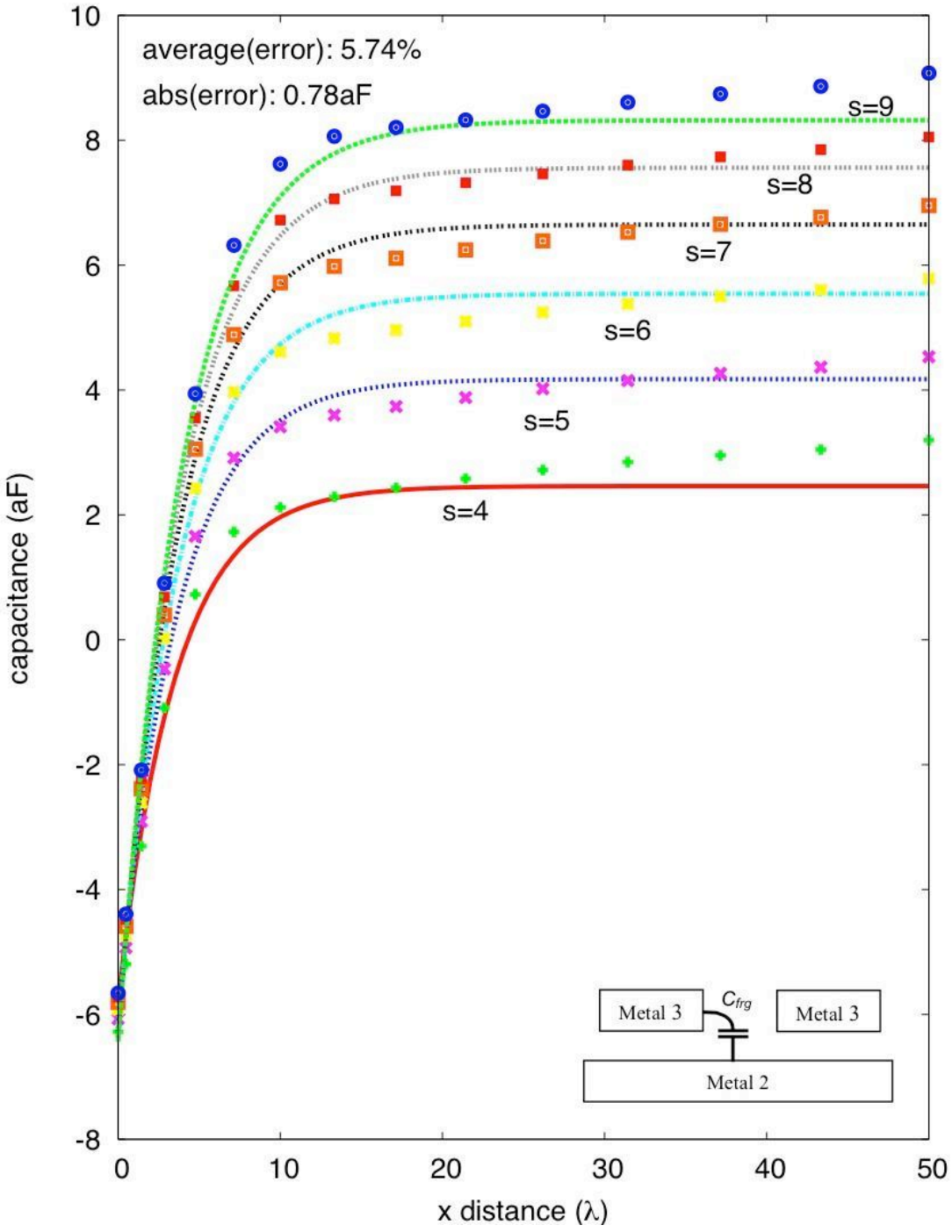

FIGURE F.11: Fringe capacitance metal 3 to metal 2 with space above per λ.

Appendix G

Magic Codes

This Appendix contains the Magic codes for real/coincident edges and pixel-based search algorithms

G.1 Real and Coincident Edge Algorithms

The following is the real and coincident algorithms, and this function has been used heavily throughout Magic.

```
--------------------------------------------------------------------------------
/* RealEdge() to calculate the real edge of tabove and tbelow */
// re->a and re->b must be initialized to zero before entering this function

int
extRealEdge(tile, re)
    Tile *tile;
    struct real_edge *re;
{
    Rect b, b2, r;
    Tile *tCheck, *tCheck2;
    re->a[0] = re->a[1] = re->b[0] = re->b[1] = 0;

    if ( tile != NULL )
    {
```

```
    re->tbelow = tile;

    if (re->tabove == NULL)
    {
        TITORECT(tile, &r);
        GEOCLIP(&re->area, &r);
        re->tabove = tile;
        re->tbelow = tile;
    }
}

/* first define 4 sides to be re->a (tabove edge) or re->b (tbelow edge) */
/* subtract the edge if same layer is connectd to tile *tabove and *tbelow */

if ( re->area.r_xtop == re->area.r_xbot )
{
    // the segment is vertical
    if (RIGHT(re->tbelow) == re->area.r_xtop)
        re->b[0] += re->area.r_ytop - re->area.r_ybot;
    if (RIGHT(re->tabove) == re->area.r_xtop)
        re->a[0] += re->area.r_ytop - re->area.r_ybot;
    if (LEFT(re->tbelow) == re->area.r_xbot)
        re->b[0] += re->area.r_ytop - re->area.r_ybot;
    if (LEFT(re->tabove) == re->area.r_xbot)
        re->a[0] += re->area.r_ytop - re->area.r_ybot;
}
else if ( re->area.r_ytop == re->area.r_ybot )
{
    // the segment is horizontal
    if (TOP(re->tbelow) == re->area.r_ybot)
    {
        re->b[0] += MIN(re->area.r_xtop, RIGHT(re->tabove)) -
            MAX(re->area.r_xbot, LEFT(re->tabove));
        for (tCheck = RT(re->tbelow); RIGHT(tCheck) > re->area.r_xbot;
            tCheck = BL(tCheck))
        {
            if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tbelow))
                continue;
            if ( BOTTOM(tCheck) > re->area.r_ytop) continue;
            if ( LEFT(tCheck) >= re->area.r_xtop) continue;
            b.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck));
            b.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck));
            re->b[0]-=(b.r_xtop - b.r_xbot);
            re->a[1]+=(b.r_xtop - b.r_xbot);
        }
    }
    if (TOP(re->tabove) == re->area.r_ybot)
```

```
{
    re->a[0] += MIN(re->area.r_xtop, RIGHT(re->tabove)) -
        MAX(re->area.r_xbot, LEFT(re->tabove));
    for (tCheck = RT(re->tabove); RIGHT(tCheck) > re->area.r_xbot;
        tCheck = BL(tCheck))
    {
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tabove))
            continue;
        if ( BOTTOM(tCheck) > re->area.r_ytop) continue;
        if ( LEFT(tCheck) >= re->area.r_xtop) continue;
        b.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck));
        b.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck));
        re->a[0]-=(b.r_xtop - b.r_xbot);
        re->b[1]+=(b.r_xtop - b.r_xbot);
    }
}
if (BOTTOM(re->tbelow) == re->area.r_ytop)
{
    re->b[0] += MIN(re->area.r_xtop, RIGHT(re->tabove)) -
        MAX(re->area.r_xbot, LEFT(re->tabove));
    for (tCheck = LB(re->tbelow); LEFT(tCheck) < re->area.r_xtop;
        tCheck = TR(tCheck))
    {
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tbelow))
            continue;
        if ( TOP(tCheck) < re->area.r_ybot) continue;
        if ( RIGHT(tCheck) <= re->area.r_xbot) continue;
        b.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck));
        b.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck));
        re->b[0]-=(b.r_xtop - b.r_xbot);
        re->a[1]+=(b.r_xtop - b.r_xbot);
    }
}
if (BOTTOM(re->tabove) == re->area.r_ytop)
{
    re->a[0] += MIN(re->area.r_xtop, RIGHT(re->tabove)) -
        MAX(re->area.r_xbot, LEFT(re->tabove));
    for (tCheck = LB(re->tabove); LEFT(tCheck) < re->area.r_xtop;
        tCheck = TR(tCheck))
    {
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tabove))
            continue;
        if ( TOP(tCheck) < re->area.r_ybot) continue;
        if ( RIGHT(tCheck) <= re->area.r_xbot) continue;
        b.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck));
        b.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck));
        re->a[0]-=(b.r_xtop - b.r_xbot);
```

```
                re->b[1]+=(b.r_xtop - b.r_xbot);
            }
        }
    }
    else
{
    /* Top */
    if (TOP(re->tabove) > TOP(re->tbelow) && TOP(re->tbelow) == re->area.r_ytop)
    {
        re->b[0]+=(re->area.r_xtop - re->area.r_xbot);
        re->b[1]+=(re->area.r_xtop - re->area.r_xbot);
        for (tCheck = RT(re->tbelow); RIGHT(tCheck) > re->area.r_xbot;
            tCheck = BL(tCheck))
        {
            // tCheck must be same as re->tbelow
            if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tbelow))
                continue;
            // BOTTOM(tCheck) can't be higher than TOP(area)
            if ( BOTTOM(tCheck) > re->area.r_ytop) continue;
            // LEFT(tCheck) can't be righter than RIGHT(area)
            if ( LEFT(tCheck) >= re->area.r_xtop) continue;
            // LEFT = most right of LEFT(area) and LEFT(tCheck)
            b.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck));
            // RIGHT = most left of RIGHT(area) and RIGHT(tCheck)
            b.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck));
            re->b[0]-=(b.r_xtop - b.r_xbot);
            re->b[1]-=(b.r_xtop - b.r_xbot);
        }
    }
    else if ( TOP(re->tabove) == re->area.r_ytop )
    {
        // tabove shrinks or tbelow bloats
        re->a[0]+=(re->area.r_xtop - re->area.r_xbot);
        // tabove shrinks or tbelow bloats
        re->a[1]+=(re->area.r_xtop - re->area.r_xbot);
        if (TOP(re->tabove) == TOP(re->tbelow))
        {
            // tbelow shrinks or tabove bloats
            re->b[1]+=(re->area.r_xtop - re->area.r_xbot);
            // tabove shrinks or tbelow bloats
            re->a[1]-=(re->area.r_xtop - re->area.r_xbot);
            for (tCheck2 = RT(re->tbelow); RIGHT(tCheck2) > re->area.r_xbot;
                tCheck2 = BL(tCheck2))
            {
                if ( TiGetTypeExact(tCheck2) != TiGetTypeExact(re->tbelow))
                    continue;
                if ( BOTTOM(tCheck2) > re->area.r_ytop) continue;
```

```
            if ( LEFT(tCheck2) >= re->area.r_xtop) continue;
            if ( RIGHT(tCheck2) <= re->area.r_xbot ||
                LEFT(tCheck2) >= re->area.r_xtop)    continue;
            b2.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck2));
            b2.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck2));
            re->b[1]-=(b2.r_xtop - b2.r_xbot);
            re->a[1]+=(b2.r_xtop - b2.r_xbot);
        }
    }
    for (tCheck = RT(re->tabove); RIGHT(tCheck) > re->area.r_xbot;
        tCheck = BL(tCheck))
    {
        // tCheck must be same as re->tabove
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tabove))
            continue;
        // BOTTOM(tCheck) can't be higher than TOP(area)
        if ( BOTTOM(tCheck) > re->area.r_ytop ) continue;
        // LEFT(tCheck) can't be righter than RIGHT(area)
        if ( LEFT(tCheck) >= re->area.r_xtop) continue;
        b.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck));
        b.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck));
        re->a[0]-=(b.r_xtop - b.r_xbot);
        if (TOP(re->tabove) != TOP(re->tbelow))
            re->a[1]-=(b.r_xtop - b.r_xbot);
        if ( TOP(re->tabove) == TOP(re->tbelow) )
        {
            for (tCheck2 = RT(re->tbelow); RIGHT(tCheck2) > b.r_xbot;
                tCheck2 = BL(tCheck2))
            {
                if ( TiGetTypeExact(tCheck2) != TiGetTypeExact(re->tbelow) )
                    continue;
                if ( BOTTOM(tCheck2) > re->area.r_ytop) continue;
                if ( LEFT(tCheck2) >= b.r_xtop) continue;
                if ( RIGHT(tCheck2) <= b.r_xbot || LEFT(tCheck2) >= b.r_xtop)
                    continue;
                b2.r_xbot = MAX(b.r_xbot,LEFT(tCheck2));
                b2.r_xtop = MIN(b.r_xtop,RIGHT(tCheck2));
                re->b[0]-=(b2.r_xtop - b2.r_xbot);
                re->a[1]-=(b2.r_xtop - b2.r_xbot);
            }
            re->b[0]+=(b.r_xtop - b.r_xbot);
        }
    }
}
/* Right */
if (RIGHT(re->tabove) > RIGHT(re->tbelow) &&
    RIGHT(re->tbelow) == re->area.r_xtop)
```

```
{
    re->b[0]+=(re->area.r_ytop - re->area.r_ybot);
    re->b[1]+=(re->area.r_ytop - re->area.r_ybot);
    for (tCheck = TR(re->tbelow); TOP(tCheck) > re->area.r_ybot;
        tCheck = LB(tCheck))
    {
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tbelow))
            continue;
        if ( LEFT(tCheck) > re->area.r_xtop) continue;
        if ( BOTTOM(tCheck) >= re->area.r_ytop) continue;
        b.r_ybot = MAX(re->area.r_ybot,BOTTOM(tCheck));
        b.r_ytop = MIN(re->area.r_ytop,TOP(tCheck));
        re->b[0]-=(b.r_ytop - b.r_ybot);
        re->b[1]-=(b.r_ytop - b.r_ybot);
    }
}
else if (RIGHT(re->tabove) == re->area.r_xtop)
{
    re->a[0]+=(re->area.r_ytop - re->area.r_ybot);
    re->a[1]+=(re->area.r_ytop - re->area.r_ybot);
    if (RIGHT(re->tabove) == RIGHT(re->tbelow))
    {
        re->b[1]+=(re->area.r_ytop - re->area.r_ybot);
        re->a[1]-=(re->area.r_ytop - re->area.r_ybot);
    }
    for (tCheck = TR(re->tabove); TOP(tCheck) > re->area.r_ybot;
        tCheck = LB(tCheck))
    {
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tabove))
            continue;
        if ( LEFT(tCheck) > re->area.r_ytop) continue;
        if ( BOTTOM(tCheck) >= re->area.r_ytop) continue;
        b.r_ybot = MAX(re->area.r_ybot,BOTTOM(tCheck));
        b.r_ytop = MIN(re->area.r_ytop,TOP(tCheck));
        re->a[0]-=(b.r_ytop - b.r_ybot);
        re->a[1]-=(b.r_ytop - b.r_ybot);
    }
}
/* Bottom */
if (BOTTOM(re->tabove) < BOTTOM(re->tbelow) &&
    BOTTOM(re->tbelow) == re->area.r_ybot)
{
    re->b[0]+=(re->area.r_xtop - re->area.r_xbot);
    re->b[1]+=(re->area.r_xtop - re->area.r_xbot);
    for (tCheck = LB(re->tbelow); LEFT(tCheck) < re->area.r_xtop;
        tCheck = TR(tCheck))
    {
```

```
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tbelow))
            continue;
        if ( TOP(tCheck) < re->area.r_ybot) continue;
        if ( RIGHT(tCheck) <= re->area.r_xbot ) continue;
        b.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck));
        b.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck));
        re->b[0]-=(b.r_xtop - b.r_xbot);
        re->b[1]-=(b.r_xtop - b.r_xbot);
    }
}
else if ( BOTTOM(re->tabove) == re->area.r_ybot)
{
    re->a[0]+=(re->area.r_xtop - re->area.r_xbot);
    re->a[1]+=(re->area.r_xtop - re->area.r_xbot);
    if (BOTTOM(re->tabove) == BOTTOM(re->tbelow))
    {
        re->b[1]+=(re->area.r_xtop - re->area.r_xbot);
        re->a[1]-=(re->area.r_xtop - re->area.r_xbot);
        for (tCheck2 = LB(re->tbelow); LEFT(tCheck2) < re->area.r_xtop;
            tCheck2 = TR(tCheck2))
        {
            if ( TiGetTypeExact(tCheck2) != TiGetTypeExact(re->tbelow))
                continue;
            if ( BOTTOM(tCheck2) > re->area.r_ybot) continue;
            if ( LEFT(tCheck2) >= re->area.r_xtop) continue;
            if ( RIGHT(tCheck2) <= re->area.r_xbot ||
                LEFT(tCheck2) >= re->area.r_xtop)    continue;
            b2.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck2));
            b2.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck2));
            re->b[1]-=(b2.r_xtop - b2.r_xbot);
            re->a[1]+=(b2.r_xtop - b2.r_xbot);
        }
    }
    for (tCheck = LB(re->tabove); LEFT(tCheck) < re->area.r_xtop;
        tCheck = TR(tCheck))
    {
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tabove))
            continue;
        if ( TOP(tCheck) < re->area.r_ybot ) continue;
        if ( RIGHT(tCheck) <= re->area.r_xbot ) continue;
        b.r_xbot = MAX(re->area.r_xbot,LEFT(tCheck));
        b.r_xtop = MIN(re->area.r_xtop,RIGHT(tCheck));
        re->a[0]-=(b.r_xtop - b.r_xbot);
        if (BOTTOM(re->tabove) != BOTTOM(re->tbelow))
            re->a[1]-=(b.r_xtop - b.r_xbot);
        if ( BOTTOM(re->tabove) == BOTTOM(re->tbelow) )
        {
```

```
            for (tCheck2 = LB(re->tbelow); LEFT(tCheck2) < b.r_xtop;
                tCheck2 = TR(tCheck2))
            {
                if ( TiGetTypeExact(tCheck2) != TiGetTypeExact(re->tbelow) )
                    continue;
                if ( BOTTOM(tCheck2) > re->area.r_ybot) continue;
                if ( LEFT(tCheck2) >= b.r_xtop) continue;
                if ( RIGHT(tCheck2) <= b.r_xbot || LEFT(tCheck2) >= b.r_xtop)
                    continue;
                b2.r_xbot = MAX(b.r_xbot,LEFT(tCheck2));
                b2.r_xtop = MIN(b.r_xtop,RIGHT(tCheck2));
                re->b[0]-=(b2.r_xtop - b2.r_xbot);
                re->a[1]-=(b2.r_xtop - b2.r_xbot);
            }
            re->b[0]+=(b.r_xtop - b.r_xbot);
        }
    }
}
/* Left */
if (LEFT(re->tabove) < LEFT(re->tbelow) &&
    LEFT(re->tbelow) == re->area.r_xbot)
{
    re->b[0]+=(re->area.r_ytop - re->area.r_ybot);
    re->b[1]+=(re->area.r_ytop - re->area.r_ybot);
    for (tCheck = BL(re->tbelow); BOTTOM(tCheck) < re->area.r_ytop;
        tCheck = RT(tCheck))
    {
        if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tbelow))
            continue;
        if ( RIGHT(tCheck) < re->area.r_xbot) continue;
        if ( TOP(tCheck) <= re->area.r_ybot) continue;
        b.r_ybot = MAX(re->area.r_ybot,BOTTOM(tCheck));
        b.r_ytop = MIN(re->area.r_ytop,TOP(tCheck));
        re->b[0]-=(b.r_ytop - b.r_ybot);
        re->b[1]-=(b.r_ytop - b.r_ybot);
    }
}
else if ( LEFT(re->tabove) == re->area.r_xbot )
{
    re->a[0]+=(re->area.r_ytop - re->area.r_ybot);
    re->a[1]+=(re->area.r_ytop - re->area.r_ybot);
    if (LEFT(re->tabove) == LEFT(re->tbelow))
    {
        re->b[1]+=(re->area.r_ytop - re->area.r_ybot);
        re->a[1]-=(re->area.r_ytop - re->area.r_ybot);
    }
    for (tCheck = BL(re->tabove); BOTTOM(tCheck) < re->area.r_ytop;
```

```
            tCheck = RT(tCheck))
        {
            if ( TiGetTypeExact(tCheck) != TiGetTypeExact(re->tabove))
                continue;
            if ( RIGHT(tCheck) < re->area.r_ybot) continue;
            if ( TOP(tCheck) <= re->area.r_ybot) continue;
            b.r_ybot = MAX(re->area.r_ybot,BOTTOM(tCheck));
            b.r_ytop = MIN(re->area.r_ytop,TOP(tCheck));
            re->a[0]-=(b.r_ytop - b.r_ybot);
            re->a[1]-=(b.r_ytop - b.r_ybot);
        }
    }
}
return (0);
}
```

G.2 Pixel-Based Search Algorithms

The following shows the pixel-based search algorithms for the right side of each tile. Left, top, and bottom sides could also be implemented similarly. The pixel search algorithm is used for lateral, fringe and substrate capacitance computations.

```
/*
 * ----------------------------------------------------------------------
 *
 * extSideRight --
 *
 * Searching to the right of the boundary 'bp', we found the tile
 * 'tpfar' which may lie on the far side of an edge to which the
 * edge bp->b_inside | bp->b_outside has sidewall coupling capacitance.
 *
 * The pixel search algorithm is implemented to compute
 * lateral capacitance. Similar method is applied to
 * extSideTop, but not to extSideLeft and extSideBottom
 * to avoid twice computations.
 *
 * Results: Returns 0 always.
 *
 * ----------------------------------------------------------------------
 */
int
```

```
extSideRight(tpfar, bp)
    Tile *tpfar;
    Boundary *bp;
{
    NodeRegion *rinside = (NodeRegion *) extGetRegion(bp->b_inside);
    NodeRegion *rfar = (NodeRegion *) extGetRegion(tpfar);
    Tile *tpnear;
    //NKH: parameters added
    int i, j, pNum, l_a, l_b, l_o, l_op, l_tp;
    Rect r, r2;
    struct real_edge se;
    Tile *above, *below, *tclose;

    if (rfar != (NodeRegion *) extUnInit && rfar != rinside)
    {
        int sep = LEFT(tpfar) - bp->b_segment.r_xtop;
        int limit = MIN(bp->b_segment.r_ytop, TOP(tpfar));
        int start = MAX(bp->b_segment.r_ybot, BOTTOM(tpfar));

        for (tpnear = BL(tpfar); BOTTOM(tpnear) < limit; tpnear = RT(tpnear))
        {
            int overlap = MIN(TOP(tpnear), limit) - MAX(BOTTOM(tpnear), start);
            if (overlap > 0)
            {
                extSideCommon(rinside, rfar, tpnear, tpfar, overlap, sep);
                tclose = tpnear;
            }
        }

        /* NKH: pixel version added */
        if (CAP_LATERAL)
        {
            for (i=start;i<limit;i++)
            {
                for(j=RIGHT(bp->b_inside);j<LEFT(tpfar);j++)
                {
                    above = below = NULL;
                    l_a = l_b = l_op = l_tp = 0;
                    r.r_ybot=i;
                    r.r_ytop=i+1;
                    r.r_xbot=j;
                    r.r_xtop=j+1;
                    r2.r_xbot=r.r_xbot-1;
                    r2.r_xtop=r.r_xtop+1;
                    r2.r_ybot=r.r_ybot-1;
                    r2.r_ytop=r.r_ytop+1;
                    se.tabove = tpfar;
```

```
se.tbelow = NULL;
se.area = r;
for (pNum = PL_TECHDEPBASE; pNum < DBNumPlanes; pNum++)
{
    if (PlaneMaskHasPlane(*bp->cd_pmask, pNum))
    {
        (void) DBSrPaintArea((Tile *) NULL,
            bp->cd_plane[pNum], &r, bp->cd_bmask,
            extRealEdge, (ClientData) &se);
        if (se.tbelow == NULL)
        {
            se.tabove = NULL;
            (void) DBSrPaintArea((Tile *) NULL,
                bp->cd_plane[pNum], &r2, bp->cd_bmask,
                extRealEdge, (ClientData) &se);
            l_o=-1; // at -1 if shrinking
        }
        else l_o=1; // if overlapped, l_o = 1 for bloating
        //define the closest tile above and below for lateral
        if (se.tbelow != NULL)
        {
            if ( TiGetTypeExact(se.tbelow)>
                TiGetTypeExact(bp->b_inside) )
            {
                if (above !=NULL)
                {
                    if ( TiGetTypeExact(se.tbelow)<
                        TiGetTypeExact(above) )
                    {
                        l_a=0;
                        above = se.tbelow;
                        l_a+=l_o*MAX(se.b[0],se.b[1]);
                    }
                    else if (l_o == 1 && l_op != 1)
                    {
                        l_a=0;
                        above = se.tbelow;
                        l_a+=l_o*MAX(se.b[0],se.b[1]);
                    }
                }
                else
                {
                    above = se.tbelow;
                    l_a+=l_o*MAX(se.b[0],se.b[1]);
                }
            }
            if ( TiGetTypeExact(se.tbelow)<
```

```
                                    TiGetTypeExact(bp->b_inside) )
                                {
                                    if (below !=NULL)
                                    {
                                        if ( TiGetTypeExact(se.tbelow)>
                                            TiGetTypeExact(below) )
                                        {
                                            l_b=0;
                                            below = se.tbelow;
                                            l_b+=l_o*MAX(se.b[0],se.b[1]);
                                        }
                                    }
                                    else
                                    {
                                        below = se.tbelow;
                                        l_b+=l_o*MAX(se.b[0],se.b[1]);
                                    }
                                }
                            }
                            l_op = l_o;
                        }
                    }
                    if ( j==RIGHT(bp->b_inside) || j==LEFT(tpfar)-1 ) l_tp+=1;
                    if ( i==start ) l_tp+=1;
                    if ( i==limit-1 ) l_tp+=1;
                    // go to lateral function if j is within region
                    extLateral(rinside, rfar, tclose, tpfar, above, l_a, below,
                        l_b, sep, l_tp);
                }
            }
        }
    }
    return (0);
}

/*
 * ----------------------------------------------------------------------------
 *
 * extGndRight --
 *
 * Searching the fringing to GND effects on the
 * right edge of each tile. Similar methods are applied to
 * extGndTop, extGndLeft, extGndBottom
 * to account for fringing effects on substrate capacitance
 *
 * Results: Returns 0 always.
 *
```

```
 * ----------------------------------------------------------------------------
 */
int
extGndRight(esws)
    extSidewallStruct *esws;
{
    NodeRegion *rsub = (NodeRegion *) extGetRegion(esws->bp->b_inside);
    int i, j, sep, pNum;
    Rect r;
    struct real_edge test;
    test.tabove = test.tbelow = esws->bp->b_inside;

    for (i=esws->bp->b_segment.r_ybot;i<esws->bp->b_segment.r_ytop;i++)
    {
        for(j=RIGHT(esws->bp->b_inside)+edgeHalo;j>=RIGHT(esws->bp->b_inside);j--)
        {
            r.r_ybot=i;
            r.r_ytop=i+1;
            r.r_xbot=j;
            r.r_xtop=j+1;
            test.area = r;

            for (pNum = PL_TECHDEPBASE; pNum < esws->plane_of_boundary; pNum++)
            {
                (void) DBSrPaintArea((Tile *) NULL, esws->bp->cd_plane[pNum],
                    &r, esws->bp->cd_bmask,
                    extRealEdge, (ClientData) &test);
            }
            (void) DBSrPaintArea((Tile *) NULL,
                esws->bp->cd_plane[esws->plane_of_boundary], &r,
                esws->edge, extRealEdge, (ClientData) &test);
        }
        if (test.tbelow != esws->bp->b_inside)
            sep = LEFT(test.tbelow) - RIGHT(esws->bp->b_inside);
        else sep = edgeHalo;

        if (sep>0) AddGndCap(rsub,sep);
    }
    return(0);
}

/*
 * ----------------------------------------------------------------------------
 *
 * extSideOverRight --
 *
 * Search fringe capacitance pixel by pixel on the right edge
```

```
 * of each tile, similar methods are applied to
 * extSideOverTop, extSideOverLeft, extSideOverBottom
 * functions to account for fringe capacitance
 *
 * Results: Returns 0 always.
 *
 * ----------------------------------------------------------------------------
 */
int
extSideOverRight(esws)
    extSidewallStruct *esws; /* Overlapping edge and plane information */
{
    int pNum, i, j, sep, l_a, l_b, l_p, side, side_x;
    Rect r;
    struct real_edge se, test;
    Tile *above, *below;
    test.tabove = test.tbelow = esws->bp->b_inside;

    /*NKH: pixel version added*/
    for (i=esws->bp->b_segment.r_ybot;i<esws->bp->b_segment.r_ytop;i++)
    {
        for(j=RIGHT(esws->bp->b_inside)+sideHalo;j>=RIGHT(esws->bp->b_inside);j--)
        {
            r.r_ybot=i;
            r.r_ytop=i+1;
            r.r_xbot=j;
            r.r_xtop=j+1;
            test.area = r;
            (void) DBSrPaintArea((Tile *) NULL,
                esws->bp->cd_plane[esws->plane_of_boundary], &r,
                esws->edge, extRealEdge, (ClientData) &test);
        }
        if (test.tbelow != esws->bp->b_inside)
            sep = LEFT(test.tbelow) - RIGHT(esws->bp->b_inside);
        else sep = 0;
        if (sep!=0)    side_x=sep;
        else           side_x=sideHalo;

        for(j=RIGHT(esws->bp->b_inside);j<RIGHT(esws->bp->b_inside)+sideHalo;j++)
        {
            above = below = NULL;
            l_a = l_b = l_p = side = 0;
            r.r_ybot=i;
            r.r_ytop=i+1;
            r.r_xbot=j;
            r.r_xtop=j+1;
            se.tabove = se.tbelow = esws->bp->b_inside;
```

```
se.area = r;
for (pNum = PL_TECHDEPBASE; pNum < DBNumPlanes; pNum++)
{
    if (PlaneMaskHasPlane(*esws->bp->cd_pmask, pNum))
    {
        esws->plane_checked = pNum;
        (void) DBSrPaintArea((Tile *) NULL, esws->bp->cd_plane[pNum],
            &r, esws->bp->cd_bmask,
            extRealEdge, (ClientData) &se);
    }
    /* define the closest tile above and below for fringe */
    if (TiGetTypeExact(se.tbelow)>TiGetTypeExact(esws->bp->b_inside))
    {
        if (above == NULL)
        {
            above = se.tbelow;
            l_a+=MAX(se.b[0],se.b[1]);
        }
        else
            if ( TiGetTypeExact(se.tbelow)<TiGetTypeExact(above) )
            {
                above = se.tbelow;
                l_a+=MAX(se.b[0],se.b[1]);
            }
    }
    if (TiGetTypeExact(se.tbelow)<TiGetTypeExact(esws->bp->b_inside))
    {
        if (below == NULL)
        {
            below = se.tbelow;
            l_b+=MAX(se.b[0],se.b[1]);
        }
        else
            if ( TiGetTypeExact(se.tbelow)>TiGetTypeExact(below) )
            {
                below = se.tbelow;
                l_b+=MAX(se.b[0],se.b[1]);
            }
    }
}
if (above != NULL)
{
    if (r.r_ytop==TOP(esws->bp->b_inside)&&r.r_ytop==TOP(above))
        l_a-=1;
    else if (r.r_ytop==TOP(esws->bp->b_inside)&&r.r_ytop<TOP(above))
        l_a+=1;
    if (r.r_ybot==BOTTOM(esws->bp->b_inside)&&r.r_ybot==BOTTOM(above))
```

```
                    l_a-=1;
                else if (r.r_ybot==BOTTOM(esws->bp->b_inside)&&
                    r.r_ybot>BOTTOM(above))    l_a+=1;
                if (r.r_ytop==TOP(esws->bp->b_inside)&&r.r_ytop<TOP(above))
                    l_p=1;
                if (r.r_ybot==BOTTOM(esws->bp->b_inside)&&r.r_ybot>BOTTOM(above))
                    l_p=1;
                if (j==LEFT(above) || j==RIGHT(above)-1)
                    side = 1;
                if (j-RIGHT(esws->bp->b_inside)==0)
                {
                    side=-1;
                    l_a-=1;
                }
                extFringe(above, l_a, esws, sep, j-RIGHT(esws->bp->b_inside),
                    l_p, side);
            }
            if (below != NULL)
            {
                if (r.r_ytop==TOP(esws->bp->b_inside)&&r.r_ytop==TOP(below))
                    l_b-=1;
                else if (r.r_ytop==TOP(esws->bp->b_inside)&&r.r_ytop<TOP(below))
                    l_b+=1;
                if (r.r_ybot==BOTTOM(esws->bp->b_inside)&&r.r_ybot==BOTTOM(below))
                    l_b-=1;
                else if (r.r_ybot==BOTTOM(esws->bp->b_inside)&&
                    r.r_ybot>BOTTOM(below))    l_b+=1;
                if (r.r_ytop==TOP(esws->bp->b_inside)&&r.r_ytop<TOP(below))
                    l_p=1;
                if (r.r_ybot==BOTTOM(esws->bp->b_inside)&&r.r_ybot>BOTTOM(below))
                    l_p=1;
                if (j==LEFT(below) || j==RIGHT(below)-1)
                    side = 1;
                if (j-RIGHT(esws->bp->b_inside)==0)
                {
                    side=-1;
                    l_b-=1;
                }
                extFringe(below, l_b, esws, sep, j-RIGHT(esws->bp->b_inside),
                    l_p, side);
            }
        }
    }
    return (0);
}
```