Development of an Architectural Design Tool for 3-D VLSI Sensors

by

Brian Tyrrell

B.S.E., University of Pennsylvania (1998)

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

Master of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2004

Author .....
Department of Electrical Engineering and Computer Science
May 7, 2004

Certified by .....
L. Rafael Reif
Associate Department Head and Professor of Electrical Engineering
Thesis Supervisor

Certified by .....
Robert K. Reich
Assistant Leader, Technology Group
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

This work was sponsored by the United States Air Force under Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Development of an Architectural Design Tool for 3-D VLSI Sensors

by

Brian Tyrrell

Submitted to the Department of Electrical Engineering and Computer Science
on May 7, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

Three dimensional integration schemes for VLSI have the potential for enabling the development of new high-performance architectures for applications such as focal plane sensors. Due to the high costs involved in 3-D VLSI fabrication and the fabrication complexity of 3-D integration, analysis of the design and process tradeoffs for a particular application is essential. An architectural and topological design tool is presented that enables the high-level analysis and optimization of sensor architectures targeted to a variety of 3-D VLSI process options. This design tool is based on an inference chain evaluation framework, and allows for a high-level structural representation of a circuit architecture to be considered in conjunction with low-level process models. Approximation strategies for projecting circuit area and performance are incorporated into the inference chain relations.

Thesis Supervisor: L. Rafael Reif
Title: Associate Department Head and Professor of Electrical Engineering

Thesis Supervisor: Robert K. Reich
Title: Assistant Leader of the Lincoln Laboratory Advanced Imaging Technology Group

Acknowledgments

First, I would like to thank the Lincoln Scholars Program (LSP) Committee at MIT Lincoln Laboratory for its support of this work. I am especially indebted to Dr. Robert O'Donnell, who encouraged me to pursue graduate study through the LSP. A special thanks also goes out to my advisors, Prof. Rafael Reif of the MIT Electrical Engineering and Computer Science Department and Dr. Robert Reich of Lincoln Laboratory.

I also owe a debt of gratitude to many Lincoln staff members in the Advanced Silicon Technology and Advanced Imaging groups who provided much needed assistance and feedback throughout the research process. A particular thanks goes out to Dr. Robert Berger, Dr. Charles Stevenson, Dr. Brian Aull, Dr. David Shaver, Bruce Wheeler, Antonio Soares and Keith Warner. Just as important are those who provided critical administrative support, especially Susan Moriarty. The library staff at Lincoln Laboratory also provided a tremendous resource and helped shave off valuable hours by streamlining the research process.

Finally, without the support of my family, academic work would have been impossible. My wife, Lanaanne, and sons Jacob, Jonathan and Jeremiah have put up with my long hours and crankiness for quite some time. I am also grateful for the useful feedback Lanaanne provided concerning both the content and the presentation of this thesis work.

... as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we don't know we don't know. And if one looks throughout the history of our country and other free countries, it is the latter category that tend to be the difficult ones.

—Donald Rumsfeld

12 February 2002

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Over the past several decades, transistor technology development has focused on reduction of process critical dimensions (CD) and supply voltages. This has been primarily driven by high-volume digital applications. For many analog applications requiring a large dynamic range, further scaling beyond the 0.25 µm-node has yielded diminishing benefits. Now, as device-to-device interconnect limitations begin to dominate in digital VLSI systems, new technologies are emerging that may also prove beneficial for many analog and mixed signal applications. One especially promising new technology is three-dimensional (3-D) integration.

Three-dimensional integration schemes for VLSI have the potential for enabling the development of new high-performance sensor architectures for applications such as focal planes, chemical and biological agent detectors and micronodes. In particular, 3-D VLSI technology is well suited to problems that can benefit from the use of parallel signal detection and processing paths, as well as those for which various stages of the signal path are best implemented in different technologies.

Because of the high costs involved in 3-D VLSI fabrication and the modular nature of 3-D integration, analysis of the design and process tradeoffs for a particular

application is essential. An architectural and topological design tool is developed to enable the high-level analysis and optimization of sensor architectures targeted to a particular set of 3-D VLSI process options. The ultimate goal of this research is the development of a tool that can be used in the conceptual phases of a design to analyze the tradeoffs inherent in the available architectural and process options. This information is particularly useful in cost-benefit analyses of proposed systems, and in selection of appropriate wafer processes for each tier of a potential 3-D integrated die. The ultimate goal is to facilitate the determination of the most promising approach for a particular application.

By its very nature, electronic design automation (EDA) for 3-D mixed signal circuits is an important but technically difficult problem. This work will provide the groundwork for a 3-D EDA tool. Further enhancements by future investigators are to be expected.

1.2 Concise Background

Over the past several years, increasing attention has been directed toward the three-dimensional integration of VLSI circuits. This development has primarily been driven by the demands of digital applications, but tangible benefits are projected for analog applications as well. It has been demonstrated that the three-dimensional stacking of multiple active device layers can significantly reduce the number of long global and semi-global interconnects.[35, 36] Since digital VLSI systems are increasingly limited by the performance of the back end, significant speed and power benefits are anticipated.

There are a number of 3-D integration schemes presently under development. One approach uses low temperature silicon epitaxy to achieve vertical integration.[12] Although there exists the potential for high vertical connectivity with this method, thermal budget considerations would likely limit such processes to a small number of

active device layers. An alternative method uses low temperature wafer bonding of processed wafers to form a multi-tiered structure.[9, 37] Since each tier of the final die corresponds to an individually processed wafer, a tremendous degree of process flexibility is possible with this method. For example, multiple material systems may be used, thermal budgets for each tier are essentially independent, and the number of active device tiers is limited primarily by the yield and reliability of the bonding process and other mechanical and thermal constraints. The particular 3-D processes considered in this study are discussed in further detail in chapter 2.

Wafer-bonded 3-D processes may offer considerable benefits for analog and mixed signal designs. 3-D stacking of state-of-the-art digital CMOS devices on top of analog-optimized wafers with larger gate CD has been getting increasing attention.[25] One possible early application of wafer-bonded processes is in the area of solid state imagers. This is an area of considerable ongoing research.[9, 24, 28, 29] The EDA design aid presented here finds its intended application in this area of integrated imagers.

The EDA infrastructure necessary for enabling design of commercial 3-D integrated circuits is not yet in place for digital applications. There has been extensive research pertaining to the modeling of interconnects in such systems.[16, 35, 36] Based on these interconnect models, some digital place and route algorithms have been implemented.[33, 34] For full-custom design, a 3-D extension of the MAGIC CAD tool has been developed.[4] However, very few EDA design aids exist for analog and mixed signal design, even for conventional 2-D integration.

The conceptual stages of 3-D imager design are well suited to automation. Any proposed imager architecture can be described as a set of signal processing paths, each composed of well-understood sub-blocks. For most 2-D imager designs, the topology of the imager is easily anticipated, thus eliminating the need for semi-automated pre-design. In this conventional case, the size of the pixel places a strict limit on the complexity of the per-pixel circuitry, thus limiting the design options. With the introduction of 3-D integration technology, the set of achievable architectures

increases tremendously since more of the signal processing can be implemented within the pixel itself. This leads to a new set of design tradeoffs, not the least of which is that between the parallelism of the signal processing and the number of tiers of active devices.

The purpose of the described tool is to facilitate the exploration of these new tradeoffs in the initial stages of architectural design. It will allow circuit architectures to be defined at a high level so that their performance may be evaluated prior to the more expensive phases of the design cycle. It will also aid in the determination of an initial floor plan that makes good use of the process flexibility that wafer bonding technology offers the circuit designer.

For most imaging systems, the signal processing path is generally described as follows.[27] As illustrated in the signal flow diagram of figure 1-1, the incident radiation, here indicated as "light," generates an analog response in the detector. This is typically subject to analog conditioning prior to analog to digital conversion (ADC) and subsequent digital processing. In some applications, particular components of this signal flow are eliminated. For example, in Geiger mode avalanche photodiode based photon counting and LADAR imagers, the photodetector is designed to produce a full digital swing, which can then serve as a direct input to digital signal processing.[6]

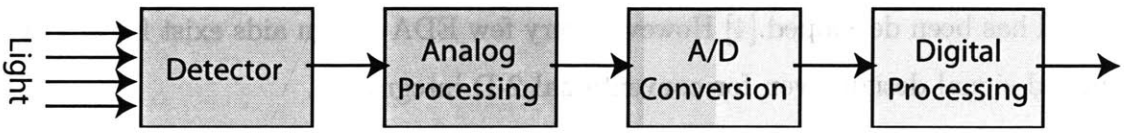

Figure 1-1: Typical signal flow path for an imager system

The detector and the input to the analog processing are always implemented in a per-pixel manner. The remainder of the analog processing stage, the ADC and the digital processing may be implemented in a per-pixel manner, shared by a set of pixels, or implemented once for the array. In addition, the functions of each of these blocks may be broken up into a per-pixel component or a shared component.

Increased functionality within the pixel can provide the benefits of improved signal to noise ratio, increased speed, and reduced power. The primary constraint limiting the addition of new functionality to the pixel is the maximum pixel area. 3-D integration can increase the available area for per-pixel processing, but this capability comes at the expense of an increased process cost and decreased yield.

A simplified illustration suggesting some of the tradeoffs inherent in the design of an imager is shown in figure 1-2.[27] Expressions for estimated power consumption of the analog processing, ADC, and digital processing are given. Note that the optimization of this system requires an understanding of the coefficients K_A, K_{ADC}, and K_D, which are technology and implementation dependent. Such an optimization is necessary in determining what aspects of signal processing are best managed at the analog front end or the digital back end.

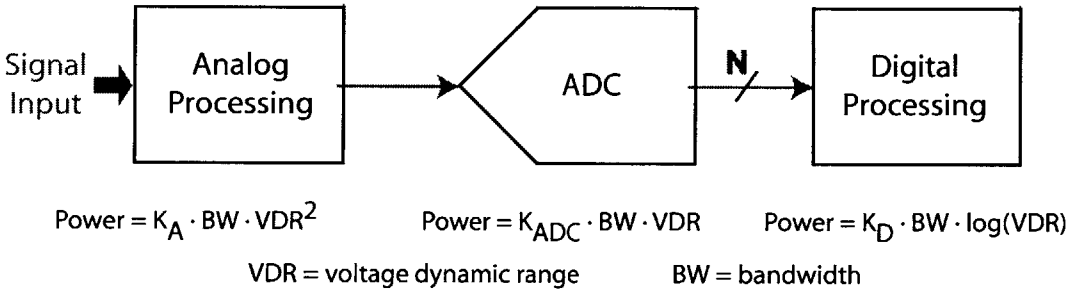

Figure 1-2: Power consumption for a typical signal path

In a 3-D design, new possibilities are introduced that affect this tradeoff. In particular, the analog processing may be made much more sophisticated and the ADC may be implemented at the pixel. It is even possible to place digital processing immediately behind the pixel array, allowing for massively parallel inputs.

1.3 Objective of This Thesis Work

Due to the high costs involved in 3-D VLSI fabrication and the modular nature of 3-D integration, analysis of the design and process tradeoffs for a particular appli-

cation is essential. This work investigates the incorporation of technology analysis into the front end architectural and topological design toolset to enable the high-level analysis and optimization of sensor architectures targeted to a particular set of 3-D VLSI process options. By combining Verilog-AMS behavioral modeling [1] with inference chain based technology extrapolation [11], automated analysis of technology constraints within the context of the circuit design process is implemented.

The intended inputs to this EDA flow are based on the combination of a parameterized Verilog-AMS behavioral model and a set of associated GTX parameters and rules. (The GTX technology extrapolation framework is described in section 2.3.1.) Process data is likewise specified in terms of additional GTX parameters and rules. In a practical setting, a process library may be envisioned that contains a set of available process options, thus allowing for the exploration of optimal process technologies for each wafer tier. Sub-block models are constructed to be parameterized according to the available process technology space. The library of sub-block models may consist of a combination of existing precharacterized reusable intellectual property (IP) and generalized topological models that use empirical performance metrics to estimate the achievable performance of a proposed subsystem. Likewise, it is also feasible to allow for certain sub-block selection operations to be performed using an inference chain method similar to that used for deriving the behavioral parameters.

The intended output files consist of Verilog-AMS cases that are suitable inputs to conventional simulation engines, along with a set of inference chain derived metrics that provide the designer with an initial insight into the system performance even prior to behavioral modeling. After a number of iterations, a proposed system-level floor plan may be derived that suggests the optimum technology for each tier of the 3-D wafer stack, the optimum placement of subblocks within the stack, projected performance and other meaningful information.

The ultimate goal of this work is the development of a tool that can be used in the conceptual phases of a design to analyze the tradeoffs inherent in the available

architectural and process options. This tool may then be used in determination of the most promising approach for a particular application. This information is particularly useful in cost-benefit analyses of proposed systems, and in selection of appropriate wafer processes and circuit architecture for each tier of a potential 3-D integrated die.

1.4 Technical Approach

This work focuses on the development of an EDA solution that combines behavioral modeling with technology extrapolation to facilitate the tradeoff analysis required for architectural design of 3-D imagers. An attempt will be made to build upon much of the existing analog/mixed signal hardware description language (HDL) infrastructure. For that reason, a generalized output interface has been implemented that allows for projected circuit parameters to be used in conjuction with HDL representations in languages such as Verilog-AMS [1]. The technology extrapolation infrastructure of GTX, the MARCO GSRC Technology Extrapolation System [11] is used for inference chain evaluation. GTX is described further in section 2.3.1.

The EDA flow that has been implemented is illustrated in figure 1-3. The requisite "general knowledge" concerning 3-D integration, interconnect schemes, optimization algorithms etc. is represented in the form of GTX parameters and rules. Additional library data provides specific parameters and rules for a particular set of processes and circuits. By supplying process selections, design options, constraints, and application-specific models, a designer may initiate the inference chain implemented as part of the general knowledge, thus evaluating relevant descriptive parameters pertaining to predicted system performance. Some of these parameters may be used by an HDL model builder or SPICE model builder to allow for behavioral modeling. The designer then closes the feedback loop of the design flow by altering inputs based on knowledge gained.

In this work, the focus is on development of the "general knowledge" and "HDL

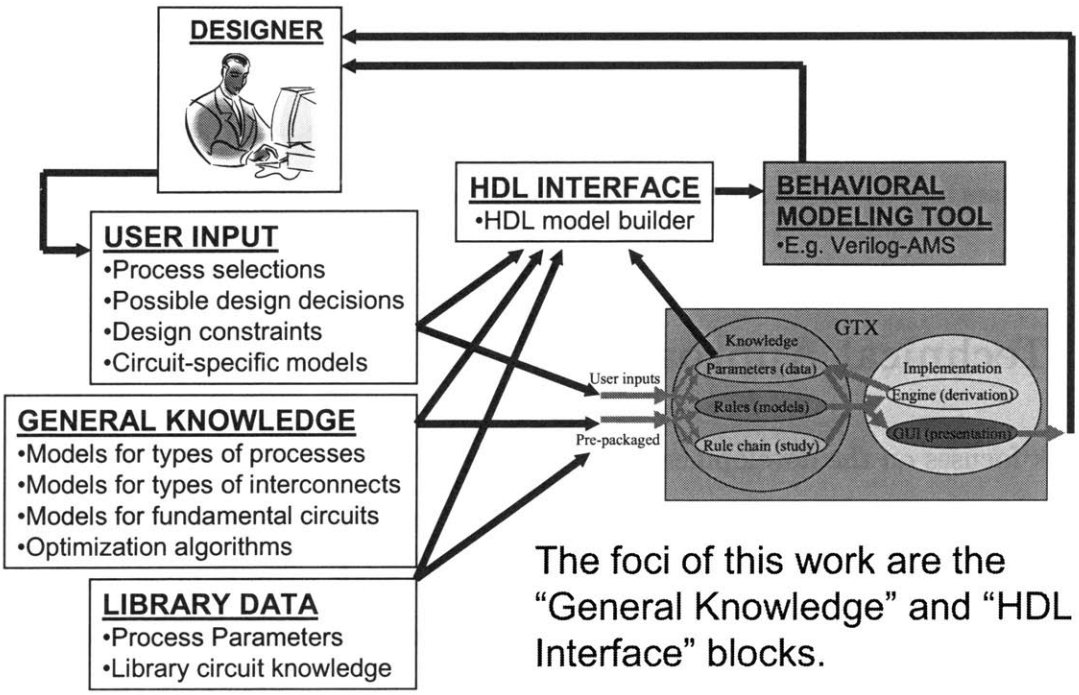

Figure 1-3: EDA flow for conceptual design of 3-D sensors

interface" blocks of the design flow. Particular emphasis is given to the formulation of approximation startegies that allow for rapid projection of circuit performance parameters for various process and implementation configurations. Throughout this work, elements of the design flow are evaluated in two ways. First, assumptions and approximations are validated using accepted modeling techniques. In particular, Synopsys HSPICE and Silvaco Exact2 have been respectively selected as a circuit level simulator and a field solver. The utility of the developed EDA flow is demonstrated through its application to actual design situations. Thus, case studies are included that focus on particular aspects of the proposed flow.

1.5 Nomenclature

1.5.1 Terminology

Since 3-D VLSI technology is still in the early stages of development, much of the applicable terminology is not well established. The following definitions are used throughout this thesis work. In general, the terminology will follow that used in the design guide for the MIT Lincoln Laboratory 3-D FDSOI process.[2]

3-D via	See inter-tier via.
design layer	A set of polygons in a layout database representing a common design purpose. For example, the design layer *METAL1* may define features on the first metal level of a VLSI process.
device plane	A plane of circuit features on levels corresponding to active area, implants, polysilicon gates, etc. which form active devices.
inter-tier via	A via that connects interconnect planes on two tiers.
interconnect plane	A plane of circuit features on metal and via levels.
level	A fabricated set of features formed in a particular lithography step.
masking metal	In a post-bond 3-D via process, a metal layer that serves as both a hard mask in 3-D via formation and as an electrical contact. See section F.1.
mating metal	In a post-bond 3-D via process, the metal layer onto which a 3-D via lands. See section F.1. In a pre-

	bond via process, the metal layer onto which each via from a Cu bond pad lands. See section F.2.
pre-bond 3-D via process	A 3-D integration process in which part of the inter-tier via is formed on each of the wafers prior to bonding. The two surfaces are subsequently mated, forming the interconnected tiers. See section F.2.
post-bond 3-D via process	A 3-D integration process in which inter-tier vias are etched after the wafers corresponding to the connected tiers have been bonded. See section F.1.
stratum	See tier.
tier	A combination of a device plane and one or more proximate interconnect planes. A three-dimensional circuit is formed through the stacking of multiple tiers. In some sources, the term *stratum* is used in a synonomous manner.

In cases where the level of an intra-tier or inter-tier via must be referenced using a single number, it will be referenced using the level on which it lands, i.e. the lowest of the two level numbers. For example, a via connecting the first and second metal levels may be described as "via1." Likewise an inter-tier via connecting tiers two and three may be described as "3Dvia2." This convention will be useful in cases where index numbers must be used to refer to via levels. A similar situation arises in referencing an alignment step, particularly for the case of tier-to-tier alignment. In this case the index number of the referenced alignment will be the lowest tier number. Thus for an alignment of the second and third tiers of a 3-D stack, the alignment will be described as "number 2."

Discussion of the GTX inference-chain based technology extrapolation system also requires a few definitions [11]:

implementation	For purposes of discussion of the GTX system, "implementation" refers to the derivation engine and the user interface.
knowledge	For purposes of discussion of the GTX system, "knowledge" refers to parameters, rules, and rule chains.
parameter	Parameters contain data. Each has a particular name, data type, and associated units.
rule	A rule is a potential inference between parameters, i.e. a model.
rule chain	A rule chain is a study in which a set of rules is used to obtain a particular result.

1.5.2 Typographical Conventions

In this work, software code is presented in `courier` font. Numerous languages are used, including Perl, Verilog-AMS, SPICE and GTX. In cases where the purpose of the included code is to demonstrate syntactic details, the following additional conventions are used:

1. Lower case words are used to denote syntactic categories. These may contain underscores. Examples include: `expression`, `list_of_arguments`.

2. Bold face words are used to denote reserved keywords, operators, and syntacticaly required punctuation. Examples include: **module**, **parameter**.

3. A vertical bar is used to separate alternative items.

4. Square brackets enclose optional items, except when printed in bold, in which case they represent themselves.

5. Braces enclose a repeated item unless they apear in bold, in which case they stand for themselves. Such a repeated item may appear zero or more times, with the repetitions occuring from left to right.

6. If the name of a category starts with an *italicized* ("slanted" for TeX purists) part, it is equivalent to the category name without the italicized part. However, the italicized part conveys some additional semantic information. For example *node*_`identifier` is an `identifier` used to identify a node.

7. When GTX rules and parameters are referenced, they are presented in `courier` font, with *slanted* text representing a numeric index such as a tier number. Tokens used in string parameters are also presented in `courier`.

In addition to these conventions, the languages used in this work each have their own conventions for variable names, etc. These are documented in the various reference works for these particular languages.

Chapter 2

Essential Background

By its very nature, EDA software stitches together the somewhat independent developments in fabrication technology and circuit design. Hence, it is necessary to include some background on each. First, two example 3-D integration processes are discussed. Following this is a discussion of image sensor architectures, from the optical system and analog front end through the subsequent A/D conversion and digital signal processing. Lastly, the state-of-the-art EDA capabilities on which this work is based, including the GTX framework and Verilog-AMS are reviewed and discussed.

2.1 3-D Fabrication Technologies

Two 3-D integration processes are considered in this work. These are discussed in detail in appendix F. Both processes are based on wafer bonding of silicon-on-insulator (SOI) tiers. In the pre-bond method, illustrated in figure 2-1(a), inter-tier vias are formed prior to wafer bonding. The inter-tier interconnect is formed by mated copper bondpads. This pre-bond method is under development at the Microsystem Technology Laboratories (MTL) at MIT[19, 36, 37]. In the post-bond method, illustrated in figure 2-1(b), inter-tier vias are formed after wafer bonding. In this case, one tier must be completely etched through to form the 3-D via. This method was devel-

oped at MIT Lincoln Laboratory (MITLL)[9, 2] and is presently available for circuit prototyping.

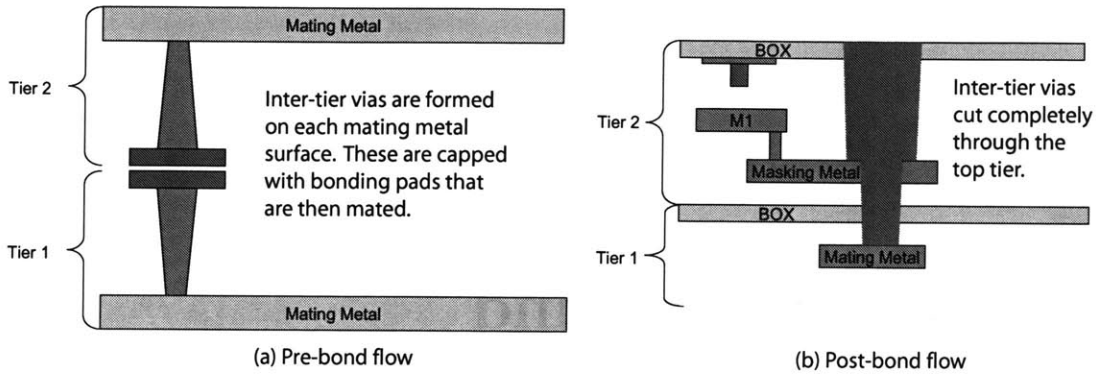

Figure 2-1: Two 3-D integration methods

The method of 3-D via formation has a direct effect on the available area on each tier for pixel implementation. See appendix F for more details.

2.2 Image Sensor Architectures

The essential function of an image sensor is to collect photons over a spatial extent and to convert those photons into a meaningful set of electrical signals. Other types of sensors perform a similar function, i.e. converting physical effects into electrical signals through the use of an array of transducers. Throughout this work, the focus is directed on a subset of image sensors, but much of what is developed may be applied to a wide range of applications.

2.2.1 Focal Plane Array Design Constraints

A focal plane array (FPA) is an array of phototransducers and associated circuitry that is placed at the focal point of the optical axis of an imaging system. FPA designs tend to be targeted toward particular applications, and thus vary widely in architecture. This section presents a brief and partial survey of constraints driving

FPA technology. Further information is widely available in the literature.[18, 27, 31, 45, 51, 52]

Each imaging application requires sensitivity to a particular range of wavelengths. The radiation of interest may range from x-rays to long-wave infrared (LWIR). The transducer must be selected according to its wavelength sensitivity. In addition, the minimum useful pixel pitch is determined by the resolution of the optical system, which is a function of both the lens design and the wavelength. For infrared telescopes, the minimum diameter of the first Airy disk can be found using the following equation:[31, pages 60–63]

$$d = 2.44 \times \lambda_{co} \times (f/\#) \tag{2.1}$$

where λ_{co} is the cut-off wavelength of the optical system and $(f/\#)$ is the effective F/number of the optical system. This is simply a statement of the Rayleigh resolution criterion. Thus for an f/1.8, 8-11-µm LWIR telescope, $d = 2.44 \times 11\mu m \times 1.8 = 48\mu m$. So for this case, the FPA pixel size has a lower limit of about 48 µm. Note that a cost tradeoff exists between the complexity of the optical system and the size of the pixel.

For visible wavelengths (400 nm to 700 nm), a similar tradeoff exists.[48, pages 157–173] However, the complexity of the optical system is often limited by the nature of the targeted applications. For most consumer visible imaging applications, it is generally accepted that the minimum useful pixel pitch is approximately 5 µm.[51]

The photon flux and image contrast also vary with the application. Visible applications may require sensitivity to low light levels, capacity for collection of large photon counts or, in many cases, a compromise between the two. For IR applications, the image is often a relatively low-contrast scene superimposed on a very high background pedestal which also contributes to detector shot noise.[43] The small bandgap of the IR detector materials and the need to minimize electronic noise often makes cryogenic cooling of the IRFPA essential. Thus the analog FPA circuitry must be

designed to operate in a low-temperature ambient.

Many other application-driven constraints are encountered in FPA design.[27] Large-format imagers present new design and process challenges, particularly with respect to circuit yield. In some cases, integrated multi-color focal planes with multiple transducer structures are required. Curved FPAs are useful for wide-field imaging. Some applications require very short integration times or very high frame rates. For low light level imaging, or high temporal resolution imaging, digital photon counting FPA architectures may be preferred.[6]

These application specific constraints determine the pixel modules that must be placed at the front end of the signal processing path. If low-noise spatial transfer of signals or binning of charge is required, charge-coupled devices (CCDs) may be implemented. Fixed pattern and $\frac{1}{f}$ noise may be mitigated through the use of correlated double sampling. For imaging applications where there is a low-contrast image superimposed on a bright pedestal, background subtraction may be required. Both spatial and temporal filters may be included. In some cases, photon counting or ramp-and-fire architectures may be employed.

Given the vast array of applications and architectures, it is necessary that any EDA tool or methodology be sufficiently general to address specific design challenges. Thus, there is a need to generalize the various functional blocks of FPA modules and to construct a framework whereby an imager chip may be represented functionally and structurally in terms of these modules. In this work, the starting point for this representation will utilize the Verilog-AMS hardware description language (HDL).

2.2.2 Analog-to-Digital Converter Technologies

Recall from figure 1-2 that the signal flow path includes analog processing, ADC, and digital processing. In any performance projection, the power consumption for each of these must be considered. For the case of the analog processing, the circuit design is strongly dependent on the application. The digital processing, while also application

dependent, is usually easily understood through conventional digital VLSI considerations. It is the ADC stage of the signal flow that in many cases will provide the most opportunities and challenges during the conceptual phase of a highly-integrated FPA design.

For a given technology, ADC power dissipation is empirically proportional to both the sampling rate and resolution. Recognizing this in 1993, the ISSCC Program Committee suggested a performance metric to normalize out these quantities.[22] This was further revised to take into consideration the difference between the stated number of bits, i.e. the number of output leads, and the effective number of bits as determined by the signal-to-noise ratio (SNR) and distortion.[47] For high-speed converters, the SNR and distortion typically become degraded as the frequency of operation increases, thus requiring the use of effective resolution bandwidth instead of the sampling rate.[20] However, these high sampling rates are not typically encountered in image sensors, and so the following equation shall be used for determining the ADC quantization energy for use as a performance metric:

$$E_Q = \frac{P_{ADC}}{2^{B_{eff}} \cdot F_s} \tag{2.2}$$

where P_{ADC} is the ADC power dissipation, B_{eff} is the effective number of bits, with SNR and distortion taken into account, and F_s is the sampling rate.

2.3 Foundational EDA Infrastructure

Before developing any new EDA solution, four questions must be answered:

1. What aspects of the problem can be addressed using the existing software infrastructure?

2. What methodologies are needed to apply existing EDA tools to the problem?

3. Are there aspects of the proposed new methodologies that require development of new user interfaces, library formats, scripts, and other EDA aids?

4. Are there aspects of the problem that the existing EDA tools cannot solve adequately or efficiently? That is, must new tools be developed?

There are several existing tools that are promising components of a 3-D sensor architectural design solution. These are described below.

2.3.1 The GTX Framework

The ability to integrate multiple process technologies into a 3-D process opens the door to many new design opportunities. This flexibility comes at a cost. A designer is forced to make process technology decisions near the start of the design cycle, but now many aspects of the process are available as design variables. These decisions relate to questions such as:

1. What is the optimum number of tiers?

2. What process technology should be selected for each tier?

3. How do design rules for inter-tier vias affect the achievable pixel pitch?

4. How does the selection of 3-D via process affect the cost/performance tradeoffs?

All of these questions essentially pertain to the interaction of process technology and design. The MARCO GSRC Technology Extrapolation System (GTX) [11] was developed to answer similar questions regarding the development of the integrated electronics infrastucture itself. GTX serves a parallel function to technology roadmaps, allowing engineers in the many disciplines and industries that are engaged in the evolution of VLSI to understand technology trends. As illustrated in figure 2-2, the GTX architecture aims to separate "knowledge" from "implementation," thus allowing for a wide range of applications.

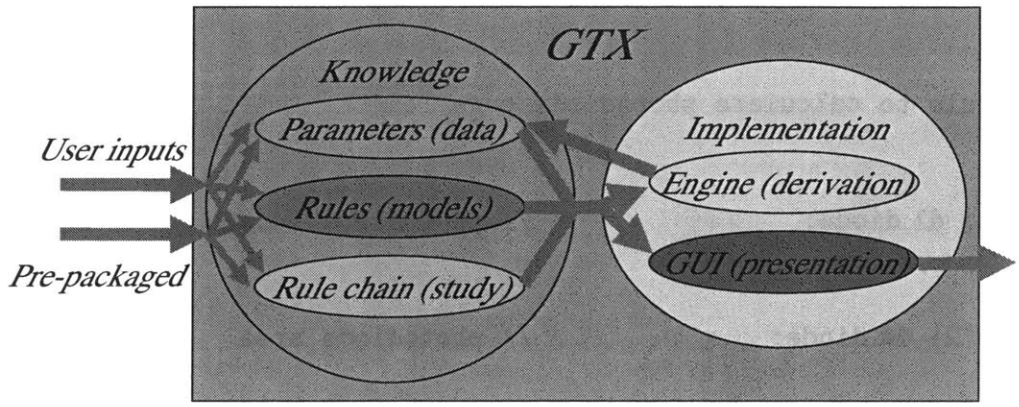

Figure 2-2: Structure of the GTX framework [11]

GTX allows for prediction of achievable design using a high level of abstraction. By designing an appropriate rule chain and applying it to known parameters, a wide range of studies may be performed. Knowledge can be specified using ASCII code, thus allowing for easy implementation of custom investigations. For rules that are too complex for ASCII specification, compiled "code rules" are also possible.

The following simple example illustrates the structure of GTX. Suppose an application requires a photodiode of a particular area. Suppose also that an estimate of the length of a photodiode edge is needed to project the required pixel pitch. We might define a parameter for that photodiode edge length as follows:

```
#parameter dl_diode
#type double
#units {m}
#default
   1e-6
#description
   photodiode length
#endparameter
```

The GTX engine might use a rule such as the following to determine `dl_diode`.

```
#rule RULE_dl_diode
```

```
#description
   example rule to calculate photodiode edge length
#output
   double {m} dl_diode;                    // photodiode length
#inputs
   double {m^2} dA_diode;                  // photodiode area
#body
   sqrt{dA_diode}
#endrule
```

Because rules must be explicitly derived for each calculated parameter, this tool is not well suited to the problem of circuit design. Rather, it is a useful tool for exploring achievable design when relationsips between parameters are reasonably well understood. For particular subcircuit modules, however, deriving a relationship between technology parameters and achieveable performance is feasible. In addition, when rules may be specified in a sufficiently general way as to allow input parameters to be derived from structural representations of circuits, high-level architectural considerations may be explored. Therefore, it is beneficial to combine this technology extrapolation capability with that of a behavioral modeling tool.

2.3.2 Verilog-AMS Capabilities

The Verilog-AMS HDL[1] is a behavioral language for analog and mixed-signal systems. This HDL allows a designer to create and use high-level behavioral descriptions and structural descriptions of systems and components. Modules may be described mathematically in terms of ports and external parameters. Verilog-AMS may be used to describe systems in many disciplines: electrical, mechanical, fluidic, and thermodynamic.

Among the various disciplines for which this HDL is useful, there exist conservative disciplines that specify potential and flow and follow Kirchhoff's flow law, and non-conservative disciplines that only specify a potential nature. One use of only the potential nature is in a signal flow model.

In chapter 6, an interface that allows modeling languages such as Verilog-AMS to be used in conjunction with GTX will be presented.

Chapter 3

GTX Model Development for 3-D VLSI Processes

In this section, several GTX models for 3-D VLSI are presented that will serve as a starting point for more complex design studies. The goal is to investigate the potential of incorporating models such as these into the design flow. In some cases, models are fully implemented and demonstrated, while in other cases, the modeling approach is only proposed. In a practical implementation, most of this process knowledge would be incorporated into process library files, with relevant parameters pertaining to process options available to a designer for studies of circuit implications.

3.1 New GTX Literals and Parameters

In order to promote consistent parameter naming, the GTX documentation outlines a convention for parameter names.[10] Parameter names are composed of "literals" divided by underscores. The allowed literals are grouped in different "literal lists": preposition literals, principal literals, place literals, qualifier literals, adverbial literals, index literals, and unit literals. A parameter has the following form:

```
[preposition]_principal_{[qualifier]_place}_{qualifier}
   _[adverbial]_[index]_[unit]
```

For this work, several new literals are defined. These will be described as necessary when new parameters are discussed. Symbols corresponding to certain parameters are defined for use in describing rules in terms of mathematical equations. These definitions appear in the appropriate parameter listing table.

3.2 Design Rule Modeling

In a conventional circuit design methodology, design rules and process parameters are treated as constant. However, in 3-D VLSI multiple process technologies and multiple process options within a particular technology may be applied to a proposed design. Hence, design rules and process parameters must exist as sets of discrete options.

Design rules for the technologies of interest will constrain the layout of a proposed circuit, making necessary a means of investigating their impact on achievable design. It is particularly important to understand the effect of misalignment on design rules and the requirements for exclusion zones to provide inter-tier vias with adequate clearance. These issues are discussed in detail in sections F.1 and F.2 of the appendix. In this section, implementation of the GTX model and related analysis is presented.

In addition to serving as inputs to models, design rules serve to constrain the exploration space. GTX provides "constraint rules" that ensure that computational resources are not wasted on invalid cases. When a constraint rule is violated, GTX discontinues computation for that particular input parameter set and discards any data obtained for that case. For example, a constraint might be set that particular metal spacings must not violate a minimum metal spacing design rule.

3.2.1 Process Flow Representation

Process flow description parameters provide information on attributes such as the process type, production volume and level order. Some flow parameters that relate to design rule modeling are listed in table 3.1. These parameters tend to be qualitative in nature.

Intra-tier Process Description Parameters

The fabrication process for each individual tier is represented using several parameters. The `k_levels_TX` parameter lists the process levels for tier X. These levels align to the levels listed in `k_alignto_levels_TX`. Inter-tier vias connecting to tier X align to the levels specified by `k_alignto_vias_intertier_TX`. Alignment analysis is described further in sections 3.2.3, 3.2.4 and 3.2.5. Inter-tier vias land on tier X at the levels indicated by `k_3Dvias_stopat_TX`.

3-D Process Flow Type

As outlined in sections F.1 and F.2, there are multiple methods of forming 3-D vias. Each of these cases has a particular method in which certain design rules must be derived. The parameter `k_3Dflowtype` selects the appropriate method for deriving these rules. In this work, two values of `k_3Dflowtype` are supported: `pre-bond` and `post-bond`.

For the case where `k_3Dflowtype=post-bond`, it is assumed that all inter-tier vias cut through the higher-number tier and land on the lower-number tier. For a flow in which a tier may serve as a landing tier for post-bond inter-tier vias from both directions, a new value for `k_3Dflowtype` would be necessary, along with additional parameters to represent the new degree of freedom of via directionality.

Parameter	Type	Interpretation
k_3Dflowtype	string	Selects pre-bond or post-bond 3-D via flow
k_3Dvias_cuttingthru_levels_T*X*	vector <double>	Each element indicates which inter-tier vias cut through the corresponding level in k_levels_T*X* (binary values converted to double for vector storage) 000 if none 100 if *X* only 010 if unmasked part *X-1* only 011 if masked part *X-1* only 110 if *X* and unmasked part *X-1* 111 if *X* and masked part *X-1* (landing via is considered cut through, masking metal is considered cut through by unmasked part)
k_3Dvias_stopat_T*X*	string	Comma-delimited list of levels on which inter-tier vias *X* and *(X-1)* land
k_alignmethod_tiers_orientation_*X*	string	Selects orientation of aligned tiers *X* and *X+1*
k_alignto_levels_T*X*	string	Comma-delimited list of levels to which each level in k_levels_T*X* aligns
k_alignto_vias_intertier_T*X*	string	Comma-delimited list of levels to which intertier vias from above and below align
k_direct_prebond	string	Selects whether pre-bond via bondpads are formed directly on the mating metal
k_level_masking_*X*	string	Masking metal for post-bond inter-tier via *X*
k_levels_T*X*	string	Comma-delimited list of levels on tier *X*
k_volumemodel	string	Selects error propagation method for design rule derivation

Table 3.1: Process flow description parameters

Additional Inter-tier Via Options

For the case where `k_3Dflowtype=pre-bond`, when bondpads are formed on the face side of a tier, they may be formed either directly on the top-level metal, (as is the case for tier 2 in figure F-10) or on top of an overglass layer and connected to the top-level metal by vias. The latter case is similar to that of an inter-tier via making contact to the underside of the first-level metal, except the inter-tier via cuts through the face side of the tier. The `k_direct_prebond` parameter is a boolean that allows for selection between these two cases.

For the case where `k_3Dflowtype=post-bond`, a masking metal level is needed in addition to a mating metal level. This is indicated for inter-tier via X by the parameter `k_level_masking_`X. The masking metal for inter-tier via X is always on tier *(X+1)*.

Orientation of Tiers

For tier-to-tier alignments, the parameter `k_alignmethod_tiers_orientation_`X selects one of four possible cases for the orientation of the aligned tiers: `face-face`, `back-back`, `face-back` and `back-face`. The orientation of the lowest number tier is described by the first token in the string value. The index X in the parameter name corresponds to the number of the lowest tier, which is also equivalent to the inter-tier alignment number.

The `k_alignmethod_tiers_orientation_`X parameter is constrained by the condition that the first token of `k_alignmethod_tiers_orientation_`*(X+1)* cannot be the same as the second token of `k_alignmethod_tiers_orientation_`X. Error checks are implemented to ensure that this constraint is met.

3.2.2 Inter-tier Via Interactions

The rule `DPS3D_k_3Dvias_cuttingthru_levels_TX` uses process description parameters to determine which process levels on each tier potentially interact (geometrically) with inter-tier vias. For each level on tier *X*, three booleans are computed to form a three-digit binary number. The most significant bit (MSB) of binary result, i.e. bit 2, is set to one (1) if inter-tier via *X* cuts through or lands on the space allocated to that particular level. Bit 1 is set to one (1) if inter-tier via *X-1* cuts through or lands on the space allocated to that particular level. For `k_3Dflowtype=post-bond`, bit 0 is one (1) if the masked part of via *X-1* cuts through the level in question. For the masking metal level, bit 0 is set to zero (0), i.e. the level is considered to interact with the wider, unmasked inter-tier via feature. For `k_3Dflowtype=pre-bond`, bit 0 is always zero (0). The three-digit binary result for each level is converted to type `double` so that it may be stored as a `vector<double>`, with indices corresponding to levels in `k_levels_TX`.

3.2.3 Quantitative Process Parameters

Alignment Parameters

Table 3.2 lists quantitative parameters used in calculating intra-tier and inter-tier alignments. Note that the parameters with the prefix "`dspace_error_loc_`" indicate placement errors for certain classes of levels in `k_levels_TX`, with respect to corresponding levels indicated within the `k_alignto_levels_TX` parameter. These have a statistical interpretation determined by `k_volumemodel`. For intra-tier alignments, the `dspace_error_loc_` parameters may be used by the rule `DPS3D_k_aligntol_levels_TX` to generate the parameter `k_aligntol_levels_TX`. The `k_aligntol_levels_TX` parameter is a vector of error vectors corresponding to the placement error of each level listed in `k_levels_TX`. For cases where the `DPS3D_k_aligntol_levels_TX` rule is not adequate, the `k_aligntol_levels_TX` parameter may be directly specified. Error

vectors are discussed in section 3.2.4.

The parameter `k_alignmentmatrix_TX` is a two-dimensional matrix of error vectors relating the relative position errors of all levels in `k_levels_TX`. For both matrix dimensions, index $i = 0$ corresponds to the wafer flat level, and other indices correspond to the level indicated by `k_levels_TX`$[(i-1)]$. This matrix is generated by the rule `DPS3D_alignmentmatrix_TX` which is discussed in section 3.2.5.

Feature Sizes, Spacings, and Thicknesses

Tables 3.3 and 3.4 lists parameters that are used to define relevant feature widths, spacings, and thicknesses. Table 3.5 also contains parameters used in calculating particular derived physical design rules. In many cases, it is useful to derive these physical sizes from drawn sizes based on an additional set of rules. However, for this study, physical sizes are specified directly for the sake of simplicity. Selected parameters are graphically indicated in figures 3-1 and 3-2. The default parameters in table 3.3 are used by rules such as `DPS3D_dw_min_levels_TX` and `DPS3D_dspace_min_levels_TX` to generate design rule parameters such as those listed in table 3.4. In some situations, it is desirable to set the design rule parameters directly, in which case such rules are not applied. Derivation of inter-tier via design rules is discussed in section 3.2.6.

3.2.4 Error Propagation and Production Volume

Design rules for the back end of semiconductor processes largely have their basis in the propagation of process error parameters. As levels are subsequently fabricated, process errors that affect alignment of features propagate. In addition to this propagating component of placement error, there is an additional registration component of feature placement error that relates the alignment feature positions on a particular level to other feature positions on the same level. Although registration error is properly modeled as level-dependent, for the purposes of this study, this intra-level error component is included in the value of a number of global "margin" parameters,

Parameter	Type	Symbol	Units	Interpretation
dspace_error_loc _APall_TX	vector <double>	$\epsilon_{apALL:tX}$	m	Placement error vector for each active and polysilicon (gate) level on tier X
dspace_error_loc _bondpad_TX	vector <double>	$\epsilon_{bp:tX}$	m	Placement error vector for each bondpad formed on tier X (pre-bond process)
dspace_error_loc _intertier_X	vector <double>	$\epsilon_{tiers:3DvX}$	m	Tier-tier alignment error vector for tiers X and $X+1$
dspace_error_loc _Mall_TX	vector <double>	$\epsilon_{mALL:tX}$	m	Placement error vector for each metal level on tier X
dspace_error_loc _toflat_default	vector <double>	$\epsilon_{toflat:default}$	m	Default alignment of first level to wafer flat
dspace_error_loc _Vall_TX	vector <double>	$\epsilon_{vALL:tX}$	m	Placement error vector for each via level on tier X
dspace_error_loc _via_intertier _all	vector <double>	ϵ_{3DvALL}	m	Placement error vector for inter-tier vias
k_alignmentmatrix _TX	vector <vector <vector <double>>>	$\boldsymbol{k}_{alignmatrix:tX}$	m	Matrix of error vectors relating all levels in k_levels_TX to each other. For both matrix dimensions, index $i = 0$ is for the wafer flat level, other indices are for k_levels_TX[$i-1$]
k_aligntol _levels_TX	vector <vector <double>>	$\boldsymbol{k}_{aligntol:tX}$	m	Vector of error vectors indicating the placement error of each level in k_levels_TX relative to the corresponding level in k_alignto_levels_TX

Table 3.2: Process parameters relating to alignment

Parameter	Type	Symbol	Units	Interpretation
df_min_via_default	double	$x_{min:V:default}$	m	Default minimum via or contact size
df_phys_bondpad_via_intertier_default	double	$x_{bp:3dv:default}$	m	Default bondpad size for pre-bond inter-tier vias
df_phys_via_intertier_default	double	$x_{3dv:default}$	m	Default inter-tier via size
dspace_min_active_default	double	$d_{min:A:default}$	m	Default minimum active spacing
dspace_min_metal_default	double	$d_{min:M:default}$	m	Default minimum metal spacing
dspace_min_poly_default	double	$d_{min:P:default}$	m	Default minimum poly spacing
dspace_min_via_default	double	$d_{min:V:default}$	m	Default minimum via spacing
dw_min_active_default	double	$x_{min:A:default}$	m	Default minimum active width
dw_min_poly_default	double	$x_{min:P:default}$	m	Default minimum poly width
dw_min_metal_default	double	$x_{min:M:default}$	m	Default minimum metal width

Table 3.3: Process parameters relating to default widths and spacings for certain classes of levels

Parameter	Type	Symbol	Units	Interpretation
df_min_bondpad _via_intertier_X	double	$x_{min:bp:3DvX}$	m	Minimum bondpad size for inter-tier via X (pre-bond process)
df_phys_bondpad _via_intertier_X	double	$x_{bp:3DvX}$	m	Physical bondpad size for inter-tier via X (pre-bond process)
df_phys_via _intertier_X	vector <double>	$\{x_{3DvXa}, x_{3DvXb}\}$	m	Inter-tier via X physical feature sizes (For pre-bond vias, first element is for via from bondpad to landing on tier $X+1$, second element is for via from bondpad to landing on tier X. For post-bond, first element is litho-defined via cut, second element is physical cut in masking metal.)
dspace_min_bondpad _via_intertier_X	double	$d_{bp:3DvX}$	m	Minimum spacing for bondpads of inter-tier via X
dspace_min_levels_TX	vector <double>	$d_{min:tX}$	m	Vector of minimum spacings for corresponding levels in k_levels_TX
dt_box_TX	double	$x_{box:tX}$	m	Buried oxide thickness on tier X
dt_overglass_TX	double	$x_{overglass:tX}$	m	Overglass thickness on tier X
dw_min_levels_TX	vector <double>	$x_{min:tX}$	m	Vector of minimum widths for corresponding levels in k_levels_TX
loc_levels_TX	vector <vector <double>>	$z_{levels:tX}$	m	Vector of pairs indicating relative z location of top and bottom of each level in k_levels_TX

Table 3.4: Process parameters representing feature widths, spacings, and thicknesses

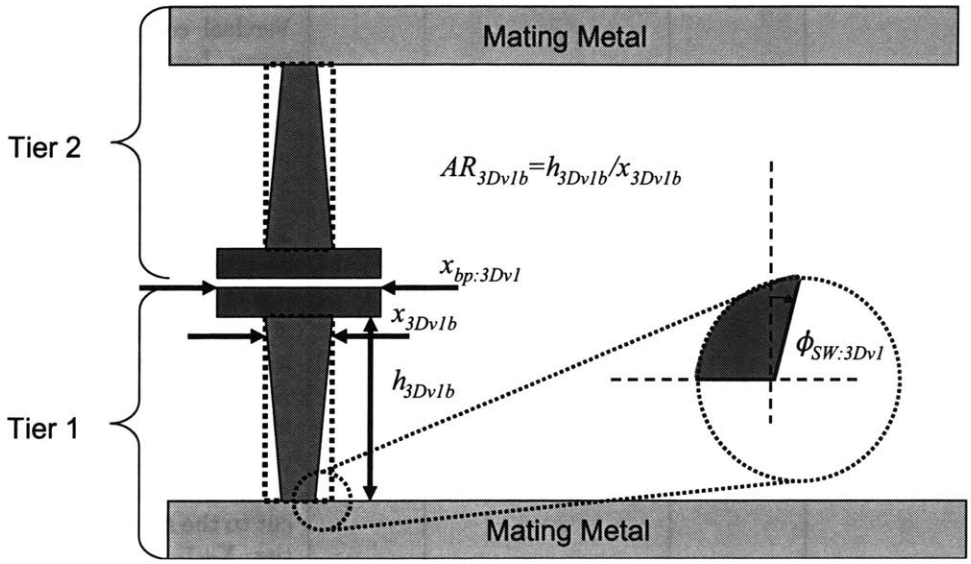

Figure 3-1: Illustration of dimensional parameters for pre-bond inter-tier via implementation

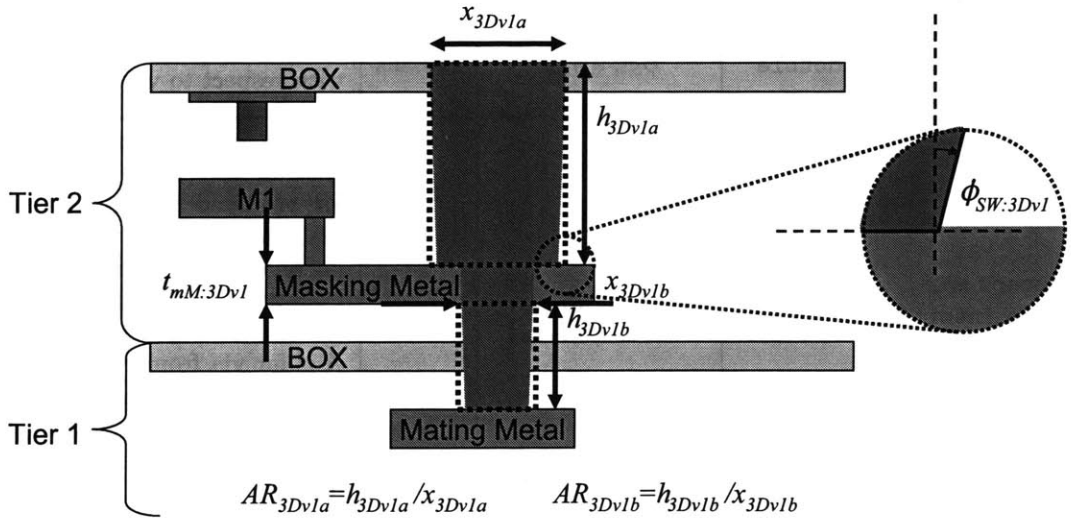

Figure 3-2: Illustration of dimensional parameters for post-bond inter-tier via implementation

Parameter	Type	Symbol	Units	Interpretation
dA_min_landed _bondpad	double	$A_{min:landed:bp}$	m^2	Minimum area of Cu bondpad that must be landed
dh_level_TX _towidest_Y	vector <double>	$z_{levels:tX:open3DvY}$	m	Vertical component of distance between each level in k_levels_TX and the widest part of the interacting section of inter-tier via Y
dh_via _intertier_X	vector <double>	$\{h_{3DvXa}, h_{3DvXb}\}$	m	Two-element vector $\{A, B\}$ describing height of inter-tier via. (For pre-bond via A =height from bondpad to landing on tier $X+1$ and B =height from bondpad to landing on tier X. For post-bond via, A =height from the litho-defined via cut to the masking metal on tier $X+1$, and B =height from the masking metal to landing on tier X. Masking metal and bondpad thickness is excluded.)
dspace_grid	double	d_{grid}	m	Layout grid snap constraint
k_angle_sidewall _via_intertier _X	double	$\phi_{SW:3DvX}$	radians	Inter-tier via sidewall angle with respect to vertical
k_aspect_max _via_intertier _X	vector <double>	$\{AR_{max:3DvXa}, AR_{max:3DvXb}\}$		A pair of maximum aspect ratios $\{A, B\}$, where for pre-bond case A is for via from bondpad to landing on tier $X+1$, B is for via from bondpad to landing on tier X; for post-bond case A is for via from cut opening down to masking metal, B is for via from masking metal on tier $X+1$ to landing on tier X
k_aspect_phys _via_intertier _X	vector <double>	$\{AR_{3DvXa}, AR_{3DvXb}\}$		A pair of physical aspect ratios A,B, where format is similar to that for k_aspect_max_via _intertier_X

Table 3.5: Additional process parameters relating to physical measurements

Parameter	Type	Symbol	Units	Interpretation
dspace_bpmargin	double	$d_{bpmargin}$	m	Additional bondpad size to be added to that which is determined by contact area considerations (may include additional allowance for certain feature size error and/or registration error)
dspace_minins	double	d_{minins}	m	Minimum required spacing for electrically isolated features (may include additional allowance for certain feature bloats and/or registration error)
dspace_minland	double	$d_{minland}$	m	Minimum landing margin for well-landed vias (may include additional allowance for feature size error and/or registration error)

Table 3.6: margin parameters

which are listed in table 3.6. These include dspace_minins, dspace_minland, and dspace_bpmargin. The model may be extended by making each of these parameters a vector<double> of level-dependent margins. In either case, errors that propagate with level-to-level alignment serve to describe the position of the centroid of the alignment features on a particular level. Then the intra-tier margin parameters describe the position of individual polygons with respect to the alignment centroid.

For the purposes of the parameter set used in this work, alignment error is considered as one component of "propagating feature placement error." Thus, alignment error, feature shift due to chemical-mechanical polishing (CMP), etc., are all represented by a single parameter prefixed with "dspace_error_loc_." Each dspace_error_loc_ parameter is of type vector<double>, where each element of the vector corresponds to an error component that is to be treated according to a particular statistical

method. The `k_volumemodel` parameter specifies the interpretation of these vector elements. For the case of inter-tier placement error, the parameter `dspace_error_loc_intertier_X` includes die-to-die feature drift on the landing metal level that is not included in other placement error components.

Each `dspace_error_loc_` vector is represented mathematically by symbols of the form:

$$\epsilon_{levels:location}^{count=N_{contributions}} = \left[\epsilon_{1:levels:location} \quad \epsilon_{2:levels:location} \quad \cdots \quad \epsilon_{N_\epsilon:levels:location} \quad N_{contributions}\right] \tag{3.1}$$

where $N_{contributions}$ is used to indicate that there are multiple identical contributions of the first N_ϵ elements of $\epsilon_{levels:location}$ to any subsequent computation. The subscript "*location*" may refer to either an inter-tier via or a tier. Note that in the symbol $\epsilon_{levels:location}^{count=N_{contributions}}$, "$count = N_{contributions}$" is a superscript, not an exponent. The `k_volumemodel` parameter is represented as a string of $N_\epsilon+1$ comma-delimited tokens, each token corresponding to an element of the `dspace_error_loc_` vector, which also must always be of length $N_\epsilon + 1$, where N_ϵ is a global constant. The $N_\epsilon + 1$ token of `k_volumemodel` must be set to "`count`" because the $N_\epsilon + 1$ element of each `dspace_error_loc_` vector must contain a local value for $N_{contributions}$. The units for error vector parameters are specified as "`{m}`" because the data values (as opposed to the dimensionless count values) are to be specified in meters.

Tokens 1 through N_ϵ of the `k_volumemodel` parameter allow the error propagation method to be selected based on the production volume. Net errors are derived using a call to a net error function netErr(). This has been implemented in Perl, and is called by an **extern_file** call within a GTX rule `DPS3D_netError()`, or by a Perl subroutine call within another GTX rule that uses an **extern_file** call. The command line executable for the netErr() function is:

perl netErr.pl -volumemodel `k_volumemodel`
-alignmethod `k_alignmethod_`*applicable*

-errorlist `list_of_error_vectors`

Case I: `k_alignmethod` = "`incremental`"

When `k_alignmethod` = "`incremental`," it is assumed that each error vector represents one in a series of alignment steps, with each level aligned to the previous level. Let $k_{i:volmod}$ be the i^{th} token of `k_volumemodel`, i.e. the i^{th} element of k_{volmod}. Thus in the present implementation:

$$k_{i:volmod} \in \{\text{"}\texttt{range}\text{"},\ \text{"}\texttt{threesigma}\text{"}\} \text{ for } i \in \{1, 2, ..., N_\epsilon\} \tag{3.2}$$

Let $\boldsymbol{L}$ be a vector of N_{errors} error vectors of the form $\epsilon_{levels:location}$. That is, each row corresponds to an error vector $\epsilon_{levels:location}$ as follows.

$$\boldsymbol{L} = \begin{bmatrix} \epsilon_{1,1} & \epsilon_{2,1} & \cdots & \epsilon_{N_\epsilon,1} & N_{1:contributions} \\ \epsilon_{1,2} & \epsilon_{2,2} & \cdots & \epsilon_{N_\epsilon,2} & N_{2:contributions} \\ \vdots & \vdots & & \vdots & \vdots \\ \epsilon_{1,N_{errors}} & \epsilon_{2,N_{errors}} & \cdots & \epsilon_{N_\epsilon,N_{errors}} & N_{N_{errors}:contributions} \end{bmatrix} \tag{3.3}$$

Then netErr() returns the sum of the net propagated `range` and `threesigma` errors:

$$\text{netErr}(k_{volmod}, \boldsymbol{L}) = \sum_{\substack{i=0 \\ k_{i:volmod} \\ =\texttt{range}}}^{N_\epsilon} \sum_{j=1}^{N_{errors}} (N_{j:contributions} \times \epsilon_{i,j}) + 3 \sqrt{\sum_{\substack{i=0 \\ k_{i:volmod} \\ =\texttt{threesigma}}}^{N_\epsilon} \sum_{j=1}^{N_{errors}} N_{j:contributions} \times \left(\frac{\epsilon_{i,j}}{3}\right)^2} \tag{3.4}$$

When the `range` volume model token is used, the corresponding error contributions are given as "worst-case" values and simply add. The `range` model is especially useful for low-volume prototyping situations in which available statistics are limited.

However, in developing design rules, a tradeoff exists between yield and performance. For high-volume cases, the overall yield tolerance should be considered if performance is to be maximized. Although many different yield constraints are required in practice, for this work it will be considered sufficient to specify a three-sigma control limit on a particular error contribution. This case is indicated when `k_volumemodel` is set to `threesigma`. Adding variable multipliers for sigma is a trivial extension.

The vector representation of error contributions allow for a case such as the following. Suppose alignment error contains both worst-case systematic misalignment and an additional random error component. Then the net misalignment may be calculated using a hybrid of the `range` and `threesigma` methods. This requires representation of each error contribution as a pair, i.e. `{systematic_part,random_part}`. It is important to note that a source of error considered random for one situation may look systematic for another. For example, when the number of lots to be produced is very small, if an error contribution represents a lot-to-lot random variation then this must be treated as a worst-case systematic error. However, for high-volume production, this may be suitably represented as a random variation.

Case II: `k_alignmethod` = "`global`"

When `k_alignmethod` = "`global`," it is assumed that a number of levels are aligned to the same global alignment reference level. In this case, the netErr() function operates

differently, returning only a worst-case value:

$$\text{netErr}(\boldsymbol{k}_{volmod}, \boldsymbol{L}) = \max \left[\begin{array}{l} \sum_{\substack{i=0 \\ k_{i:volmod}=\texttt{range}}}^{N_\epsilon} \epsilon_{i,1} + 3 \sqrt{\sum_{\substack{i=0 \\ k_{i:volmod}=\texttt{threesigma}}}^{N_\epsilon} \left(\frac{\epsilon_{i,1}}{3}\right)^2} \\ \sum_{\substack{i=0 \\ k_{i:volmod}=\texttt{range}}}^{N_\epsilon} \epsilon_{i,2} + 3 \sqrt{\sum_{\substack{i=0 \\ k_{i:volmod}=\texttt{threesigma}}}^{N_\epsilon} \left(\frac{\epsilon_{i,2}}{3}\right)^2} \\ \vdots \\ \sum_{\substack{i=0 \\ k_{i:volmod}=\texttt{range}}}^{N_\epsilon} \epsilon_{i,N_{errors}} + 3 \sqrt{\sum_{\substack{i=0 \\ k_{i:volmod}=\texttt{threesigma}}}^{N_\epsilon} \left(\frac{\epsilon_{i,N_{errors}}}{3}\right)^2} \end{array} \right] \tag{3.5}$$

3.2.5 Alignment Methods

In a process flow, level alignment may be represented as an "alignment tree" structure, as illustrated in figure 3-3. The alignment tree for the levels within a given tier is constructed using the information in `k_alignto_levels_T`X. The alignment of the inter-tier vias interacting with tier X is described by `k_alignto_vias_intertier_T`X. Note that the levels to which inter-tier vias align is not necessarily the same as the levels indicated by `k_3Dvias_stopat_T`X.

The rule `DPS3D_alignmentmatrix_T`X uses the alignment tree information to generate an alignment matrix `k_alignmentmatrix_T`X for each tier by partially applying the netErr() function to the applicable error vectors. In this case, the outer sum of the incremental `k_alignmentmatrix_T`X is not performed, so that the result remains in the form of an error vector. This partial netErr() function has been given the

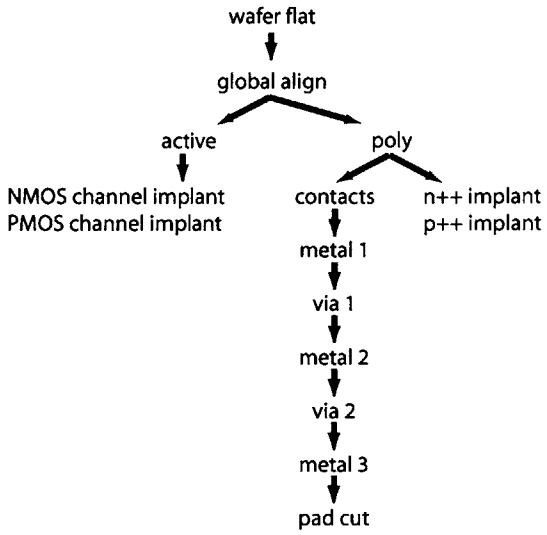

Figure 3-3: Example of alignment tree

operator name netErrVec(). Thus,

$$(\text{netErrVec}(k_{volmod}, \boldsymbol{L}))\,[i] = \begin{cases} \sum_{j=1}^{N_{errors}} (N_{j:contributions} \times \epsilon_{i,j}) & k_{i:volmod} = \texttt{range} \\ 3\sqrt{\sum_{j=1}^{N_{errors}} N_{j:contributions} \times \left(\frac{\epsilon_{i,j}}{3}\right)^2} & k_{i:volmod} = \texttt{threesigma} \end{cases} \tag{3.6}$$

By definition, the count element for the result is always equal to one.

$$(\text{netErrVec}(k_{volmod}, \boldsymbol{L}))\,[N_\epsilon] = 1 \tag{3.7}$$

For example, given the alignment tree in figure 3-3, the error vector relating

metal 1 and active is given by:

$$\epsilon_{(active,metal1)} = \text{netErrVec}\left(\boldsymbol{k}_{volmod}, \begin{bmatrix} \epsilon_{(active,globalalign)} \\ \epsilon_{(poly,globalalign)} \\ \epsilon_{(contacts,poly)} \\ \epsilon_{(metal1,contacts)} \end{bmatrix}\right) \tag{3.8}$$

3.2.6 Calculation of Inter-tier Via Design Rules

Inter-tier Via Height Parameters

The inter-tier via width and spacing rules are derived based on fundamental process parameters. The height measurements $\{h_{3DvXa}, h_{3DvXb}\}$ of inter-tier via X are calculated using the `DPS3D_dh_via_intertier_`X rule. The result is stored as parameter `dh_via_intertier_`X. The `DPS3D_dh_via_intertier_`X rule is implemented using the Perl script **calc3DViaHeight.pl**, which uses orientation and level thickness information to determine `dh_via_intertier_`X.

Physical Inter-tier Via Size Parameters

Aspect ratio and alignment considerations are then used to determine inter-tier via size. The `DPS3D_df_phys_via_intertier_`X rule includes an **exclude_file** call to **calcPhys3DViaSize.pl**. For the case where `k_3Dflowtype=pre-bond`:

$$x_{3DvXa} = \begin{cases} x_{bp:3DvX} & \texttt{k\_direct\_prebond=true} \\ & \text{and tier } X{+}1 \text{ is face down} \\ \max\left(\frac{h_{3DvXa}}{AR_{max:3DvXa}}, x_{3Dv:default}\right) & \text{otherwise} \end{cases} \tag{3.9}$$

$$x_{3DvXb} = \begin{cases} x_{bp:3DvX} & \texttt{k\_direct\_prebond=true} \\ & \text{and tier } X \text{ is face up} \\ \max\left(\frac{h_{3DvXb}}{AR_{max:3DvXb}}, x_{3Dv:default}\right) & \text{otherwise} \end{cases} \tag{3.10}$$

When `k_3Dflowtype=post-bond`, and the inter-tier via aligns to *alignto_level*:

$$x_{3DvXb} = \max\left(\frac{h_{3DvXb}}{AR_{max:3DvXb}}, x_{3Dv:default}\right) \tag{3.11}$$

$$\begin{aligned} x_{3DvXa:\boldsymbol{SUB1}} &= x_{3DvXb} \\ &+ 2\,\mathrm{netErr}\left(\boldsymbol{k}_{volmod}, \begin{bmatrix} \boldsymbol{k}_{alignmatrix:tX}\,[masking\_level, alignto\_level] \\ \epsilon_{3DvALL} \end{bmatrix}\right) \\ &+ 2d_{minland} + 2h_{3DvXa}\tan\left(\phi_{SW:3DvX}\right) \end{aligned} \tag{3.12}$$

$$x_{3DvXa} = \max\left(\frac{h_{3DvXa}}{AR_{max:3DvXa}}, x_{3DvXa:\boldsymbol{SUB1}}\right) \tag{3.13}$$

Once the physical inter-tier via size is known, the physical aspect ratios $\{AR_{3DvXa}, AR_{3DvXb}\}$ are readily determined.

Physical Bondpad Size

The rule `DPS3D_df_phys_bondpad_via_intertier_X` uses $A_{min:landed:bp}$ to determine the minimum bondpad size for pre-bond vias. The bondpads of inter-tier via X are assumed to be circles of radius $r_1 = r_2 = \frac{x_{bp:3DvX}}{2}$ with centers separated by $d = \mathrm{netErr}\left(\boldsymbol{k}_{volmod}, \left[\epsilon_{tiers:3DvX}\right]\right)$. This is a conservative approximation for the case where bondpads have rounded corners and misalignment may be realized radially in any direction. For cases of partial overlap, let C be the simple closed curve bounding the intersection of the two circles. Then the area of intersection is given by:

$$A = \oint_C x\,dy \tag{3.14}$$

This evaluation of this integral is described in detail in appendix C.

The rule `DPS3D_calcMinRadiusBondpad` uses an iterative finite-difference algorithm to invert the area function derived in apendix C. The built-in tolerance of

this algorithm is 1 nm, and the initial value for the bondpad size is $x_{bp:3Dv:default}$. Let the operator name minBpRad() describe the `DPS3D_calcMinRadiusBondpad` rule. minBpRad() is called by `DPS3D_df_phys_bondpad_via_intertier_`X to evaluate bondpad size parameter `df_phys_bondpad_via_intertier_`X as follows:

$$x_{bp:3DvX} = \max\left(\left(\text{minBpRad}\left(A_{min:landed:bp}, \epsilon_{tiers:3DvX}\right) + d_{bpmargin}\right), x_{min:bp:3DvX}\right) \tag{3.15}$$

Physical Bondpad Spacing

The bondpad spacing `dspace_min_bondpad_via_intertier_`X is determined as follows:

$$d_{bp:3DvX} = d_{minins} + \text{netErr}\left(\boldsymbol{k}_{volmod}, \left[\epsilon_{tiers:3DvX}\right]\right) \tag{3.16}$$

3.2.7 Design Grid

In practice, it is often necessary to snap all polygons to a fixed design grid. This can significantly impact design rules. The parameter `dspace_grid` allows grid snapping to be included in process models. The rule `DPSLIB_snapToGrid` may be used as a called rule to force any parameter to be snapped to the next largest grid position.

3.2.8 Determination of Exclusion Regions

With the requisite parameters defined, it is possible to compute the size of the exclusion regions necessary to accomodate the inter-tier vias. Table 3.7 describes the exclusion region parameters to be determined. The name of the rule to calculate each of these parameters is given by "`DPS3D_`" followed by the name of the parameter to be calculated.

For the following equations, let $\boldsymbol{k}_{cutthru:tX}[level, b]$ be the b-th bit of the item of `k_3Dvias_cuttingthru_levels_T`X corresponding to *level*. Inter-tier vias align to *alignto_level* and stop on *stopat_level*.

Parameter	Type	Symbol	Units	Interpretation
`df_exclude_TX _via_intertier_Y`	`vector <double>`	$\boldsymbol{x}_{excl:tX:3DvY}$	m	Exclusion zones for tier X due to inter-tier via Y, for each level in `k_levels_TX`

Table 3.7: Calculated exclusion region parameters.

Pre-bond Inter-tier Via Flow

When `k_3Dflowtype=pre-bond`, the exclusion zones on tier X due to inter-tier via Y are described by the following equations. If $Y = X$, let $x_{3DvY} = x_{3DvYb}$ and $\boldsymbol{k}'_{cutthru:tX}[level] = \boldsymbol{k}_{cutthru:tX}[level, 2]$; however, if $Y = X - 1$, let $x_{3DvY} = x_{3DvYa}$ and $\boldsymbol{k}'_{cutthru:tX}[level] = \boldsymbol{k}_{cutthru:tX}[level, 1]$. If $k'_{cutthru:tX}[level] = 0$ then:

$$\boldsymbol{x}_{excl:tX:3DvY}\,[level] = 0 \tag{3.17}$$

Otherwise, if $level = stopat\_level$, then let:

$$d_{netErr} = \text{netErr}\left(\boldsymbol{k}_{volmod}, \begin{bmatrix} \epsilon_{3DvALL} \\ \boldsymbol{k}_{alignmatrix:tX}\,[level, alignto\_level] \end{bmatrix}\right) \tag{3.18}$$

If $level = stopat\_level$, then:

$$\begin{aligned} &\boldsymbol{x}_{excl:tX:3DvY}\,[level] \\ &\quad = x_{3DvY} - 2\boldsymbol{z}_{levels:tX:open3DvY}\,[level]\tan\left(\phi_{SW:3DvY}\right) \\ &\qquad + 2\left(d_{minland} + \boldsymbol{d}_{min:tX}\,[level] + d_{netErr}\right) \end{aligned} \tag{3.19}$$

Otherwise,

$$\boldsymbol{x}_{excl:tX:3DvY}[level] = x_{3DvY} - 2\boldsymbol{z}_{levels:tX:open3DvY}[level]\tan(\phi_{SW:3DvY}) + 2(d_{minins} + d_{netErr}) \quad (3.20)$$

Post-bond Inter-tier Via Flow

When `k_3Dflowtype=post-bond`, the exclusion zones on tier X due to inter-tier via Y are described by the following equations.

If $Y = X$, then let $x_{3DvY} = x_{3DvYb}$ and $\boldsymbol{k}'_{cutthru:tX}[level] = \boldsymbol{k}_{cutthru:tX}[level, 2]$. If $Y = X - 1$, then let $x_{3DvY} = x_{3DvYa}$, $\boldsymbol{k}'_{cutthru:tX}[level] = \boldsymbol{k}_{cutthru:tX}[level, 1]$ and $\boldsymbol{k}_{maskedpart:tX}[level] = \boldsymbol{k}_{cutthru:tX}[level, 0]$. Let *masking_level* be the level of the masking metal on tier *(Y+1)*. If $Y = X$, let:

$$d_{netErr} = \text{netErr}\left(\boldsymbol{k}_{volmod}, \begin{bmatrix} \epsilon_{tiers:3DvY} \\ \boldsymbol{k}_{alignmatrix:tX}[level, alignto\_level] \end{bmatrix}\right) \quad (3.21)$$

If $level = stopat\_level$, then:

$$\boldsymbol{x}_{excl:tX:3DvY}[level] = x_{3DvY} - 2\boldsymbol{z}_{levels:tX:open3DvY}[level]\tan(\phi_{SW:3DvY}) + 2(d_{minland} + \boldsymbol{d}_{min:tX}[level] + d_{netErr}) \quad (3.22)$$

Otherwise:

$$\boldsymbol{x}_{excl:tX:3DvY}[level] = x_{3DvY} - 2\boldsymbol{z}_{levels:tX:open3DvY}[level]\tan(\phi_{SW:3DvY}) + 2(d_{minins} + d_{netErr}) \quad (3.23)$$

If $Y = X - 1$ and $\boldsymbol{k}_{maskedpart:tX}[level] = 1$ then let:

$$d_{netErr} = \text{netErr}\left(\boldsymbol{k}_{volmod}, \left[\boldsymbol{k}_{alignmatrix:tX}[level, masking\_level]\right]\right) \quad (3.24)$$

and

$$\boldsymbol{x}_{excl:tX:3DvY}[level] = x_{3DvY} - 2\boldsymbol{z}_{levels:tX:open3DvY}[level]\tan(\phi_{SW:3DvY}) + 2(d_{minins} + d_{netErr}) \quad (3.25)$$

Otherwise, for $\boldsymbol{k}_{maskedpart:tX}[level] = 0$, let:

$$d_{netErr} = \text{netErr}\left(\boldsymbol{k}_{volmod}, \begin{bmatrix} \epsilon_{3DvALL} \\ \boldsymbol{k}_{alignmatrix:tX}[level, alignto\_level] \end{bmatrix}\right) \quad (3.26)$$

If $level = masking\_level$:

$$\boldsymbol{x}_{excl:tX:3DvY}[level] = x_{3DvY} - 2\boldsymbol{z}_{levels:tX:open3DvY}[level]\tan(\phi_{SW:3DvY}) + 2(d_{minland} + \boldsymbol{d}_{min:tX}[level] + d_{netErr}) \quad (3.27)$$

Otherwise:

$$x_{excl:tX:3DvY}[level] = x_{3DvY} - 2z_{levels:tX:open3DvY}[level]\tan(\phi_{SW:3DvY}) + 2(d_{minins} + d_{netErr}) \quad (3.28)$$

3.2.9 Case Study: Active-Dominated vs. Interconnect-Dominated Circuits

Even without extensive circuit knowledge, it is possible to gain insight into design and process integration questions using the process models previously discussed. Consider the following two cases: Suppose we have two tiers, each implemented in a five-metal process. For both cases, let `k_3Dflowtype=post-bond`. Assume all intra-tier levels align incrementally, that `k_volumemodel` has only one component, which may be either `range` or `threesigma`, and that each intra-tier alignment contributes a 100-nm placement error of the type indicated by `k_volumemodel`. Also assume that the inter-tier via size is fixed at 1 µm, and that inter-tier via sidewalls are vertical.

For case 1:

```
k_alignmethod_tiers_orientation_1 = face-face
k_alignto_vias_intertier_T2       = ...,metal1
k_level_masking_1                 = metal1
```

For case 2:

```
k_alignmethod_tiers_orientation_1 = face-back
k_alignto_vias_intertier_T2       = ...,metal5
k_level_masking_1                 = metal5
```

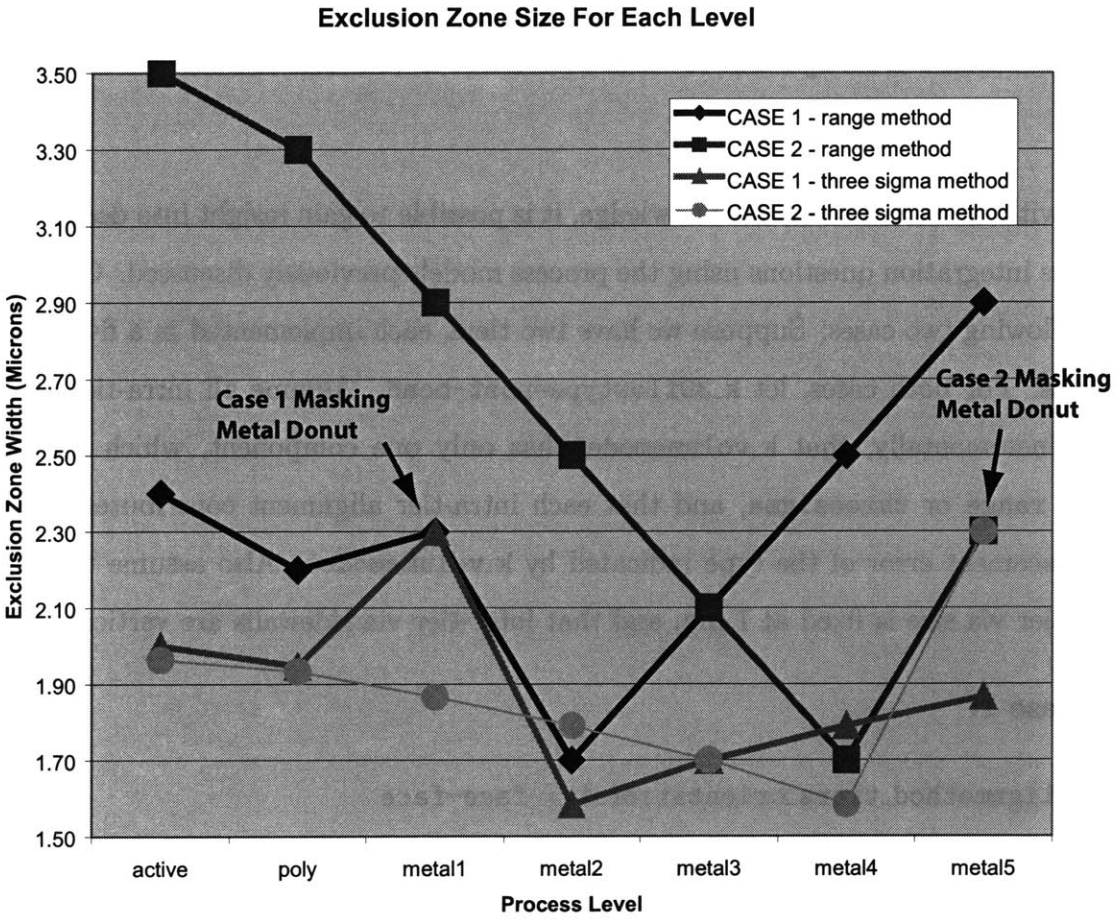

Figure 3-4: Example exclusion zone study

Figure 3-4 shows the resulting tier *(X+1)*=2 exclusion zone width on each level for each case. Curves for both `range` and `threesigma` interpretations of placement error are included. Note that there is a local maximum at the masking metal level due to the masking metal donut. Also, note that if tier *(X+1)* is interconnect-dominated, bond *X* should be `face-back`, otherwise, bond *X* should be `face-face`. For example, if tier *(X+1)* is a reduced-skew clock tree tier [42], then tier *(X+1)* should be face up. On the other hand, if tier *(X+1)* is used for dense logic, then active area is more valuable than higher-level interconnect, and so tier *(X+1)* should be face down.

3.3 Summary

To summarize, table 3.8 summarizes the key GTX rules implemented for the purpose of process modeling. Appendix D includes suggestions for future improvements to these process models.

Rule	Description
DPSLIB_areaIntersectCircles	Determine area of intersection of two arbitrary circles
DPSLIB_interiorAngle	Determine angle in triangle from three known sides using law of cosines
DPSLIB_netError	netErr() function
DPSLIB_snapToGrid	Snap design rule parameter to design grid
DPS3D_calcMinRadiusBondpad	Calculate minimum bondpad radius necessary to meet landing area constraint
DPS3D_df_exclude_TX_via_intertier_Y	Determine exclusion zone on tier X due to inter-tier via Y
DPS3D_df_phys_bondpad_via _intertier_X	Determine physical inter-tier via bondpad size
DPS3D_df_phys_via_intertier_X	Determine physical inter-tier via size
DPS3D_dh_level_TX_towidest_Y	Determine vertical component of distance between level on tier X and opening of inter-tier via Y
DPS3D_dh_via_intertier_X	Determine height parameters for inter-tier via X
DPS3D_dspace_min_bondpad_via _intertier_X	Determine minimum inter-tier via bondpad spacing
DPS3D_dspace_min_levels_TX	Create vector of minimum spacings using default parameters
DPS3D_dw_min_levels_TX	Create vector of minimum widths using default parameters
DPS3D_k_alignmentmatrix_TX	Create matrix of relative placement errors
DPS3D_k_alignto_levels_TX	Use defined masking metal and mating metal parameters to determine inter-tier via alignment levels
DPS3D_k_aligntol_levels_TX	Create alignment tolerance vector using dspace_error_loc parameters
DPS3D_k_aspect_phys_via_intertier_X	Determine physical aspect ratio for inter-tier via
DPS3D_k_3Dvias_cuttingthru_levels_TX	Create vector describing inter-tier via interaction with intra-tier levels

Table 3.8: GTX rules for modeling 3-D processes

Chapter 4

Interconnect Modeling for Conceptual Design

The projection of interconnect performance for sensor architectures is facilitated by the regularity of the design. In this chapter, a modeling approach is described that takes advantage of regularity by dividing wires into three categories. Within a pixel, there is wiring that is local to the pixel subcell; this includes the inter-tier interconnects. The design of the local wiring is a full-custom problem, limiting the projections that can be made from a high level of abstraction. For functionality shared within a neighborhood of pixels, a channel routing scheme will be assumed. For global buses, the wiring is also essentially channel routing, but a particular channel position is reserved for all (or almost all) cells in the array, thus simplifying the analysis. Models for such interconnect schemes are readily available in the literature.[16, 33, 34, 35, 36]

The modeling approach is as follows. First, usages are defined for each interconnect level. Then for levels with global and semi-global wires, signal sets are assigned to the appropriate levels. Performance parameters for these interconnects are then evaluated. The result is the construction of an "interconnect skeleton" onto which a sensor circuit is then built.

During the conceptual design phase, it is necessary to apply various "rules-of-

thumb" to the problem of estimating potential circuit performance. These approximations must not be computationally intensive, and must accommodate a lack of specific circuit information. In this chapter, approximation methods for estimating droop voltage and parasitic capacitance in a non-existent circuit are presented. The validity of the assumptions underlying these models is discussed.

4.1 Level Representation

4.1.1 Ordering of Levels

In the conceptual design phase, the desired orientation of tiers is often unknown. Thus it is desirable to specify level-specific properties in tier-specific vectors such that the vectors are ordered according to `k_levels_T`*X*. However, for determination of circuit performance, it is necessary to consider the proposed integrated stack. Hence, a new list of levels, `k_levels_ordered`, is used. This comma-delimited list of level names is derived using the rule `DPS3D_k_levels_ordered`, which uses information contained in `k_levels_T`*X*, `loc_levels_T`*X* and `k_alignmethod_tiers_orientation_`*Y* to order levels according to the midpoint of their vertical locations. Level names in `k_levels_ordered` are of the form "T*X*:*levelname*". `DPS3D_k_levels_ordered` has in its body an **extern_file** call to **orderLevels.pl**.

Several rules are defined to convert sets of tier-specific vectors into vectors that are consistent with `k_levels_ordered`. These rules utilize the following Perl scripts: **orderStringsByLevel.pl**, **OrderLocationsByLevel.pl**, **orderVectorsByLevel.pl**, and **orderVectorVectorsByLevel.pl**.

4.1.2 Definition of Level Usages

For each tier, a parameter `k_usage_levels_T`*X* is defined. This parameter is a comma-delimited string containing tokens listed in table 4.1.

Usage	Interpretation
`randomlocal`	Local wiring for pixel subcircuit
`randomglobal`	Channel routing for subcircuit that is larger than the pixel size
`randomglobalx`	Channel routing for subcircuit that is larger than the pixel size, running only in horizontal direction
`randomglobaly`	Channel routing for subcircuit that is larger than the pixel size, running only in vertical direction
`gridx`	Grid wiring for global signals to pixel array, running in horizontal direction
`gridy`	Grid wiring for global signals to pixel array, running in vertical direction
`coplanar`	Coplanar waveguide for global signals
`groundplane`	Ground plane
`unused`	Dielectric only

Table 4.1: Level usages in `k_usage_levels_TX`

Because wires on different tiers may be coupled, it is necessary to construct parameters describing the fully-integrated stack. The parameter `k_ordered_usage_levels_interconnect` is a comma-delimited list of ordered interconnect levels, starting at the outer level of tier 1 and ending at the outer level of the highest number tier. Elements are ordered according to `k_levels_ordered`.

The inter-layer dielectric thicknesses between each level listed in `k_levels_ordered` is listed in the `vector<vector<double>>` parameter `dt_interLD_between_levels_ordered`. This parameter is a matrix with both indices arranged according to `k_levels_ordered`.

The `randomlocal` usage may be used to represent either local wiring or local device active areas. Various global usages are defined. For the `randomglobal` usage and its constrained variants, it is useful to apply system-level interconnect models. If levels

with these usages correspond to digital logic, the models of [16, 33, 34, 35, 36] are easily applied in GTX. For analog and mixed signal design, it is most likely the case that these levels correspond to functional blocks with well-understood performance. It is important to understand unwanted signal and thermal coupling which may introduced when such blocks are introduced into a system-level design.

4.1.3 Example of Interconnect Skeleton: Snapshot Mode Imager with Very Short Integration Time

To illustrate how level usages may be applied to the construction of an interconnect skeleton, consider the sensor architecture of figure 4-1. This represents an imager concept developed at MIT Lincoln Laboratory.[26, 42]. In this proposed design, a co-planar waveguide is used to deliver a low-skew shutter signal to the pixel array. Because of the very short integration time, it is critical that skew be minimized if all pixels are to capture data simultaneously. Hence, tier 3 serves as a clock distribution tier.

Based on the analysis presented in 3.2.9, the orientation of tiers has been set to correspond to `k_alignmethod_tiers_orientation_2 = back-back`. This is because tier 2 is active-dominated and tier 3 is interconnect-dominated. In addition to the clock distribution tier, other inputs and outputs are routed on metals 2 and 3 of tier 2 in a grid fashion. These include select signals for the readout multiplexer, supply lines, reset levels, analog output lines, and other global nets.

4.2 Representation of Grid-Type Interconnect

Table 4.2 lists parameters that describe global signals wired through each pixel. The parameter `k_signals` describes known signals on global wires, particularly grid wires. The coordinates of these wires are described by `loc_wires`. Inter-tier via signals are defined in parameter `k_signals_intertier`, and the location of intertier vias is

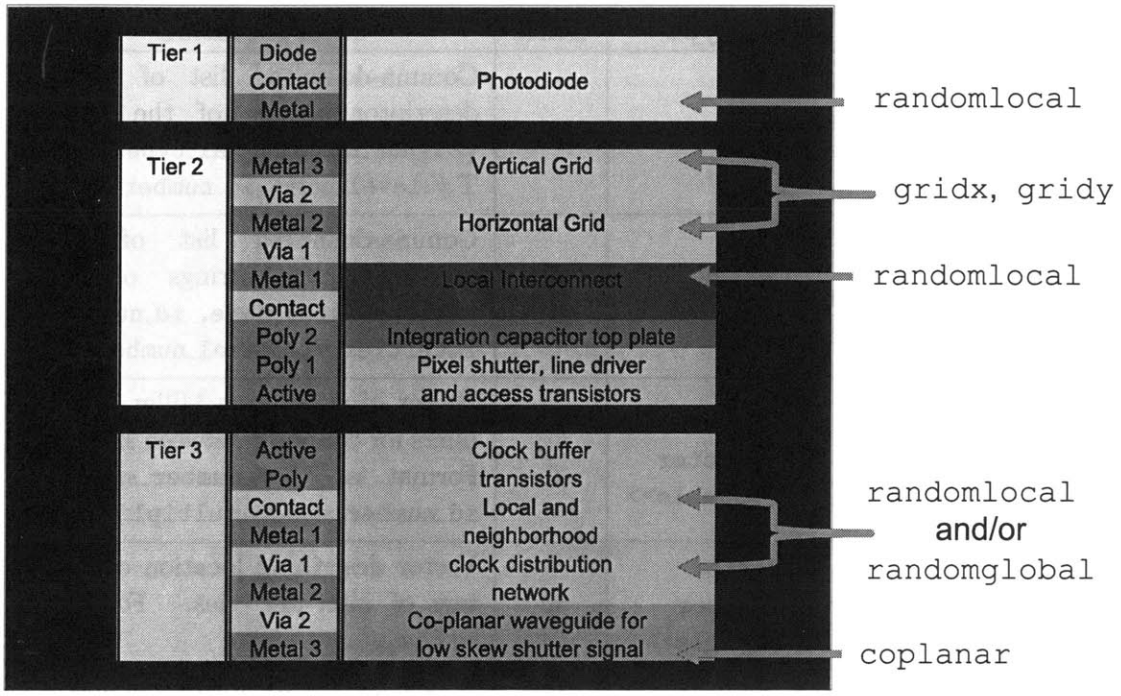

Figure 4-1: Example of interconnect skeleton: snapshot-mode imager with short integration time

specified by `loc_vias_intertier`.

Each signal is defined using an ID number. The base ID number ID is an integer that refers to the electrical net. Because of the single-precision floating point representation of net ID numbers in `k_signals`, ID numbers are restricted to integers from 1 to 9999. (GTX type `double` seems to actually correspond to single-precision floating point. See appendix A for details.) For nets comprised of both horizontal and vertical grid wires (such as mesh supplies), $ID + 0.1$ corresponds to the horizontal wire only and $ID + 0.2$ corresponds to the vertical wire only.

Since imaging arrays tend to have very wide buses (e.g. row selects and column lines), it is useful to exploit this regularity by letting $ID + 0.01$ correspond to an analogous net in adjacent cell (-1,0). Likewise $ID + 0.02$, $ID + 0.03$ and $ID + 0.04$ correspond to analogous nets in adjacent cells (+1,0), (0,-1) and (0,+1), respectively.

ID zero (0) is reserved for generic ground. The generic ground net is used when a

Parameter	Type	Units	Interpretation
k_signals	string		Comma-delimited list of signal descriptor strings of the form: {signal_name,signal_type, T#:level_name ,id_number }
k_signals_intertier	string		Comma-delimited list of signal descriptor strings of the form: {signal_name, id_number, intertier_via_level_number}
k_miller_signals	vector <vector <double>>		Vector of worst-case Miller multipliers for coupling between signals. Format is: {id_number_*signal1*, id_number_*signal2*,multiplier}
loc_vias_intertier	vector <vector <double>>	m	Vector describing location of centers of inter-tier vias. Form is $\{ID,x,y\}$.
loc_wires	vector <vector <double>>	m	Vector describing location of wires. Level of wire is from k_signals. For each signal, a vector of the form $\{ID,x1,x2,y1,y2\}$ is included. For nets with wires running in only the horizontal (vertical) direction, $x1 = x2 = -1$ ($y1 = y2 = -1$).

Table 4.2: Signal parameters

defined ground wire does not have an assigned net number. For defined ground rails with net numbers, the defined ID is used and thus that ground is not generic.

Each signal has a *signal_type* property. Allowed signal types include "supply", "wire_elmore", "wire_slew". "supply" signals are droop-constrained. Signal types "wire_elmore" and "wire_slew" are delay-constrained and slew-constrained wires, respectively.

4.3 Supply Rail Droop Modeling

Even in cases where overall pixel functionality is not affected, supply droop is one possible source of fixed pattern noise in an array sensor. Hence, droop voltage constraints may determine the minimum width of certain interconnect features. Two methods of analyzing the power supply grid have been implemented. In the first method, a direct nodal analysis is performed. Alternatively, the second method uses a superposition of voltage divider solutions.

In this simplified analysis, only the direct current (DC) component of droop is considered. It is assumed that throughout the pixel grid, sufficient decoupling capacitance is implemented to provide a charge reservoir that makes the transient or alternating component of droop a local problem. In this case, average current drawn throughout the grid creates a DC droop across a resistive network.

For each interconnect level, a resistivity ρ_{level} is defined. Resistivity is represented as a `vector<double>` parameter `rho_levels_TX` with units Ω-m. For non-interconnect levels, $\rho_{level} = \boldsymbol{\rho}[level] = -1$. The resistance of a wire on *level* is given by:

$$R = \frac{\rho_{level}\ell}{hw} \tag{4.1}$$

where ℓ, w and h are the length, width, and height of the wire, respectively. Height h is determined from `dt_mod_levels_X`, which is derived from `loc_levels_TX` using rule `DPS3D_dt_mod_levels_X`. This allows for generation of the sheet resistance parameter `r_sheet_levels_TX`, which is a `vector<double>` with symbol $\boldsymbol{R}_{\square}$.

For some levels, such as diffusions in bulk, it is inconvenient to specify a resistivity. These levels generally would also have zero thickness according to `loc_levels_TX`. When `loc_levels_TX` indicates zero thickness, the corresponding element of `dt_mod_levels_TX` is set to a thickness of 1 m instead of zero. This allows the sheet resistance to be directly specified in the appropriate element of the `rho_levels_TX` parameter.

$\boldsymbol{R}_{\square}$ is expressed in units of Ω. When $\boldsymbol{R}_{\square}[level] \leq 0$, a non-conductive *level* is

indicated and the exact value of $R_{\square}$ is meaningless. If $\boldsymbol{h}$ is a vector of wire heights for each level, then for each interconnect level:

$$\boldsymbol{R}_{\square}[level] = \frac{\boldsymbol{\rho}[level]}{\boldsymbol{h}[level]} \tag{4.2}$$

The parameter `r_sheet_levels_interconnect` is a `vector<double>` that combines `r_sheet_levels_TX` for all tiers. The rule `DPS3D_r_sheet_levels_ordered` is used to construct this parameter. Levels are ordered according to `k_levels_ordered`.

Supply rails may run horizontally, vertically, or as a grid. These cases are considered below. Table 4.3 lists some of the parameters used in this analysis.

Parameter	Type	Symbol	Units	Interpretation
`df_pixels_array`	`vector <double>`	$\{x_{pixels,x}, x_{pixels,y}\}$	m	Horizontal and vertical pixel size
`dt_interLD _between_levels _ordered`	`vector <vector <double>>`	t_{ILD}	m	ILD thickness between levels in `k_levels_ordered`
`dt_mod_levels _ordered`	`vector <double>`	$\boldsymbol{h}_{levels}$	m	Thickness of levels, ordered according to `k_levels_ordered` (set to 1 m to indicate zero thickness)
`k_ordered _usage_levels _interconnect`	`string`			Comma-delimited list of ordered interconnect levels, starting at the outer level of tier 1 and ending at the outer level of the highest number tier.
`k_usage_levels_TX`	`string`			Level usages for tier X
`num_pixels_array`	`vector <double>`	$\{N_{pixels,x}, N_{pixels,y}\}$		Number of pixels in horizontal and vertical dimensions
`num_dummy_pixels _array`	`vector <double>`	$\{N_{dummy,x}, N_{dummy,y}\}$		Number of dummy pixels added to each array edge (added to horizontal and vertical dimensions)

Table 4.3: Topological parameters

4.3.1 Nodal Analysis of 1-D Supply Rail

Let $N_{pixels,x}$ be the number of pixels in the horizontal dimension, and $N_{dummy,x}$ be the number of dummy cells padding each side. We can represent the load currents that must be supported by a vector $\boldsymbol{I}_\ell$ of length $N_{pixels,x}$. Let $\boldsymbol{V}[i]$ be the voltage droop at node i. For a resistive line as illustrated in figure 4-2, we can apply Kirchoff's node current law:

$$\boldsymbol{I}_\ell[i] = \frac{(\boldsymbol{V}[i-1] - \boldsymbol{V}[i]) + (\boldsymbol{V}[i+1] - \boldsymbol{V}[i])}{R_{segment}} \tag{4.3}$$

$R_{segment}$ is the resistance of one pixel segment of wire. For the wire corresponding to *signal* on *level*:

$$R_{segment} = \frac{R_\square[level]x_{pixels,x}}{w[signal]} \tag{4.4}$$

Figure 4-2: Resistor line segments for 1-D supply grid modeling

The droop voltage at node i is given by:

$$\boldsymbol{V}[i] = \frac{(\boldsymbol{V}[i-1] + \boldsymbol{V}[i+1]) - \boldsymbol{I}_\ell[i]R_{segment}}{2} \tag{4.5}$$

The boundary condition for the supply line is modeled as if a voltage source were placed at both ends. Since $\boldsymbol{V}[i]$ is defined as droop voltage, this source is ground.

$$\boldsymbol{V}[-N_{dummy,x}] = 0 \tag{4.6}$$

$$\boldsymbol{V}[N_{pixels,x} + N_{dummy,x} - 1] = 0 \tag{4.7}$$

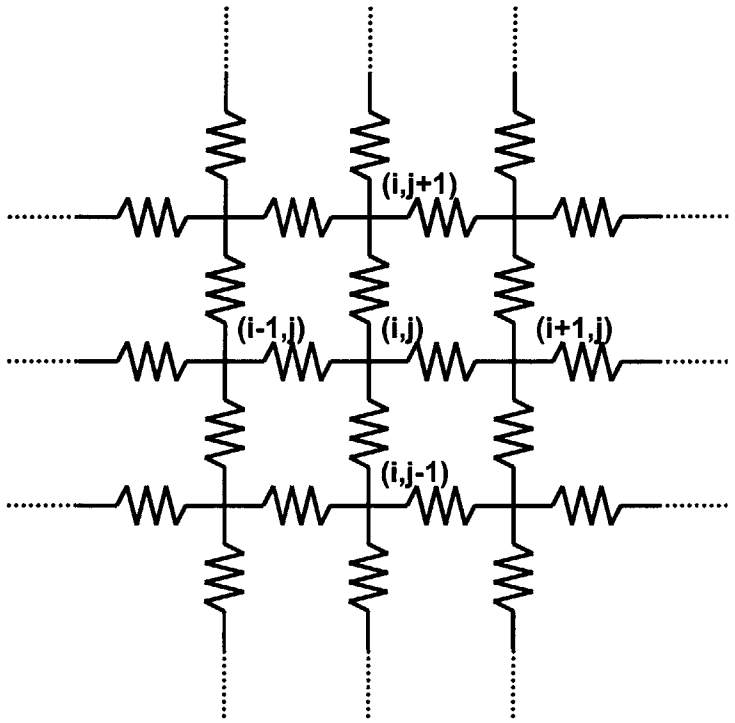

Figure 4-3: Resistor grid for 2-D supply grid modeling

4.3.2 Nodal Analysis of 2-D Supply Rail

Consider the resistive grid shown in figure 4-3. Let $\{N_{pixels,x}, N_{pixels,y}\}$ be the number of pixels in each dimension, and $\{N_{dummy,x}, N_{dummy,y}\}$ be the number of dummy pixels added to the edges in each dimension. We can represent the load currents that must be supported by a matrix $\boldsymbol{I}_\ell$ of dimensions $\{N_{pixels,x}, N_{pixels,y}\}$. Let $\boldsymbol{V}[i,j]$ be the voltage droop at node (i,j). Let $\{R_{segment,x}, R_{segment,y}\}$ be the segment resistance in the horizontal and vertical dimension. For a resistive grid as illustrated in figure 4-3, we can apply Kirchoff's node current law:

$$\boldsymbol{I}_\ell[i,j] = \frac{(\boldsymbol{V}[i-1,j] - \boldsymbol{V}[i,j]) + (\boldsymbol{V}[i+1,j] - \boldsymbol{V}[i,j])}{R_{segment,x}} + \frac{(\boldsymbol{V}[i,j-1] - \boldsymbol{V}[i,j]) + (\boldsymbol{V}[i,j+1] - \boldsymbol{V}[i,j])}{R_{segment,y}} \quad (4.8)$$

$$R_{segment,x}R_{segment,y}\boldsymbol{I}_\ell[i,j] = R_{segment,y}(\boldsymbol{V}[i-1,j]+\boldsymbol{V}[i+1,j]-2\boldsymbol{V}[i,j]) + R_{segment,x}(\boldsymbol{V}[i,j-1]+\boldsymbol{V}[i,j+1]-2\boldsymbol{V}[i,j]) \quad (4.9)$$

$$\boldsymbol{V}[i,j] = \frac{1}{2(R_{segment,x}+R_{segment,y})}\Big(R_{segment,y}(\boldsymbol{V}[i-1,j]+\boldsymbol{V}[i+1,j]) + R_{segment,x}(\boldsymbol{V}[i,j-1]+\boldsymbol{V}[i,j+1]) - R_{segment,x}R_{segment,y}\boldsymbol{I}_\ell[i,j]\Big) \quad (4.10)$$

The boundary condition for the supply grid is modeled as if a voltage source were placed at the border. Since $\boldsymbol{V}[i,j]$ is defined as droop voltage, this source is ground.

$$\begin{aligned}\boldsymbol{V}[-N_{dummy,x},j] &= 0\\ \boldsymbol{V}[(N_{pixels,x}+N_{dummy,x}-1),j] &= 0\end{aligned} \qquad \text{for all } j \quad (4.11)$$

$$\begin{aligned}\boldsymbol{V}[i,-N_{dummy,y}] &= 0\\ \boldsymbol{V}[i,(N_{pixels,y}+N_{dummy,y}-1)] &= 0\end{aligned} \qquad \text{for all } i \quad (4.12)$$

4.3.3 Voltage Divider Superposition Analysis of 1-D Supply Rail

A simpler formulation of the 1-D supply rail nodal analysis may be derived by considering each current source separately. Let N_{I_ℓ} be the number of pixel locations i for which there is a local current source, i.e. $\boldsymbol{I}_\ell[i] \neq 0$. Let i_n be the pixel position of the n-th current source. If the positions of the first and last pixel in the row are $-N_{dummy,x}$ and $N_{pixels,x}+N_{dummy,x}-1$, respectively, and the resistance of one pixel segment of wire is given by $R_{segment}$, then for the boundary conditions

$$\boldsymbol{V}[-N_{dummy,x}] = 0 \quad (4.13)$$

$$\boldsymbol{V}[N_{pixels,x}+N_{dummy,x}-1] = 0 \quad (4.14)$$

The net resistance to the left of the current source is given by

$$R_{left} = R_{segment}\,(i_n + N_{dummy,x}) \tag{4.15}$$

and the net resistance to the right of the current source is given by

$$R_{right} = R_{segment}\,(N_{pixels,x} + N_{dummy,x} - 1 - i_n) \tag{4.16}$$

The voltage at the current source n is given by:

$$\boldsymbol{V_n}[i_n] = -\boldsymbol{I}_\ell[i_n]\,(R_{left} \parallel R_{right}) \tag{4.17}$$

The remaining voltages $\boldsymbol{V_n}[i]$ for any pixel location i may then be determined by a simple linear interpolation. The net droop due to all current sources is:

$$\boldsymbol{V}[i] = \sum_{n=1}^{N_{I_\ell}} \boldsymbol{V_n}[i] \tag{4.18}$$

The result is equivalent to that given by direct nodal analysis.

4.3.4 Voltage Divider Superposition Analysis of 2-D Supply Rail

For the 2-D case, using a superposition of voltage divider solutions is not as straightforward, but a very good approximation of supply droop may be made using an approach that is similar to that used in the case of the 1-D supply rail. Let N_{I_ℓ} be the number of pixel locations (i, j) for which $\boldsymbol{I}_\ell[i, j] \neq 0$. Let (i_n, j_n) be the location of the n-th current source. Let the horizontal positions of the first and last pixel in each row be $(-N_{dummy,x})$ and $(N_{pixels,x} + N_{dummy,x} - 1)$, respectively; and let the vertical positions of the first and last pixel in each column be $(-N_{dummy,y})$ and $(N_{pixels,y} + N_{dummy,y} - 1)$, respectively. Let the resistance of one pixel segment of

horizontal wire be given by $R_{segment,x}$ and let the resistance of one pixel segment of vertical wire be given by $R_{segment,y}$.

Resistance will be expressed in terms of a geometric mean resistance $\bar{R}_G$. Let

$$\bar{R}_G = \sqrt{R_{segment,x} R_{segment,y}} \tag{4.19}$$

$$\alpha = \frac{R_{segment,x}}{\bar{R}_G} = \sqrt{\frac{R_{segment,x}}{R_{segment,y}}} \tag{4.20}$$

$$\therefore \quad \alpha^{-1} = \frac{R_{segment,y}}{\bar{R}_G} = \sqrt{\frac{R_{segment,y}}{R_{segment,x}}} \tag{4.21}$$

Consider a washer-shaped resistor with inside radius r_a and outside radius r_b on a material with sheet resistance $R_\square$. The resistance between the inside edge and outside edge of the washer shape is given by

$$R_{washer} = \int_{r_a}^{r_b} \frac{R_\square}{2\pi r} dr = \frac{R_\square}{2\pi} \ln\left(\frac{r_b}{r_a}\right) \tag{4.22}$$

The resistor described in equation 4.22 will be used to approximate the supply grid surrounding each current source n. The center of the washer has a radius corresponding to a half-pixel. Since $\bar{R}_G$ is in units of Ω/pixel, distances are expressed in pixel counts. The distances from the pixel at (i_n, j_n) to each array edge are:

$$d_{right} = N_{pixels,x} + N_{dummy,x} - 1 - i_n \tag{4.23}$$

$$d_{left} = N_{dummy,x} + i_n \tag{4.24}$$

$$d_{top} = N_{pixels,y} + N_{dummy,y} - 1 - j_n \tag{4.25}$$

$$d_{bottom} = N_{dummy,y} + j_n \tag{4.26}$$

$$r_{pixel} = 0.5 \tag{4.27}$$

Note that in the washer resistor equation 4.22, the resistance is dominated by the

Reimann sum terms corresponding to smaller radii. Assume that the current source n is far enough away from the edge of the array that current flows radially for enough of a distance (i.e. radius) as to dominate the resistance. Then it is possible to define four effective resistors based on quarter washers. For d_{right}, d_{left}, d_{top} and d_{bottom} as the respective distances from the center of pixel (i_n, j_n),

$$R_{right} = \frac{2\alpha\bar{R}_G}{\pi} \ln\left(\frac{d_{right}}{rpixel}\right) + \frac{\bar{R}_G}{2} \tag{4.28}$$

$$R_{left} = \frac{2\alpha\bar{R}_G}{\pi} \ln\left(\frac{d_{left}}{rpixel}\right) + \frac{\bar{R}_G}{2} \tag{4.29}$$

$$R_{top} = \frac{2\bar{R}_G}{\alpha\pi} \ln\left(\frac{d_{top}}{r_{pixel}}\right) + \frac{\bar{R}_G}{2} \tag{4.30}$$

$$R_{bottom} = \frac{2\bar{R}_G}{\alpha\pi} \ln\left(\frac{d_{bottom}}{r_{pixel}}\right) + \frac{\bar{R}_G}{2} \tag{4.31}$$

The voltage at the current source n is given by

$$\boldsymbol{V_n}[i_n, j_n] = -\boldsymbol{I}_\ell[i_n, j_n]\,(R_{left} \parallel R_{right} \parallel R_{top} \parallel R_{bottom}) \tag{4.32}$$

The washer approximation may then be used to estimate the voltage droop at pixel coordinates (u, v) due to current source n. If $u = -N_{pixels,x}$, $v = -N_{pixels,x}$, $u = N_{pixels,x} + N_{dummy,x} - 1$, or $v = N_{pixels,y} + N_{dummy,y} - 1$ then the pixel (u, v) is along the grounded border, and there is zero droop. Otherwise, for $u \neq i_n$ and $v \neq j_n$, $\boldsymbol{V_n}[u, v]$ may be estimated as follows. Define vector $\vec{r}$ as extending from (i_n, j_n) to (u, v). Then $\vec{r}$ is given by

$$\vec{r} = (u - i_n)\ \hat{i} \quad + \quad (v - j_n)\ \hat{j} \tag{4.33}$$

$$= r_x\ \hat{i} \quad + \quad r_y\ \hat{j} \tag{4.34}$$

The effective distance along $\vec{r}$ to the outside edge of the "washer" is modeled using four elipse quarters, one for each quadrant. The semimajor and semiminor axes of

these ellipse quarters correspond to the effective Manhattan distance to ground from (i_n, j_n). For the array border located at a distance of

$$d_{x1} = \begin{cases} d_{right}, & \text{if } r_x > 0 \\ d_{left}, & \text{otherwise} \end{cases} \tag{4.35}$$

$$d_{y1} = \begin{cases} d_{top}, & \text{if } r_y > 0 \\ d_{bottom}, & \text{otherwise} \end{cases} \tag{4.36}$$

ground is considered to either be at the array border, or when the perpendicular resistance dominates the parallel resistance. That is,

$$\frac{\alpha \bar{R}_G d_{x2}}{(d_{top} + d_{bottom})} = \frac{2\bar{R}_G}{\alpha d_{x2}} \frac{d_{top} d_{bottom}}{(d_{top} + d_{bottom})} \tag{4.37}$$

$$\frac{\bar{R}_G d_{y2}}{\alpha(d_{left} + d_{right})} = \frac{2\alpha \bar{R}_G}{d_{y2}} \frac{d_{left} d_{right}}{(d_{left} + d_{right})} \tag{4.38}$$

$$d_{x2} = \alpha^{-1} \sqrt{2 d_{top} d_{bottom}} \tag{4.39}$$

$$d_{y2} = \alpha \sqrt{2 d_{left} d_{right}} \tag{4.40}$$

and d_{x2} and $dy2$ then serve to limit d_x and d_y.

$$d_x = \min\left(\text{ceil}(\alpha^{-1}\sqrt{2 d_{top} d_{bottom}}), d_{x1}\right) \tag{4.41}$$

$$d_y = \min\left(\text{ceil}(\alpha\sqrt{2 d_{left} d_{right}}), d_{y1}\right) \tag{4.42}$$

where ceil() is the ceiling function.

The axes of the ellipse defining the outside edge of the washer correspond to d_x and d_y. Likewise, the inner edge of the "washer" is modeled by forming a circle with radius r_{pixel}. All of these ellipses are centered on (i_n, j_n). Let $\phi = \angle(\vec{r})$. The distance

from (i_n, j_n) to the outer washer edge is given by:

$$r_{outer1} = \frac{d_x d_y}{\sqrt{d_x^2 \sin^2(\phi) + d_y^2 \cos^2(\phi)}} \tag{4.43}$$

Alternatively, an outer edge may be defined that extends into the corners. That is,

$$r_{outer2} = \min\left(\frac{d_x}{\cos\phi}, \frac{d_y}{\sin\phi}\right) \tag{4.44}$$

Using r_{outer2} may be more conservative when α is very far from 1, but the droop maps tend to be more rectangular than elliptical. A compromise may also be made, with r_{outer} defined as a function (such as a geometric mean) of r_{outer1} and r_{outer2}. For the remainder of this discussion, let $r_{outer} = r_{outer1}$.

An adjusted resistance coefficient $\alpha_\angle$ is derived from α and ϕ.

$$\alpha_\angle = \frac{1}{\sqrt{\alpha^{-2} \sin^2(\phi) + \alpha^2 \cos^2(\phi)}} \tag{4.45}$$

Finally, a voltage division factor corresponding to the divider circuit is obtained.

$$k_{div} = \frac{\frac{1}{8} + \frac{\alpha_\angle}{2\pi} \ln\left(\frac{|\vec{r}|}{r_{pixel}}\right)}{\frac{1}{8} + \frac{\alpha_\angle}{2\pi} \ln\left(\frac{r_{outer}}{r_{pixel}}\right)} \tag{4.46}$$

Note that the washer does not extend all the way to the center of the origin pixel. The $\frac{1}{8}$ terms in equation 4.46 correspond to the resistances within the origin pixel. Now the voltage droop may be estimated for any (u,v):

$$\boldsymbol{V_n}[u, v] = \begin{cases} \boldsymbol{V_n}[i_n, j_n]\,(1 - k_{div}), & \text{if same sign as } \boldsymbol{V_n}[i_n, j_n] \\ 0, & \text{otherwise} \end{cases} \tag{4.47}$$

And the net voltage droop due to all current sources is given by:

$$\boldsymbol{V}[u,v] = \sum_{n=1}^{N_{I_\ell}} \boldsymbol{V_n}[u,v] \tag{4.48}$$

4.3.5 Input Parameters for Droop Voltage Modeling

Supply Signal Properties

The parameter `k_signals_supply` describes supply signals. This is a comma-delimited list of supply signal descriptor strings of the form: {*signal_name*, *id_number*, *horiz_level*, *vert_level*} This parameter may be generated using the rule `DPS3D_k_signals_supply`.

Definition of Load Signals

For both 1-D and 2-D droop modeling, a load pattern $\boldsymbol{I_\ell}$ is required. This corresponds to the worst possible load pattern on a given supply. $\boldsymbol{I_\ell}$ is represented as a vector of load matrices, i.e. a `vector<vector<vector<double>>>` with name `I_load_signals_supply_array`. To aid the user in creating this load array, a rule `DPS3D_I_load_signals_supply_array` has been implemented. This rule creates the load pattern types described in the string parameter `k_load_pattern_type`. Parameter `k_load_pattern_type` is a comma-delimited string, with allowable tokens: `horiz_lines`, `vert_lines` and `points`. These tokens are ordered according to parameter `k_signals_supply`.

The rule `DPS3D_I_load_signals_supply_array` creates current sources positioned periodically in the pixel array, based on the input parameters `k_period_loads_array` and `k_offset_loads_array`. These `vector<vector<double>>` inputs are vectors of pairs containing the appropriate properties for the horizontal and vertical directions. The magnitude of each current source on the supplies listed in `k_signals_supply` is given in the `vector<double> I_load_signals_supply_pixel`. For all of these parameters, the outer vector is ordered according to `k_signals_supply`.

Interpretation of Load Patterns

The current source load pattern is determined based on the anticipated circuit operation. In cases such as for "ripple read" line drivers, the load pattern consists of non-zero current sources located in every pixel in a single row at any given time. In this case, power buses typically run perpendicular to the row of current sources, and a 1-D droop analysis is sufficient. In other cases, such as a snapshot-mode imager with both horizontal and vertical multiplexer elements in the pixel, or in the case of a distributed clock signal, the current source representation and power supply network is two-dimensional in nature, requiring a 2-D droop model. As previously mentioned, it is assumed that sufficient decoupling capacitance will be implemented to ensure validity of a quasi-DC approximation.

4.3.6 Droop Voltage Evaluation Using Direct Nodal Analysis

The rule `DPS3D_v_droop_signals_supply_array` is used to compute droop voltages on each supply using the nodal analysis method described in sections 4.3.1 and 4.3.2. This rule takes as inputs the following parameters:

`k_signals_supply`	the supply list
`I_load_signals_supply_array`	the load patterns
`dw_wire_signals_supply`	vector of wire width pairs of the form $\{horiz\_width, vert\_width\}$
`num_pixels_array`	number of pixels in x and y dimensions
`num_dummy_pixels_array`	number of dummy pixels on each border in x and y dimensions
`k_levels_ordered`	the level list
`r_sheet_levels_ordered`	sheet resistances
`df_pixels_array`	horizontal and vertical pixel size
`k_frac_v_simtolerance_droop`	tolerance parameter for numerical method

The rule `DPS3D_v_droop_signals_supply_array` includes a **extern_file** call to **calcDroopMap.pl**. This selects the appropriate droop model (1-D or 2-D) and computes a matrix of droop voltages. The result is reported as parameter `v_droop_signals_supply_array`. In many cases, only the worst-case droop is desired. This is determined using the rule `DPS3D_v_max_droop_supply`. The result is a `vector<double> v_max_droop_supply` that is ordered according to `k_signals_supply`.

By their very nature, the nodal analysis equations in sections 4.3.1 and 4.3.2 are identical to those used in a full SPICE simulation of the supply grid. For this reason, both SPICE simulations and any valid numerical implementation of the nodal analysis equations will converge to the same result.

Let N_{pix} be the total number of pixels under consideration. For the 1-D case in the horizontal direction,

$$N_{pix} = N_{pixels,x} + 2N_{dummy,x} \tag{4.49}$$

For the 2-D case,

$$N_{pix} = (N_{pixels,x} + 2N_{dummy,x}) \times (N_{pixels,y} + 2N_{dummy,y}) \tag{4.50}$$

The voltages $\boldsymbol{V}[\boldsymbol{i}, \boldsymbol{j}]$ are described by N_{pix} equations in N_{pix} unknowns, all forming a sparse matrix. For the 1-D case, the matrix is tridiagonal, save for the boundary conditions. For the 2-D case, the matrix is a banded cube matrix, save for the boundary conditions. Numerous numerical methods exist for solving such sparse matrices, and a detailed investigation of these would be beyond the scope of this work.

For the purposes of the implementation in **calcDroopMap.pl**, a Gauss-Seidel successive overrelaxation method was applied. The number of iterations required for this method is essentially proportional to the maximum separation between pixels, i.e., a change in voltage recorded at any pixel must propagate to all other

pixels. Thus for the 1-D case, the computation time is roughly proportional to N_{pix}^2, and for the 2-D case, the computation time is roughly proportional to $N_{pix}^{1.5}$ (for a square array with equal $R_{segment,x}$ and $R_{segment,y}$). It was verified that the `DPS3D_v_droop_signals_supply_array` results converged to the same values predicted by SPICE.

Other approaches could potentially offer considerable improvements. For example, a SPICE-like Newton-Raphson method could be implemented. Alternatively, a SPICE engine could be directly called by the rule, using a wrapper function written in Perl or some other suitable scripting language. In addition, since Perl does not provide the most efficient matrix manipulation functions, a more optimized implementation of this computation is desirable.

4.3.7 Droop Voltage Evaluation Using Voltage Divider Superposition Analysis

For the projections of supply droop that are required for conceptual design, the voltage divider superposition method described in sections 4.3.3 and 4.3.4 are more interesting. This is because it often is not necessary to calculate the entire droop map since knowledge of the current source distribution may be used to choose interesting observation points. For the 2-D case, since the voltage droop decreases hyperbolically, if current sources are sufficiently sparse, then the global maximum of $|\boldsymbol{V}[u,v]|$ will correspond to a local maximum occurring at the location of a current source. If current sources are somewhat less sparse, then it also is important to look at positions in between to make sure that the sum of several tails does not exceed the local maximum at each (i_n, j_n) =the location of current source n.

Figure 4-4 (a) shows the nodal solution for voltage droop in a 32×32 array with one 0.1 mA source at $(4,4)$, as calculated by Synopsys HSPICE. $R_{segment,x} = R_{segment,y} = 0.64\Omega$ Note that in these simulation results, the pixel location indices of the plot are given by $((u+N_{dummy,x}),(v+N_{dummy,y}))$, with $N_{dummy,x} = N_{dummy,y} = 3$.

The voltage divider approximation is plotted in figure 4-4 (b). The error, expressed as a percentage of the SPICE-calculated peak is plotted in figure 4-4 (c). Positive error corresponds to an overly conservative estimate. Note that in the center of the array, the droop voltage tail is softer in figure 4-4 (b) than in figure 4-4 (a), thus giving rise to positive error. Likewise, there is some negative error near the edge. Both effects are due to the radial currents approximation and the approximations for ground location. The observed result is desirable, since it is important to be overly conservative in computing superposition at the center of the array, even if at the expense of an underestimation near the edge. The model could be potentially improved through the addition of fitting parameters to account for the treatment of a rectangle as an ellipse and the nonuniform current flow.

Figure 4-5 shows the same plots for the case where the current source is located at $(12, 12)$. The error is less than 4% of the voltage peak. This is to be expected since the source is located closer to the center of the array.

The effect of unequal $R_{segment,x}$ and $R_{segment,y}$ is shown in figure 4-6. Here $R_{segment,x}$ is held fixed at 0.64Ω and $R_{segment,y}$ is changed. The voltage divider model successfully predicts the change in ellipse eccentricity. Figure 4-7 is a plot of worst pixel droop versus resistor ratio. There is good agreement for all ratios of $0.1 \leq (\alpha^{-2} = \frac{R_y}{R_x}) \leq 100$. As α^{-2} increases beyond 100, it is important to note that the 2-D droop model begins to aproach the 1-D droop model, with the larger R_y considered open.

When the single current source is replaced by a periodic 4×4 array of 0.1 mA current sources, the droop profile is as illustrated in figure 4-8. The droop peaks are predicted within 10%.

As previously discussed, the advantage to the superposition of dividers method is that the computational complexity scales as $N_{I_\ell} N_{obs}$, where N_{obs} is the number of observation points. For example, if in a 2-D array, it is desirable to observe the droop voltage at every current source location and immediately between every pair

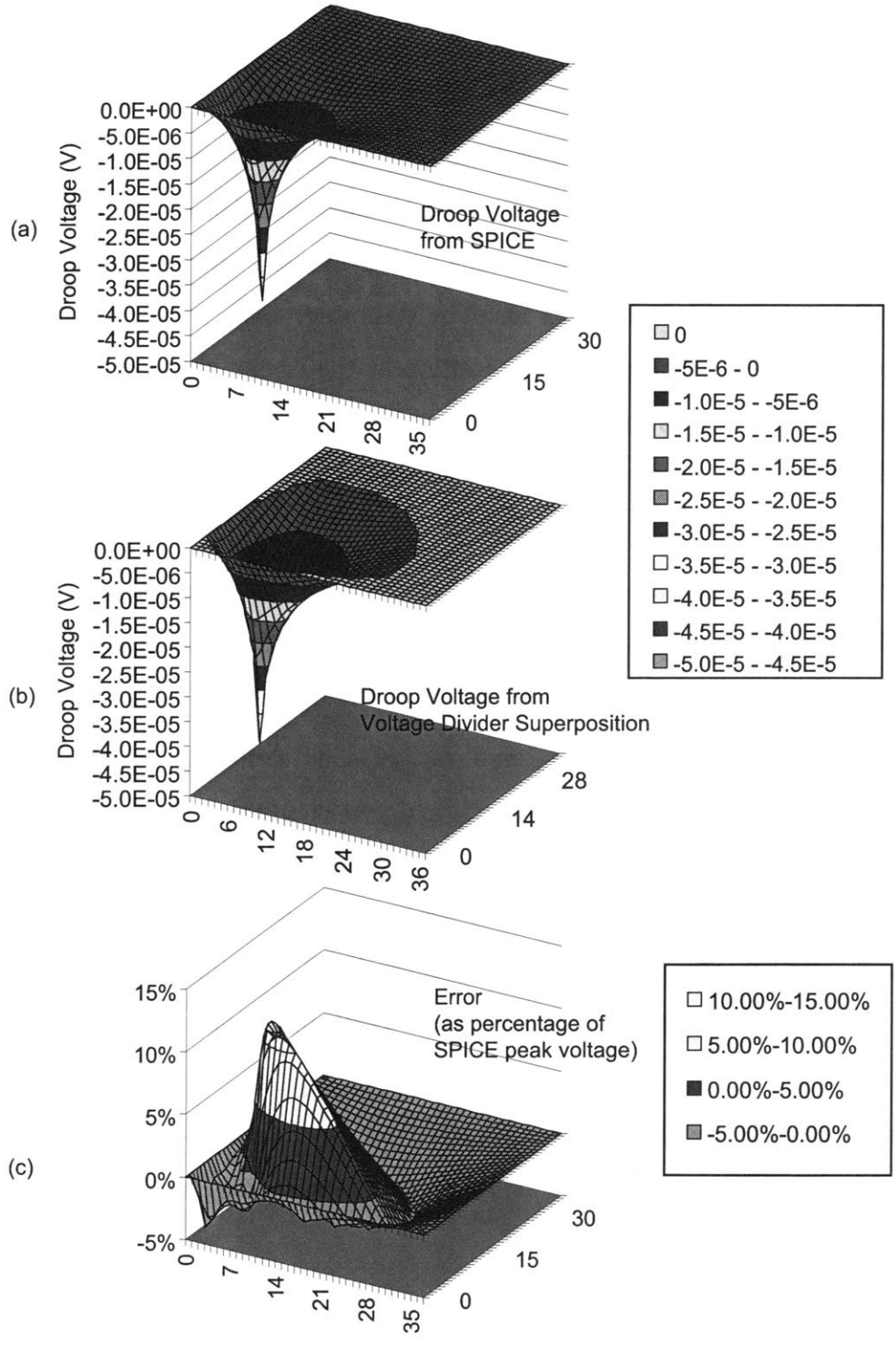

Figure 4-4: 32×32 array with one 0.1 mA source at (4,4)

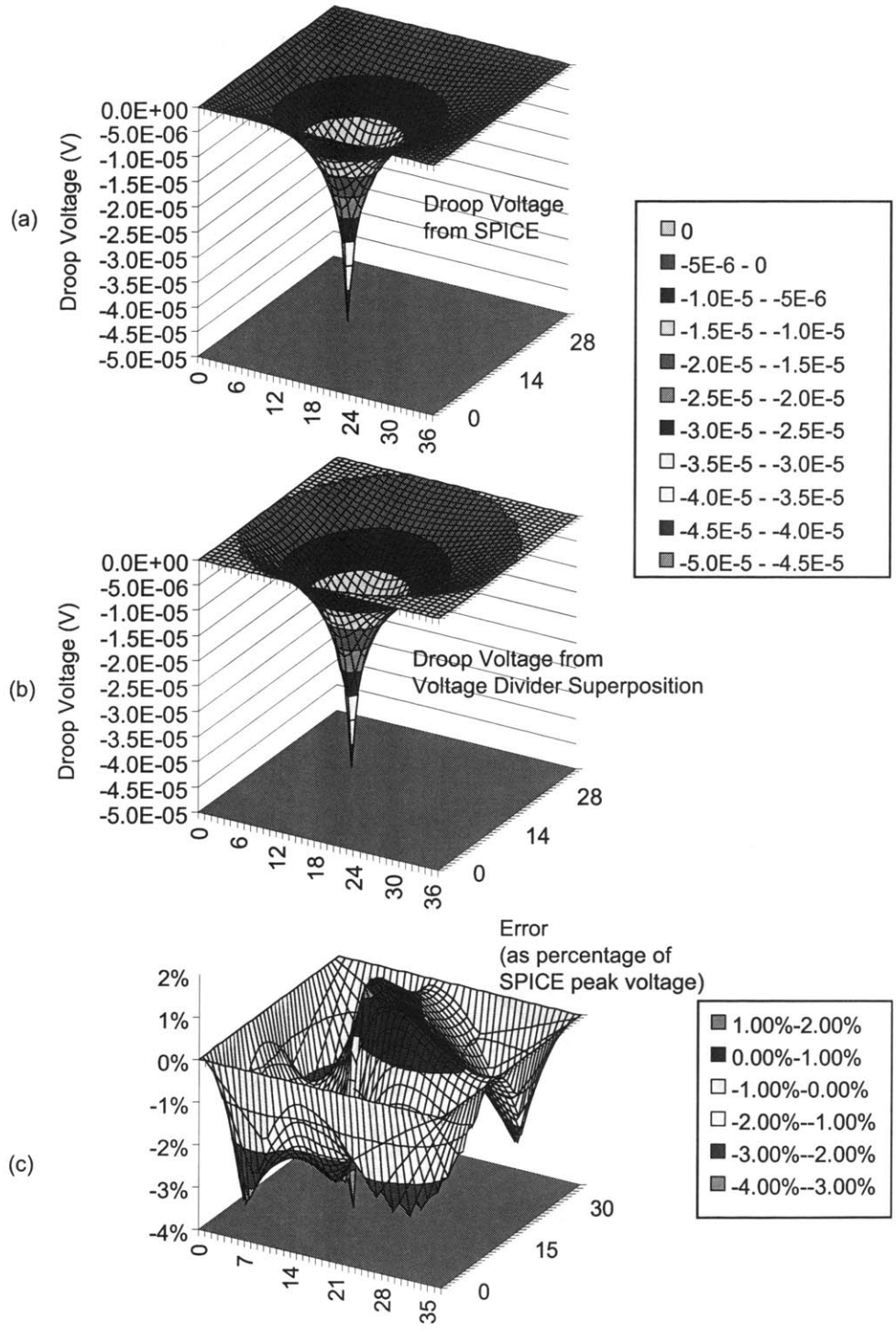

Figure 4-5: 32×32 array with one 0.1 mA source at (12,12)

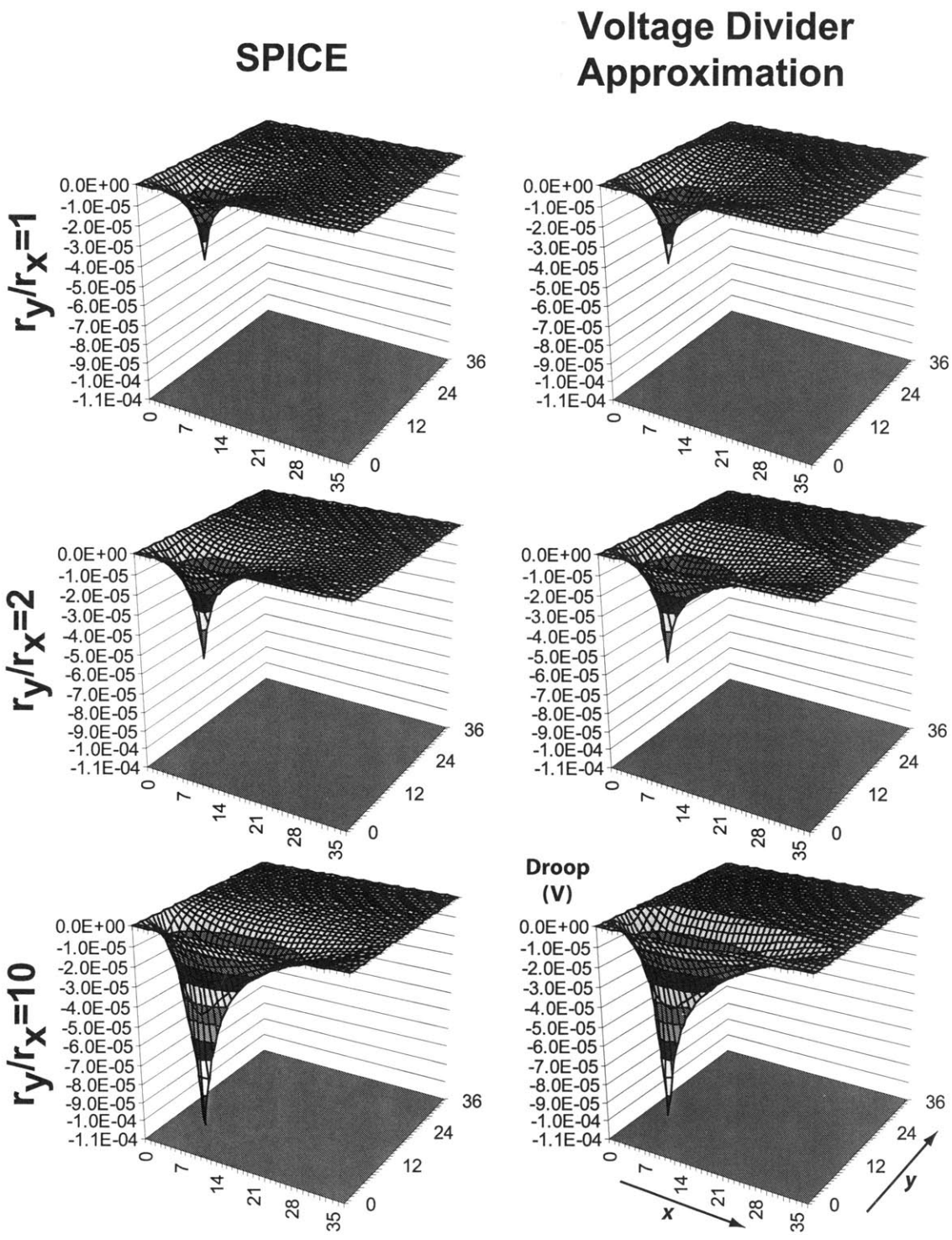

Figure 4-6: 32×32 array with one 0.1 mA source at (7,7) and various resistor ratios

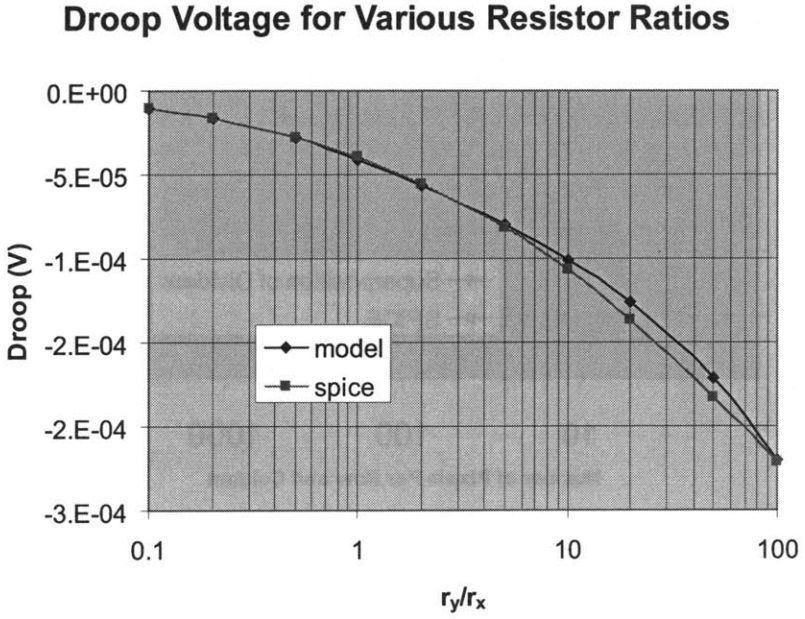

Figure 4-7: Droop peak for various resistor ratios

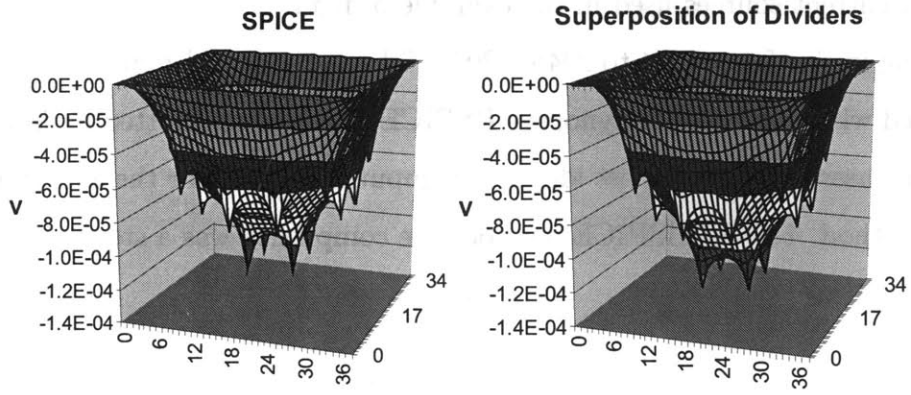

Figure 4-8: 32×32 pixel array with a 4×4 array of 0.1 mA sources

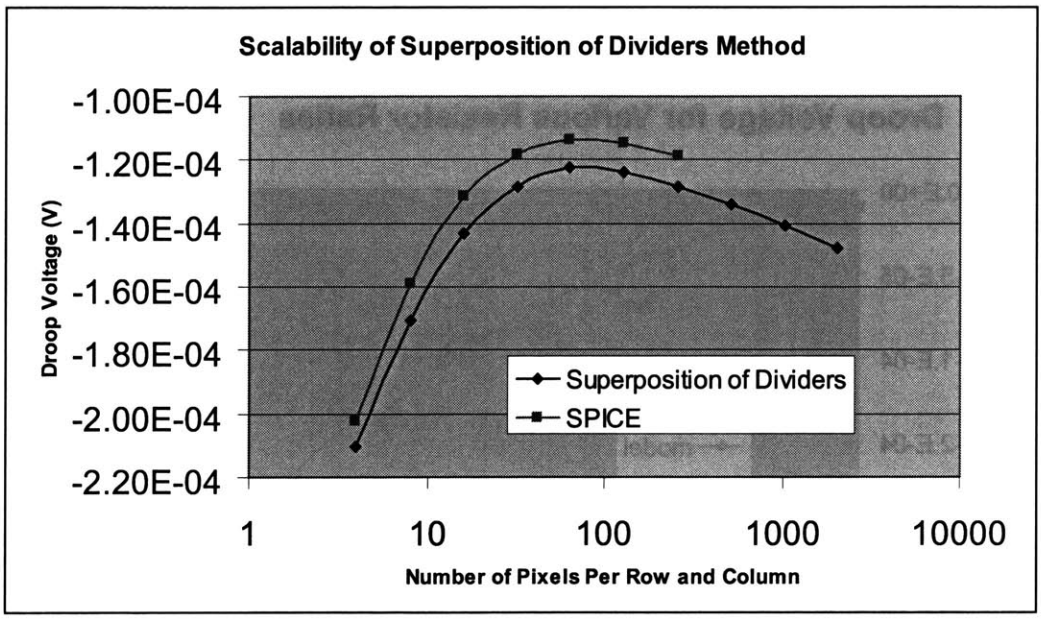

Figure 4-9: Scaling of droop calculations to large arrays

of current sources in the horizontal or vertical direction, then the algorithm scales as $3N_{I_\ell}^2$. One caveat is that it is essential to avoid running into I/O bottlenecks due to the inefficiencies of GTX and Perl.

The scalability of the superposition of dividers approach was also explored. The same sixteen current sources used in the example of figure 4-8 were evenly placed in arrays ranging in size from 4×4 to 2048×2048. The worst case droop was computed and compared with results from Synopsys HSPICE. Results are plotted in figure 4-9. Note that all cases required almost identical computation time for the superposition of dividers method. For the HSPICE method, the complexity was a steep function of N_{pix}.

4.4 Parasitic Capacitance Modeling

For the purposes of parasitic capacitance modeling, a number of empirical models for geometric primitives are applied. These are reviewed in detail in appendix E. The

interconnect structure is decomposed into the following primitives: parallel wires over one or between two groundplanes, crossovers, overlapping parallel lines on different levels and nonoverlapping parallel lines on different levels.

Interconnect capacitances for grid-type wiring are represented in F/pixel. The relative permittivity of the dielectric material ϵ_{rel} is represented as `eps_rel_ox_interLD`. (Note that here ϵ is used to indicate permittivity, not placement error.) This is multiplied by vacuum permittivity ϵ_0, `eps_0` to give ILD permittivity ϵ_{ox}, `eps_ox_interLD`. In this implementation, only one insulating material is supported.

4.4.1 Miller Multipliers

In some cases, two coupled wires may carry signals that potentially have complementary slopes. For these cases, Miller multiplication [30] must be considered in interpreting capacitance estimates. For the purposes of interconnect placement and for analysis of wires coupled to loosely-defined signals, a worst-case Miller multiplier parameter `k_miller_signals` is used. For the purposes of analysis or well-defined signal lines, the coupling capacitance is explicitly evaluated for use in custom simulations. For example, where coupled digital signals can be complementary, the corresponding multiplier is set to 2.

4.4.2 Evaluation of Parasitic Capacitances

The rule `DPS3D_C_parasitic_signals` contains an **extern_file** call to **calcParasiticCap.pl**, which computes the parasitic capcitances for all defined grid signals. Parameter `C_parasitic_signals` is a `vector<vector<double>>` with the parasitic capacitance between nets represented in the form $\{ID_1,ID_2,C_p\}$. The net numbers $ID + 0.01$, $ID + 0.02$, $ID + 0.03$ and $ID + 0.04$ are used to represent coupling to adjacent cells.

`DPS3D_C_parasitic_signals` makes use of several control parameters. Parameter `k_select_parasitic_c` is a comma-delimited string with tokens representing those

parasitic models which are to be applied. These correspond to equations presented in appendix E, and to subroutine names in the Perl implementation of these models, **capacitanceFormulaeREQUIRE.pl**. Allowed tokens include:

`C_12l_0gp_sl`	Line-to-line capacitance, lines on same level, no ground plane
`C_12g_1gp_sl`	Line-to-ground capacitance, lines on same level, one ground plane
`C_12l_1gp_sl`	Line-to-line capacitance, lines on same level, one ground plane
`C_12g_2gp_sl`	Line-to-ground capacitance, lines on same level, two ground planes
`C_12l_2gp_sl`	Line-to-line capacitance, lines on same level, two ground planes
`C_12l_1gp_ol`	Line-to-line capacitance, overlapping lines on different levels, one ground plane
`C_12l_1gp_nol`	Line-to-line capacitance, nonoverlapping lines on different levels, one ground plane
`C_12l_1gp_xov`	Line-to-line crossover capacitance, different levels, one ground plane

It is readily apparent from the equations presented in appendix E that approximations are needed to enable modeling of complex structures using the specified base models. The parameter `k_cap_options` allows for selection of some of these approximations. For example, for nearbody capacitance calculations, the provided models assume wires of identical width. Parameter `k_cap_options` may be provided with the tokens `nearbody_mode_avgw`, `nearbody_mode_minw`, and `nearbody_mode_maxw` to select the width value that is used for nearbody capacitance computation.

It is also desirable to limit output to parasitic capacitance values that exceed a given threshold. The parameter `k_cap_filter` specifies the minimum reported per-pixel parasitic capacitance.

Extraction Flow

The called function **calcParasiticCap.pl** is actually a wrapper function that uses subroutines defined in **calcParasiticCapREQUIRE.pl**. This allows the workhorse function to be applied to other rules, such as is necessary for wire placement optimization and table building. The wrapper function reads the input parameters, performs some basic error checks, and creates hashes. A list of interconnect levels is generated and ordered according to vertical location in the physical stack.

For levels with grid usages, adjacent pixel signals are added. These are named according to the conventions outlined in section 4.2. Only the nearest nearbody wire on each level of each adjacent pixel is included. With the full set of wires of interest defined, the workhorse subroutine **&calcParasitCap** is called.

The implementation of **&calcParasitCap** assumes nearbody-dominated capacitance. To validate this assumption, a 2-D field solver simulation was performed using Silvaco Exact2. A structure consisting of three parallel lines and two ground planes was considered. (This is identical to the structure illustrated in figure E-2(b) in appendix E.) $S = 0.3$ µm, $W = 2S = 0.6$ µm, $T = 0.6$ µm, and $H_1 = H_2 = H$. Parameter H was varied from 0.1 µm to 5 µm. The values for parameters S, W, and T roughly correspond to the process parameters for the MITLL 3-D FDSOI technology,[2] applied to the case of minimum-spaced wires of the minimum contacted width. Figure 4-10 shows the simulated capacitances. The "nearbody capacitance" trace corresponds to the capacitance between the center wire and one of the outer wires. The "capacitance to ground plane" corresponds to each capacitance between one outer wire and one ground plane. Note that the total capacitance is strongly dominated by the nearbody component when $H = 2t_{ILD} + t_{metal}$. For this reason, and for simplicity of implementation, level-to-level capacitances are only computed for adjacent pairs of used metal levels. For example, if nets exist on the first and third metal level, but not on the second, then capacitances between the nets on metal 1 and metal 3 are computed. If, however, there are also nets on metal 2, then capacitances

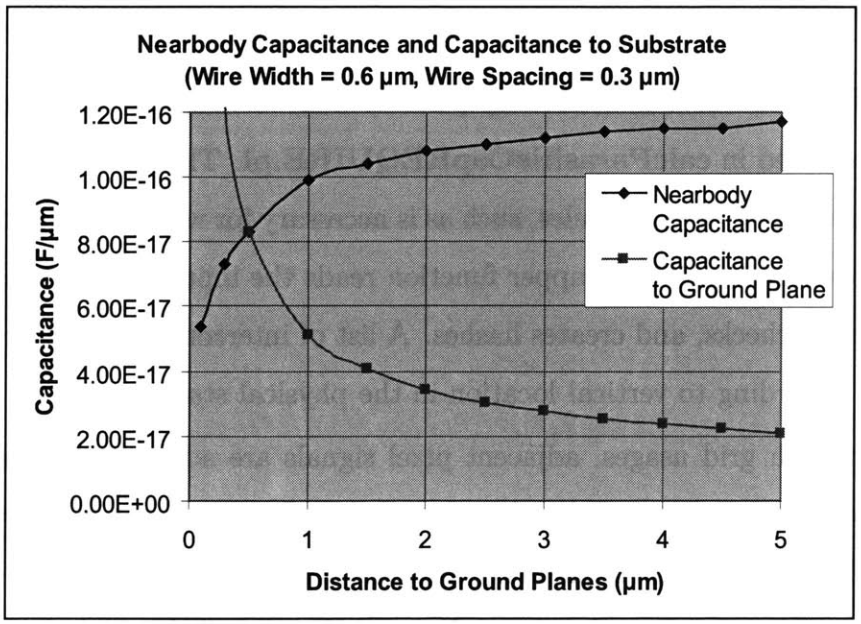

Figure 4-10: Simulated nearbody capacitance and capacitance to substrate for various ILD thicknesses

between nets on metal 1 and metal 3 are ignored. Implementing a more sophisticated capacitance extraction engine with shielding models is a straightforward extension that is beyond the scope of this work.

For adjacent pairs of used metal levels, capacitance calculation is based on level usage. For parallel or perpendicular grid interconnects, the models described in appendix E are readily applied. For non-grid interconnects, certain assumptions must be made. One approach to modeling dense random local and global interconnect is to replace it with a ground plane. This possibility was explored using a set of Silvaco Exact2 field solver simulations. A set of layouts was prepared consisting of three parrallel wires with $W = 0.6$ µm and $S = 0.3$ µm. These were placed 1 µm above either a conducting grating or a groundplane. The conducting gratings were 0.3-µm half-pitch conducting wires of height 1 µm, directly formed on a conducting substrate. Both the horizontal and vertical grating cases were simulated. Silicon dioxide was assumed to exist between all features. Table 4.4 shows the field solver results. Note

Case	Solver Type	Center Wire Capacitance (aF/μm)	Outer Wire Capacitance (aF/μm)
Parallel grating, with grating space centered on center wire	2-D	35.8	65.6
Parallel grating, with grating line centered on center wire	2-D	36.2	65.4
Perpendicular grating	3-D	31.8	52.2
Groundplane	2-D	34.3	52.4
Groundplane	3-D	33.4	52.9

Table 4.4: Capacitances of wires over gratings and groundplanes

that although the capacitance is slightly higher for the grating case, the groundplane still allows for an acceptable approximation.

Other 3-D Circuit Parasitic Capacitances

Figure 4-11 shows some additional parasitic capacitances that exist in a 3-D circuit. Suppose the structure pictured occurs within each pixel. Also, for the purposes of this discussion, assume that figure 4-11 gives a cross-sectional view of three parallel grid wires. Two of these are on metal 1 and one is on metal 2. Since this structure is repeated many times per row or column, the parasitic capacitances shown are multiplied by the number of pixels traversed.

Capacitor C_1 represents the capacitance between a grid wire and a stacked intra-tier via. This capacitance may be larger than a per-pixel crossover capacitance of perpendicular grid lines. Capacitors C_2 and C_3 are between grid wires and the inter-tier via itself. The worst case values for C_2 and C_3 are a function of alignment error $\epsilon_{(metal1,3Dv1)}$. Conventional parasitic extraction algorithms often neglect alignment error in consideration of intra-tier parasitics such as C_6, where misalignment can sig-

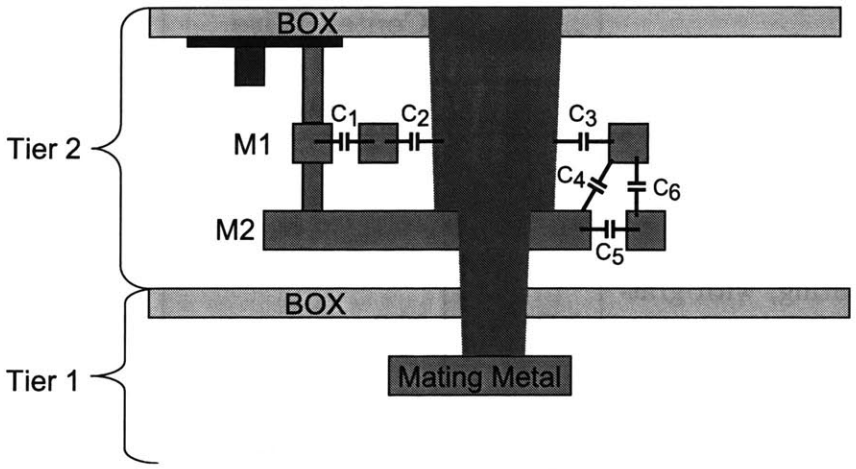

Figure 4-11: Some parasitic capacitances in a 3-D circuit

nificantly affect the net parasitic capacitance between long parallel wires on different levels. When error propagation of a variable number of misalignments is included, the effect is even more severe. Fortunately the rules developed in chapter 3 are easily applied to this problem.

In addition to the capacitances to the inter-tier via, capacitors C_4 and C_5 occur between grid wires and the masking metal level. This is an intra-tier problem, so misalignment is not as much of a concern (save for some possible shielding effects). An in-depth study of parasitics arising from various inter-tier via structures and misalignment conditions may form an interesting piece of future work.

4.4.3 Case Study: Slew-limited Line Driver

Consider an imager that consists of a 1024 × 1024 pixel array, with three rows of dummy pixels around the border. This is implemented in the MITLL process, with design rules and process parameters as specified in [2]. The relevant part of the pixel is illustrated in figure 4-12. In each pixel, there is a 100-μm^2 subcircuit that has dense local interconnect on tier 2, metal 3. On metal 2, there are four global `gridy` interconnects: VDD, GND, ROW, and RST. The metal 2 VDD and GND supply

lines are 1 µm in width, while ROW and RST are `wire_elmore` lines 0.6 µm in width. On metal 3, there are four global `gridx` interconnects. These include 1-µm VDD and GND lines, a 0.6-µm VRST reset level line and a 0.6-µm COL analog output line. This case study focuses on the driver for the COL line.

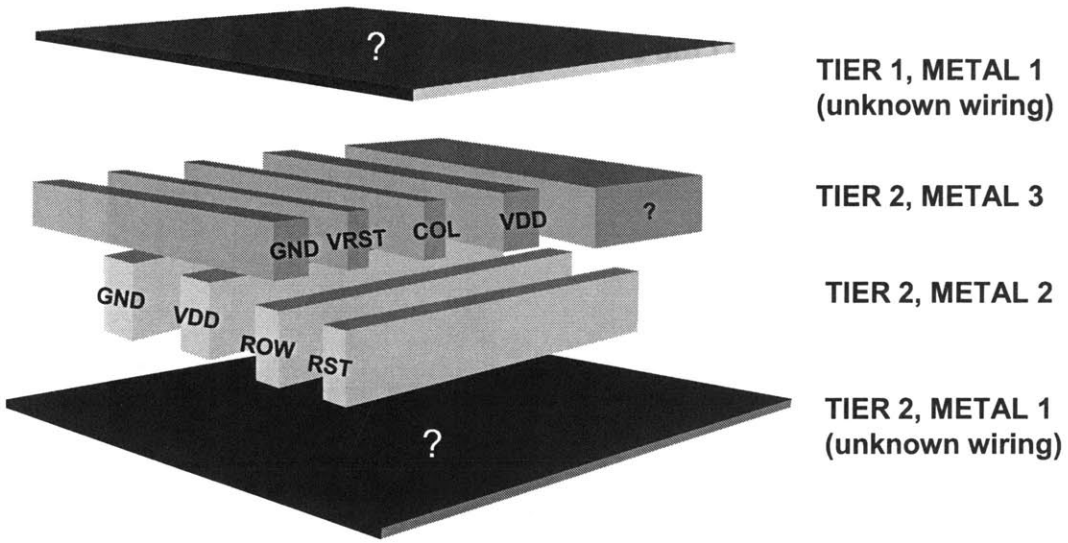

Figure 4-12: Interconnects in an example pixel

Assume that the imager has a frame rate f_{frame} of 60 Hz. There a column multiplexer and ADC module located at one end of the column lines with an input capacitance of 100 fF. The rows must be read out at a rate of $f_{frame} \times N_{pixels,y}$, giving 16 µs to read a row. If one-tenth of this time is budgeted for driving the column line, and the signal swing is 2 V, then the required per-pixel current is approximately given by $I_{driver} = C_{total} \times \frac{V_{swing}}{t_{allowed}} = 1.2\frac{A}{\mu F} \times C_{total}$.

C_{total} is the sum of the load capacitance and the wiring capacitance. The wiring capacitance includes two nearbody components: COL-to-VDD and COL-to-VRST. At the minimum possible pixel size of just over 13 µm, it is to be expected that the nearbody capacitance dominates. As the pixel size increases, this nearbody capacitance decreases superlinearly with the inter-wire spacing, but all per-pixel capacitances also increase linearly because of increased wire length. Figure 4-13 shows a

projection made using the GTX capacitance models discussed earlier in this chapter. Indeed there is a superlinear decrease in required driver current near the minimum possible pixel pitch and a linear increase at large pitches. The minimum occurs at about 16 µm.

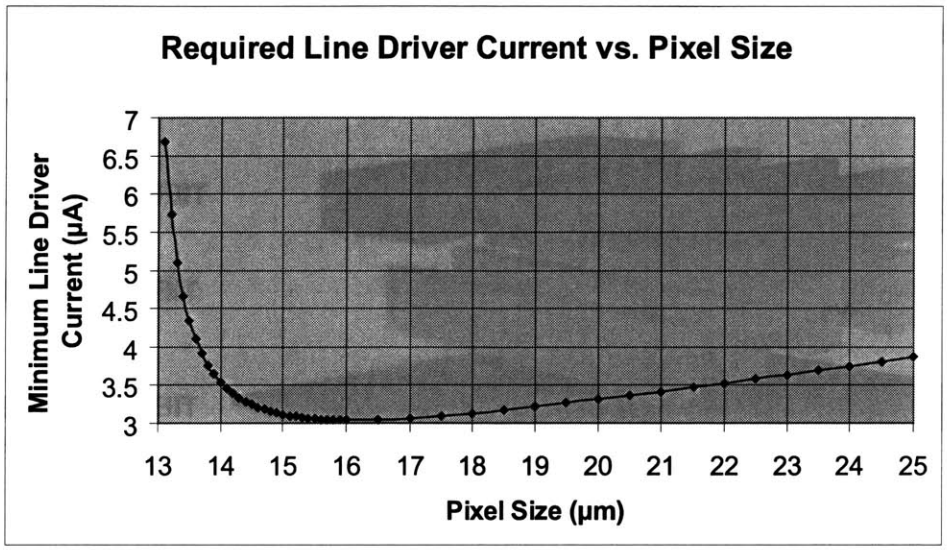

Figure 4-13: Required line driver current vs. pixel size

Chapter 5

Module Placement Optimization Using Simulated Annealing

In the previous chapters, methods have been developed to model tradeoffs between area, power, and performance. In chapter 3, a model relating the area consumption of inter-tier interconnect to fundamental process parameters – including tier counts and metal level counts – was presented. Chapter 4 discussed the modeling of interconnect structures that frequently occur in sensor circuits. These models may be applied to a minimization heuristic based on simulated annealing.[39]

5.1 Optimization Heuristic

In this section, a basic placement optimizer is presented. In this case, the cost function is equal to the pixel area. It is straightforward to extend this approach to a more sophisticated cost funtion, based in part on the previously-discussed models for capacitance, voltage droop, slew, etc. In the present implementation, the optimizer routine is implemented in an **extern_file** call rather than GTX code. With some of the minor improvements to the GTX framework suggested in appendix A, implementing the identical algorithm with a cost function based on called rules previously

developed is a relatively easy task.

Consider a pixel circuit comprised of a set of numbered modules.

$$M_i \in M = \{0, 1, 2, \ldots N_M\} \tag{5.1}$$

These modules are interconnected by a set of numbered port nets.

$$P_i \in P = \{0, 1, 2, \ldots N_P\} \tag{5.2}$$

A GTX parameter `k_nets_modules` is defined as a `vector<vector<double>>` with each inner vector of the form $\{M_i, P_{i,1}, P_{i,2}, \ldots\}$ where $P_{i,j}$ is the j^{th} port of module M_i. In practice, `k_nets_modules` could be easily derived from a structural HDL description. The area of each module is specified by the `vector<vector<double>>` parameter `dA_modules`, which has inner vectors of the form $\{M_i, A_i\}$ where A_i is the area of module M_i. The initial location of these modules is specified by the parameter `k_loc_modules_initial`, which is a `vector<vector<double>>` with inner vectors of the form $\{M_i, T_i\}$, where T_i is the tier number on which module M_i resides. The final locations will be reported as `k_loc_modules_final`, which has the same format.

The simulated annealing heuristic uses a temperature parameter `k_temp_sa` which defines the initial probability p_{init} that a randomly-chosen movement of a module from one tier to another that results in higher cost will be accepted. Moves that result in lower cost are always accepted. The parameter `num_moves_pertempstep` specifies the number of moves N_{moves} attempted before the temperature is lowered by a factor τ_{temp} (`k_tau_temp`) so that the probability of acceptance of a higher-cost move during temperature step k is $p_k = p_{k-1}\tau_{temp}$. `k_tau_temp` is subject to the constraint $0 < \tau_{temp} < 1$. The annealing is considered complete when three successive temperature reductions fail to achieve at least a 1% improvement in cost.

Since some modules (such as a photodiode) must reside on a particular tier, a parameter `k_mobility_modules` is included. This is a `vector<vector<double>>` with

inner vectors of the form $\{M_i, \mu_i\}$, where $0 \leq \mu_i \leq 1$ describes the mobility of a module. When $\mu_i = 0$, T_i is fixed. For $0 < \mu_i \leq 1$, the effective probability of acceptance of a costly move is given by $p_{eff} = \mu_i p_k$.

The areas of the exclusion zones due to the interaction of inter-tier vias with intra-tier levels are specified by two parameters, `dA_exclude_cut_3Dvia` and `dA_exclude_land_3Dvia`. In this simple implementation, it is assumed that N_{tiers} (`num_tiers`) tiers are stacked usind a post-bond via process. All have identical orientation, i.e. either `face-back` or `back-face`. Inter-tier via stacking is allowed. `dA_exclude_cut_3Dvia` represents the exclusion zone due to via X on tier *(X+1)*, and `dA_exclude_land_3Dvia` represents the exclusion zone due to via X on tier X. It is expected that `dA_exclude_cut_3Dvia` and `dA_exclude_land_3Dvia` be derived from the more complete exclusion information in `df_exclude_TX_via_intertier_Y`, as discussed in chapter 3.

The simulated annealing rule was defined as `DPS3D_k_loc_modules_final_sa`. Rules `DPS3D_dA_pixel_initial` and `DPS3D_dA_pixel_final` report the initial and final pixel areas in `dA_pixel_initial` and `dA_pixel_final`, respectively.

5.2 Case Study: Application of Simulated Annealing to a LADAR Imager

To demonstrate the capabilities of `DPS3D_k_loc_modules_final_sa`, a laser radar (LADAR) imager currently under development at MITLL was considered.[6, 7] Figure 5-1(a) illustrates the intended application. A photon triggers an avalanche in a geiger-mode avalanche photodiode (APD). This is detected by a digital timing circuit, and the digitally-encoded photon flight time is read out. A three-tier 3-D implementation of this imager is presently in fabrication. This 3-D implementation is illustrated in figure 5-1(b).

All pixels are serviced by a high-speed clock signal. A local buffering of this

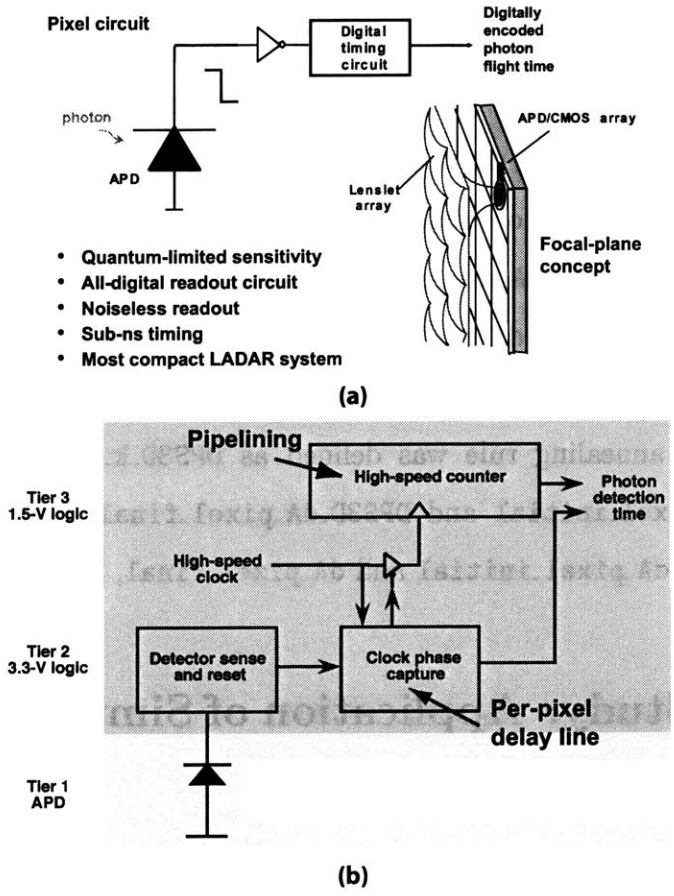

Figure 5-1: Concept (a) and 3-D implementation (b) of LADAR imager [6, 7]

clock controls a high-speed pseudorandom counter and a clock-phase capture circuit. When an avalanche of the APD is detected, the counter is halted and the clock phase is sampled. The resulting time measurement is read out. Since there are many of these timing circuits in parallel, the depth of an image may be measured based on the photon time of flight.

5.2.1 LADAR Modules

Table 5.1 lists modules in the LADAR pixel of [7]. Not listed are decoupling capacitors of area 90 μm^2 that occur on every tier (other than the APD tier.) Modules requiring additional decoupling on the same tier have the required decoupling area incorporated into the module area. Tier 1 is reserved for the APD alone. The APD is set up to have zero area so that its area does not put a lower limit on the pixel size.

In some cases, the nets listed in the table do not directly correspond to circuit nets. For example, DATAIO is actually both the nets DATAIN and DATAOUT of a pixel. Since these must be daisy-chained together for serial readout, they are combined into a single net for this placement analysis. In other cases, inverters have been neglected. This is based on the assumption that certain modules can be modified to accept either a true or a complement signal with minimal impact on area.

5.2.2 Floorplan Construction Using Actual Design Parameters

For the initial experiment, it was assumed that the tier 1 to tier 2 bond was `face-face`, and that the tier 2 to tier 3 bond was `back-face`. `dA_exclude_cut_3Dvia` was set to 18 μm^2 and `dA_exclude_land_3Dvia` was set to 40 μm^2. These values correspond to the design rules used for the actual layout. It was assumed that modules could be placed on any tier (other than the APD tier) without penalty. The initial temperature was set to $p_{init} = 0.95$, and the decay constant was set to $\tau_{temp} = 0.1$. N_{moves}

Module	Area (μm^2)	Nets
APD	0	PHOT
ARM INV	13	ARM, $\overline{\text{ARM}}$
ARM/DISARM	100	PHOT, ARM, $\overline{\text{ARM}}$, DISARM
FIRE INV	17	PHOT, FIRE
VBC LOGIC	120	FIRE, $\overline{\text{SCLK}}$, VBC
SCLK INV	13	SCLK, $\overline{\text{SCLK}}$
VERNIER CKT	300	Q9, CLK, DATAIO
MUX XOR 1	55	DATAIO, Q0, Q1
MUX XOR 2	55	Q0, Q1, Q2, Q7, Q8
FF1	35	Q1, Q2, CLK
FF2	35	Q2, Q3, CLK
FF3	35	Q3, Q4, CLK
FF4	35	Q4, Q5, CLK
FF5	35	Q5, Q6, CLK
FF6	35	Q6, Q7, CLK
FF7	35	Q7, Q8, CLK
FF8	35	Q8, Q9, $\overline{\text{SCLK}}$
CLK LOGIC DRV	240	CLK, SCLK, FIRE

Table 5.1: Modules in the LADAR pixel

was set to 500. The results from ten passes are shown in table 5.2

Note that the simulated annealing optimization results in comparable area utilization than the full-custom designer's floorplan. The actual full-custom layout was in a 35 µm × 35 µm (1225 μm^2) pixel. It included about 300 μm^2 of space that did not contain any active, poly, or local interconnect features. Compared with the calculated 906 μm^2, there is a 35% overhead. Based on the results in table 5.2, it can be concluded that a well-designed simulated annealing experiment may provide a good early insight into what a full custom designer might do later on in the design flow. Since several possible arrangements of modules are generated, a range of area

Module	Actual MITLL Layout Tier	Tier for Each Pass									
		1	2	3	4	5	6	7	8	9	10
APD	1	1	1	1	1	1	1	1	1	1	1
ARM INV	2	2	2	3	2	2	2	2	2	2	2
ARM/DISARM	2	2	2	3	2	2	2	2	2	2	2
FIRE INV APD	2	2	2	2	2	2	2	2	2	2	2
VBC LOGIC	2	2	3	2	2	2	3	2	3	3	3
SCLK INV	2	3	3	3	3	2	3	3	3	3	3
VERNIER CKT	2	3	3	3	2	3	3	2	3	3	3
MUX XOR 1	3	3	3	2	3	3	3	3	2	2	2
MUX XOR 1	3	2	3	2	3	3	3	3	2	2	2
FF1	3	2	3	2	3	3	3	3	2	2	2
FF2	3	2	2	2	3	3	3	3	2	2	2
FF3	3	2	2	2	3	3	3	3	2	2	2
FF4	3	2	2	2	3	3	2	3	2	2	2
FF5	3	2	2	2	3	3	2	3	2	2	2
FF6	3	2	2	2	3	3	2	3	2	2	2
FF7	3	2	3	2	3	3	3	3	2	2	2
FF8	2	3	3	2	3	2	3	3	3	2	2
CLK LOGIC DRV	3	3	2	3	3	2	2	3	3	3	3
Calculated Area (μm^2)	9 0 6	8 9 8	8 5 3	8 7 5	8 9 8	8 1 7	8 8 1	8 9 8	8 7 0	8 5 3	8 5 3

Table 5.2: Simulated annealing results for actual LADAR design conditions

requirements may be anticipated.

5.2.3 Effect of Number of Tiers

As previously discussed, it is possible to investigate the effect of various process and circuit parameters on a specified cost function. For the case of the LADAR focal plane, it is important to understand the dependence of pixel size on process parameters. Figure 5-2 shows the LADAR pixel size versus number of tiers for three different process options.

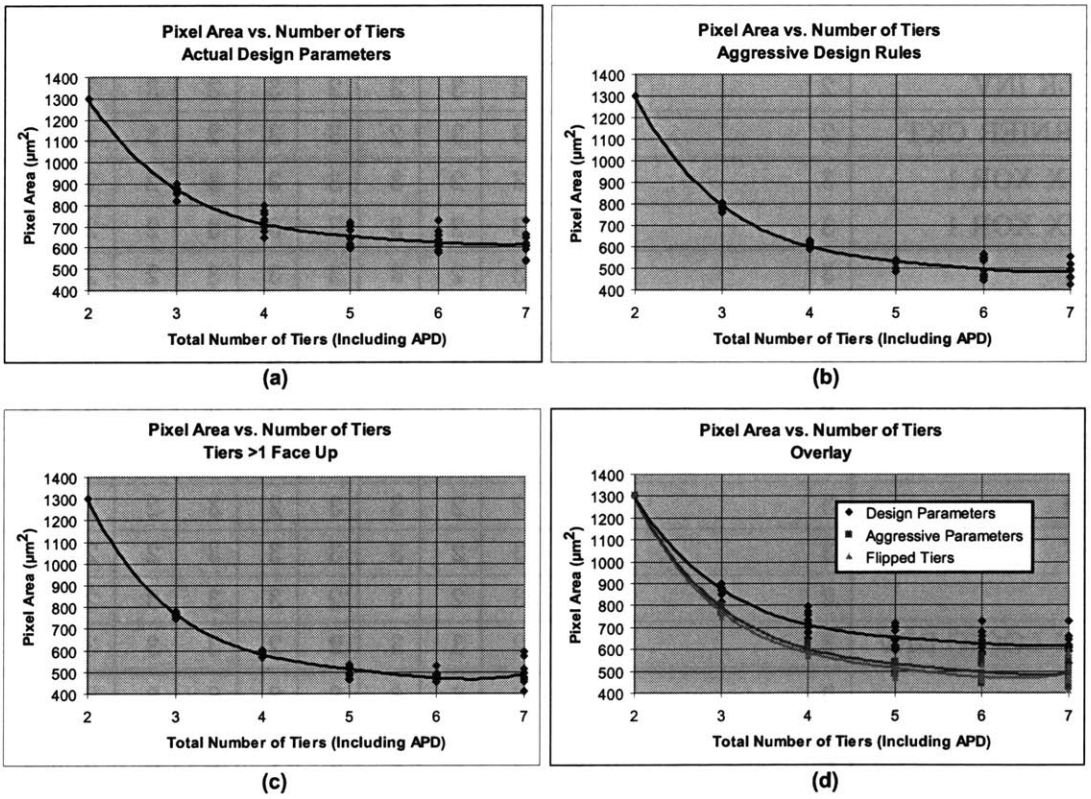

Figure 5-2: LADAR imager pixel size vs. number of tiers for three different sets of design rules

Figure 5-2(a) illustrates the case of the design rules that were actually used for the MITLL LADAR design. These parameters are the same as in section 5.2.2. The plot shows the results of ten different simulated annealing optimizations for each number

of tiers. The total number of tiers includes the APD tier. Note that after four or five tiers, there is little benefit. This is due to the high degree of interconnectivity within the pixel circuit.

Figure 5-2(b) shows the case where a more aggressive set of design rules is applied. In this case, `dA_exclude_cut_3Dvia` was reduced to 9 μm^2 and `dA_exclude_land_3Dvia` was reduced to 16 μm^2. There is a modest (about 15%) improvement in pixel size for each number of tiers. Also, the spread in the solution areas is reduced, suggesting a more easily achievable design. In figure 5-2(c), all of the tiers are bonded face-back, i.e. all tiers are face up. This allows inter-tier vias to land on the top-level metal, thus leaving room for underlying circuitry. With `dA_exclude_cut_3Dvia` set to 18 μm^2 and `dA_exclude_land_3Dvia` set to 0 μm^2, the results are similar to those for the aggressive design rules case. Figure 5-2(d) shows an overlay of the three cases.

5.2.4 Summary

In this chapter, the use of simulated annealing to explore design possibilities was demonstrated. In general, a cost function may be derived using many of the models developed in previous chapters. The goal of conceptual design is to describe a circuit concept at as high a level of abstraction as necessary, and then model the dependencies on design decisions. In chapters 3 and 4, foundational models for process and circuit behavior were presented. This chapter adds to this foundation a means of understanding how a design that exists only in abstract form might be implemented subject to those process and design constraints. GTX provides the inference chain structure necessary to represent the cause and effect relationships that are encountered in circuit design. With a well-established knowledge library, tradeoff studies may be conducted easily and efficiently.

Chapter 6

Verilog-AMS and SPICE Interface

Recall from the proposed EDA flow illustrated in figure 1-3 that the design loop is closed in two ways. First, the designer may directly interpret and act on GTX output parameters. Second, GTX parameters may be inserted into behavioral models that are then used to evaluate system performance. Not shown in figure 1-3 is a third possible approach, namely the incorporation of GTX result parameters into a low-level SPICE simulation. Depending on the application, GTX parameters pertaining to resistance, capacitance, droop voltage, supply voltage, power density, thermal conductivity, transitor properties, etc. may be germane to a behavioral model.

In this chapter, the construction of behavioral and low-level models based on result parameters is briefly discussed. Although there may be limited pedagogical value to an extensive discussion of scripting, it is certainly valuable to consider strategies for effective utilization of derived parameter data.

6.1 Template Files: Verilog-AMS Output Example

Since there are several different types of simulation tools that may be used, a general template approach was selected. These templates are processed by **buildModel.pl**.

The following example shows a (partial) template file that might be used in modeling column drivers. In this case, there is a source follower within a pixel that has a current specified by the parameter `I_driver_pixel`. The total capacitance of the column line is described by the parameter `C_line_pixel`. The load capacitance on a column is given by `C_load_column`. The slew rate is calculated by dividing the available current by the total capacitance.

```
begin header
valfile valuefile1.val
suffix .v
%%drivecurrent%% &lookupval("I_driver_pixel");
%%collinecap%% &lookupval("C_line_pixel");
%%colloadcap%% &lookupval("C_load_col");
%%sr%% %%drivecurrent%%/(%%collinecap%%+%%colloadcap%%);
end header

module column_line(out, in);
    inout out, in;
    electrical out, in;

    analog begin
        V(out) <+ slew(V(in), %%sr%%);
    end

endmodule
```

Note that the above template file starts with header information. "**begin header**" is syntactically required to denote the start of the header section. All parameters to

be used in the subsequent macro must be defined in the header. The first line after the start of the header must be of the form

valfile `values_filename`

where `values_filename` is an output dump created by GTX. The next line is of the form

suffix `filename_extension`

where `filename_extension` specifies the extension of the output files.

After these required lines, the parameter definitions follow. These are evaluated sequentially. Parameters always begin and end with "%%". Parameter definition lines have the form

`parameter_name expression`

where the expression may include Perl operators. Several special operators are defined to allow for retrieval of data from the values file. For parameter values of type `bool`, `double` or `int`, the operator

&lookupval("`parameter_name`"**);**

retrieves the appropriate scalar from `parameter_name` in the values file. For values such as total net capacitances, where the parameter is a `vector<vector<double>>` with inner vector elements of the form **{**`id_number,value`**}**, the operator

&lookupval_1index("`parameter_name`"**,**`id_number`**);**

is provided. Likewise for parameters such as net-to-net capacitances and Miller multipliers, the operator

&lookupval_2index("`parameter_name`"**,**`id_number1,id_number2`**);**

allows for retrieval of values corresponding to two index elements. If no value corresponding to the specified index elements is found, the operator returns a null value. The following example shows how such a null value might be converted to zero so that it may be substituted into the macro.

```
%%collinecap%% &lookupval_1index("C_total_signals",3)*1;
```

Note that in the example template, the slew rate parameter **%%sr%%** is computed based on the values for previously-defined parameters. The only requirements for the expression are that it must be free of white space characters, and it must be Perl-compatible after all substitutions have been made. Thus, the user has considerable flexibility to implement complex functions. For example, a user-defined subroutine may be called.

```
%%paramA%% require("userroutine.pl");&usersub(%%paramB%%);
```

In many cases, the GTX values file contains multiple sets of output parameters that relate to varied input parameters. When **buildModel.pl** encounters multiple sets of parameters, it generates multiple versions of the output file, one for each parameter set.

6.2 Template Files: SPICE Deck Output Example

The same template format may be used to generate SPICE netlists. Consider a subcircuit with five nets. These have node ID numbers 0-4.

```
begin header
valfile test4.val
suffix .sp
%%GND%% 0;
%%VDD%% 1;
```

```
%%COL%% 2;
%%COLBAR%% 3;
%%VRST%% 4;

%%C_COL_GND%% &lookupval_2index("C_parasitic_signals",
                                %%GND%%,%%COL%%)*1;

%%C_COL_VDD%% &lookupval_2index("C_parasitic_signals",
                                %%VDD%%,%%COL%%)*1;

%%C_COL_VRST%% &lookupval_2index("C_parasitic_signals",
                                %%COL%%,%%VRST%%)*1;

%%C_COLBAR_GND%% &lookupval_2index("C_parasitic_signals",
                                %%GND%%,%%COLBAR%%)*1;

%%C_COLBAR_VDD%% &lookupval_2index("C_parasitic_signals",
                                %%VDD%%,%%COLBAR%%)*1;

%%C_COLBAR_VRST%% &lookupval_2index("C_parasitic_signals",
                                %%COLBAR%%,%%VRST%%)*1;

%%C_COL_COLBAR%% &lookupval_2index("C_parasitic_signals",
                                %%COL%%,%%COLBAR%%)*1;


end header


* SPICE deck for parasitic capacitance
C_COL_GND COL GND %%C_COL_GND%%
C_COL_VDD COL VDD %%C_COL_VDD%%
C_COL_VRST COL VRST %%C_COL_VRST%%
C_COLBAR_GND COLBAR GND %%C_COLBAR_GND%%
C_COLBAR_VDD COLBAR VDD %%C_COLBAR_VDD%%
C_COLBAR_VRST COLBAR VRST %%C_COLBAR_VRST%%
C_COL_COLBAR COL COLBAR %%C_COL_COLBAR%%
```

Once processed, this might produce a SPICE deck that resembles the following.

```
* SPICE deck for parasitic capacitance
C_COL_GND COL GND 3e-013
C_COL_VDD COL VDD 1e-013
C_COL_VRST COL VRST 4e-016
C_COLBAR_GND COLBAR GND 1e-015
C_COLBAR_VDD COLBAR VDD 0
C_COLBAR_VRST COLBAR VRST 0
C_COL_COLBAR COL COLBAR 1e-015
```

In addition to capacitances, parameters corresponding to device sizes, device models, temperatures, voltages, currents, resistances, etc. may be derived through the use of GTX models.

Chapter 7

Conclusion

7.1 Summary

Conceptual design has historically started with a mixture of creativity, insight, and "back of the envelope" calculations. Unfortunately, neither the capabilities of a designer nor the size of an envelope improve exponentially with Moore's law. It is improvements in EDA that have allowed for the nonrecurring engineering costs of the design cycle to be kept in check.

This work has laid the groundwork for a new approach to the early stages of the design cycle. By leveraging the inference chain framework originally developed for improving strategic planning in the semiconductor industry, a designer may gain new insight into the costs and benefits of various approaches to very specific problems. This is especially beneficial when, as in the case of 3-D integration, the designer has many degrees of freedom. With an adequate library of process and circuit knowledge, a problem may be simultaneously addressed at both a high and low level of abstraction.

The goal of conceptual design is to describe a circuit concept at as high a level of abstraction as necessary, and then model the dependencies on design decisions. In chapter 3, process models were developed that were later used to project area con-

sumption for various conditions. Chapter 4 added to this a subset of the many basic models necessary to draw the connection between implementation and performance. Then, in chapter 6, these process and circuit projection models were tied to a conventional high-level modeling approach through the use of interface scripts. Finally, in chapter 5, simulated annealing was used to derive plausible layouts suitable for improving the designer's understanding of the consequences of design and process decisions.

Several simple case studies were presented to demonstrate how with limited information, one might gain insight into the relationship between design and process parameters. The particularities of the inter-tier via flow were related to area consumption for particular circuit architectures. For the LADAR example of section 5.2, it was observed that for one particular circuit approach, additional tiers yield diminishing returns. This suggests that future work to devise an alternative counter architecture for large tier counts might yield useful results. However, process cost and yield are both strong functions of the number of tiers, and those economic issues can and should be modeled in conjunction with the circuit approaches.

In the line driver example in section 4.4.3, a relationship between power dissipation and area utilization was explored. However, area utilization is a function of process parameters, which can be variable. Likewise, using the droop voltage rules in chapter 4 in conjunction with clock tree distribution models reviewed in [42], it is possible to model the impact of power rail sizing on clock skew. In short, through the use of inference chains describing process and circuit parameter relationships, a paper study of a design concept may be performed before area, power, time or money is budgeted. Perhaps the greatest benefit to this is an elucidation of the questions that need to be asked throughout the design process.

7.2 Future Work

Many opportunities exist for future work in this area. It is recommended that the parameters, rules and scripts developed as part of this work be made available to designers on future MITLL 3-D multiproject runs. In addition, contributions to a design-oriented GTX knowledge library should be encouraged. As with any software developed for academic purposes, the models, rules, and scripts prepared in this work are not fully mature. Only application to actual design challenges will allow this work to transition from an academic exercise to a readily available EDA approach. Described below are some specific recommendations relating to the further development of an inference-chain based conceptual design tool.

7.2.1 Interface Improvements

As it is presently implemented, the GTX user interface is not optimized for use in a pre-design flow. For such a tool to be broadly accepted by circuit designers, a seamless interface between HDL representations and inference chain relations would be necessary. In addition, a visualization aid capable of graphically representing the interaction of process and design constraints would be especially valuable. In the interim, there are many improvements to GTX that could facilitate design studies. Some of these are noted in appendix A.

7.2.2 Enhancements to Process Models

Appendix D presents some observations regarding the limitations of the process models presented in chapter 3. Because 3-D integration is still in the early stages of development, both the general knowledge and library knowledge must follow developments in the technology.

7.2.3 Enhancements to Circuit Performance Projection Models

In the conceptual design phase, it is necessary to use models that require the fewest assumptions. Although simple models are often the most applicable, it is essential that all critical effects are at least crudely represented. In chapter 4, the parasitic capacitance models did not model the alignment-dependent parasitic capacitances to inter-tier via structures. Proper representation of alignment-dependent parasitics will require future study. A more nuanced model for the effects of unknown random interconnects is also desirable. For the superposition-based droop models, refinement of the determiniation of "effective ground" is required. Also, investigation of the applicability of resistor reduction techniques to this problem would be beneficial. Any droop modeling algorithm must be also combined with some means of determining decoupling capacitance requirements.

Development of a library of module performance models is also required. Such a library would require a parameterization of functional blocks such as photatransducers, integrating amplifiers and ADC modules, according to process parameters. In addition, local power dissipation of particular modules should be used to construct a thermal "circuit." Through the use of scripts such as those described in chapter 6, an HDL such as Verilog-AMS could then be applied to the projected thermal circuit during the architectural design phase.

7.2.4 Generalization of Simulated Annealing Cost Function

In the simulated annealing rule described in chapter 5, the cost function is simply the area utilization. This should be generalized in such a way that the cost function is itself a called rule. However, for such a called rule to be used within an external simulated annealing rule, improvements to the inference chain engine are necessary.

7.2.5 Improvement of Computational Efficiency

Finally, in this work, **extern_file** rules were implemented in Perl because GTX provides input parameters to external rules in ASCII format. Computational efficiency could be significantly improved through the use of a more streamlined programming language and programming style. Improvements to the input/output capabilities of GTX would also help enable the use of very large matrices as inputs to an external rule.

Appendix A

Discovered GTX Flaws and Limitations

In the course of this research, several limitations of the GTX software were discovered. In some cases, the applicability of GTX to certain problems is limited as a result.

1. Cannot use a rule that returns type `vector<double>` as a called rule using `#calledrules`. Attempting to do so causes GTX to crash.

2. An insufficient number of vector operations have been implemented. One useful function would be vector concatenation. Part of this deficiency would be addressed by solving item 1.

3. It would be desirable to allow a rule to apply a called rule with input type *inputtype* and output type *outputtype* to each element of an input of type `vector<`*inputtype*`>`, thus returning an output of type `vector<`*outputtype*`>`. Perhaps some sort of **for each ()** syntax could be used?

4. Vectors can currently only hold type `double`.

5. Type `double` is not really double precision. The number of mantissa and exponent digits retained appears to correspond to single-precision floating point

representation.

6. The GTX plotting utility crashes if the x-axis range is limited by a constraint rule.

Appendix B

GTX Disclaimer

The following license notice for the GTX software is included here to ensure compliance with GTX copyright requirements.

RANTIES, whether express or implied, including warranties of merchantability or fitness for a particular purpose or noninfringement, with respect to this software and supporting documentation. Providers have NO obligation to provide ANY support, assistance, installation, training or other services, updates, enhancements or modifications related to this software and supporting documentation.

Providers shall NOT be liable for ANY costs of procurement of substitutes, loss of profits, interruption of business, or any other direct, indirect, special, consequential or incidental damages arising from the use of this software and its documentation, whether or not Providers have been advised of the possibility of such damages.

Appendix A: there are no restrictions as to copyright, distribution or fee policies regarding software independently developed for use with GTX software.

Appendix C

Calculation of Area of Intersection of Two Circles

Consider two circles, as shown in figure C-1. Let A be equal to the area of intersection of the two circles, i.e. the area of the shaded region.

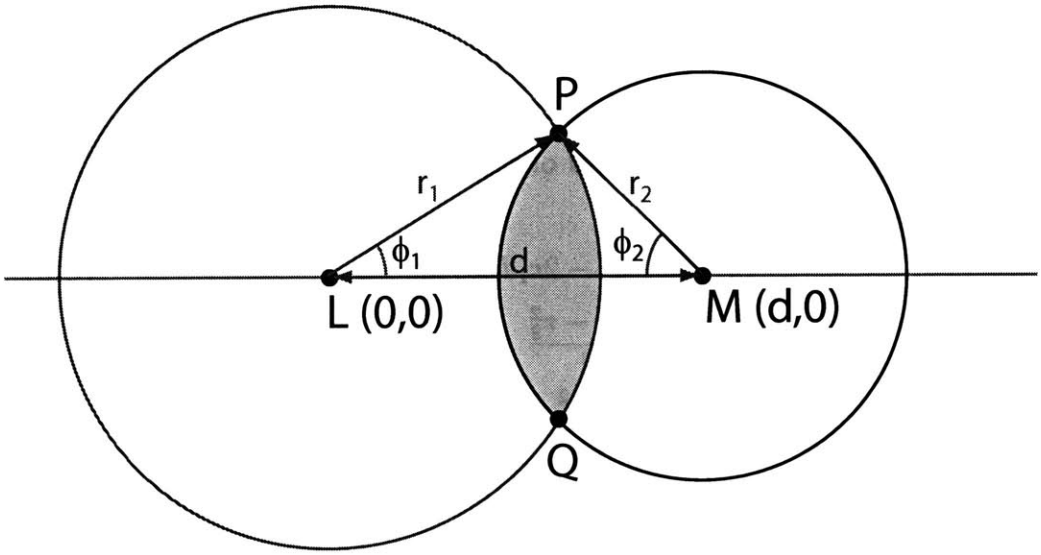

Figure C-1: Two intersecting circles of arbitrary size and spacing

Assume that both circles have centers on the x-axis. If $d \geq r_1 + r_2$ then $A = 0$ because the circles do not overlap. If $(\min(r_1, r_2) + d) \leq \max(r_1, r_2)$ then one circle

lies entirely within (or coincident with) the other. Thus $A = \pi \times (\min(r_1, r_2))^2$.

For cases of partial overlap, let C be the simple closed curve bounding the intersection of the two circles. Then the area of intersection is given by:

$$A = \oint_C x\,dy \tag{C.1}$$

If C_1 and C_2 are the right and left-hand arcs respectively, and C is defined by the superposition of C_1 and C_2, then:

$$A = \oint_C x\,dy = \int_{C_1} x\,dy + \int_{C_2} x\,dy \tag{C.2}$$

Let ϕ_1 be the angle $\angle PLM$. C_1 is described by:

$$x = r_1 \cos\phi \qquad -\phi_1 \leq \phi \leq \phi_1 \tag{C.3}$$

$$y = r_1 \sin\phi \qquad -\phi_1 \leq \phi \leq \phi_1 \tag{C.4}$$

The angle ϕ_1 is determined using the law of cosines.

$$r_2^2 = r_1^2 + d^2 - 2r_1 d\cos\phi_1 \tag{C.5}$$

$$\cos\phi_1 = \frac{r_1^2 + d^2 - r_2^2}{2r_1 d} \tag{C.6}$$

$$\phi_1 = \cos^{-1}\left(\frac{r_1^2 + d^2 - r_2^2}{2r_1 d}\right) \tag{C.7}$$

Thus,

$$\begin{aligned}\int_{C_1} x\,dy &= \int_{-\phi_1}^{\phi_1} r_1^2 \cos^2\phi\,d\phi \\ &= r_1^2\left(\phi_1 + \frac{\sin 2\phi_1}{2}\right)\end{aligned} \tag{C.8}$$

Likewise, let ϕ_2 be the angle $\angle PML$. C_2 is described by:

$$x = d + r_2 \cos\phi \qquad (\pi - \phi_2) \leq \phi \leq (\pi + \phi_2) \tag{C.9}$$

$$y = r_2 \sin\phi \qquad (\pi - \phi_2) \leq \phi \leq (\pi + \phi_2) \tag{C.10}$$

The angle ϕ_2 is also determined using the law of cosines.

$$r_1^2 = r_2^2 + d^2 - 2r_2 d\cos\phi_2 \tag{C.11}$$

$$\cos\phi_2 = \frac{r_2^2 + d^2 - r_1^2}{2r_2 d} \tag{C.12}$$

$$\phi_2 = \cos^{-1}\left(\frac{r_2^2 + d^2 - r_1^2}{2r_2 d}\right) \tag{C.13}$$

Thus,

$$\begin{aligned}\int_{C_2} x\,dy &= \int_{\pi-\phi_2}^{\pi+\phi_2} (d + r_2\cos\phi)\,(r_2\cos\phi)\;d\phi \\ \int_{C_2} x\,dy &= \int_{\pi-\phi_2}^{\pi+\phi_2} \left(r_2 d\cos\phi + r_2^2\cos^2\phi\right)\;d\phi \\ &= -2r_2 d\sin\phi_2 + r_2^2\left(\phi_2 + \frac{\sin 2\phi_2}{2}\right)\end{aligned} \tag{C.14}$$

Finally, the area of intersection is evaluated by substituting the results from C.8 and C.14 into C.2.

Appendix D

Known Limitations To 3-D Process Models for GTX and Suggestions for Future Development

The following future improvements to the 3-D Process Models for GTX described in chapter 3 are suggested.

1. The parameter `k_angle_sidewall_via_intertier_`X is implemented as a `double` corresponding to all sidewall angles in a particular inter-tier via. In practice, there are multiple sidewall angles. For example, in a post-bond via flow, the sidewall angle for the masked part of the inter-tier via is often different from that of the unmasked part. The existence of two distinct sidewall angles may be represented by making `k_angle_sidewall_via_intertier_`X a parameter of type `vector<double>` of the form $\{\phi_{SW:3DvXa}, \phi_{SW:3DvXb}\}$.

2. For post-bond via processes, it would be desirable to allow for representation of metal levels formed after inter-tier via definition. This would allow for high-level metal to physically reside above an inter-tier via cut.

3. The current representation of post-bond via directionality is very limiting. It

would be desirable to allow for post-bond inter-tier vias to land on both sides of an intermediate tier.

4. If it were possible to represent information pertaining to multiple tiers in a single vector, then studies in which number of tiers is varied would be facilitated. However, this is made difficult by the GTX limitation described in appendix A, item 1.

5. Registration error would be better modeled if the parameters `dspace_minins`, `dspace_minland` and `dspace_bpmargin` were made into vectors describing level-dependent margins. The same comment applies to expansional errors in tier-to-tier alignment.

6. Additonal multipliers for sigma in `k_volumemodel` would be desirable. Perhaps "`threesigma`" could be replaced with "`sigma`X" where $X = 3$ is the desired multiplier. The netErr() function would then parse the token of `k_volumemodel` and treat the sigma multiplier as a parameter rather than a constant.

Appendix E

Summary of Parasitic Capacitance Model Equations

Parasitic capacitance modeling is extensively discussed in the literature.[5, 8, 13, 14, 15, 17, 21, 23, 32, 38, 40, 41, 44, 46, 49, 50, 53, 54]. Models used in this work are summarized in this appendix.

E.1 Notation

For the purposes of describing these models, we can define the following parameters. Subscripts are used to denote the respective metal levels.

$\frac{C}{\epsilon}$ = normalized capacitance

ϵ = dielectric permittivity

W = width of metal line

T = thickness of metal line

H = thickness of dielectric layer between metal levels

S = clear spacing between parallel lines on same level

D = adjacent wire spacing for lines on different levels

E.2 Survey of Empirical Models

Chern et al. proposed a set of empirical capacitance models for dense multilevel metal stuctures.[13] In this model, the three capacitances illustrated in figure E-1 are evaluated by decomposition of a layout into primitive structures. The models in [13] have the following range of validity.

$$0.3 \leq \frac{W}{H} \leq 10, \tag{E.1}$$

$$0.3 \leq \frac{S}{H} \leq 10 \rightarrow \text{cut off } \frac{S}{H} = 10, \tag{E.2}$$

$$0.3 \leq \frac{T}{H} \leq 10 \tag{E.3}$$

Delorme et al. have reported an improved model for capacitances of one, two and three lines over a ground plane. These better represent the narrow wires and thick dielectrics of deep submicron processes. They are valid for $\frac{W}{H}, \frac{T}{H}, \frac{S}{H} > 0.02$. However the upper limits for these ratios is significantly lower than that of [13].

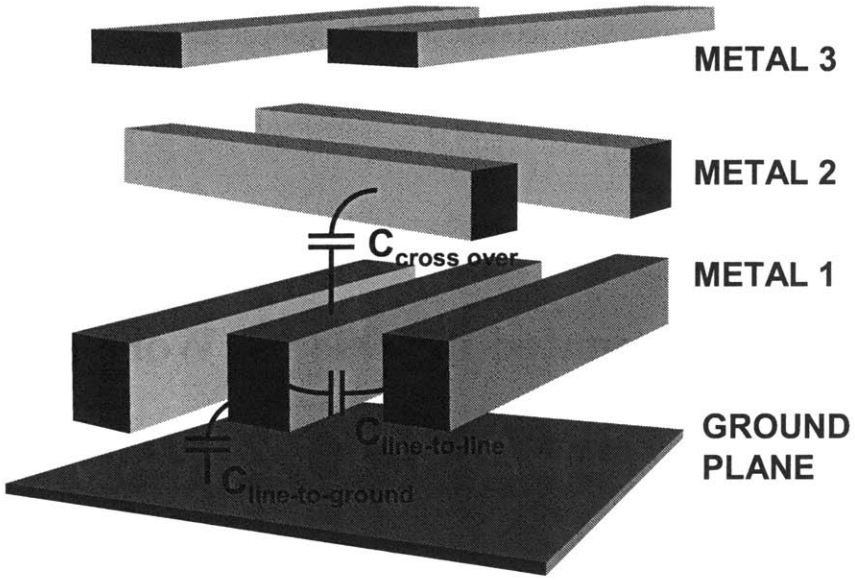

Figure E-1: Parasitic capacitance components

Wong et al. introduced a set of five capacitance models [54]. In addition to parallel lines on a ground plane, parallel lines between ground planes and crossover lines on different levels, additional models for nonoverlapping parallel lines on different levels and parallel lines on different levels are included. The range of validity is as follows:

$$0.12 \leq \frac{W}{H} \leq 4.9, \tag{E.4}$$

$$0.12 \leq \frac{S}{H} \leq 27, \tag{E.5}$$

$$0.12 \leq \frac{D}{H} \leq 27, \tag{E.6}$$

$$0.05 \leq \frac{T}{H} \leq 3.2 \tag{E.7}$$

Although the models of Wong et al. are not as deeply scalable as those of Delorme et al., they will be primarily used for this work. This decision was made because a complete set of models is available in the literature and the models are applicable to the existing MITLL prototype 3-D technology.[2, 3] More recent formulae for interconnect capacitances can also be found in the literature.[44, 53] Today, however,

many improvements to empirical models are of a proprietary nature. As a result, detailed descriptions of model equations are not as readily available.

E.3 Models Presented in 1998 by Wong et al.[54]

Figure E-2 shows the primitive geometries modeled by [54]. In this presentation, the various capacitance terms are indicated using the notation of figures E-1. For the case of parallel lines on the same level over a ground plane, as illustrated in figure E-2(a):

$$\frac{C_{line-to-ground}}{\epsilon} = \frac{W}{H} + 2.977\left(\frac{T}{H}\right)^{0.232} \tag{E.8}$$

$$\frac{C_{line-to-line}}{\epsilon} = \left(0.229\left(\frac{W}{S}\right) + 1.227\left(\frac{T}{S}\right)^{1.384}\right)\left(\frac{H}{T}\right)^{0.398} \tag{E.9}$$

As $S \rightarrow \infty$, equation E.9 reduces to zero, corresponding to a single line structure. For parallel lines between two ground planes, as illustrated in figure E-2(b):

$$\frac{C_{line-to-ground}}{\epsilon} = \frac{W}{H} + 1.637\left(\frac{T}{H_1}\right)^{0.216} + 1.637\left(\frac{T}{H_2}\right)^{0.216} \tag{E.10}$$

$$\frac{C_{line-to-line}}{\epsilon} = \left(0.053\left(\frac{W}{S}\right) + 1.348\left(\frac{T}{S}\right)^{1.597}\right)\left(\frac{H}{T}\right)^{0.628} \tag{E.11}$$

where

$$H = \frac{H_1 H_2}{H_1 + H_2} \tag{E.12}$$

For the case of overlapping parallel lines on different levels, as illustrated in figure E-2(c):

$$\frac{C_{line-to-ground}}{\epsilon} = 1.25\left(\frac{W_1 - S_{ov}}{H_1 + H_2 + T_2}\right) + 2.919\left(\frac{T_1}{H_1 + H_2 + T_2}\right)^{0.25} \tag{E.13}$$

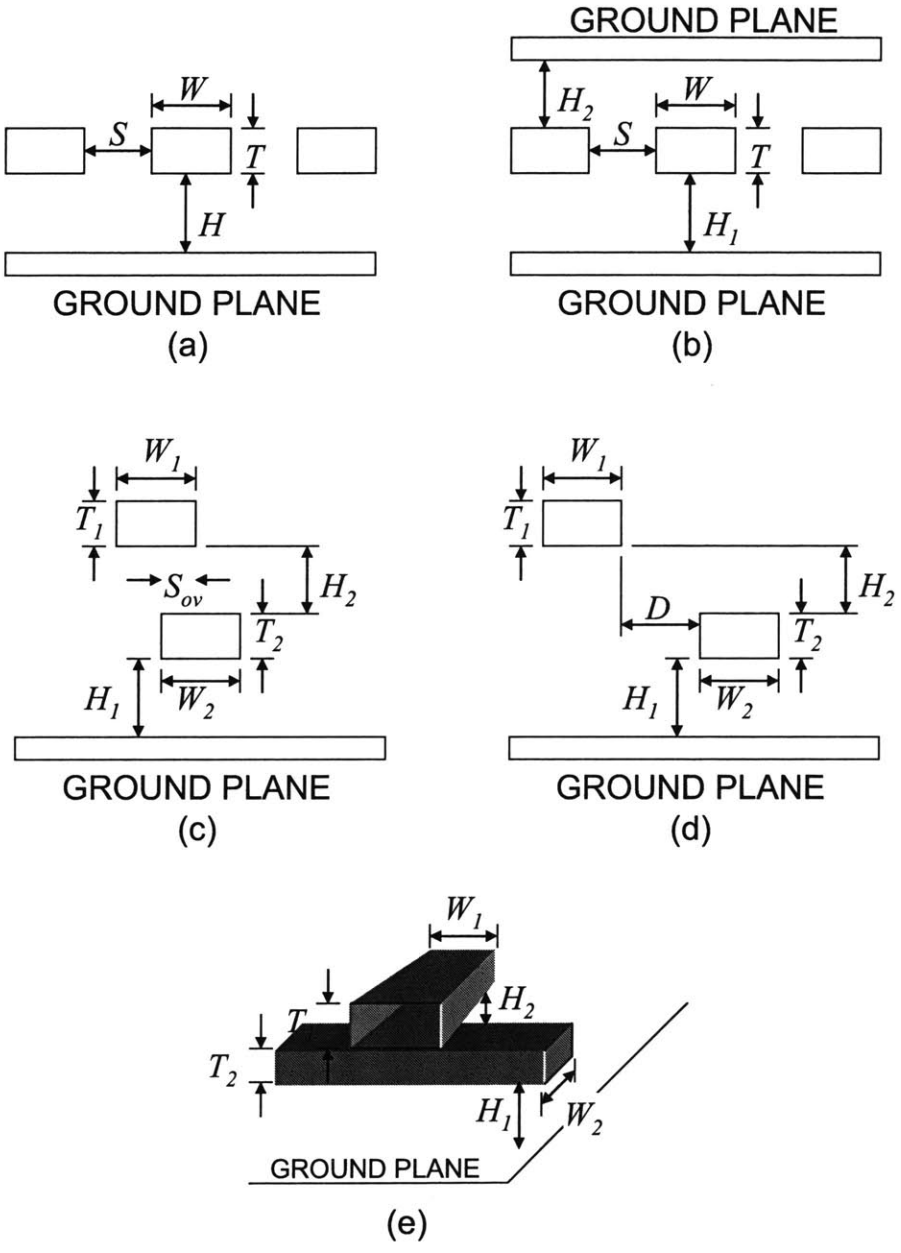

Figure E-2: Primitive capacitance structures [54]

$$\frac{C_{line-to-line}}{\epsilon} = 0.906\left(\frac{S_{ov}}{H_2}\right) + \left(\frac{H_1}{H_2}\right)^{0.649}\left[0.198\left(\frac{T_1}{W_2 - S_{ov} + 1}\right) - 0.447\left(\frac{T_1}{W_2 - S_{ov} + 1}\right)^2 + 2.514\left(\frac{T_2}{W_1 - S_{ov} + 1}\right) - 2.883\left(\frac{T_2}{W_1 - S_{ov} + 1}\right)^2\right] \quad \text{(E.14)}$$

For the case of nonoverlapping parallel lines on different levels, as illustrated in figure E-2(d):

$$\frac{C_{line-to-ground}}{\epsilon} = 1.16\left(\frac{W_1}{H_1 + H_2 + T_2}\right) + 2.704\left(\frac{T_1}{H_1 + H_2 + T_2}\right)^{0.204} \quad \text{(E.15)}$$

$$\frac{C_{line-to-line}}{\epsilon} = \left(\frac{H_1}{H_2}\right)^{0.446}\left[0.7076\left(\frac{W_1}{W_1 + 2D}\right)\left(\frac{T_2}{H_2}\right)^{0.247} + 0.162\left(\frac{W_2}{W_2 + 2D}\right)\left(\frac{T_1}{H_2}\right)^{0.458}\right] \quad \text{(E.16)}$$

For the case of crossover lines on different levels, as illustrated in figure E-2(e):

$$\frac{C_{line-to-line}}{\epsilon} = 3.285\left(\frac{W_1 W_2}{H_2}\right) + W_1\left[4.505\left(\frac{T_2}{T_2 + 0.2H_2}\right) - 4.348\left(\frac{T_2}{T_2 + 0.2H_2}\right)^2\right] + W_2\left[4.505\left(\frac{T_1}{T_1 + 0.2H_2}\right) - 4.348\left(\frac{T_1}{T_1 + 0.2H_2}\right)^2\right] + 1.532\left[T_2\left(\frac{W_1}{W_1 + 0.5H_2}\right)^{2.56} - T_1\left(\frac{W_2}{W_2 + 0.2H_2}\right)^{2.54}\right] \quad \text{(E.17)}$$

And, as previously mentioned, equations E.8 through E.17 are valid for parameter ranges given by E.4 through E.7.

E.4 Nearbody capacitors with no ground plane

In some cases, when $H \gg S$ the interconnect is best modeled as if there were no groundplane. For the case of two parallel wires with no groundplane, the following equation is useful[49]

$$\frac{C_{line-to-line}}{\epsilon} = \begin{cases} \frac{T}{S} + \frac{\pi\left(1-0.0543\left(\frac{W}{S}\right)\right)}{2\ln\left(1+\frac{S}{W}+\sqrt{\frac{S}{W}\left(\frac{S}{W}+2\right)}\right)} + 0.735, & W > 2T \\ \frac{2T-W}{2S} + \frac{2\pi}{2\ln\left(1+\frac{S}{W}+\sqrt{\frac{S}{W}\left(\frac{S}{W}+2\right)}\right)}, & W \leq 2T \end{cases} \tag{E.18}$$

Appendix F

3-D Fabrication Technologies

Two 3-D integration processes are discussed in this section. Both are based on wafer bonding. The first, developed at MIT Lincoln Laboratory (MITLL)[2, 9], uses inter-tier vias that are etched after the wafers corresponding to the connected tiers have been bonded. The second, under development at the Microsystem Technology Laboratories (MTL) at MIT, uses copper wafer bonding[19, 36, 37]. In this case, part of the inter-tier via is formed on each of the wafers prior to bonding. The two surfaces are subsequently mated, forming the interconnected tiers. The method of 3-D via formation has a direct effect on the available area on each tier for pixel implementation.

F.1 The MITLL Post-bond 3-D Via Process

Note: Much of this discussion of the 3-D assembly process flow has been taken from the MITLL process documentation.[2, pages 86–89]

The MITLL 3-D process is based on a fully-depleted silicon-on-insulator (FDSOI) technology. The 3-D stacking process may be applied to a wide range of applications and substrates. For this discussion, an imager application will be assumed. This example structure consists of two FDSOI circuit tiers and one bulk photodiode tier.

Prior to 3-D stacking, these FDSOI and bulk wafers are processed independently. These three initial wafers are illustrated in figure F-1.

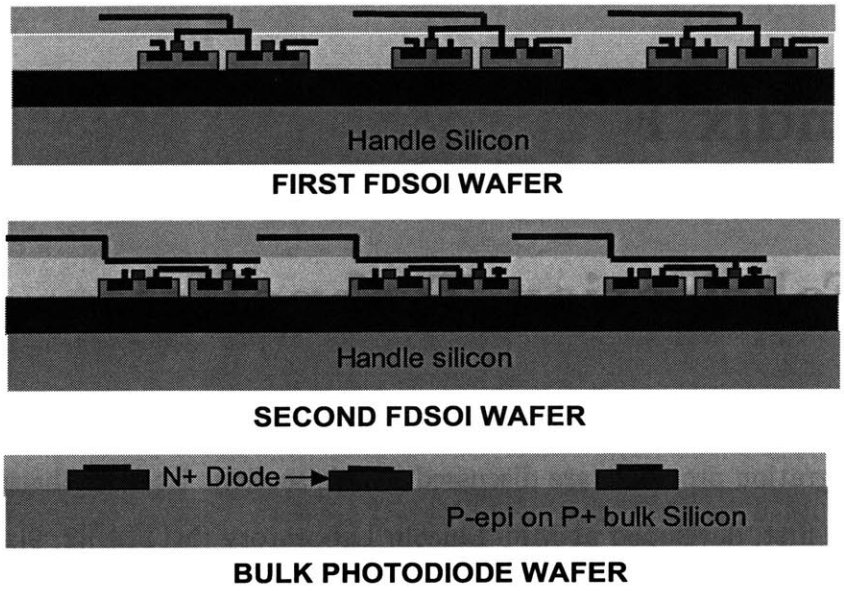

Figure F-1: Individually processed SOI and bulk wafers

For this particular structure, the photodiode wafer will be used to form tier 1 of the final 3-D structure. The FDSOI circuits will be on tiers 2 and 3. To eliminate confusion, it is necessary to define a particular side of the 3-D assembly that will be considered to be the top. For this case, tier 1 is defined as being on the top tier of the 3-D stack. As the following discussion of the assembly process proceeds, it will become apparent that if tier 1 is considered to be the top tier, then tier 1 is flipped with respect to the final 3-D assembly, and tiers 2 and 3 are not flipped.

The 3-D process flow proceeds as follows. The assembly is done upside-down and inverted at the end. First the wafer from which tier 1 is to be formed, in this case the photodiode wafer, is selected to serve as the substrate for the assembly process. The wafer from which tier 2 is to be formed is inverted, aligned, and bonded to the photodiode layer as indicated in figure F-2.

The silicon substrate is removed from the tier 2 wafer, exposing the buried oxide

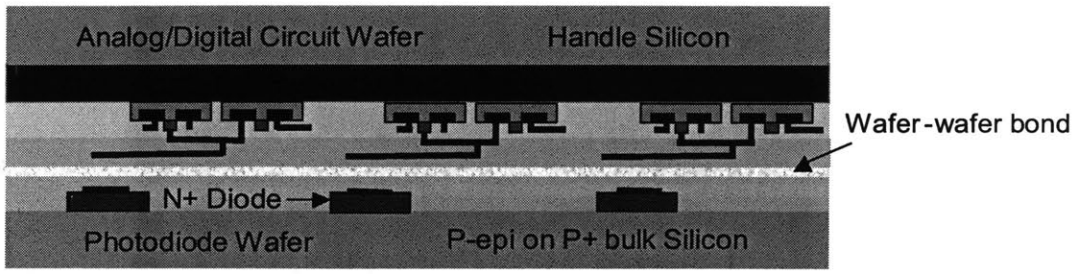

Figure F-2: First wafer bonding step

(BOX). Then 3-D vias are etched, and tungsten is deposited and planarized using chemical-mechanical polishing (CMP). In the resulting structure, shown in figure F-3, the 3-D vias connect the top-level metal of the tier 2 FDSOI circuit to the top-level metal of the tier 1 photodiode circuit.

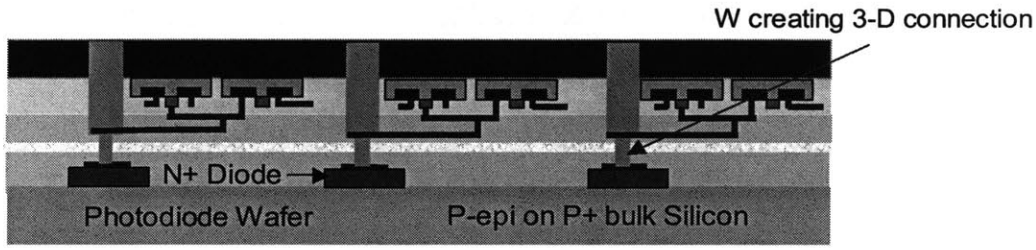

Figure F-3: First set of 3-D vias

The same process is then repeated for tier 3. The wafer that is to be used to form tier 3 is inverted, aligned, and bonded to the tier 1 - tier 2 assembly, as shown in figure F-4. The silicon substrate is removed, 3-D vias are etched and tungsten is deposited and planarized using CMP. In the resulting structure, shown in figure F-5, the 3-D vias connect the top level metal of tier 3 to the first level metal of tier 2.

The entire assembly is then inverted and bonded to a carrier wafer. The silicon of the tier 1 photodiode wafer is thinned. Bond pads are etched through the top of the structure to the tier 1 first-level metal layer. The completed imager structure is illustrated in figure F-6.

The inter-tier vias used to connect the multiple tiers of the 3-D circuit are presently

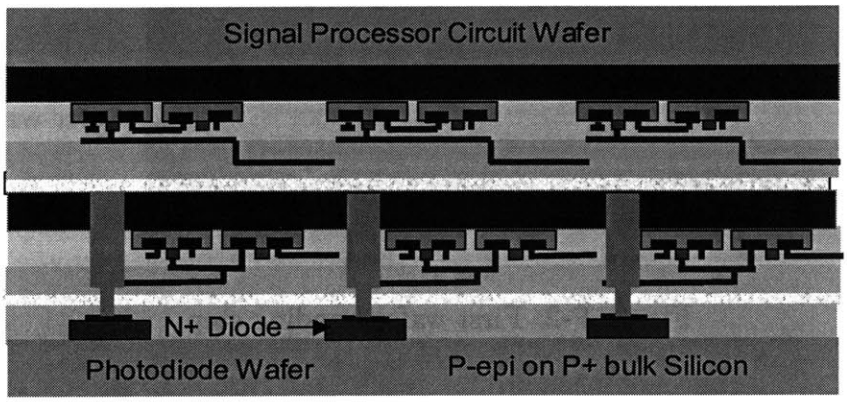

Figure F-4: Second wafer bonding step

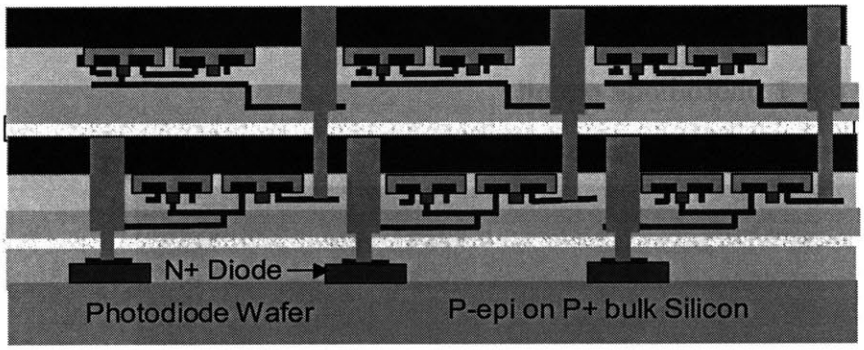

Figure F-5: Second set of 3-D vias

on the order of a micron. When compared to indium bump bond technology, this represents an improvement in area utilization of two to three orders of magnitude. Advantages of this 3-D integration technology include better circuit to interconnect ratio, high density interconnect between wafer tiers, and reduced digital system power. Multiple material systems and process technologies may also be integrated into the same 3-D system.

Table F.1 provides a listing of design rules relevant to the MITLL 3-D process.[2, pages 61–63][3, page 2] There are two sets of constraints, one corresponding to a conservative design rule set, and one that is more aggressive, in anticipation of future tool capabilities, and perhaps at the cost of decreased yield. In the MITLL rules set, 3-D vias are defined by features on the *3DCUT* layer of the tier through which the

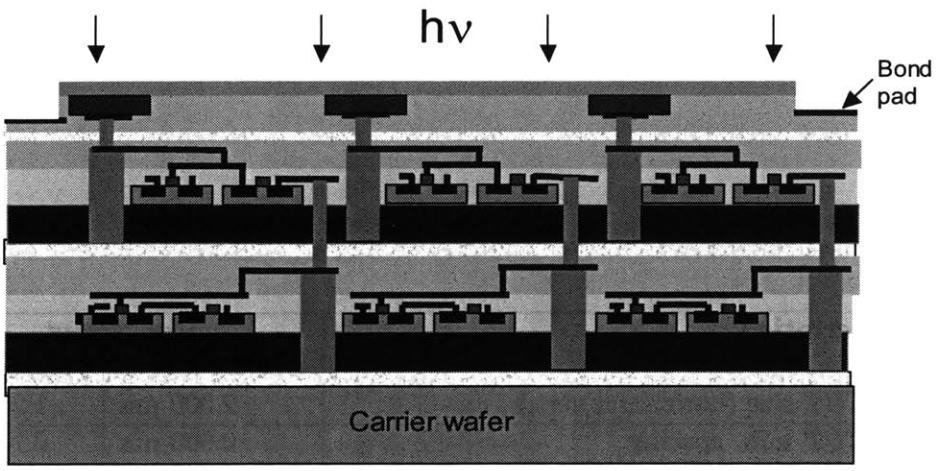

Figure F-6: 3-D imager configured for backside imaging

via is cut and a corresponding feature on the *3DLAND* layer of the tier onto which the 3-D via lands. For these rules, "masking metal" refers to the top-level metal of the tiers that are not flipped. This masking metal must include a donut feature as specified by these rules to ensure proper formation of the 3-D via. The donut serves as a hard mask during the 3-D via etch and also provides electrical contact to the via. The rules listed in table F.1 assume that the layout of neither tier is flipped. (I.e., the rules apply to the situation of tiers 2 and 3 of the structure in figure F-6.) Also, note that these rules (with the exception of 41.11 and 42.13) pertain to physical verification of the layout database for a single tier, not a multi-tier stack.

Figure F-7 shows example layouts of 3-D vias implemented using these design rules. From these illustrations, it is readily apparent that the primary limitation of the 3-D integration technology is wafer-to-wafer misalignment. Misalignment is due to a combination of effects, each of which may contribute translational, rotational, or expansional errors. This poor alignment tolerance forces the implementation of a large landing pad on the mating metal level. For the case where the landing is made to the underside of the first metal level as opposed to the top level metal, a substantial exclusion zone within which active devices may not be instantiated is required. This

Rule No.	Description	Constraint	
		Conservative	Aggressive
41.1,2	*3DCUT* size (horiz. and vert.)	2.000 µm	1.250 µm
41.3,4	*3DCUT* min. spacing	0.900 µm	0.700 µm
41.5,6	Min. masking metal surround on *3DCUT*	0.250 µm	0.125 µm
41.7,8	*3DCUT* surround on donut opening in masking metal (only size allowed)	0.250 µm	0.125 µm
41.9,10	Min. *3DCUT* spacing to *3DLAND* on same tier	2.750 µm	1.650 µm
41.11	*3DCUT* not corresponding to *3DLAND* on mating tier	prohibited	
41.12*	Min. *3DCUT* spacing to *POLY*	1.000 µm	–
41.13*	Min. *3DCUT* spacing to *ACTIVE*	1.075 µm	–
41.14*	Min. *3DCUT* spacing to non-masking metal	0.800 µm	–
42.1,2	*3DLAND* size (horiz. and vert.)	2.000 µm	1.250 µm
42.3,4	*3DLAND* min. spacing on common mating metal	0.900 µm	0.700 µm
42.5,6	*3DLAND* min. spacing on isolated mating metal	3.850 µm	2.100 µm
42.7,8	Min. mating metal surround on *3DLAND*	1.750 µm	0.875 µm
42.9,10	Min. *3DLAND* spacing to *ACTIVE* or *POLY*	2.250 µm	1.375 µm
42.13	*3DLAND* not corresponding to *3DCUT* on mating tier	prohibited	

Table F.1: Version 5.11 (Jan. 2002) and *version 6.mosaic (July 2003) MITLL 3-D process design rules

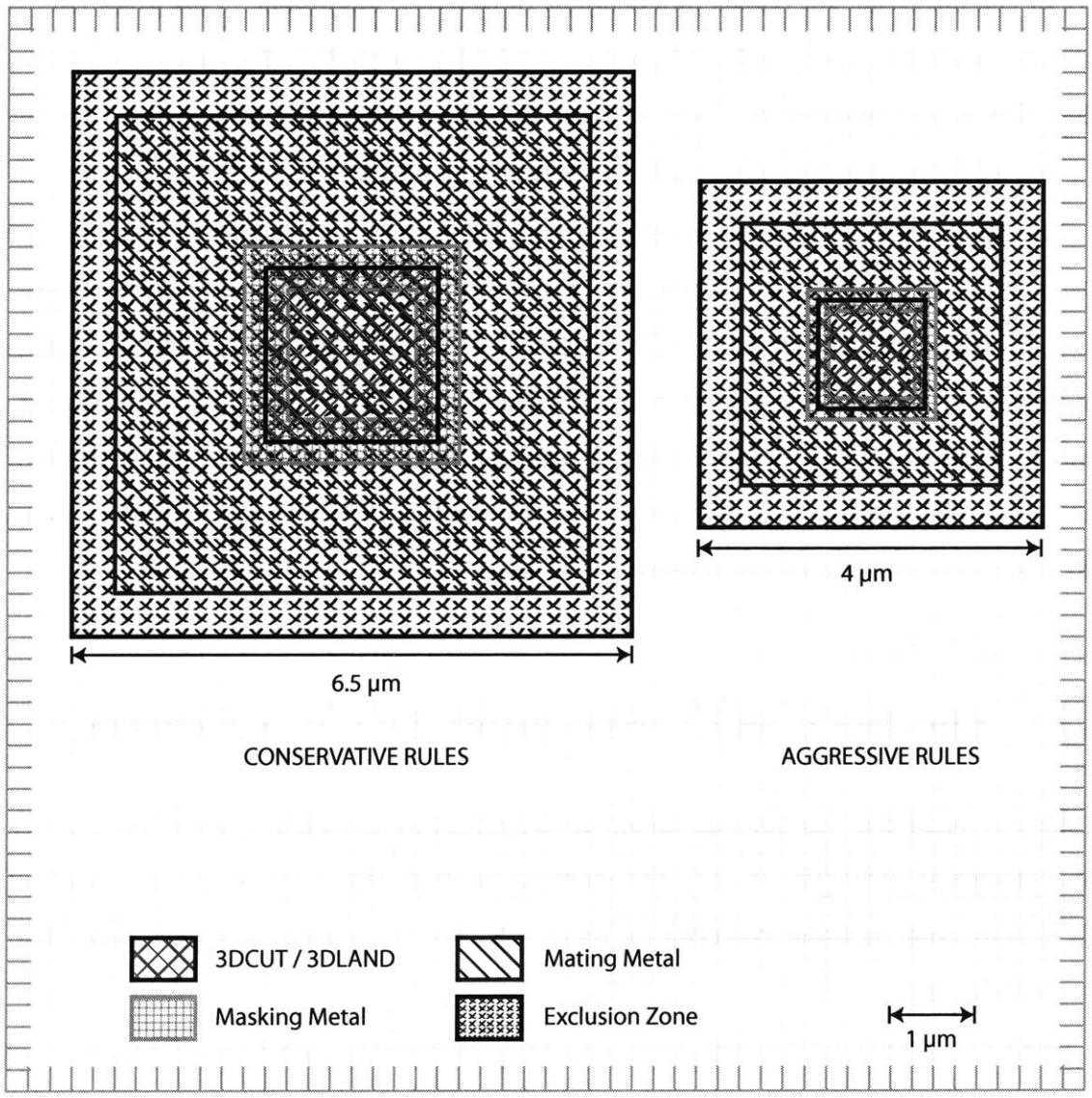

Figure F-7: 3-D via layout examples

is not necessary when the inter-tier via lands on the top side of the top metal level. On the tier through which the inter-tier via cuts, *3DCUT* aligns directly to the masking metal, making the exclusion zone on that tier for layers through which the narrow part of the inter-tier via passes independent of *3DCUT* alignment. This gives rise to a smaller exclusion zone, defined by rules 41.12-14 in table F.1. For the conservative rules, this is on the order of 1 µm, but could be reduced to something on the order of a conventional metal-metal spacing in a more aggressive rule set.

These exclusion zones significantly reduce the available circuit area on each tier and thus limit the versatility of post-bond 3-D via processes. However, manufacturability concerns make post-bond 3-D via processes significantly more realizable than pre-bond 3-D via processes. It is reasonable to expect that the availability of post-bond 3-D via processes will lead that of pre-bond 3-D via processes. An analysis of 3-D circuit design and process tradeoffs must therefore consider any exclusion zones that are a consequence of the integration scheme.

F.2 The MIT MTL Pre-bond 3-D Via Process

In the pre-bond 3-D via process under development at the Microsystem Technology Laboratories at MIT [19, 36, 37], the starting point is also a set of initial processed SOI wafers, such as those pictured in figure F-8. For the sake of this discussion, two tiers are assumed.

Prior to wafer bonding, one of these wafers is inverted and bonded to a carrier wafer. This wafer will correspond to tier 1. The handle silicon of the tier 1 wafer is removed, and vias are etched to make contact with an interconnect level on this first tier. This is illustrated in figure F-9. Copper bonding pads are formed on each wafer, as shown in figure F-10 In the present MIT MTL implementation [37], the contact pads consist of a 300-nm Cu layer and a 50-nm Ta layer. These are passivated using plasma-enhanced chemical vapor deposited (PECVD) oxide followed by planarization

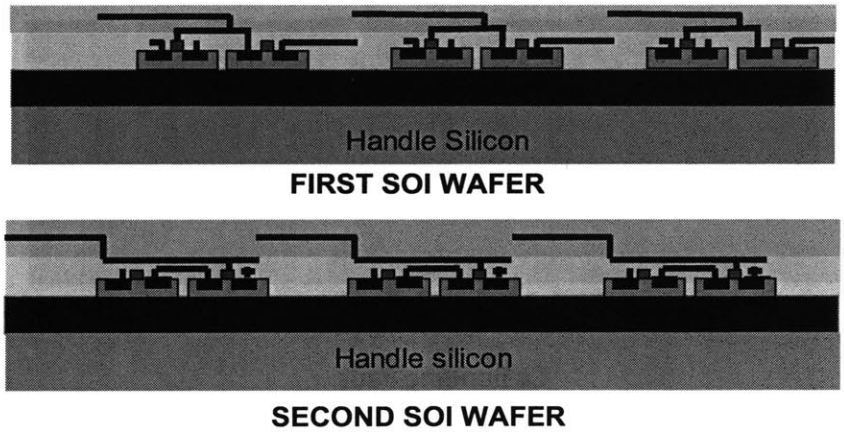

Figure F-8: Initial set of processed SOI wafers

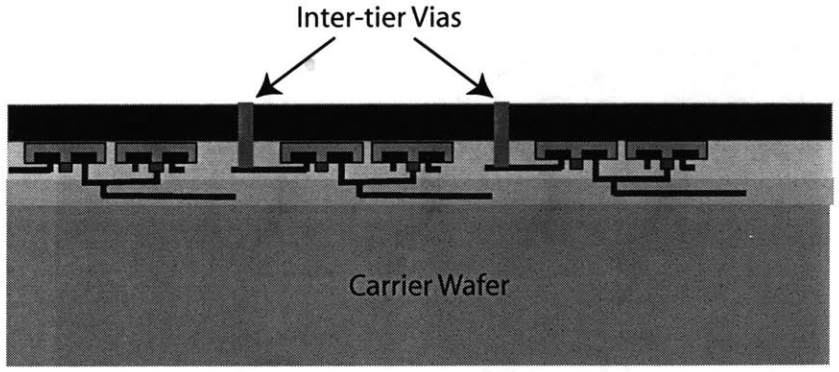

Figure F-9: Definition of inter-tier vias

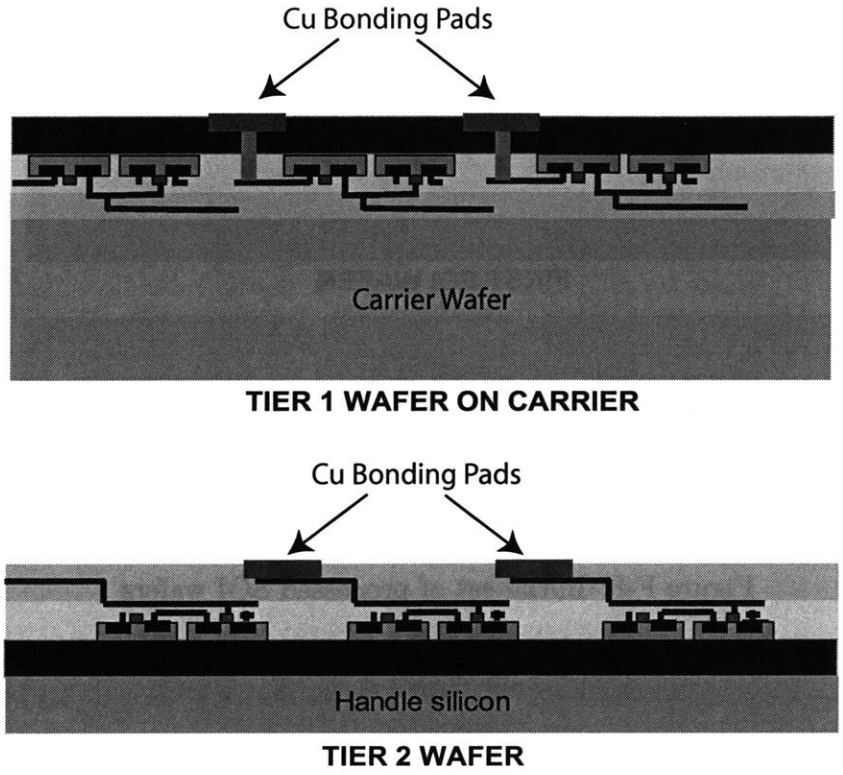

Figure F-10: Formation of Cu bond pads

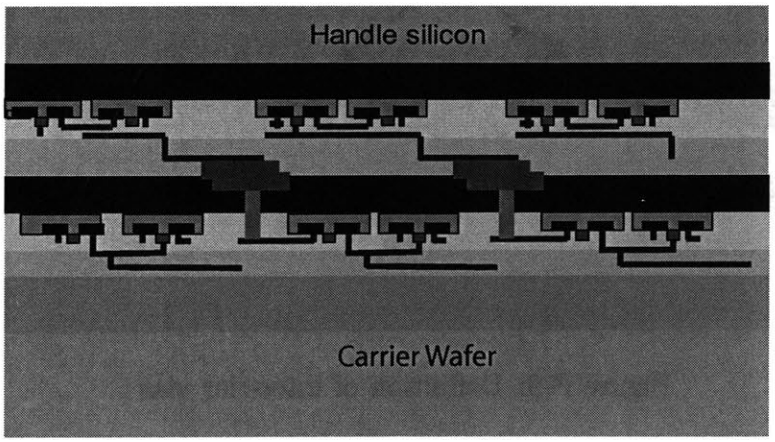

Figure F-11: Two-tiered structure after Cu-Cu bonding

using CMP. The two surfaces are subsequently mated using low-temperature Cu-Cu thermocompression, forming the interconnected tiers. The bonded structure is illustrated in figure F-11

Since the MIT MTL process is not yet available for circuit prototyping, formal design rules are not available. However, it is possible to form an understanding of the origins of future design rules based on process considerations. In [37], 500-µm inter-tier vias are used to achieve a via aspect ratio between 2:1 and 3:1. For the sake of consistency, these will be indicated by the drawn layer *3DCUT*. This via is directly aligned to a *METAL1* level, thus requiring a *METAL1* surround comparable to that for a standard via. Because the 500-µm inter-tier via size is determined by via etch aspect ratio, it is to be expected that the via size will scale according to buried oxide (BOX) thickness. Hence, it can be assumed that the size of this via is on the order of a standard intra-tier local interconnect via. The required *ACTIVE* and *POLY* exclusion zone is determined by *3DCUT*-to-(*ACTIVE* OR *POLY*) alignment error plus applicable CD error. In many cases, this results in an exclusion zone that is only slightly larger than the minimum contact-to-gate spacing.

On each tier, the copper bonding pad size and spacing is determined in large part by alignment considerations. In [37] the wafer-to-wafer alignment system has a 3-µm 3-σ misalignment. This is projected to improve. Since the origin of this misalignment error is the same as that seen for the MITLL process, it is useful to compare the resulting Cu bond pad minimum size and spacing to rules listed in table F.1. For the MITLL post-bond 3-D via process, the alignment tolerance is ± 2 µm for the conservative case and ± 1 µm for the aggressive case. Subtracting expected tapering of the inter-tier via, and adding additional CD variation and photomask registration considerations one obtains Rule 42.7,8: "min. mating metal surround on *3DLAND*."

For the case of Cu-Cu bond pads, the rule for minimum pad size is a function of both alignment tolerance and landing tolerance. For the sake of a simple example, assume one-dimensional misalignment. If w_{min} is the minimum width of a Cu bond

pad, k_{land} is the fraction of the pad width (in one dimension) that must be landed and Δx_{align} is worst-case misalignment, then $w_{min} \approx \frac{\Delta x_{align}}{1-k_{land}}$. In section 3.2.6, a more realistic calculation of minimum bond pad size that takes into account two-dimensional misalignment is discussed.

The rule for minimum pad spacing is also determined primarily by the alignment tolerance. To this one must add additional margin for other process effects to ensure electrical isolation of independent inter-tier connections. In addition to the sizing and spacing rules, a Cu bond pad density rule is required to ensure mechanically reliable bonding.

To summarize, the MIT MTL process flow potentially offers significant reductions in the size of the circuit exclusion zone in the vicinity of the inter-tier via. 3-D via density, however, is still limited by the wafer-wafer alignment tolerance. Further integration work is necessary before this process is available for large circuit prototyping.

Appendix G

Availability of Code Produced as Part of This Work

To facilitate further development of GTX resources pertaining to 3-D design, the GTX "knowledge" produced as part of this work will be made available to the public. This includes parameter files, rule files and scripts. A link to this archive will be posted at the following World Wide Web address:

`http://www-mtl.mit.edu/~reif/project/projects.htm`

Public release of this material is subject to a timetable to be determined by applicable release review procedures. It is estimated that the code will be made available to the public in mid-summer of 2004.

Bibliography

[1] Verilog-AMS language reference manual: Analog & mixed-signal extensions to verilog HDL, version 2.1. Accellera, 1370 Trancas St., #163. Napa, CA 94558, January 2003.

[2] MITLL 0.18 μm low power FDSOI CMOS process design guide, version 5.11. Advanced Silicon Technology Group, MIT Lincoln Laboratory, 244 Wood St. Lexington, MA 02421, January 2002.

[3] MITLL 0.18 μm low power FDSOI CMOS process design guide, version 6.mosaic. Advanced Silicon Technology Group, MIT Lincoln Laboratory, 244 Wood St. Lexington, MA 02421, July 2003.

[4] Syed M. Alam, Donald E. Troxel, and Carl V. Thompson. A comprehensive layout methodology and layout-specific circuit analyses for three-dimensional integrated circuits. In *Proc. IEEE ISQED*, pages 246–251, 2002.

[5] Narain D. Arora, Kartik V. Raol, Reinhard Schumann, and Llanda M. Richardson. Modeling and extraction of interconnect capacitances for multilayer vlsi circuits. *IEEE Trans. Computer-Aided Design*, 15(1):58–67, January 1996.

[6] B. F. Aull, A. H. Loomis, J. A. Gregory, and D. J. Young. Geiger-mode avalanche photodiode arrays integrated with CMOS timing circuits. In *IEEE 56th Ann. Device Res. Conf. Dig.*, pages 58–59, 1998.

[7] Brian F. Aull and Alvin Stern. Unpublished data: 3-D LADAR imager. Advanced Imaging Technology Group, MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02421, 2001 - 2004.

[8] Erich Barke. Line-to-ground capacitance calculation for VLSI: A comparison. *IEEE Trans. Computer-Aided Design*, 7(2):295–298, February 1988.

[9] J. Burns, L. McIlrath, C. Keast, C. Lewis, A. Loomis, K. Warner, and P. Wyatt. Three-dimensional integrated circuits for low-power, high-bandwidth systems on a chip. In *Proc. IEEE ISSCC*, pages 268–269,453, 2001.

[10] Andrew Caldwell, Andrew B. Kahng, Igor Markov, and Mike Oliver. *MARCO GSRC GTX Documentation.* MARCO Gigascale Silicon Research Center (GSRC), http://www.gigascale.org/gtx/, rev. 2.1 edition, December 2001.

[11] Andrew E. Caldwell, Yu Cao, Andrew B. Kahng, Farinaz Koushanfar, Hua Lu, Igor L. Markov, Michael Oliver, Dirk Stroobandt, and Dennis Sylvester. Gtx: The marco gsrc technology extrapolation system. In *Proc. DAC*, 2000.

[12] Victor W. C. Chan, Philip C. H. Chan, and Mansun Chan. Three dimensional CMOS integrated circuits on large grain polysilicon films. In *Proc. IEEE IEDM*, pages 161–164, 2000.

[13] Jue-Hsien Chern, Jean Huang, Lawrence Arledge, Ping-Chung Li, and Ping Yang. Multilevel metal capacitance models for CAD design synthesis systems. *IEEE Electron Device Lett.*, 13(1):32–34, January 1992.

[14] Umakanta Choudhury and Alberto Sangiovanni-Vincentelli. Automatic generation of analytical models for interconnect capacitances. *IEEE Trans. Computer-Aided Design*, 14(4):470–480, April 1995.

[15] Jason Cong, Lei He, Andrew B. Kahng, David Noice, Nagesh Shirali, and Steve H.-C. Yen. Analysis and justification of a simple, practical 2 1/2-D capacitance extraction methodology. In *Proc. DAC*, pages 627–632, June 1997.

[16] Jeffrey A. Davis, Raguraman Venkatesan, Alain Kaloyeros, Michael Beylansky, Shukri J. Souri, Kaustav Banerjee, Krishna C. Saraswat, Arifur Rahman, Rafael Reif, and James D. Meindl. Interconnect limits on gigascale integration (GSI) in the 21st century. *Proc. IEEE*, 89(3):305–324, March 2001.

[17] N. Delorme, M. Belleville, and J. Chilo. Inductance and capacitance analytic formulas for VLSI interconnects. *Electronics Lett.*, 32(11):996–997, 23 May 1996.

[18] P. N. J. Dennis. *Photodetectors: An Introduction to Current Technology.* Updates in Applied Physics and Electrical Technology. Plenum Press, New York, New York, 1986.

[19] A. Fan, A. Rahman, and R. Reif. Copper wafer bonding. *Electrochem. Solid State Lett.*, 2(10):534–536, October 1999.

[20] Covert Geelen. A 6b 1.1GSample/s CMOS A/D converter. In *IEEE ISSCC Dig. Tech. Papers*, pages 128–129. IEEE, February 2001.

[21] Ashok K. Goel. *High-Speed VLSI Interconnections: Modeling, Analysis and Simulation.* Wiley Series in Microwave and Optical Engineering. John Wiley & Sons, Inc., New York, New York, 1994.

[22] Frank Goodenough. Analog technology of all varieties dominate ISSCC. *Electronic Design*, 44(4):96–111, 19 February 1996.

[23] Susumu Kurosawa. An analytical modeling of three primary wiring capacitive components for multi-layer interconnect structure. *IEICE Trans. Fundamentals Electronic Comm. Comp. Sci.*, E78-A(12):1793–1798, December 1995.

[24] Kang Wook Lee, Tomonori Nakamura, Katsuyuki Sakuma, Ki Tae Park, Hiroaki Shimazutsu, Nobuaki Miyakawa, Ki Yoon Kim, Hiroyuki Kurino, and Mitsumasa Koyanagi. Development of three-dimensional integration technology for highly parallel image-processing chip. *Jpn. J. Appl. Phys. Part 1*, 39(4B):2473–2477, April 2000.

[25] Bob Markunas. Mixing signals with 3-D integration. *Semiconduct. Int.*, 25(13):63–68, November 2002.

[26] Advanced Imaging Technology Group, MIT Lincoln Laboratory. Unpublished data: Snapshot mode fast imager. MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02421, 2001 - 2003.

[27] Solid State Division, MIT Lincoln Laboratory. DARPA advanced imager study. MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02421, September 2002.

[28] Lisa G. McIlrath. A low power, low noise, ultra-wide dynamic range CMOS imager with pixel-parallel A/D conversion. In *IEEE Symp. VLSI Circuits Dig. Tech. Papers*, pages 24–27, 2000.

[29] Lisa G. McIlrath and Paul M. Zavracky. An architecture for low-power real time image analysis using 3D silicon technology. In *Proc. SPIE*, volume 3362, pages 184–195, April 1998.

[30] J. M. Miller. Dependence of the input impedance of a three-electrode vacuum tube upon the load in the plate circuit. *National Bureau of Standards Scientific Papers*, 15:367–385, June 1919.

[31] John Lester Miller. *Principles of Infrared Technology: A Practical Guide to the State of the Art.* Van Nostrand Reinhold, New York, New York, 1994.

[32] Zhen-Qiu Ning, Patrick M. DeWilde, and Fred L. Neerhoff. Capacitance coefficients for vlsi metallization lines. *IEEE Trans. Electron Devices*, ED-34(3):644–649, March 1987.

[33] Michiroh Ohmura. 3-D router for irregular channels. In *Proc. IEEE ISCAS*, pages 1692–1695, 1997.

[34] Michiroh Ohmura. An initial placement algorithm for 3-D VLSI. In *Proc. IEEE ISCAS*, volume 6, pages 195–198, 1998.

[35] Arifur Rahman, Andy Fan, James Chung, and Rafael Reif. Wire-length distribution of three-dimensional integrated circuits. In *Proc. Int. Interconnect Technol. Conf.*, pages 233–235, 1999.

[36] Arifur Rahman and Rafael Reif. System-level performance evaluation of three-dimensional integrated circuits. *IEEE Trans. VLSI Syst.*, 8(6):671–678, December 2000.

[37] Rafael Reif, Andy Fan, Kuan-Neng Chen, and Shamik Das. Fabrication technologies for three-dimensional integrated circuits. In *Proc. IEEE ISQED*, pages 33–37, 2002.

[38] Albert E. Ruehli and Pierce A. Brennan. Capacitance models for integrated circuit metallization wires. *IEEE J. Solid-State Circuits*, SC-10(6):530–536, December 1975.

[39] Rob A. Rutenbar. Simulated annealing algorithms: An overview. *IEEE Circuits Devices Mag.*, 5(1):19–26, January 1989.

[40] T. Sakurai and K. Tamaru. Simple formulas for two-and-three-dimensional capacitances. *IEEE Trans. Electron Devices*, ED-30(2):183–185, February 1983.

[41] Takayasu Sakurai. Closed-form expressions for interconnection delay, coupling, and crosstalk in vlsi's. *IEEE Trans. Electron Devices*, 40(1):118–124, January 1993.

[42] Erica M. Salinas. Application of three-dimensional circuit integration to global clock distribution. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, January 2004.

[43] Dean A. Scribner, Melvin R. Kruer, and Joseph M. Killiany. Infrared focal plane array technology. *Proc. IEEE*, 79(1):66–85, January 1991.

[44] Franco Stellari and Andrea L. Lacaita. New formulas of interconnect capacitances based on results of conformal mapping method. *IEEE Trans. Electron Devices*, 47(1):222–231, January 2000.

[45] M. A. Trishenkov. *Detection of Low-Level Optical Signals: Photodetectors, Focal Plane Arrays and Systems.* Solid-State Science and Technology Library. Kluwer Academic Publishers, Boston, Massachusetts, 1997.

[46] N. P. van der Meijs and J. T. Fokkema. VLSI circuit reconstruction from mask topology. *Integration*, 2(2):85–119, June 1984.

[47] R. H. Walden. Analog-to-digital converter technology comparison. In *IEEE GaAs IC Symposium Tech. Dig.*, pages 217–219. IEEE, October 1994.

[48] Ching-Chun Wang. *A Study of CMOS Technologies for Image Sensor Applications.* PhD dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, September 2001.

[49] Yun Wang and Yan Zhou. IC package simulation extractor. Cs 597 final project, University of Bridgeport, Computer Science and Engineering Department, University of Bridgeport, December 2002. http://www.bridgeport.edu/sed/projects/cs597/Fall_2002/yunwang.

[50] William T. Weeks. Calculation of coefficients of capacitance of multiconductor transmission lines in the presence of a dielectric interface. *IEEE Trans. Microwave Theory Tech.*, MTT-18(1):35–43, January 1970.

[51] Hon-Sum Wong. Technology and device scaling considerations for CMOS imagers. *IEEE Trans. Electron Devices*, 43(12):2131–42, December 1996.

[52] Hon-Sum Philip Wong. CMOS image sensors – recent advances and device scaling considerations. In *Proc. IEEE IEDM*, pages 201–4, December 1997.

[53] Shyh-Chyi Wong, Trent Gwo-Yann Lee, Dye-Jyun Ma, and Chuan-Jane Chao. An empirical three-dimensional crossover capacitance model for multilevel interconnect VLSI circuits. *IEEE Trans. Semicond. Manufacturing*, 13(2):219–227, May 2000.

[54] Shyh-Chyi Wong, Patrick S. Liu, Jien-Wen Ru, and Shi-Tron Lin. Interconnection capacitance models for vlsi circuits. *Solid State Electronics*, 42(6):969–977, June 1998.