ERNI-3D: A Technology-Generic Tool for Interconnect Reliability Projections in 3D Integrated Circuits

by

Syed Mohiul Alam

B.S., Electrical Engineering
The University of Texas at Austin, 1999

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2001

Author . . . .
Department of Electrical Engineering and Computer Science
May 15, 2001

Certified by . . . .
Donald E. Troxel
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by . . . .
Carl V. Thompson
Stavros Salapatas Professor of Electronic Materials
Thesis Supervisor

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

ERNI-3D: A Technology-Generic Tool for Interconnect Reliability Projections in 3D Integrated Circuits

by

Syed Mohiul Alam

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Recent developments in semiconductor processing technology has enabled the fabrication of a single integrated circuit (IC) with multiple device-interconnect layers or wafers stacked on each other. This approach is commonly referred to as the 3D integration of ICs. Although there has been significant research on the impact of 3D integration on chip size, interconnect delay, and overall system performance, the reliability issues in the 3D interconnect arrays are largely unknown. In this research, a novel Reliability Computer Aided Design (RCAD) tool ERNI-3D has been developed for reliability analysis of interconnects in a 3D IC. Using this tool, circuit designers can get interactive feedback on the reliability of their circuits associated with electromigration, 3D bonding, and joule heating. Based on a joint probability distribution, a full-chip reliability model combines all reliability figures from different components to give a useful number for the designers' reference. This initial version of ERNI-3D treats 3D circuits with two wafers or device-interconnect layers in the stack. However, the data-structures and algorithms in the tool are generic enough to make it compatible with 3D circuits with more than two device-interconnect layers, and to allow incorporation of more sophisticated reliability models in the future. Since 3D integration technology is not yet widespread, and no CAD tool supports IC layouts for such a technology, a novel layout methodology has been implemented in 3DMagic by extending MAGIC, a widely used layout editor in academia. Apart from the CAD tool work, this research has also led to the development of, and interesting experiments with, some 3D circuits for testing ERNI-3D. The test circuits investigated are a 3D 8-bit adder and an FPGA.

Thesis Supervisor: Donald E. Troxel
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Carl V. Thompson
Title: Stavros Salapatas Professor of Electronic Materials

Acknowledgments

I would like to thank Prof. Donald E. Troxel and Prof. Carl V. Thompson for their guidance and support throughout the project. There were numerous discussion sessions leading to this productive research. I am grateful to Prof. Rafael Reif, Prof. Anantha Chandrakasan, Arifur Rahman, and Andy Fan for discussion sessions on 3D integration technology. I also acknowledge the support of Yonald Chery for providing feedback on software development during various phases of this project. The software infra-structure of ERNI-3D is based on Yonald's earlier work on MAJIC with ERNI. I am thankful to Stefan Riege for helping me understand the reliability issues in integrated circuits associated with electromigration. Stefan developed the electromigration reliability models used in ERNI and ERNI-3D during his Ph.D. dissertation. I would also like to thank Nisha Checka and Charlotte Lau for using 3DMagic to design the 3D FPGA in their course project.

There are several people who helped me with the writing of this thesis. Shamik Das, Raiyan T. Zaman, and Syed Masudul Alam reviewed some of the chapters and gave useful feedback. Syed Masudul Alam also assisted me in drawing some of the figures.

I am indebted to support and encouragement I have received over the years from my parents. They inspired me to choose graduate school after my undergraduate education.

This research is funded by MARCO Focused Research Center for Interconnects under contract number B12-D00-S1. The project is carried out at Microsystems Technology Laboratories.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A novel Reliability Computer Aided Design (RCAD) tool ERNI-3D for the reliability analysis of interconnects in a three-dimensional integrated circuit (3D IC) has been developed in this research. Using this tool, circuit designers can get interactive feedback on the reliability of their circuits, as it is affected by electromigration, 3D bonding, and joule heating. Since 3D integration technology is not yet widespread, and no CAD tool supports IC layouts for such a technology, a novel layout methodology has been implemented in 3DMagic by extending MAGIC, a widely used layout editor in academia.

This chapter provides a brief introduction to 3D integration technology and reliability issues in ICs, along with previous work in these areas. It also presents the organization of the thesis.

1.1 Three Dimensional Integrated Circuit (3D IC)

The main idea behind 3D integration is to have multiple device layers, such as transistors, along the third axis (z axis) and lower the interconnect lengths by connecting the layers vertically. This has been accomplished by bonding multiple wafers fabricated with different or similar technologies [1] as well as by fabricating multiple device, such as CMOS, layers on the same wafer [2, 3].

Although the concept of 3D integration emerged as early as 1979 [4], significant research work was done only after early 90's. Technology scaling posed limitations on overall system performance by degrading the interconnect delay and increasing the number of longer global and semi-global interconnects in a chip. The need for a long-term solution to enhance performance in successive technology generations became apparent. 3D integration is an attractive solution to this problem as it can significantly reduce the number of long wires by mapping a 2D circuit into different layers [5]. Moreover, the total number of repeaters for long and intermediate wires will also decrease, resulting in higher density and lower interconnect-limited chip area. Because of these reasons, high-performance microprocessors and programmable logic devices are attractive applications for 3D integration.

Another promising advantage of 3D integration is its ability to integrate completely different technologies in a single 3D chip. Future system-on-chip (SOC) applications will consist of digital, analog, and RF/optical networking units on the same die [6]. Using 3D integration, each unit can be fabricated on separate wafers with its optimized process technology, and then integrated vertically to form a 3D SOC [1].

1.2 Interconnect Reliability Concepts

The conventional approach to meet reliability goals in an integrated circuit has been to design the circuit using simple and conservative design rules based on current density in a wire segment. However, this simplicity and conservatism lead to limited performance in newer technology generations [7]. There is a phenomenal increase in total number of wire segments due to higher wiring requirements for increased number of transistors per chip. Moreover, reliability concerns have grown significantly as narrower interconnects are stressed with high currents, and joule heating from power dissipation has increased in high frequency circuits. The particular reliability aspects that ERNI-3D focuses on are electromigration, the effect of increased joule heating, and bond reliability in bonded 3D ICs.

Electromigration is the transport of the atoms of a conducting material due to momentum transfer from flowing electrons. The rate of electromigration is directly related to the current density in a segment of an interconnect tree in an integrated circuit. An interconnect tree consists of continuously connected high-conductivity metal within one layer of metallization. Trees usually have multiple branches connected to other interconnect trees in a different metallization layer through vias or contacts. Therefore, the reliability of a segment of an interconnect tree also depends on the current densities in the linked segments [7].

As the processing technology for ICs allows fabrication at smaller length scale, a large number of narrower interconnects now contain ever higher current densities. Thus, the electromigration effects are severe. The compressive stress at the anode side can cause the conducting metal to extrude. The extrusion can create a short circuit with nearby interconnects. Moreover, the compressive stress in Copper (Cu) interconnects causes the liner materials to crack, and the Cu can diffuse throughout the inter-layer dielectric material. Tensile stress at the cathode end can also lead to formation of a void or a gap. A large void increases the resistance of a conducting line, and can cause the line to fail at some critical resistance. Such failure can occur in both Aluminum and Copper interconnects.

Another important reliability concern in integrated circuits is the effect of increased joule heating from power dissipation. For reliable operation of interconnects, the junction temperature is required to be less than 100^oC [8]. However, today's high performance microprocessors already run at temperatures close to 100^oC [9]. Such a high temperature causes the electromigration effects to be severe as the median time to failure due to electromigration lowers with increasing temperature [10]. Moreover, the interconnect geometries in the circuit can deform in a very high temperature, causing functional failure through short-circuits. The task of dissipating heat has already become a challenge in conventional 2D circuits, and it surely will be a concern in 3D ICs. Several solutions, such as low-power circuit design, use of materials with higher thermal conductivity, innovative cooling techniques with heat-sinks or forced-air, and thermally-efficient packaging, are under active investigation [8, 11, 12, 13].

Finally, the reliability concept that is unique to 3D ICs is the reliability of bonds in a wafer bonding technology. Apart from the issue of proper alignment of different wafers in the fabrication process, the reliability also depends on bond strength and quality [1, 14]. Moreover, the conducting points at the bonding surface, shortening the interconnects at the adjacent layers, will be under high electrical and thermal stress during the operation of a circuit. Under such conditions, overall system failure can be affected by interface fracture. Therefore, it is important to quantify bond strength and quality given a particular wafer bonding technology, and also to investigate the reliability models for the inter-wafer vias at the bonding surface [15].

1.3 Related Work

1.3.1 3D Integration using Wafer Bonding

Wafer bonding technology using copper, developed at MIT, has shown promise for a successful 3D integration with two or more device layers [1]. In this technology, each device-interconnect layer is fabricated separately on different wafers with the same or different processing technologies, and then the wafers are bonded with each other at low temperature ($< 450^oC$) with a bonding layer of Cu. For example, in a 3D IC with back-to-front bonding, the top wafer is attached to a handle wafer and thinned down from the back side (the Si substrate side). Then the back side of the top wafer is bonded to the front side of the bottom wafer. The handle wafer is released after the bonding process. Using a similar technique, the wafers can be bonded in other orientations, such as front-to-front or back-to-back. The inter-wafer vias can be etched before or after the bonding process. Figure 1-1 shows a cross-section of a 3D IC formed by Cu-Cu front-to-front wafer bonding.

Some variations of wafer bonding technique exist because of the different materials that can be used for bonding. At the Rensselaer Polytechnic Institute, it has been demonstrated that bonds can be facilitated by using a polymeric "glue" layer [16]. Then the wafer sandwich is thinned from the sides to permit etching of inter-wafer vias of reasonable size. Irrespective of the bonding technique, the main advantage of wafer bonding technology is that it allows nearly conventional fabrication steps for

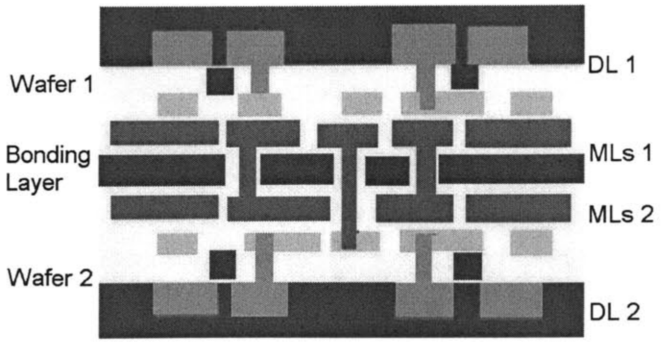


Figure 1-1: *Cross-section of a 3D IC with Cu-Cu front-to-front wafer bonding. Here DL and MLs correspond to device and metal layers, respectively.*

processing each wafer. The individual wafers are aligned using an infra-red (IR) process or tools such as the Electronics Vision model EV640 during bonding. However, there are some limitations in wafer bonding because of the lack of precision in the alignment of wafers and ability to etch deep inter-wafer vias.

1.3.2 Electromigration Reliability in Networked Interconnects (ERNI)

There has been significant research on electromigration reliability models [10, 17]. Several efficient techniques have been developed for conducting reliability analysis on integrated circuits during design-time [18, 19]. Conservative electromigration models for straight lines have existed for many years. Recently, an interconnect tree has been identified as the basic reliability unit, and more efficient electromigration models have been developed [7]. Moreover, a hierarchical approach to circuit-level reliability analyses has enabled the incorporation of these newly-developed models in the RCAD tools [19, 20].

The RCAD tool ERNI (Electromigration Reliability in Networked Interconnects), recently developed at MIT [21], allows process-sensitive and layout-specific reliability assessments of a fully or partially laid-out integrated circuit. Figure 1-2 shows the flow diagram of ERNI's operations. First, the interconnect trees are extracted from a

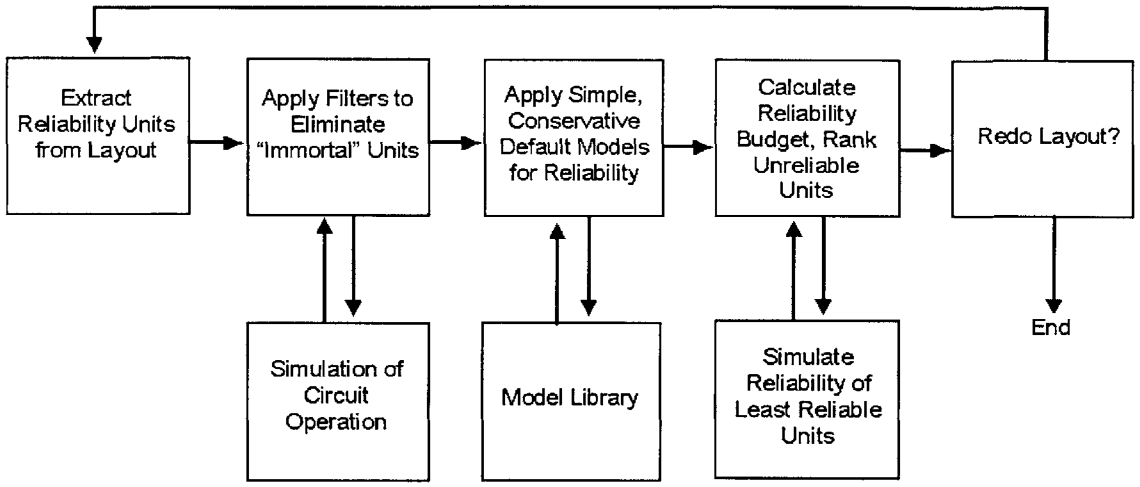

Extract Reliability Units from Layout
Apply Filters to Eliminate "Immortal" Units
Apply Simple, Conservative Default Models for Reliability
Calculate Reliability Budget, Rank Unreliable Units
Redo Layout?
Simulation of Circuit Operation
Model Library
Simulate Reliability of Least Reliable Units
End

Figure 1-2: *Flow diagram of hierarchical circuit-level reliability analysis in ERNI.*

circuit layout and categorized into mortal and immortal (highly reliable) trees based on the current density-length product (jL_{max}). Further computation goes on with only the mortal trees, and a reliability figure for each tree, in terms of median time to failure, is obtained after applying the default electromigration model [7]. ERNI is also expected to interact with MIT/EmSim. MIT/EmSim evaluates the electromigration-induced stress on interconnect trees from ERNI and predicts more accurate reliability figures based on the Korhonen Model [22, 7].

The code for ERNI is written in Java 2 (JDK 1.2). It is a client extension to MAJIC, a layout parser and viewer also written entirely in Java. Lots of data-structures and algorithms in MAJIC are based on MAGIC, an IC layout editor developed at UC Berkeley and widely used in academia [23]. Using MAJIC, users can view a circuit layout, and apply reliability analyses by choosing different filter algorithms from the "ERNI" menu. Figure 1-3 shows a screen-shot of MAJIC with the display of available options at the "ERNI" menu. Currently, ERNI operates on 2D IC layouts with multiple metallization layers created using MAGIC.

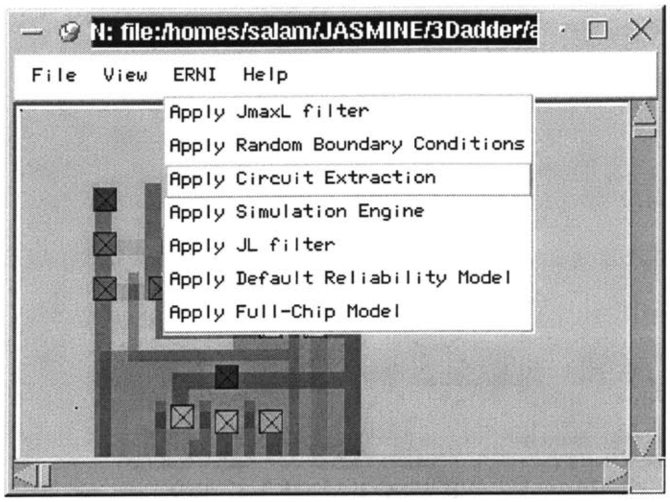


Figure 1-3: *A screen-shot of MAJIC with a view of menu items for reliability analyses in ERNI.*

1.4 Thesis Statement

Although there has been significant research on the impact of 3D integration on chip size, interconnect delay, and overall system performance, the reliability issues in 3D interconnect arrays are fairly unknown. In this research, we have developed a framework for reliability analysis in 3D circuits by extending ERNI to ERNI-3D. For completeness, we have also developed a novel layout methodology for designing 3D ICs, and implemented this methodology in MAGIC. ERNI-3D treats 3D circuits laid out in 3DMagic, the version of MAGIC capable of 3D IC layout and extraction, and applies reliability models for assessing electromigration and bond reliability, and the effects of increased joule heating. A full-chip reliability model then combines all the reliability figures of different components to give a useful number for the designers' reference. This initial version of ERNI-3D works with 3D circuits with two wafers or device-interconnect layers in the stack. However, the data-structures and algorithms in the tool are generic enough to make it compatible with 3D circuits with more than two device-interconnect layers in the future.

1.5 Organization of Thesis

This chapter has established the concept of 3D integration and associated reliability issues. Chapter 2 presents a novel 3D circuit layout methodology for CAD tools. It also discusses the implementation of such methodology in MAGIC. Chapter 3 describes the architecture and design of two 3D test circuits, an 8-bit adder and an FPGA, developed during this research. Chapter 3 also includes interesting results from the performance comparison of 2D and 3D FPGAs.

The design and architecture of the RCAD tool ERNI-3D is detailed in chapter 4. There are several sections on 3D layout parsing, interconnect tree extraction, and methods or functions for reliability analyses. Chapter 5 presents the procedures for, as well as results of, reliability analyses on the 8-bit adder and FPGA using ERNI-3D. After that, possible extensions of this research and other suggestions on future work are discussed in chapter 6. Finally, chapter 7 summarizes the research and discusses its implications.

For reference as well as better understanding of 3D circuits and design methodology, the thesis also includes four appendices. Appendix A includes an excerpt from the technology file which is at the heart of enabling 3D circuit design in MAGIC. A comprehensive tutorial for using 3DMagic, the version of MAGIC with 3D design capabilities, on Athena[1] can be found in Appendix B. Appendix C has a list of preliminary design rules for the layout of 3D circuits in 3DMagic. Finally, Appendix D includes an overview of the Java source code of MAJIC, the parent application of ERNI-3D.

[1]Athena is MIT's UNIX-based campus-wide academic computing facility.

Chapter 2

Layout Methodology for 3D ICs

2.1 Overview

As discussed in the previous chapter, there has not been any significant development on standardizing CAD methodologies for 3D ICs. However, the need for such standardization is critical as the topic of 3D integration is receiving more and more attention in both academia and industry. More importantly, the final product of this research, ERNI-3D, is a layout-specific RCAD tool. Therefore, we need to have comprehensive CAD methodologies for 3D ICs to layout our test circuits as well as define the input method for ERNI-3D.

This chapter proposes a novel CAD methodology for the layout of 3D ICs [24]. The proposed methodology enables the layout of individual device-interconnect layer keeping in mind the position and orientation of vertical/3D contacts. Moreover, the layout methodology can be easily incorporated into existing IC layout tools.

2.2 3D Trees and Inter-wafer Vias

In wafer bonding technology, each device-interconnect layer is fabricated separately on different wafers with the same or different technologies, and then the wafers are bonded with each other with a bonding layer of copper or polymeric adhesives. When the bonding is complete, 3D ICs have vertical interconnects of significantly higher length than vias or contacts in the 2D integration of ICs (a single device layer and multiple metal layers on a single chip). Moreover, the 3D circuits have two different types of vertical interconnects as shown in figure 2-1. The inter-wafer vias connect

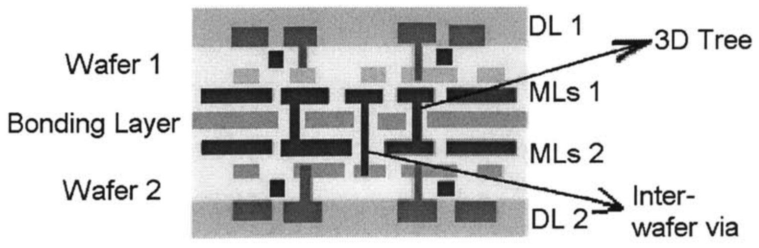


Figure 2-1: *Cross-section of a 3D IC with the 3D trees and inter-wafer vias. Here DL and MLs correspond to device and metal layers, respectively.*

multiple interconnect trees in different wafers. At the bonding surface, the adjacent metallization layers of two wafers can be connected with vertical Cu lines. The vertical Cu lines create a new type of tree, referred to as "3D tree", which expands between two different wafers in a 3D circuit.

To facilitate the layout of 3D ICs, all types of inter-wafer vias or contacts are generalized into three major categories. Figure 2-2 shows the three categories of contacts along with their connectivity in a wafer. Since in wafer bonding technology, different

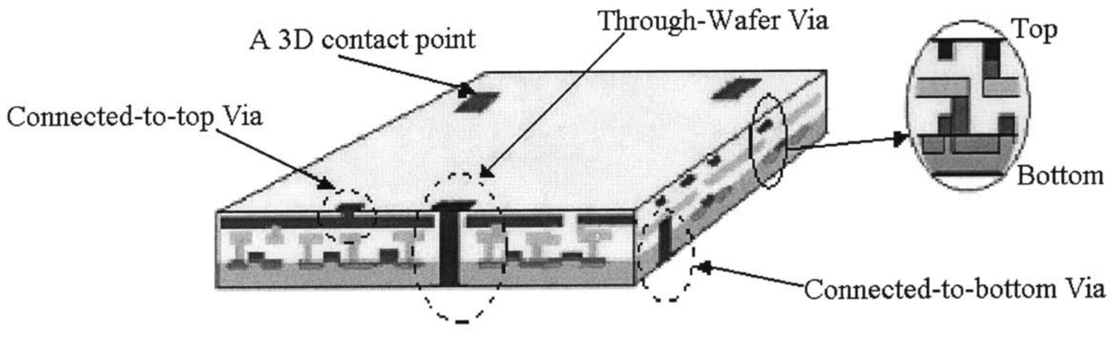


Figure 2-2: *Different types of vias/contacts for 3D ICs.*

wafers are fabricated separately before bonding, layout for each wafer can be done separately with inter-wafer contact information embedded in the layout. Moreover, three categories of vias are sufficient for defining almost all types of interconnection between wafers in 3D ICs. A detailed description of each category is as follows.

2.2.1 Connected-to-top Vias

This type of via connects a metal layer to a 3D contact point at the top[1] of a wafer. In figure 2-2, the via is connected with Metal2, the topmost metallization layer of this particular wafer. A connected-to-top via can also connect other metal layers, such as Metal1 or Metal3 to a 3D contact point at the top. When two wafers are bonded on front-to-front, as shown in figure 2-1, a 3D tree is formed if connected-to-top type vias of the topmost metal layers from the two wafers share the same 3D contact point at the bonding surface.

2.2.2 Connected-to-bottom Vias

The connected-to-bottom type via extends between a metal layer and a 3D contact point at the bottom[2] of a wafer. In figure 2-2, a connected-to-bottom via connects Metal1 to a 3D contact point. When two wafers are bonded back-to-front, a 3D tree is formed if a connected-to-bottom via from the top wafer and a connected-to-top via of the topmost metal layer from the bottom wafer share the same 3D contact point. Similarly, two connected-to-bottom type vias of different wafers with the same 3D contact point can also create a 3D tree when they are bonded back-to-back.

2.2.3 Through-wafer Vias

Through-wafer vias extend through the whole wafer without being electrically connected to any interconnect or active layer of the wafer. This type of via is particularly useful when more than two wafers are bonded to create a 3D IC. For example, in a three-wafer 3D circuit, an interconnect layer of the top wafer can be connected with

[1]Both "top" and "front" refer to metal side (opposite of the Si substrate side) of the wafer.
[2]Both "bottom" and "back" refer to the Si substrate side of the wafer.

that of the bottom wafer through a through-wafer via of the middle wafer. Moreover, Cu filled through-wafer vias can be designed as heat conductors to overcome the effect of increased Joule heating in a 3D IC. However, such application of through-wafer vias is still an area of experimental research [15]. To represent a through-wafer via that is also electrically connected to an interconnect layer in a wafer, a designer can add both connected-to-top and connected-to-bottom type vias at the same position on that interconnect layer.

2.3 Implementation in MAGIC (3DMagic)

MAGIC is an interactive VLSI circuit layout editor developed at UC Berkeley [23, 25]. It is widely used within the academic community. Moreover, its well-documented source code and wide variety of features make it an excellent vehicle with which to conduct VLSI and CAD research. To implement the 3D IC layout methodology in MAGIC, we have developed a new technology file (scmos3D.tech26) to support all the new layers and inter-wafer vias. The technology file also contains preliminary design rules and a model for electrical connectivity. Appendix A includes an excerpt from the 3D technology file with the additions emphasized in italics.

Several entries for display styles are also added in the MAGIC style files, mos.7bit.dstyle5 and mos.24bit.dstyle5. These entries define the new color maps and styles for the mask layers representing metal, poly, ndiffusion, and so on. Since MAGIC is a technology independent layout editing tool, a 3D circuit can be designed on any MAGIC (version 6.4 or higher) with the display style file installed in the proper path and technology file specified with the -T techfile flag at the command line [26]. More discussion on the implementation and usage of the layout methodology is as follows.

2.3.1 Graphical User Interface

MAGIC consists primarily of an internal data-structure representation of a 2D layout with a graphical user interface (GUI) to manipulate and view a circuit design [27]. However, MAGIC also supports viewing multiple layouts on different windows and editing a layout on the edit window [28]. Using this feature, the layout of different wafers in a 3D circuit can be done on different windows with new abstract layers (discussed in the next section) used to specify the inter-wafer vias. The major issue with this approach is in managing the alignment of different layouts and inter-wafer vias with shared 3D contact points.

The alignment issue in 3D layouts has been resolved by introducing two area markers: mainboundary or mbnd for specifying wafer size, and 3Dcontactpoint or 3Dconp for specifying the position of an inter-wafer via. Figure 2-3 shows the usage

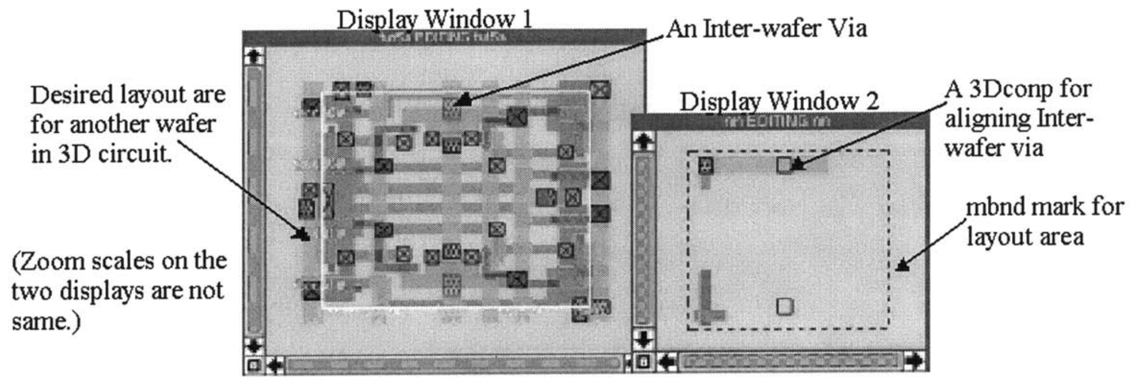

Display Window 1
An Inter-wafer Via
Desired layout are for another wafer in 3D circuit.
Display Window 2
A 3Dconp for aligning Inter-wafer via
mbnd mark for layout area
(Zoom scales on the two displays are not same.)

Figure 2-3: *GUI support for 3D circuit layout in 3DMagic.*

of the area markers in MAGIC. In Display Window 1, one layer of a 3D circuit has been designed with the abstract layers indicating the inter-wafer vias. Then Display Window 2 is opened to layout the other wafer using MAGIC commands such as ":openwindow", ":select", and ":edit" [28]. The positions of inter-wafer vias and desired layout area for the second wafer are marked in Display Window 1 with the box tool to retrieve their grid coordinates (":box"). Now the 3Dcontactpoints and mainboundary are painted in Display Window 2 at the corresponding positions (":paint 3Dconp", ":paint mbnd"). Further layout can be done on Display Window 2 with respect to the area markers to design the second wafer of the 3D circuit.

To automate the detection and placement of the area markers, a MAGIC feature, called "feedback", can be used very efficiently. Table 2.1 shows the commands and

Table 2.1: *Alignment of 3D layouts using the MAGIC feature "feedback".*

Command	**Description**
Display Window 1	
feedback add mbnd outline	Adds the main boundary mark for 3D layouts
feedback add iwvia1	Adds the inter-wafer via, named iwvia1, mark. Any name can be used instead of iwvia1 to suit designer's need
feedback add iwvia2	Adds the inter-wafer via mark for iwvia2
. . . . .	Adds more inter-wafer via marks
feedback save afile	Saves all the area markers in file "afile"
Display Window 2	
source afile	Paints all the area markers from file "afile". Corresponding name and style, such as outline, right-cross, etc., of the area markers are also reserved.

corresponding descriptions for making 3D layouts using the "feedback" feature. A 3D circuit laid out in such a way will have multiple files. A strategy for layout-file management that also incorporates orientation[3] of each wafer in the bonding process is discussed at section 2.3.3.

2.3.2 Abstract Layers

Abstract layers are the boxes in color that are drawn on the edit window to represent a particular mask layer, such as metal1, metal2, poly, ndiffusion, and so on. Abstract layers are defined in the technology file in the section "types." Moreover, an abstract layer that represents a contact or a via also has further definition at the section "contact." The "types" section introduces an abstract layer to its plane, such as

[3]In the fabrication process of a 3D circuit, a wafer can go at the top or bottom of the stack and can also be flipped during the bonding process.

oxide, well, metal1, metal2 etc., and then the "contact" section lists a series of other abstract layers that an abstract layer of type contact connects. These sections are shown in Appendix A.

Two abstract layers representing contact types have been defined in the technology file as the connected-to-top vias. The two contacts connect either metal2 or metal3 to a 3D contact point at the top of the wafer. A connected-to-top via from metal1 is not defined, since in a scalable CMOS process with at least three levels of metal, circuits with only one metal layer is very uncommon. Figure 2-4 shows the two connected-

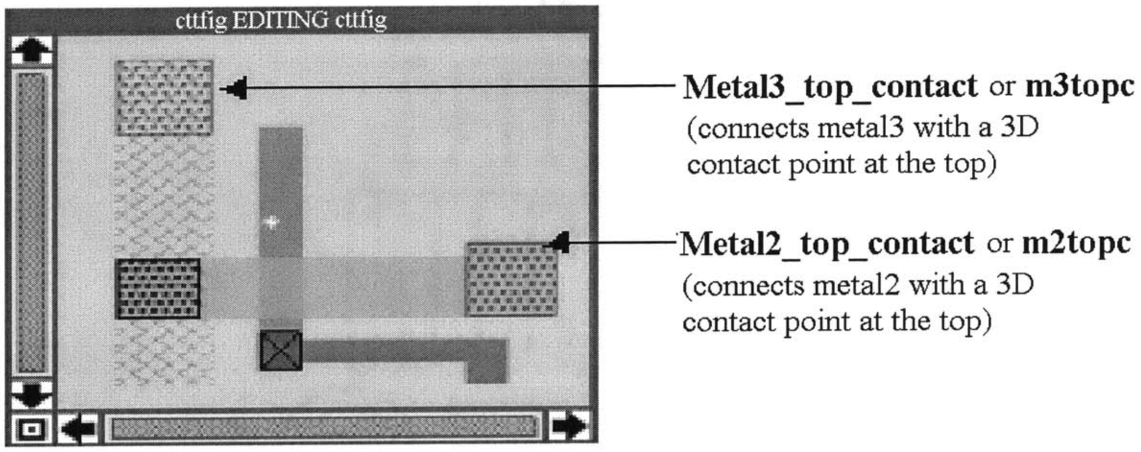

cttfig EDITING cttfig
Metal3_top_contact or **m3topc**
(connects metal3 with a 3D contact point at the top)
Metal2_top_contact or **m2topc**
(connects metal2 with a 3D contact point at the top)

Figure 2-4: *Connected-to-top type vias in 3DMagic.*

to-top type vias: Metal2_top_contact or m2topc, connecting metal2 and a 3D contact point, and Metal3_top_contact or m3topc, connecting metal3 and its corresponding 3D contact point.

Similarly for the connected-to-bottom category of vias, the two new abstract layers are Poly_bottom_contact or pbcon and Metal1_bottom_contact or m1bcon. Poly_bottom_contact directly connects poly1 with a 3D contact point at the bottom of the wafer. However, in order to connect Metal1 with a 3D contact point at the bottom, it is necessary to paint Poly_bottom_contact and Metal1_bottom_contact on top of each other in any order. This is because the scheme in the current technology files only allows contacts for connecting at most three layers, and the metal1 and 3D contact point at the bottom layer are two layers apart. Figure 2-5 shows a

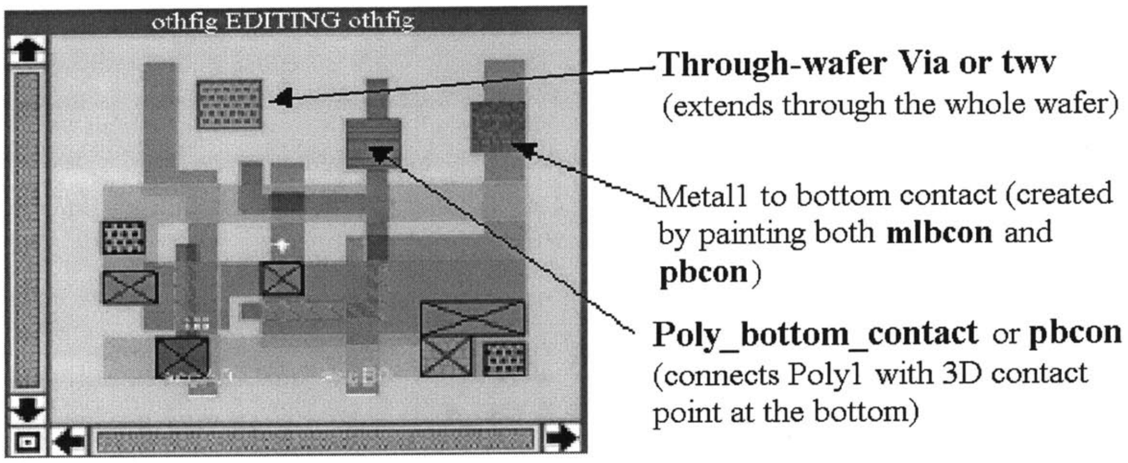

othfig EDITING othfig
Through-wafer Via or twv
(extends through the whole wafer)
Metal1 to bottom contact (created by painting both **mlbcon** and **pbcon**)
Poly_bottom_contact or **pbcon**
(connects Poly1 with 3D contact point at the bottom)

Figure 2-5: *Connected-to-bottom and Through-wafer vias in 3DMagic.*

Poly_bottom_contact and a stacked contact between Metal1 and a 3D contact point at the bottom created by painting both pbcon and m1bcon.

It is not necessary to paint separate layers for 3D contact points. Drawing a contact on the MAGIC edit window will create the internal image of the corresponding 3D contact point at either the topmost or bottommost layer of the layout. Proper connectivity information is also added in the "connect" section of the technology file to enable the built-in hierarchical circuit extractor successfully extract each layout of a 3D circuit. In the "drc" (Design Rule Checker) section of the technology file, some preliminary design rules are defined in terms of λ^4 (the "drc" section is shown in Appendix A). Tables C.2 and C.1 in Appendix C also show a list of 3D layout rules with relevant descriptions. However, these rules may vary depending on the bonding process, and a designer may need to observe absolute micron rules for the contacts. Finally, an abstract layer named through-wafer via or twv has been defined to indicate an extension through the whole wafer (figure 2-5). The connectivity information and design rules ensure that the twv is electrically isolated from all other layers.

[4] λ is the smallest grid spacing in MAGIC.

2.3.3 Strategy for Layout Management

A completely laid out 3D circuit in 3DMagic will consist of multiple files: two or more layout files of format *.mag* for different wafers, and one or more text file[5] containing the area markers. These files can be easily managed using a simple directory scheme. For example, all the files for a 3D adder laid out in two wafers can be stored as shown in figure 2-6. The directory is named *adder8*, under which two layout files,

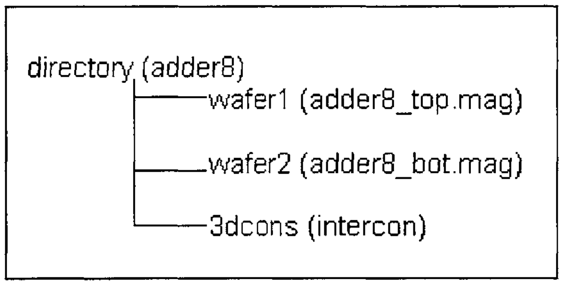


Figure 2-6: *Layout-file structure for a 3D adder.*

adder8_top.mag and *adder8_bot.mag*, and one text file for the 3D area markers, *intercon*, are stored. It is also important to note that layout files have suffix, such as *_top* or *_bot*, where *_top* indicates that the corresponding layout is for the top wafer in the 3D stack, and similarly *_bot* indicates a layout for the bottom wafer. In this layout methodology, by default the wafers are not flipped during the bonding process. Therefore, to indicate that a particular layout is for the wafer that is also flipped in the 3D stack, another suffix *_flp* is required. This way a designer can incorporate necessary information on the orientation of wafers for the wafer bonding process, and also add inter-wafer vias accordingly. Figure 2-7 shows the different 3D bonding orientations with corresponding layout-file structures of the 3D adder. The layout management scheme can be easily extended for 3D circuits with more than two wafers by indicating the middle wafers with different suffixes, such as *_md1*, *_md2*, etc. starting from the top. For example, in case of a 3D FGPA with three wafers in the stack, there will be three *.mag* files under the directory, *3dfpga*. The *.mag* files can be named

[5]These text files are created with "feedback save afile" command.

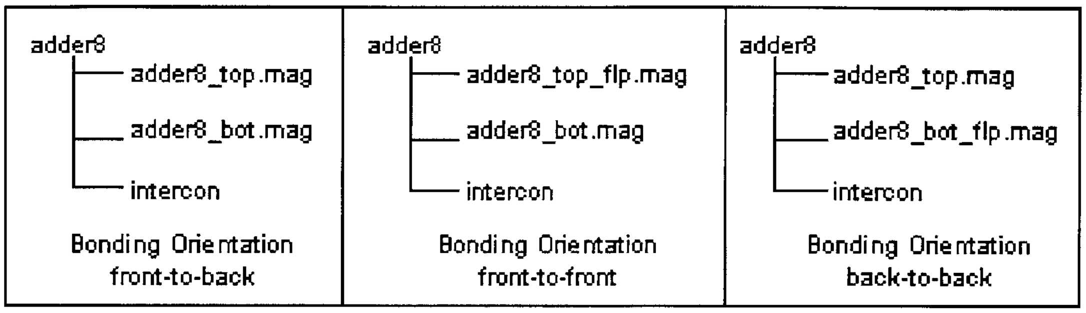


Figure 2-7: *Layout-file structures for a 3D adder with different bonding orientations.*

3dfpga_top.mag, *3dfpga_md1.mag*, and *3dfpga_bot.mag*. The bonding orientation for all the wafers is front-to-back.

2.3.4 Circuit Extraction and Verification

Circuit extraction and verification are integral parts of CAD tools. MAGIC can extract circuits from layouts with the extraction parameters retrieved from the technology file. In the initial version of 3DMagic, circuit extraction is supported in a primitive way. All the layouts for a 3D circuit are dumped into a single layout cell (using the command ":dump cellname") and extracted with the command ":extract all". The 3D connections between layouts are automatically done if the same label is used (using ":label") for inter-wafer vias of different layouts that requires to be connected.

Currently there is no experimental data on extraction parameters for the inter-wafer and through-wafer vias. Both the extraction process and accuracy will be improved in the future when more extraction parameters become available. The extraction process creates a file with an extension, *.ext*, from the layouts. A spice netlist can be derived from the *.ext* file with the command "ext2spice" [29]. Finally, the spice netlist can be simulated with HSpice or Spice with a proper input vector for functional verification of the circuit.

2.4 Using 3DMagic in Athena

3DMagic is available for the users of MIT's academic computing facility, Athena. 3DMagic can be used for course[6] projects as well as for designing 3D test circuits for research purposes. Appendix B includes a tutorial on using 3DMagic in Athena. During the fall semester of 2000, several students taking the course, Analysis and Design of Digital Integrated Circuits (6.374), used 3DMagic for a research project investigating the advantages of 3D integration. The 3D FPGA discussed in the next chapter is an outcome of that project. Moreover, the availability of a CAD tool for 3D layouts has set a research direction where design and verification of actual circuits are integral parts of system-level modeling and analysis.

[6]6.374-Analysis and Design of Digital Integrated Circuits, and 6.371-Introduction to VLSI Systems.

Chapter 3

3D Test Circuits

3.1 Overview

The main purposes of 3D test circuits are to provide a debugging tool for ERNI-3D and to enable a preliminary simulation of the RCAD tool with actual circuits. Both 3D adder and FPGA heavily exploit the principle of hierarchy in IC design. The topmost level is divided into several modules of regular structure. Moreover, the modules are simple enough for manually keeping track of the number of interconnect trees and transistors. As a result, the 3D test circuits are both simple enough for serving the main purposes, and large enough to simulate a reasonable computational load for an RCAD tool. This chapter describes the architecture and design of the test circuits in detail. Moreover, some interesting results from performance comparison of the 3D FPGA with its 2D counterpart are also presented at the end.

3.2 3D Adder

An adder is one of the basic datapath operators for every processor ranging from conventional microprocessor to digital signal processors and network processors. Moreover, this type of circuit takes advantage of structured design. Since n-bit data is being processed, n identical circuits can be combined to form the operator. A similar technique is exploited in the design of a 3D 8-bit adder that consists of two wafers in the 3D wafer stack. A detailed description of the design follows.

3.2.1 Architecture and Implementation

There is a large variety of adder implementations to meet the density and timing constraints for different applications [30]. For the first test circuit, we chose to implement a ripple carry adder where n blocks are cascaded with each other to form an n-bit adder. Each block is a 1-bit adder that generates sum and carry outputs from the two input bits and previous carry input. The carry signal propagates through the whole adder, and the result is ready only when the final block produces its output.

Given that A and B are the adder inputs, C is the carry input, SUM is the sum output, and $CARRY$ is the carry output, a 1-bit adder implements the following equations.

$$\begin{aligned} CARRY &= AB + C(A+B) \qquad (3.1) \\ SUM &= ABC + (A+B+C)\overline{CARRY} \end{aligned}$$

The equations are derived from a truth table of an adder [30], and simplified by reusing $CARRY$ for generating SUM. Instead of realizing the equations with logic gates, we followed a fully static CMOS transistor-level design approach and implemented the 1-bit adder with 28 transistors. The transistor schematic for this implementation is shown in figure 3-1.

According to conventional or 2D implementation, eight 1-bit adder cells would be cascaded to form an 8-bit ripple carry adder. For the 3D 8-bit adder, we used an architectural technique referred to as "datapath folding". "Datapath folding" means that the operation in the whole datapath is sliced and distributed among different wafers or layers in a serial fashion. For the 3D 8-bit adder with two wafers in a stack, the bottom wafer adds the least significant nibble[1], and then the top wafer calculates the most significant nibble. The $CARRY$ signal from the bottom wafer propagates to

[1]Nibble stands for a 4-bit binary number.

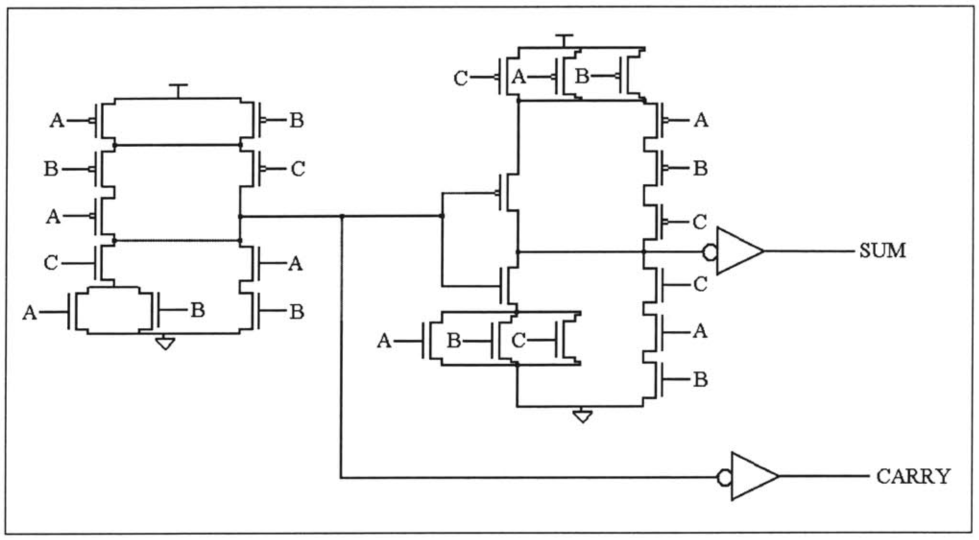


Figure 3-1: *Schematic of a 28-transistor fully static CMOS adder.*

the top wafer through an inter-wafer via. There are other inter-wafer vias connecting Vdd and GND lines. Figure 3-2 shows the architecture of the 3D 8-bit adder.

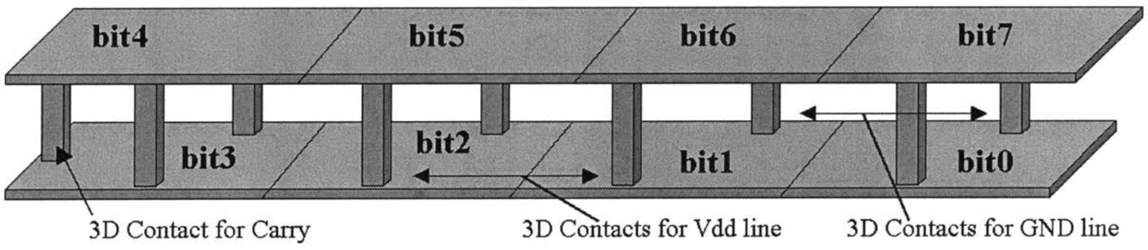


Figure 3-2: *Architecture of the 3D 8-bit adder.*

3.2.2 3D Layout

First, a 1-bit adder was laid out in MAGIC. Apart from the active and poly layers, two levels of metallization layer, metal1 and metal2, are used for routing. The input signals A, B, and C are routed with polysilicon wires, and the GND and Vdd signals are routed with metal1. Each transistor has a W/L ratio[2] of 3/2. The layout is very compact with an area of $(167 \times 48)\lambda^2$. Figure 3-3 shows the layout of the 1-bit adder. Now, using 3DMagic, the layout is replicated according to the structure shown

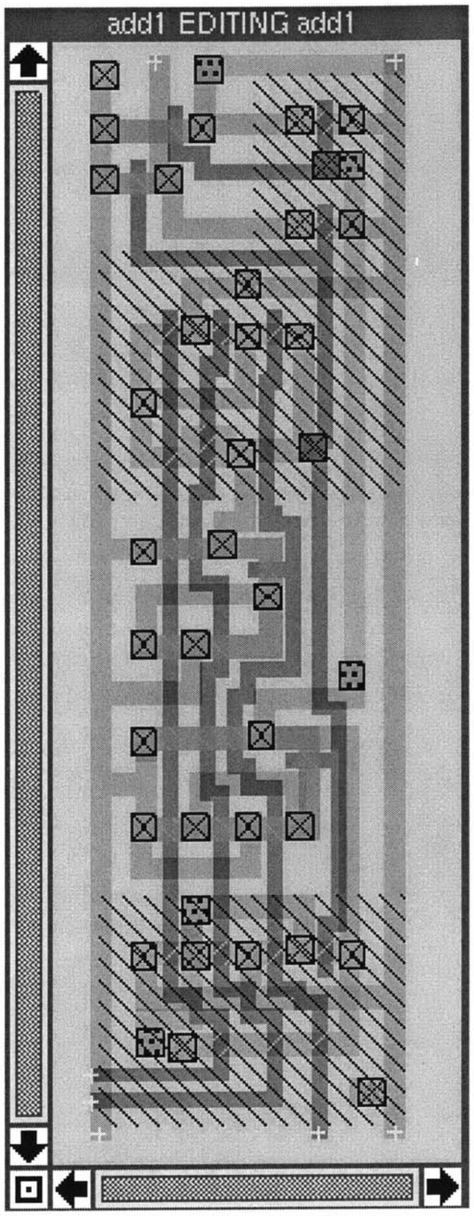


Figure 3-3: *Layout of a 1-bit adder in MAGIC.*

on figure 3-2. The layout for the bottom wafer, named *adder8_bot.mag*, has several

[2]W/L refers to width/length ratio of a transistor.

connected-to-top type vias for connecting GND and Vdd lines with those of the top wafer and $CARRY$ signal of bit3 with C signal of bit4 on the top wafer. Similarly, the layout for the top wafer, named *adder8_top.mag*, has connected-to-bottom types vias at those positions. The area markers for the inter-wafer vias are stored in a text file, *intercon*. Figure 3-4 shows part of the layout files with the inter-wafer vias marked

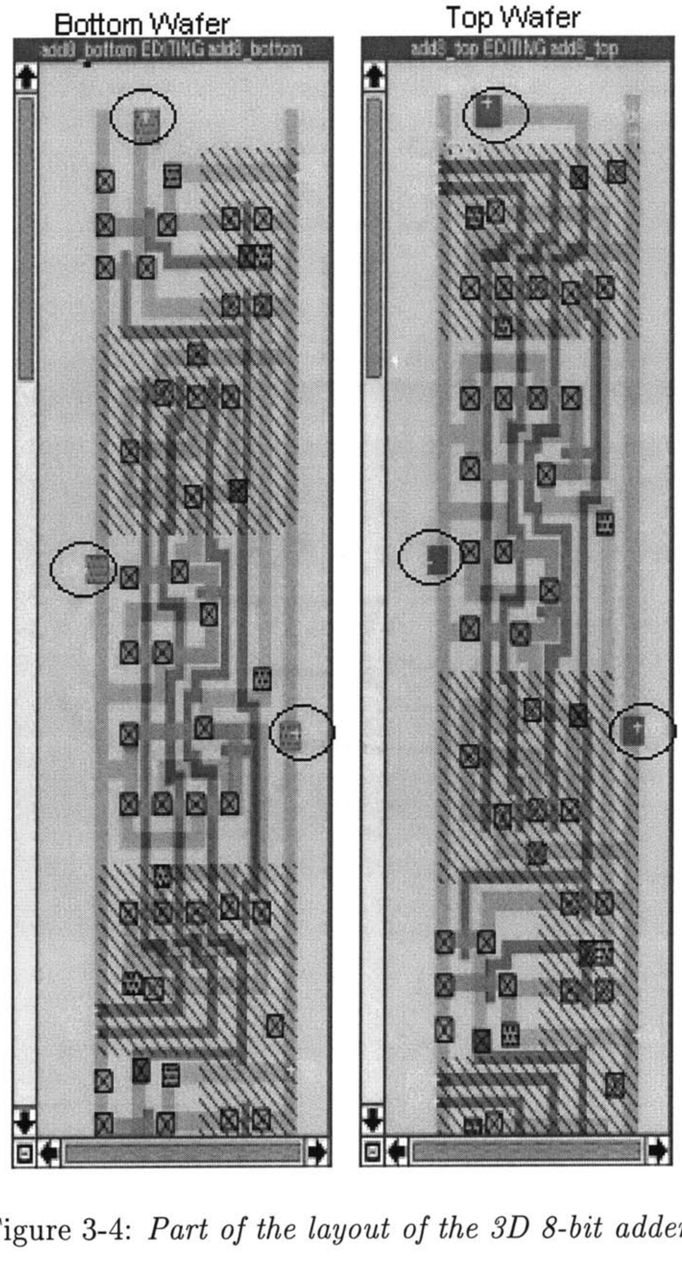


Figure 3-4: *Part of the layout of the 3D 8-bit adder.*

in circles. According to the file management strategy described in section 2.3.3, the

3D 8-bit adder has a bonding orientation of front-to-back, and none of the wafers are flipped in the 3D stack.

3.2.3 Functional Verification

The 3D adder is extracted using the technique discussed in section 2.3.4. Then a *.sim* file, named *adder8.sim*, is generated from the *.ext* file with the MAGIC command "ext2sim". The *adder8.sim* file is simulated with IRSIM, a switch-level simulator, for verifying the correct operation of the circuit [31]. A detailed procedure for using IRSIM can be found in reference [32]. Figure 3-5 shows the two 8-bit input signals, *Ain* and *Bin*, carry signal *Ci*, and corresponding *SUM* with carry output *Co*. It can be verified from the figure that the 3D adder is functioning properly.

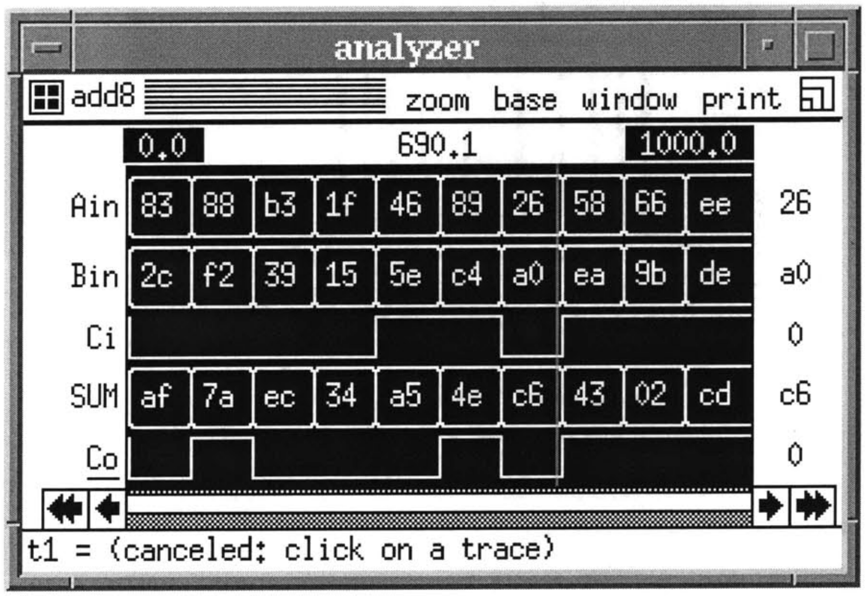

analyzer
add8
zoom base window print
0.0
690.1
1000.0
Ain 83 88 b3 1f 46 89 26 58 66 ee 26
Bin 2c f2 39 15 5e c4 a0 ea 9b de a0
Ci 0
SUM af 7a ec 34 a5 4e c6 43 02 cd c6
Co 0
t1 = (canceled: click on a trace)

Figure 3-5: *IRSIM simulation of the 3D adder.*

3.3 3D FPGA

FPGA (field-programmable gate array) based integrated circuit design has become very common because of its shorter design time and cost. In an FPGA based design, designers first derive logic equations from behavioral-level description of a circuit or system. Then the logic equations are optimized and mapped into a programmable logic architecture. The programmable logic architecture usually consists of an array of configurable logic blocks connected with both vertical and horizontal programmable interconnects. Figure 3-6 shows an FPGA architecture. The mapping of a logic equation into an FPGA involves configuring the logic blocks and routing the interconnects. This mapping can be permanent in anti-fuse based technology as well as reconfigurable in SRAM based technology where the values stored in memory cells set the states of pass-transistors [33].

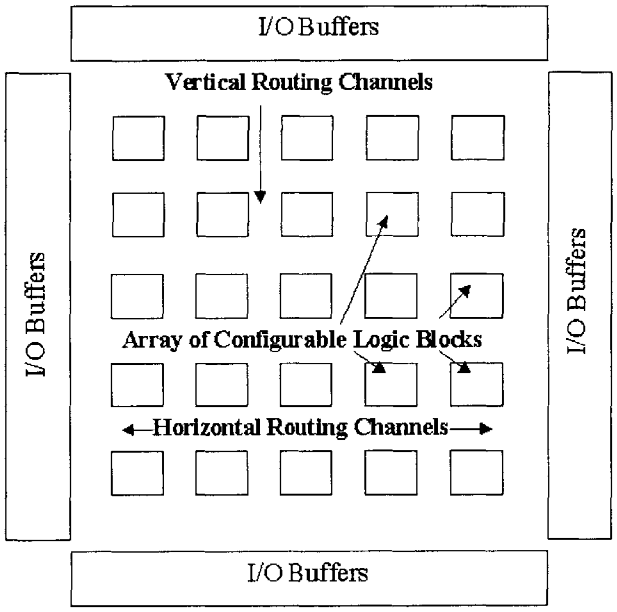

I/O Buffers
Vertical Routing Channels
I/O Buffers
I/O Buffers
Array of Configurable Logic Blocks
Horizontal Routing Channels
I/O Buffers

Figure 3-6: *Conventional FPGA architecture.*

It is apparent from the FPGA architecture that the total area is heavily wiring-limited. In fact, studies show that interconnects and configurable memory account for more than 90% of the chip area, and interconnects account for 40% to 80% of the overall delay [34, 35, 36]. As discussed in section 1.1, 3D integration technology has the potential for shortening interconnect length by mapping active devices or transistors in different wafers and connecting them vertically. Therefore, FPGAs are very attractive applications for such a technology. There has been significant research on system-level modeling of 3D FPGAs for predicting the improvements in performance, density, and power dissipation [14]. Because of the availability of CAD methodologies for 3D circuits derived in this research, several students[3] have taken the research to next step and done simulation on actual circuit layouts. It is important to understand the architecture and layout of a 2D FPGA for making a fair comparison with its 3D counterpart. Therefore, following sections describe the design and architecture of both 2D and 3D FPGAs.

3.3.1 Triptych: 2D FPGA Architecture

The 2D FPGA architecture investigated in this research is known as the Triptych architecture [37, 38]. Triptych is a sea-of-gates type structure constructed with an array of Routing and Logic Blocks (RLBs) instead of conventional configurable logic blocks. Figure 3-7 shows the routing structure in the Triptych architecture. The RLBs can be configured to implement a logic function as well as to act as a signal router. As a result, resource utilization in Triptych architecture is much higher than that of conventional FPGAs. In addition, an RLB configured as a router can provide short and direct connection between its neighbors. Each RLB receives inputs from the four neighboring RLBs (N, NW, S, SW), and routing channel (see figure 3-7). The functional block inside the RLB (figure 3-8) selects one input from N, or NW, one input from S, or SW, and another from the routing channel. Then the functional block implements a function of these inputs according to the control-bits used for mapping. The result is sent to the routing channel and four other RLBs in latched or unlatched form. The original RLB design, developed by [37, 38], is shown in figure 3-8.

[3]Nisha Checka and Charlotte Lau, from 6.374-Analysis and Design of Digital Integrated Circuits, Fall 2000.

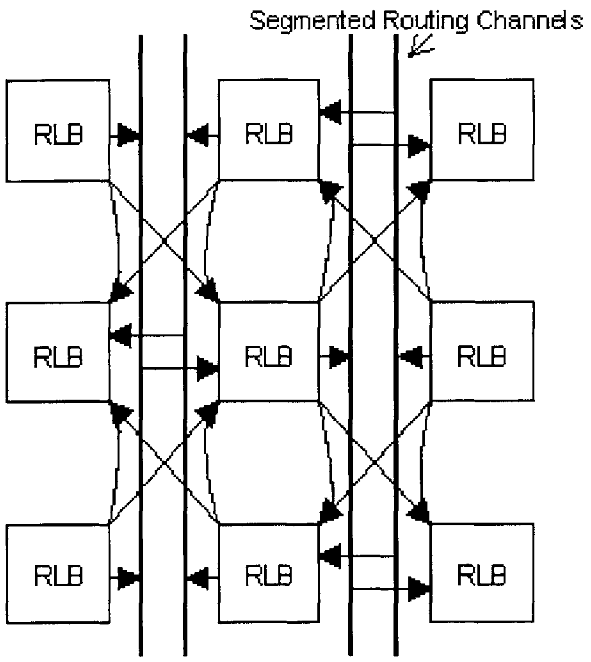


Figure 3-7: *Routing structure in the Triptych architecture.*

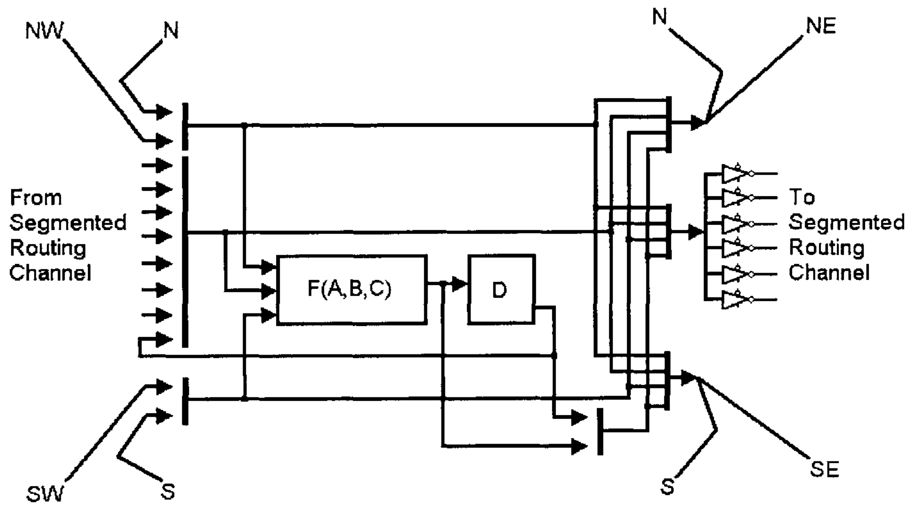


Figure 3-8: *Block diagram of an RLB.*

3.3.2 Rothko: 3D FPGA Architecture

The 3D FPGA architecture investigated, known as Rothko, is based on Triptych and was proposed at Northeastern University [39, 40]. In Rothko, the sea-of-gates structure is extended to multiple layers or wafers. The functional block inside each RLB now selects the first input from N, NW, Above, or Below[4], second input from S, SW, Above, or Below, and third input from the routing channel. Similarly, the output from an RLB feeds into two other RLBs (Above, Below) from top and bottom layers in addition to the routing channel and four RLBs (N, NW, S, SW) in the same layer. Figure 3-9 shows the routing structure of a layer in Rothko.

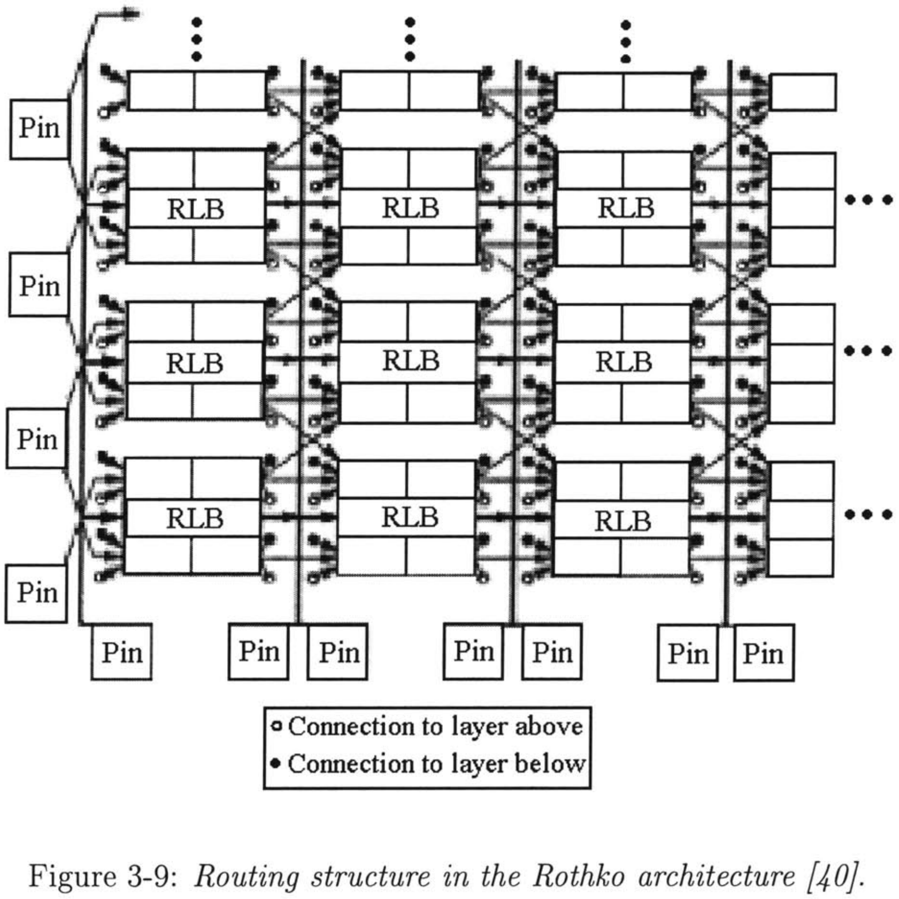


Figure 3-9: *Routing structure in the Rothko architecture [40].*

[4]Above and Below stands for inputs from the RLBs in top and bottom layers respectively.

3.3.3 8-bit Encryption Processor in 2D and 3D FPGAs

An 8-bit encryption processor was implemented in both 2D and 3D FPGAs by laying out the circuits using MAGIC and 3DMagic. The 2D FPGA has 96 (16×6) blocks of RLBs in a single layout, and the 3D version has slightly higher number of RLBs (112) to facilitate the routing in two layouts. The functional unit in an RLB consists of pass-transistor based logic circuitry, and eight control bits are fed from the values stored in a register. Figure 3-10 shows the layouts of 2D and 3D FPGAs.

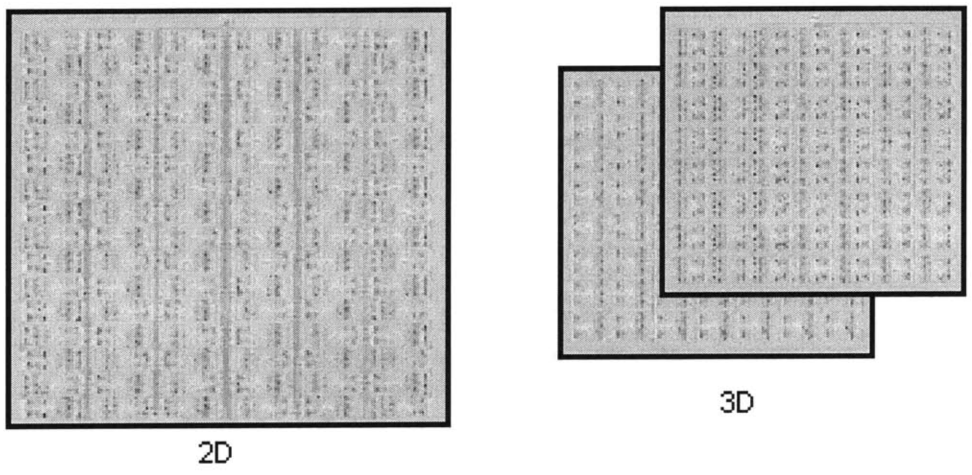


Figure 3-10: *Layouts of 2D and 3D FPGAs.*

3.3.4 System-level Modeling for FPGAs

Researchers at MIT[14] and University of Virginia[41] have estimated key advantages of 3D integration in FPGAs with theoretical analysis. Using stochastic wire-length distribution models, it is demonstrated that 3D FPGAs can shorten the average interconnect length by 13.8% [41]. This is partly because, long planar interconnections between configurable logic blocks are replaced with shorter vertical interconnects. Shorter wire means lower capacitance naturally leading to lower power dissipation and interconnect delay. Studies show that improvements in interconnect delay can be as much as 45% for short interconnects and 60% for long interconnects [14]. Moreover, by 3D integration with $2-4$ layers and the same clock frequency as a 2D FPGA, the reduction in power dissipation in a 3D FPGA is 35% – 55%.

Several key parameters, such as number of interconnects, interconnect length, and improvement in interconnect delay, are also estimated for the 8-bit encryption processor using theoretical analysis. The number of interconnects can be modelled as,

$$i = \frac{w}{f_{out}} \tag{3.2}$$

where i is the number of interconnect channels required between two columns, w is the average number of connections through the interconnects between two columns, and f_{out} is the average fan-out[5]. Therefore, the total number of interconnects, I_{total}, required for an entire system with N number of blocks in a square array is,

$$I_{total} = \frac{w(\sqrt{N}-1)}{f_{out}} \tag{3.3}$$

Applying above equation to the 2D design where w is approximately 8,

$$I_{2D} = \frac{8(\sqrt{N}-1)}{f_{out}^{2D}} \tag{3.4}$$

Similarly, for the 3D design where w is 3, and there are $N/2$ blocks in each layer,

$$I_{3D} = \frac{3(\sqrt{N/2}-1)}{f_{out}^{3D}} \tag{3.5}$$

Assuming the area of an FPGA is dominated by interconnects, area improvement in the 3D FPGA can be estimated using equations 3.4 and 3.5. Since the average fan-outs in 2D and 3D FPGAs depend on particular design and vary from block to block, a conservative estimate of the area reduction is 50%. Theoretical analysis also shows that average interconnect length grows as $O(n^{1/2})$ and $O(n^{1/3})$ in an n-block 2D and 3D FPGA respectively [40]. Therefore, assuming the delay is also predominantly determined by interconnects and proportional to interconnect length, for the 8-bit

[5]Fan-out of a logic gate is the total number of gate inputs that are driven by a gate output.

encryption processor implementation with n around 100,

$$\%\ improvement\ in\ delay = \frac{100^{1/2} - 100^{1/3}}{100^{1/2}} = 53\%$$

3.3.5 Performance Comparison of 2D and 3D FPGAs

SPICE netlists were extracted from the layout of 2D and 3D FPGAs and simulated using PowerMill[6]. Both circuits were simulated with parameters for $.30/.25\mu m$ technology with a power supply of $3V$. Performance parameters such as critical path propagation delay, percentage of used RLBs, and power dissipation were measured and compared. Table 3.1 shows some results from the comparison. Figure 3-11 shows the critical path delay of the 2D and 3D FPGAs.

Table 3.1: *Performance parameters for 2D and 3D FPGAs.*

	Critical Path Delay	% of RLBs used	Total Power
2D	62ns	70.8%	$2.59mW$
3D	53ns	85.7%	$2.77mW$

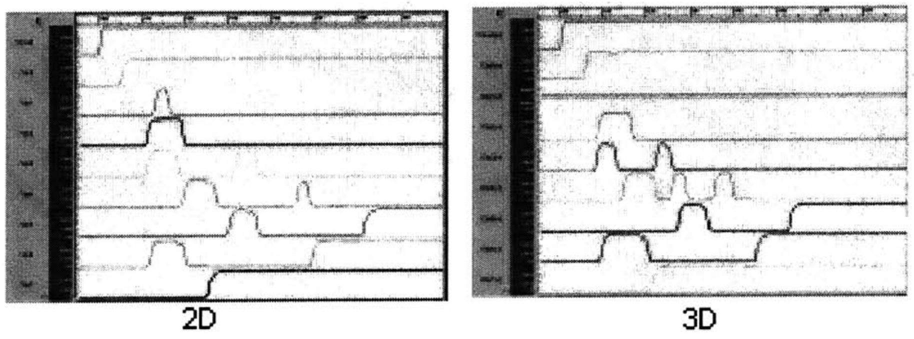


Figure 3-11: *Critical path delay of 2D and 3D FPGAs from PowerMill simulation. The first waveform shows the input signal's transition at* $100ns$*, while next eight signals are the outputs of the encryption processor. The output waveforms of 3D and 2D settle to their final values at* $153ns$ *and* $162ns$ *respectively.*

It is apparent from table 3.1 that there are some advantages in 3D FPGA, even though the improvements do not match the expectation from theoretical analysis.

[6]PowerMill is a high speed circuit simulator available from Synopsis.

The improvement in interconnect delay is only 15% contrary to 53% calculated in the previous section. The modeling analysis does not consider actual circuit being implemented, rather it assumes a standard FPGA architecture and straightforward extension of switch boxes to 3D technology. Therefore, the actual improvement may vary from the ideal case based on circuit design and implementation details in the architecture. The percentage of used RLBs for the 3D FPGA is higher than that of 2D within the area of the smallest rectangle enclosing the system. This shows that the 3D design has better resource utilization than its 2D counterpart.

The third parameter, total power dissipation, is in fact higher in 3D FPGA in terms of absolute value. The percentage of RLBs used in the 3D design is higher, and more importantly, the implementation of the functional block in an RLB has nMOS pass-transistor based logic where outputs from pass-transistors feed into inverters. This particular implementation increased static power consumption (30% of the total power in the 3D design). Therefore, a direct comparison of the total power consumption would be inappropriate in this case. A more useful metric is the difference in power dissipation due to interconnects. Experiments show that the interconnects for 3D consumes 5% less power than 2D largely because the number of routing channel tracks has been reduced.

3.3.6 Conclusion from the Performance Comparison

From the performance comparison of 2D and 3D FPGAs, it is found that the 3D design does not demonstrate an improvement at the same scale as expected from system-level modeling. System-level modeling and analysis is important for estimating the impact of integrating new technologies. Over the last several years, this type of modeling framework has been used successfully to assess the impact of technology scaling, introduction of Cu and low-k, and modification of system architecture on overall system performance. However, the actual performance of a circuit/system depends on the details in implementation. Designers need to pay special attention and exploit clever techniques to achieve the ideal improvement in system performance. It is apparent that the 3D FPGA design can be further improved (by using a circuit with no static power dissipation instead of pass-transistor based logic in an RLB, or using completely different 3D architecture) to decrease power dissipation, and interconnect delay. Therefore, research and experiment with actual circuit implementations are equally important to fully harness the advantages of a new technology. The experiment with 2D and 3D FPGAs done by the students introduced such approach in the investigation of 3D integration technology [42]. Thus the CAD methodology and tools, such as 3DMagic and ERNI-3D, will aid in the development of successful 3D integration technology.

Chapter 4

Software Development for ERNI-3D

4.1 Overview

ERNI-3D is a client extension of MAJIC, a layout viewer written in Java 2. The initial version of MAJIC (0.5, Alpha) was developed by Yonald Chery using graphics implementation ideas by Manuel Perez [21, 43]. The initial version contains ERNI as the default reliability analysis tool. MAJIC is extended to version 1.0 (Beta) supporting both ERNI and ERNI-3D as its client application. The current version selects an appropriate reliability analysis tool during run-time. When MAJIC is invoked with a directory for a 3D circuit, ERNI-3D appears as its default client. Moreover, ERNI-3D automatically recognizes the bonding orientation of a 3D circuit from the layout files, and can apply reliability models accordingly.

The software design approach that ensures such flexibility in MAJIC and ERNI-3D is detailed in this chapter. The earlier sections present general algorithms and data-structures for layout parsing and extraction of interconnect trees. Then the graphical user interface and reliability analysis algorithms specific to ERNI-3D are discussed. Appendix D includes an overview of the source code for MAJIC and provides a list of *.java* files (Java source code) with corresponding functional descriptions. This chapter also contains references to source files whenever appropriate.

4.2 Layout Parsing and Displaying with MAJIC

MAJIC reads layout files, referred to as *.mag* files, created with MAGIC and 3DMagic. A *.mag* file provides an ASCII file representation of a circuit layout with coordinates of rectangular tiles representing mask layers [23]. A set of coordinates have a header tag << layer >>, where "layer" defines the mask type of following rectangles. Table 4.1 shows a sample *.mag* file. While parsing such a file, MAJIC decomposes rectangular elements into mask layers and their positions (source code: *ParseMAGFile.java*). The

Table 4.1: *A sample* .mag *layout file.*

```
magic
tech scmos
timestamp 962374816
<< polysilicon >>
rect -21 -4 -12 -1
rect -8 -4 10 -1
<< metal1 >>
rect -27 35 -21 36
rect 8 35 20 50
<< polycontact >>
rect -12 -4 -8 0
<< labels >>
rlabel polysilicon -1 -3 -1 -3 1 opa
rlabel metal1 -10 2 -10 2 1 m1th
<< end >>
```

mask information is stored in a data-structure known as corner-stitching, introduced in MAGIC [27]. According to the corner-stitching data-structure, multiple planes, such as active, oxide, poly, and metal1, are superimposed on each other to represent a layout. Each plane contains different types of non-overlapping rectangular tiles with stitches at the four corners. Figure 4-1 illustrates a representation of three corner-stitched solid tiles in a single plane.

The Java source files defining the corner-stitching data-structure in MAJIC are *Plane.java, PlaneType.java, Tile.java,* and *TileType.java. Layout.java* implements the class that puts together multiple *Tile* and *Plane* objects to provide an internal representation of a layout. While plotting the layout on an application window, the

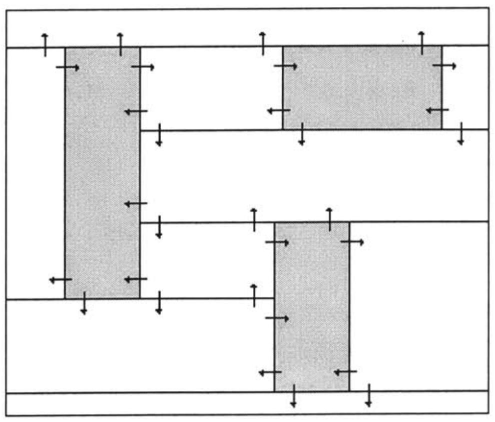


Figure 4-1: *Corner-stitched representation of a single plane with multiple tiles in a MAGIC layout. The gray areas are solid tiles that are corner-stitched to neighboring space tiles.*

Planes are accessed in a serial fashion, and the corner-stitched *Tiles* on each *Plane* are painted on the screen in an appropriate color (source code: *MAJICcomponent.java*).

A *.mag* file only provides the coordinates of mask layers in a layout. Therefore, further information on the connectivity and type of mask layers is retrieved from the technology file, scmos3D.tech26 (Appendix A). While parsing a *.mag* file, the tiles that also represent contacts and inter-wafer vias, such as, pcontact, m2c, and m2topc, are specially tagged. A contact tile has multiple representations, one at every plane that it connects. For example, m2contact connects metal interconnects from metal1 and metal2 planes, and has one tile representation at the metal1 plane and another at the metal2 plane. The Java source file, *TechDB.java*, parses the technology file to retrieve such crucial connectivity information for contacts. It also identifies the inter-wafer vias specific to 3D integration technology from their representation with abstract layers (section 2.3.2).

4.3 Extraction of Interconnect Trees

The corner-stitching data-structure in MAJIC enables several key operations of a CAD layout tool in a very efficient manner. Two critical operations are finding all the tiles in a given area and searching for neighboring tiles in a plane. Both operations are at the heart of extracting interconnect geometries from a layout. Given a particular tile, all the adjacent tiles are retrieved from the corner-stitches implemented as an ArrayList[1]. The adjacent tiles are then stored in a Vector[2] (source code: *ITree.java*) [44]. Using a depth-first search algorithm, the top left-most tile is identified, and the whole tree is built via a depth-first walk on adjacent tiles [45]. Figure 4-2 illustrates an interconnect tree extraction from corner-stitched representation of tiles.

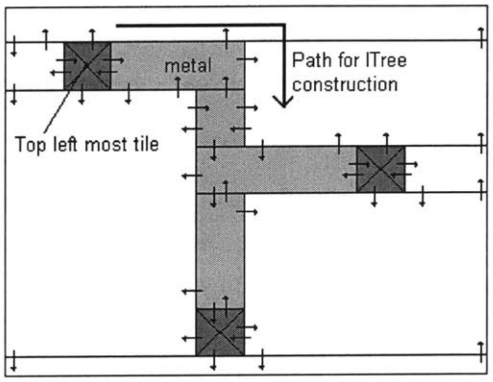


Figure 4-2: *Extraction of an interconnect tree from corner-stitched representation of tiles. Tiles marked with a cross are contact tiles and are the starting and terminating points for this particular tree.*

Setting the top left-most tile as a starting point facilitates the computation of several useful interconnect parameters along the possible paths in an interconnect tree. The current version of MAJIC calculates the length of all possible paths from the coordinates of rectangular tiles. Moreover, the parameters are calculated at the same time when a tree is being built with the depth-first walk algorithm. An *ITree* object, the internal representation of an interconnect tree, stores the parameters in a path-table for future uses. For the interconnect tree shown in figure 4-2, the path-table contains data for two possible paths, starting at the top left-most contact tile

[1]The ArrayList class in Java implements a fixed size array.

[2]The Vector class in Java implements a growable array of objects. Like an array, it contains components that can be accessed using an integer index.

and ending at the two terminating contact tiles.

In order to construct a 3D tree, the *ITree* objects from different layouts that have connected_to_top or connected_to_bottom type vias at the same positions are first identified. Two *ITree* objects of such category forms a 3D tree with an inter-wafer via connecting them vertically. The height of the inter-wafer via is retrieved from the bonding orientation and particular type of inter-wafer via in use. In every *ITree* object, a list of sub-path is computed starting from the inter-wafer via tile and ending at other terminating contact or regular tiles. To compute the longest length of a 3D path, that extends through both *ITree* objects, the height of the inter-wafer via is added to the longest lengths from the sub-paths. If this length is greater than the longest path lengths from individual *ITree* objects, it is considered the longest path for the 3D tree. Figure 4-3 shows a 3D interconnect tree representation from interconnect trees of two different planes and the path with the longest length.

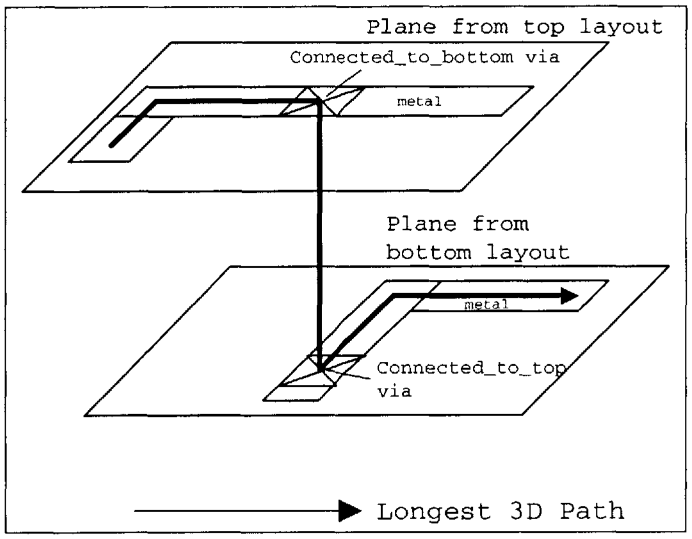


Figure 4-3: *Formation of a 3D tree with two interconnect trees from different layouts. The longest electrical path in this tree would be along the direction of the arrow.*

4.4 Client Extension: ERNI-3D

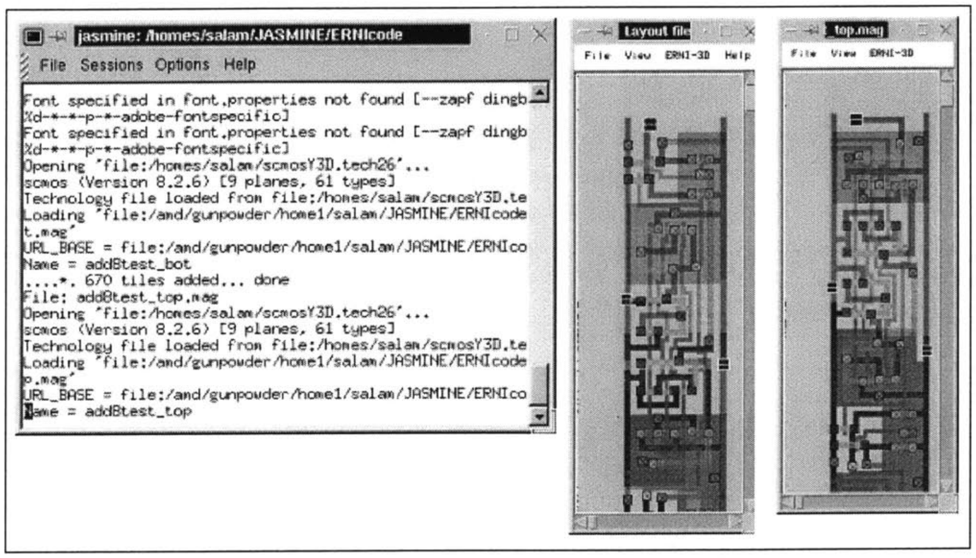
As mentioned in the overview, MAJIC automatically establishes ERNI-3D as its client extension for reliability analysis whenever a directory name for a 3D circuit layout is specified at the command prompt with a *-3d* or *-3D* flag. The major difference between MAJIC with ERNI and with ERNI-3D is the ability to handle multiple layout files in the latter one. The *MAJICapp* class implements the functionality for retrieving the layout files from a 3D circuit layout directory and interpreting the bonding orientation from file names (source code: *MAJICapp.java*). Then an instance of *MAJICframe* class is created for each layout file. The *MAJICframe* is responsible for managing the window display for a layout. It functions independently with its own data-structures and classes, such as *MAJICcomponent*, *TechDB*, *LayoutDB*, and *ITree*. While multiple instances of the *MAJICframe* class manage the different layouts in a 3D circuit, the *ERNIw3D* class maintains the interaction between the layouts (source code: *ERNIw3D.java*). The *ERNIw3D* class has access to the *MAJICapp*'s class-variable[3] "waferframes" of type Vector. "Waferframes" contains the list of current *MAJICframe* objects. *ERNIw3D* applies its computational methods to every layout of a 3D circuit, and updates the layouts by traversing the "waferframes" Vector.

Figure 4-4 shows the graphical user interface of MAJIC with ERNI-3D invoked with the 3D adder. The layout windows display the two layouts for top and bottom wafers of the 3D adder, and the other console window reports intermediate and final results from reliability analysis. The menu options available in the "ERNI-3D" menu are listed in table 4.2 with a brief note on their functionalities. Algorithms for the reliability models in ERNI-3D are detailed in the following sections.

[3]A class-variable is a static variable in a class. It is only associated with a class not with multiple instances or objects.

Table 4.2: *Menu options available at the "ERNI-3D" menu.*

#	**Menu Option**	**Description**
1	*Apply JmaxL filter*	Applies design rule current (j_{max}) to all interconnect trees and filters out the first group of immortal trees.
2	*Apply Random Boundary Conditions*	Assigns marginal voltages to the contact points in the interconnect trees that failed the $J_{max}L$ filter.
3	*Apply Circuit Extraction*	Extracts the circuit from the interconnect trees that failed the $J_{max}L$ filter
4	*Apply Simulation Engine*	Simulates the circuits extracted from the interconnect trees with assigned marginal voltages.
5	*Apply JL filter*	Applies the current density filtering algorithm on mortal trees with their calculated current density.
6	*Apply Default Reliability Model*	Applies the default reliability model on mortal trees for electromigration reliability analysis.
7	*Identify 3D Trees*	Identifies the interconnect trees from different layouts that are limbs of 3D trees.
8	*Apply 3D Bond Reliability Model*	Applies the bond reliability model associated with Cu-wafer bonding.
9	*Apply Full-Chip Model*	Applies the full-chip model to combine different reliability figures in one layout.
10	*Apply 3D-Chip Model*	Applies the 3D-chip model to combine the reliability figures from different layouts to get a single number for a 3D circuit.
11	*Reset Analysis*	Resets all reliability computation and allows a user to start over.


jasmine: /homes/salam/JASMINE/ERNIcode
File Sessions Options Help
Layout file
File View ERNI-3D Help
top.mag
File View ERNI-3D

```
Font specified in font.properties not found [--zapf dingb
%d-*-*-p-*-adobe-fontspecific]
Font specified in font.properties not found [--zapf dingb
%d-*-*-p-*-adobe-fontspecific]
Opening 'file:/homes/salam/scmosY3D.tech26'...
scmos (Version 8.2.6) [9 planes, 61 types]
Technology file loaded from file:/homes/salam/scmosY3D.te
Loading 'file:/and/gunpowder/home1/salam/JASMINE/ERNIcode
t.mag'
URL_BASE = file:/and/gunpowder/home1/salam/JASMINE/ERNIco
Name = add8test_bot
....*. 670 tiles added... done
File: add8test_top.mag
Opening 'file:/homes/salam/scmosY3D.tech26'...
scmos (Version 8.2.6) [9 planes, 61 types]
Technology file loaded from file:/homes/salam/scmosY3D.te
Loading 'file:/and/gunpowder/home1/salam/JASMINE/ERNIcode
p.mag'
URL_BASE = file:/and/gunpowder/home1/salam/JASMINE/ERNIco
Name = add8test_top
```

Figure 4-4: *Graphical User Interface of MAJIC with ERNI-3D. In this example, MAJIC is invoked with the 2-wafer 3D adder circuit.*

4.5 Reliability Algorithms Inherited from ERNI

ERNI-3D inherits the concept of *Hierarchical Reliability Analysis* from ERNI. This concept allows a computationally manageable approach for doing a layout-level reliability analysis, and was proposed by Stefan Riege, and Carl Thompson [19]. The main idea is to isolate the interconnect trees that are more prone to failure using different levels of filters. After the filtering steps, the number of such interconnect trees remaining are usually small. Then computationally heavy as well as stringent reliability models are applied to the remaining trees for accurate reliability figures. A complete description of the concept and related algorithms can be found in reference [7]. The following two sections briefly discusses the filtering algorithms and default reliability model in the *Hierarchical Reliability Analysis* and how it is incorporated in ERNI-3D.

4.5.1 Filter Steps

Both ERNI and ERNI-3D have several filtering steps to identify "immortal" trees (trees that will never fail). The concept of immortality in an interconnect tree is extended from straight, stud-to-stud interconnects and based on a critical current-density length product, $(jL)_{crit}$, necessary for void nucleation. The maximum stress difference in an interconnect tree is given by the path that has the highest sum of the jL products summing over the limbs in the path [7],

$$\Delta\sigma_{max} = \frac{Z^* e\rho}{\Omega}(jL)_{eff} \tag{4.1}$$

with,

$$(jL)_{eff} \equiv^{all\ junctions\ i,j} max \left(\sum_k j_k L_k\right) \tag{4.2}$$

where $\Delta\sigma_{max}$ is the difference in maximum stress, Z^* is the effective atomic charge number, e is the fundamental charge, ρ is the electrical resistivity, and Ω is the atomic volume used in calculating chemical potential. The sum in equation 4.2 is taken over all limbs connecting junctions i and j.

Using equation 4.1, an upper limit for electromigration-induced stress increase in an interconnect tree can be calculated. When the effective current-density length product, $(jL)_{eff}$, is less than the critical product necessary for void nucleation, $(jL)_{crit}$, the tree will be immortal. The term, $(jL)_{crit}$, is adopted from experimental values for different interconnect technologies. For example, the $(jL)_{crit}$ for Aluminum interconnects is $4 \times 10^3 A/cm$ [46].

The filtering methods in ERNI and ERNI-3D estimate $(jL)_{eff}$ in a number of ways (source code: *ERNIw3D.java*). The tasks of identifying all possible paths in an interconnect tree and computing lengths along the paths are already discussed in section 4.3. The first filtering algorithm invoked by Menu Option 1, "Apply JmaxL filter" (table 4.2) assumes j_{max} set by design practice while computing $(jL)_{eff}$. Then Menu Options $2-5$ estimate $(jL)_{eff}$ by computing current-density, j, from the assigned boundary voltages at the contact points of a tree. The filtering process with $(jL)_{eff}$ estimated using the latter approach is less conservative than filtering with $(j_{max}L)_{eff}$. The immortal trees are identified with these two filtering steps and separated from the list of trees that furthers into rigorous reliability analysis. Figure 4-5 illustrates the steps of reliability analysis with block diagrams.

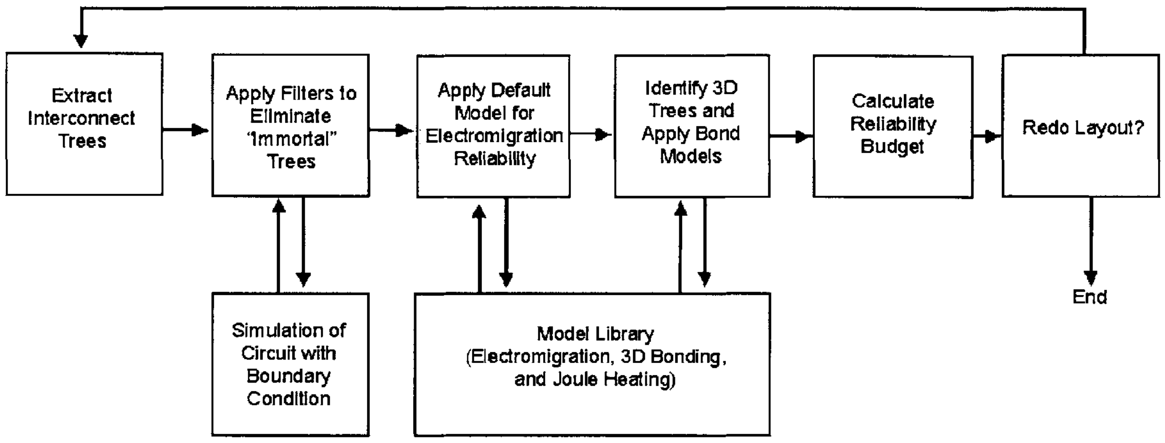

Extract Interconnect Trees
Apply Filters to Eliminate 'Immortal" Trees
Apply Default Model for Electromigration Reliability
Identify 3D Trees and Apply Bond Models
Calculate Reliability Budget
Redo Layout?
End
Simulation of Circuit with Boundary Condition
Model Library (Electromigration, 3D Bonding, and Joule Heating)

Figure 4-5: *Flow diagram of reliability analysis in ERNI-3D.*

4.5.2 Default Electromigration Reliability Model

After the filtering process, lifetimes of the mortal trees are estimated with a conservative default electromigration reliability model. Failure in an interconnect tree can occur due to the resistance increase from voiding in the event of tensile stress, or due to cracks in the passivation or liner materials, and metallic extrusions in the event of large compressive stress. A void forms and starts to grow when the tensile stress exceeds the critical stress necessary for void nucleation, σ_{nucl}, at time t_{nucl}. Eventually, an increased size in the void causes an unacceptable resistance increase at time t_{growth}. Both t_{nucl} and t_{growth} are estimated, and the longer time is taken as the time to failure due to voiding, t_{void}. Similarly, the time for extrusion due to critical compressive stress, $t_{extrusion}$, is estimated, and and the overall time to failure, t_{fail}, is taken to be the minimum of t_{void} and $t_{extrusion}$. Moreover, the median time to failure of an interconnect tree is the same as t_{fail}. The equations for computing t_{nucl}, t_{growth}, $t_{extrusion}$ are discussed in details in reference [7].

The statistical distribution of electromigration test failures has been found in many studies to approximate a lognormal distribution [7, 10]. When a random variable X is normally distributed with finite mean μ and positive variance σ^2, then a random variable T with

$$T = e^X \tag{4.3}$$

is said to have a lognormal distribution. Median time to failure, t_{50}, of T is

$$t_{50} = e^\mu \tag{4.4}$$

Therefore, mean time to failure ($MTTF$) of an interconnect tree can be calculated with

$$MTTF = \mu = \ln t_{50} \tag{4.5}$$

Thus both mean and median times to failure can be calculated using the default reliability model. The model was implemented in the first version of MAJIC (source code: *ERNI.java*), and the current version inherits the model for reliability computation associated with electromigration in 3D circuits.

4.6 Methods for New Models

MAJIC with ERNI-3D has several methods[4] defined in *ERNIw3D.java* that aid the reliability analysis of 3D interconnect trees. As mentioned in table 4.2, the new menu options are "Identify 3D Trees", "Apply 3D Bond Reliability Model", "Apply 3D-Chip Model", and "Reset Analysis". While parsing interconnect trees from layouts, the *ITree* objects that form a 3D tree are marked. The "Identify 3D Trees" option invokes the method to highlight the marked interconnect trees with white borders. Then the "Apply 3D Bond Reliability Model" invokes the method for reliability analysis with those 3D trees.

The approach to reliability analysis in a 3D tree is fundamentally different from that in 2D trees. Although the inter-wafer vias are limbs of a 3D tree connecting multiple 2D trees from different wafers, they need to be specially treated for accounting the reliability associated with bonding. A preliminary approach to reliability analysis

[4]Methods are functions or subroutines in a Java class.

in a 3D tree is described here. Figure 4-6 shows a simple 3D tree. The $MTTF$s of two 2D trees at the top and bottom can be calculated using the Default Reliability Model discussed in the previous section. The $MTTF$ of an inter-wafer via has to be calculated using sophisticated models for reliability issues associated with 3D bonding and increased joule heating [15].

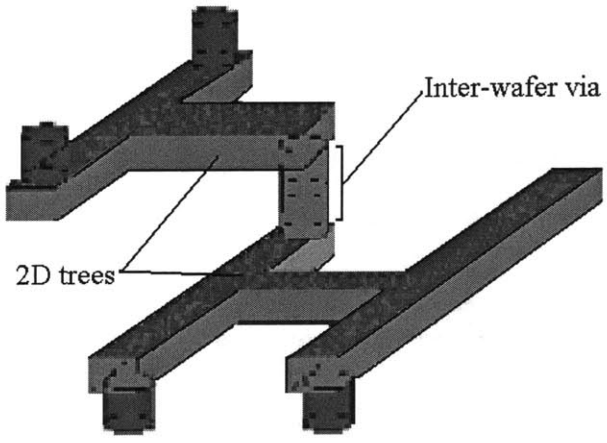


Figure 4-6: *Approach to reliability analysis in a 3D tree.*

Therefore, in general, a 3D tree will have multiple $MTTF$s from its limbs: 2D trees from different wafers and one or more inter-wafer vias. Multiple $MTTF$s from the components can be combined using the same procedure as used in 3D-chip and full-chip models. The method for 3D-chip model incorporates multiple reliability figures from different layouts to estimate the reliability of the whole circuit. Both the 3D-chip model and full-chip model, originally used in ERNI, are based on the same principal, interpolation of multiple $MTTF$s using a probability distribution, and discussed in the following section in details. Finally, the method for "Reset Analysis" does not contribute directly in computing any reliability figure. However, it aids the flow of analysis by allowing a user to reset all computations and start over without restarting MAJIC with the same circuit.

4.7 Full-Chip and 3D-Chip Reliability Models

The goal of both full-chip and 3D-chip models is to combine multiple reliability figures. The full-chip reliability model combines the $MTTF$s of multiple interconnect trees from the same layout and provides one $MTTF$ for a chip on a single wafer. The 3D-chip model again combines the different $MTTF$s of multiple wafers/layouts in a 3D circuit to report a single $MTTF$. Therefore, the same algorithm based on a probability density function, applies to both models.

Let T be the time to failure of a reliability unit, and $f_T(t)$ is the probability density function (PDF) of the continuous random variable T. $f_T(t)$ can be modelled as a PDF of an exponential random variable with λ as its positive exponent factor [47], i.e.

$$f_T(t) = \begin{cases} \lambda e^{-\lambda t} & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

It is important to note that $f_T(t)$ is not the probability function, rather $f_T(t)$ needs to be integrated over proper interval to get the probability of T falling within that interval. $P(T \geq x)$, probability that T exceeds a certain value x, would report the probability that an unit is reliable or functional within time x.

$$\begin{aligned} P(T \geq x) &= \int_x^\infty \lambda e^{-\lambda t} dt \\ &= -e^{-\lambda t}\Big|_x^\infty \\ &= e^{-\lambda x} \end{aligned} \tag{4.7}$$

Writing $P(T \geq x)$ as the probability function $R(t)$,

$$R(t) = P(T \geq t) = e^{-\lambda t} \tag{4.8}$$

$R(t)$ is a proper probability function plotted against time as in figure 4-7. $R(t)$ is the measure of probability that at time t the reliability unit is still operating, and λ defines how fast that probability changes.

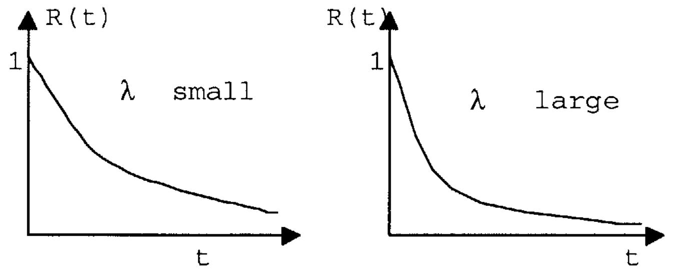


Figure 4-7: *Plot of a probability function indicating the reliability of a unit versus operating time.*

At time $t = 0$, an interconnect tree will definitely be reliable, and hence $R(t = 0) = 1$. The probability decreases afterwards in an exponential manner indicating that as the time increases, an interconnect tree prone to failure will be less reliable (lower $R(t)$) or will have a higher chance of failing. This phenomenon coincides with the actual experiments of failure, such as void formation and extrusion, in interconnect trees. Therefore, $f_T(t)$ in equation 4.6 and related probability density laws can be correctly used for modeling the reliability figures in a circuit layout.

The $MTTF$ of an interconnect tree is equivalent to the expected value of T and can be calculated from equation 4.6.

$$\begin{aligned} MTTF &= E[T] \\ &= \int_0^\infty t\lambda e^{-\lambda t} dt \\ &= (-te^{-\lambda t})\Big|_0^\infty + \int_0^\infty e^{-\lambda t} dt \\ &= 0 + \int_0^\infty e^{-\lambda t} dt \\ &= \int_0^\infty R(t) dt \\ &= -\left.\frac{e^{-\lambda t}}{\lambda}\right|_0^\infty \\ &= \frac{1}{\lambda} \end{aligned} \tag{4.9}$$

Therefore, only the $MTTF$ of an interconnect tree is sufficient to define the PDF,

$f_T(t)$, associated with it. To combine multiple $MTTF$s, a joint PDF $f_{joint}(t)$ or equivalently a joint $R_{joint}(t)$ needs to be defined. Time to failures of different interconnect trees are independent random variables. The current flow in one interconnect tree drives the electric behavior of adjacent trees. However, the filtering algorithms and reliability models described in the previous sections resolve any dependency by treating an interconnect tree as a separate reliability unit and assigning proper boundary conditions while evaluating the models. Therefore, $R_{joint}(t)$ can be calculated using the multiplication rule for independent random variables.

$$\begin{aligned} R_{joint}(t) &= e^{-\lambda_{joint} t} \\ &= R_1(t) \cdot R_2(t) \cdot R_3(t) \cdots R_n(t) \\ &= e^{-\lambda_1 t} \cdot e^{-\lambda_2 t} \cdot e^{-\lambda_3 t} \cdots e^{-\lambda_n t} \\ &= e^{-(\lambda_1+\lambda_2+\lambda_3+\cdots+\lambda_n)t} \end{aligned} \tag{4.10}$$

which suggests,

$$\lambda_{joint} = \lambda_1 + \lambda_2 + \lambda_3 + \cdots + \lambda_n \tag{4.11}$$

equivalently,

$$\frac{1}{MTTF_{joint}} = \frac{1}{MTTF_1} + \frac{1}{MTTF_2} + \frac{1}{MTTF_3} + \cdots + \frac{1}{MTTF_n} \tag{4.12}$$

Equation 4.12 shows the parallel resistor averaging property for calculation of a single $MTTF$ from multiple $MTTF$s. This produces the same formula for combining $MTTF$s as used in ERNI and proposed by Donald Troxel [20]. For simplicity, the above analysis is done with random variables with exponential distributions. However, such an analysis also applies to normal distributions. Finally, a joint median time to failure can be calculated from $MTTF_{joint}$ using equation 4.4. Both the full-chip and 3D-chip models use the parallel resistor averaging property of $MTTF$s. The methods for these two models are defined in the *ERNIw3D* class.

Chapter 5

Simulation with Test Circuits

5.1 Overview

A rigorous simulation of ERNI-3D was carried out with the 3D adder and FPGA. The purpose was to verify the functionality of ERNI-3D as well as to achieve some preliminary reliability results with the test circuits. This chapter illustrates the simulation work and discusses some of the results.

5.2 Debugging with 3D Adder

As discussed in section 3.2, the 3D adder has a hierarchical structure with simple 1-bit adder modules. The repetitive pattern allowed a manual count of interconnect geometries. Table 5.1 shows the number of interconnect trees in the 3D adder.

Table 5.1: *Number of interconnect trees in the 3D adder.*

Top wafer: adder8_top.mag	
# of interconnect trees in metal1	43
# of interconnect trees in metal2	12
Bottom wafer: adder8_bot.mag	
# of interconnect trees in metal1	42
# of interconnect trees in metal2	21
3D Structure	
# of 3D trees	3

While debugging ERNI-3D with the 3D adder, the internal data-structures responsible for storing the interconnect geometries, such as *Layout* and *ITree*, were manually traced. Such manual trace verified the accurate bookkeeping of interconnect trees in ERNI-3D.

Table 5.2: *Reliability analysis with the 3D adder.*

Top wafer: adder8_top.mag		
	Plane: metal1	
# of immortal trees	37	
# of mortal trees for computing MTTF	6	$MTTF = 1.74 year$
	Plane: metal2	
# of immortal trees	8	
# of mortal trees for computing MTTF	4	$MTTF = 2.22 years$
Bottom wafer: adder8_bot.mag		
	Plane: metal1	
# of immortal trees	35	
# of mortal trees for computing MTTF	7	$MTTF = 1.33 year$
	Plane: metal2	
# of immortal trees	17	
# of mortal trees for computing MTTF	4	$MTTF = 2.22 years$
	3D Chip Analysis	
# of 3D trees	3	
Reliability of top wafer		$MTTF = 0.975 year$
Reliability of bottom wafer		$MTTF = 0.84 year$
Reliability of 3D chip		$--$

5.3 Reliability Analysis with 3D Adder

The results of reliability analysis with the 3D adder are summarized in table 5.2. The operating voltage of the adder was set to $5V$. The analysis is done in steps according to the menu options from the "ERNI-3D" menu. The combined $MTTF$ from multiple interconnect trees is derived using the parallel resistor averaging algorithm described in section 4.7. The reliability figures of the top and bottom wafers are also computed in a similar fashion. For accuracy, a reliability budget for the 3D chip is not computed in the table. Although an $MTTF$ can be computed from the two $MTTF$s of the top and bottom wafers, the result would not reflect actual reliability as the bond reliabilities are not yet considered.

For completeness, the reliability figures from 2D counterpart of the 3D adder are shown in table 5.3. There is an interesting observation from tables 5.2 and 5.3. The goal of 3D integration is to lower the interconnect length by mapping the interconnects into different wafers. The total number of interconnect trees in the 3D adder is slightly higher, 118 in 3D versus 106 in 2D. However, it is apparent from the layouts that the 3D version contains a higher number of trees with shorter length. Since trees with shorter length (smaller jL) usually have improved electromigration reliability, the $MTTF$s from the 3D version of a circuit should be lower. The results from tables 5.2

Table 5.3: *Reliability analysis with the 2D version of 3D adder.*

Layout file: add8.mag		
	Plane: metal1	
# of trees	82	
# of immortal trees	72	
# of mortal trees for computing MTTF	10	$MTTF = 0.89 year$
	Plane: metal2	
# of trees	24	
# of immortal trees	16	
# of mortal trees for computing MTTF	8	$MTTF = 1.12 year$
	Full Chip Analysis	
Reliability of full chip		$MTTF = 0.49 year$

and 5.3 match our intuition. The $MTTF$s from metal1 and metal2 plane of the 2D circuit are lower than those of the 3D circuit, 0.89 year versus 1.74 year or 1.33 year for metal1 plane, and 1.12 year versus 2.22 years for metal2 plane. The full-chip $MTTF$ of the 2D counterpart is also lower than the $MTTF$ of top or bottom wafer in the 3D circuit. Therefore, the 3D version is more reliable considering only the electromigration effects.

5.4 Reliability Analysis with 3D FPGA

As discussed in section 3.3, the 8-bit encryption processor is mapped into Rothko FPGA architecture with two wafers in the stack. The layout for top wafer, *3dfpga_top.mag*, has a driver block and 56 Routing and Logic Blocks (RLBs). Similarly, the layout for bottom wafer, *3dfpga_bot.mag*, has 56 RLBs. There are approximately 43 inter-wafer vias for 3D connections. Both layouts are fairly large, and the reliability analysis is done in a modular fashion. First, the interconnect geometries in a single driver block and RLB are tallied, and $MTTF$s are calculated. Other $MTTF$s are calculated from the interconnect geometries dedicated to the vertical routing channels in each layout. The vertical routing channel is not necessarily a combination of long wires, rather it is a network connecting different RLBs in the array structure in each layout. In the top layout, it also feeds inputs from the driver block to selected RLBs. Moreover, all the 3D trees exist in vertical routing channels of the two layouts as the inter-wafer vias are only placed at the channels.

If $MTTF_{RLB}$ denotes the mean time to failure of a single RLB, according to the parallel resistor averaging algorithm, the mean time to failure of an array structure, $MTTF_{array}$ with 56 RLBs will be $\frac{MTTF_{RLB}}{56}$. For the top layout, three $MTTF$s are computed from the array structure, driver, and vertical routing channel, and combined to get a single reliability budget. The reliability budget for the bottom layout has contributions from the array structure and vertical routing channel. Tables 5.4 and 5.5 illustrate the reliability analysis with the 3D FPGA.

Although a reliability budget for the 2D FPGA implementation of the 8-bit encryption processor was not computed, the interconnect geometries from the layouts showed a similar observation as in the 2D and 3D adder layouts. The same RLB and driver blocks are used in the 2D version. However, the vertical routing channel in 2D has significantly longer interconnect trees as the array structure consists of 96 (16×6) RLBs. Therefore, we would expect lower $MTTF$s in the 2D version.

Table 5.4: *Reliability analysis with the 3D FPGA.*

RLB Layout: rlb.mag		
	Plane: metal1	
Total # of trees	132	
# of immortal trees	100	
# of mortal trees for computing MTTF	32	$MTTF = 3.49 years$
	Plane: metal2	
Total # of trees	26	
# of immortal trees	14	
# of mortal trees for computing MTTF	12	$MTTF = 7.53 years$
Driver Layout: driver.mag		
	Plane: metal1	
Total # of trees	32	
# of immortal trees	28	
# of mortal trees for computing MTTF	4	$MTTF = 2.57 years$
	Plane: metal2	
Total # of trees	14	
# of immortal trees	14	
# of mortal trees for computing MTTF	0	$MTTF = N/A$
Top wafer: 3dfpga_top.mag		
	Vertical Routing Channel	
	Plane: metal1	
Total # of trees	180	
# of immortal trees	85	
# of mortal trees for computing MTTF	95	$MTTF = 2.80 years$
	Plane: metal2	
Total # of trees	570	
# of immortal trees	570	
# of mortal trees for computing MTTF	0	$MTTF = N/A$
Reliability of RLB-array		$MTTF = 0.042 year$
Reliability of driver block		$MTTF = 2.57 years$
Reliability of rout. chan.		$MTTF = 2.80 years$

Table 5.5: *Reliability analysis with the 3D FPGA (continued).*

Bottom wafer: 3dfpga_bot.mag		
Vertical Routing Channel		
Plane: metal1		
Total # of trees	258	
# of immortal trees	116	
# of mortal trees for computing MTTF	142	$MTTF = 1.39 year$
Plane: metal2		
Total # of trees	628	
# of immortal trees	628	
# of mortal trees for computing MTTF	0	$MTTF = N/A$
Reliability of RLB-array		$MTTF = 0.042 year$
Reliability of rout. chan.		$MTTF = 1.39 year$

3D Chip Analysis		
# of 3D trees	43	
Reliability of top wafer		$MTTF = 0.041 year$
Reliability of bottom wafer		$MTTF = 0.0407 year$

5.5 Summary of Results

The reliability analyses with the 3D adder and FPGA demonstrate proper functionality of ERNI-3D. Some observations are also made based on the reliability budgets from the test circuits. The mean time to failures of overall systems may seem to be low (less than a year). In fact, these figures reflect the continuous operating time of the circuit and does not consider any effects of idle mode. It is also observed that electromigration reliability is improved in bonded 3D circuits as 3D mapping shortens the wire length in interconnect trees. However, no conclusion on overall reliability can be reached as this initial analyses do not fully account for bond reliability and effect of joule heating.

Chapter 6

Future Work

6.1 Overview

This chapter provides some future directions for research and development in the area of Reliability CAD (RCAD) tools. The concept of reliability computation at the circuit-layout level has been evolving for last several years. With emerging semiconductor processing technology, such as 3D integration, reliability has become a critical issue. Moreover, the availability of Reliability CAD tools will allow designers compare reliability of their circuits with different design topologies and process technologies. Therefore, there are enough opportunities for enhancement of ERNI-3D to fully meet such goals.

6.2 Incorporation of Sophisticated Reliability Models

As the processing technology for 3D integration will advance, sophisticated reliability models will be developed. Such models have to be incorporated into ERNI-3D to keep the tool up-to-date. Moreover, the current version of ERNI-3D lacks bonding models for a Cu-Cu wafer bonding process in 3D integration. The bonding models are under development [15] and will be included in the next release of ERNI-3D. The object-oriented software architecture in ERNI-3D will enable incorporation of any new models with a minimal effort. After the inclusion of new models, reliability analyses similar to the one described in the previous chapter have to be done.

6.3 Layout Editing in ERNI-3D

MAJIC, the parent application of ERNI-3D, is a layout viewer and does not allow any editing in the circuit layout. However, after running initial reliability simulations, a designer may want to change the geometries of mortal interconnect trees and rerun ERNI-3D for new reliability figures. Consequently, layout editing in MAJIC with ERNI-3D is a very useful feature. It will eliminate manual bookkeeping with interconnect trees when a layout is ported to any other layout editor, such as MAGIC, for updates and changes. However, adding such a feature in MAJIC is a nontrivial task. Significant upgrade in graphical user interface will be required for handling user input from the layout display. Moreover, the internal data-structures will require real-time update while the layout is being changed. Advanced software algorithms such as multi-threaded programming, pipelining, and semaphores, have to be employed to efficiently manage the computational load.

Table 6.1: *Comparison of features available in MAGIC, 3DMagic, and MAJIC.*

Feature	**MAGIC**	**3DMagic**	**MAJIC**
3D layout	no	yes	yes
Jsim	no	no	yes
Boundary condition	no	no	yes
Layout viewer	yes	yes	yes
Layout editor	yes	yes	no
DRC	yes	yes	no
Tech file parsing	yes	yes	yes
File saving	yes	yes	no
Language	C	C	Java
Reliability	no	no	ERNI ERNI-3D

Table 6.1 summarizes the available features of MAJIC, and provides a comparison with MAGIC and 3DMagic. Another feature that comes naturally with layout editing in CAD tools is a design rule checker. A design rule checker (DRC) verifies whether interconnect geometries meet the rules specified by a fabrication process. If designers

are allowed to edit the layout in MAJIC before rerunning ERNI-3D, MAJIC has to check for any design rule violations in the new interconnect geometries.

6.4 Reliability Analysis based on Local Temperature

While computing the median time to failure of an interconnect tree in a large circuit, the models usually assume a fixed operating temperature. This approach is more applicable for conventional or 2D integration technology. However, the effect of increased joule heating in a 3D circuit is more severe as the heat dissipation from wafer-stack poses a challenge. Therefore, the reliability models for 3D integration technology should include local temperature[1] as one of the input parameters to calculate more accurate median time to failures.

The task of calculating the actual operating temperature in a layout is not trivial. Currently, no CAD tool supports such operation. Operating temperature is directly proportional to power dissipation in a circuit. Power dissipation is calculated by simulating circuit netlists extracted from a layout. After the extraction phase, there is no direct mapping between the extracted netlist and area of the layout from which it is extracted. This fundamental approach to circuit extraction has to be extended in order to compute power dissipation by layout area, and thus estimate local temperature.

RCAD tools with the ability to compute local temperature can also facilitate the usage of innovative heat dissipation techniques in a 3D circuit. In response to managing heat dissipation, researchers are already investigating possible solutions by using through-wafer vias filled with higher thermal conductivity materials, and innovative cooling techniques with heat-pipes and forced-air. The through-wafer vias, heat-pipes, or forced-air will be more effective when they are placed in higher temperature zone in a layout. Such placement will require RCAD tools to calculate local temperature.

[1]Local temperature would indicate the operating temperature at the vicinity of an interconnect.

Chapter 7

Conclusion

7.1 Overview

In this research, a novel RCAD tool ERNI-3D has been developed for reliability analysis with bonded 3D integrated circuits. Both the concepts of CAD tools for 3D circuits and layout-level reliability assessment are very recent. There has not been any significant research in these areas before. Therefore, in addition to RCAD tool work, this research has made significant contribution to design methodology for 3D circuits. The following sections outline the achievements of this thesis work.

7.2 3DMagic and Layout Methodology for 3D ICs

One of the major outcomes of this research is the layout methodology for bonded 3D integrated circuits [24]. In bonded 3D integration technology, parts of a circuit are fabricated on different wafers, and then the wafers are bonded with copper or polymer-based glue layer to form a single chip. The circuit on each wafer can be laid out separately with inter-wafer via information embedded in a layout. The inter-wafer via information is generalized into three categories, connected-to-top, connected-to-bottom, and through-wafer vias. The three categories of vias are sufficient for defining almost all types of interconnection between wafers in a 3D stack. A strategy for layout-file management that also incorporates the orientation of each wafer in 3D bonding process is proposed. The layout methodology is implemented in 3DMagic, an extension of MAGIC originally developed at UC Berkeley and widely used in academia. A new technology file has been developed for supporting the abstract layers and preliminary design rules for the inter-wafer vias. Proper connectivity information is also added in the technology file to enable the built-in hierarchical circuit extractor successfully extract a 3D circuit layout. The alignment of multiple layouts and inter-wafer vias can be managed very efficiently with the MAGIC feature "feedback". Using 3DMagic, a 3D 8-bit adder and an 8-bit encryption processor mapped into a 3D FPGA are designed as test circuits for ERNI-3D. The 3D layout methodology is powerful enough for designing a 3D circuit of any complexity, and yet it can be incorporated into existing CAD tools with minimal changes.

7.3 RCAD Tool: ERNI-3D

The final product of this thesis, ERNI-3D, is a tool for reliability analysis associated with electromigration, 3D bonding, and increased joule heating in 3D circuits. ERNI-3D is client extension of MAJIC, a layout viewer written in Java 2 [21]. When MAJIC is invoked with a directory for a 3D circuit laid out in 3DMagic, ERNI-3D automatically recognizes the orientation in bonding process using the proposed layout-file management scheme. Then it parses the layout files and extracts both conventional and 3D interconnect trees. ERNI-3D employs the *Hierarchical Reliability Analysis* approach, and filters out a group of immortal trees (trees that will never fail) using their current-density length products (jL) [7]. After the filtering process, the stringent reliability models are applied to the remaining interconnect trees for computing their median and mean times to failure. Finally, the mean time to failures ($MTTF$s) are combined using a joint probability distribution to provide a single $MTTF$ for the whole chip.

This initial version of ERNI-3D works with 3D circuits with two wafers or device-interconnect layers in the stack. However, the data-structures and algorithms are generic enough to make it compatible with 3D circuits with more than two wafers, and to allow incorporation of more sophisticated reliability models in the future.

7.4 Simulation and Experiment with 3D Circuits

An indirect contribution of this thesis work is the introduction of simulation and experiments with actual implementation of 3D circuits. Over the past years, the investigation of performance improvement in 3D integration has been limited to system-level modeling with theoretical analysis. Availability of 3DMagic led to an experiment with 2D and 3D FPGAs [42]. Researchers implemented an 8-bit encryption processor in both 2D and 3D versions using MAGIC and 3DMagic. Spice netlists were extracted from the layouts and simulated with PowerMill for measuring critical path delay, resource utilization, and total power. The 3D FPGA demonstrated some improvement, however, not at the same scale as expected from system-level modeling. The modeling analysis does not consider actual circuit being implemented, rather it assumes a standard FPGA architecture and straightforward extension of switch boxes to 3D technology. The experiment with 2D and 3D FPGA layouts clearly demonstrated the importance of research with actual circuit implementations to fully harness the advantages of 3D integration technology. Availability of the CAD methodology for 3D integration technology and tools, such as 3DMagic and ERNI-3D, can set such research direction. In fact, researchers at MIT have adopted the 3D CAD methodology developed in this research and set a new focus on developing actual 3D circuits [48].

Appendix A

Excerpt from the Technology File

MAGIC is a technology independent layout editor. The technology file contains all necessary technology-specific information such as mask layers, design and extraction rules, and connectivity information. There is a different technology file for each technology – optical, MEMS, BiCMOS, etc – supported by MAGIC. A new technology file, scmos3D.tech36, has been developed to support 3D integration technology.

A technology file is divided into seventeen sections, and each section has the following format.

```
section name
                    content
end
```

The seventeen sections in a technology file are *tech, planes, types, styles, contact, compose, connect, connect, cifoutput, cifinput, mzrouter, drc, extract, wiring, router, plowing,* and *plot.* A detailed description on the format of technology files can be found in reference [26]. Following is the excerpt from the 3D technology file. Specific additions and changes are emphasized in the sections where entire section content is shown. Elsewhere only the additions are included.

Excerpt from file: **scmos3D.tech26**

```
tech
          scmos
end

version
          version 8.2.6
          description "MOSIS Scalable CMOS Technology for 3D IC design to
          be used for ERNI-3D"
end

planes
          well,w
          implant,i
          active,a
          metal1,m1
          metal2,m2
          metal3,m3
          oxide,ox
end
types
        well     pwell,pw
        well     nwell,nw
        well     capwell,cwell,cw
        well     highvoltnwell,hvnwell,hnwell,hnw
        well     highvoltpwell,hvpwell,hpwell,hpw
        active   polysilicon,red,poly,p
        active   electrode,poly2,el,p2
        active   capacitor,polycap,pcap,cap
        active   wellcapacitor,wellcap,wcap
```

Excerpt from file: **scmos3D.tech26**

```
active    ndiffusion,ndiff,green
active    pdiffusion,pdiff,brown
active    highvoltndiffusion,hvndiff,hndiff
active    highvoltpdiffusion,hvpdiff,hpdiff
metal1    metal1,m1,blue
metal2    metal2,m2,purple
metal3    metal3,m3,cyan
active    ntransistor,nfet
active    ptransistor,pfet
active    entransistor,enfet
active    eptransistor,epfet
active    doublentransistor,nfloating-gate,nfloatg,nfg,nffet
active    doubleptransistor,pfloating-gate,pfloatg,pfg,pffet
active    highvoltntransistor,hvnfet,hnfet
active    highvoltptransistor,hvpfet,hpfet
active    collector,coll,col,co,cl
active    emitter,emit,em
active    pbase,pb
implant   bccdiffusion,bd
active    nbccdiffusion,nbd
active    polycontact,pcontact,polycut,pc
active    ndcontact,ndiffcut,ndc
active    pdcontact,pdiffcut,pdc
active    highvoltndcontact,hndiffcut,hndc
active    highvoltpdcontact,hpdiffcut,hpdc
active    capcontact,ccontact,capc,cc
active    electrodecontact,econtact,ec,poly2contact,p2c
active    collectorcontact,colcontact,colc,coc,clc
```

```
        active      emittercontact,emitcontact,emc
        active      pbasecontact,pbcontact,pbc
        active      nbccdiffcontact,nbdc
        metal1      m2contact,m2cut,m2c,via,v
        metal2      m3contact,m3cut,m3c,via2,v2
        metal3      m2_Top_Contact,m2topc
        metal3      m3_Top_Contact,m3topc
        active      psubstratepcontact,pwcontact,pwc,psc
        active      nsubstratencontact,nwcontact,nwc,nsc
        active      psubstratepdiff,pohmic,ppd,psd
        active      nsubstratendiff,nohmic,nnd,nsd
        active      highvoltpsubcontact,hpwcontact,hpsc
        active      highvoltnsubcontact,hnwcontact,hnsc
        active      highvoltpsubdiff,hpohmic,hpsd
        active      highvoltnsubdiff,hnohmic,hnsd
        active      nplusdopping,ndoping,ndop
        active      pplusdopping,pdoping,pdop
        oxide       throughwafervia,twv,twvia,greenwaffle
        well        3Dcontactbottom,3Dconb
        implant     polybottomcontact,pbcon
        active      m1bottomcontact,m1bcon
        metal3      pad
        oxide       glass
end
contact
            ec     poly2   metal1
            cc     cap     metal1
            pc     poly    metal1
            ndc    ndiff   metal1
```

Excerpt from file: **scmos3D.tech26**

```
    pdc       pdiff     metal1
    nsc       nsd       metal1
    psc       psd       metal1
    clc       col       metal1
    emc       emit      metal1
    pbc       pbase     metal1
    nbdc      nbd       metal1
    m2c       metal1    metal2
    m3c       metal2    metal3
    pad       metal2    metal3    glass
    m2topc    metal2    metal3    twv
    m3topc     metal3   twv
    pbcon     3Dconb    bd        poly
    m1bcon    bd        poly      m1
end
styles
        styletype  mos
        twv        4
        twv        47
        m3topc     22
        m3topc     47
        m2topc     21
        m2topc     47
        pbcon      1
        pbcon      49
        m1bcon     8
        m1bcon     48
end
```

Excerpt from file: **scmos3D.tech26**

```
compose
            paint   pbcon     m1bcon   pbcon
            paint   m1bcon    pbcon    m1bcon
end
connect
           m1,m1bcon/m1                          m1,m1bcon/m1
           m2,m2c/m2,m3c/m2,                     m2,m2c/m2,m3c/m2,
           m2topc/m2,pad                         m2topc/m2,pad
           m3,m3c/m3,m3topc/m3                   m3,m3c/m3,m3topc/m3
           poly,pc/a,pbcon/a,m1bcon/a            poly,pc/a,pbcon/a,m1bcon/a
           m3topc                                m3,m3c/m3,m3topc/m3
           m3topc                                twv
           m2topc                                m2,m2c/m2,m3c/m2,pad,m2topc/m2
           m2topc                                twv
           pbcon                                 poly,pc/a,nfet,pfet,wcap,pbcon/a,m1bcon/a
           pbcon                                 3Dconb
           m1bcon                                poly,pc/a,nfet,pfet,wcap,pbcon/a,m1bcon/a
           m1bcon                                m1,m1bcon/m1
end
cifoutput
            ··· no change in section content ···
end
cifinput
           ··· no change in section content ···
end
mzrouter
            ··· no change in section content ···
end
```

Excerpt from file: **scmos3D.tech26**

```
drc
    width twv 6 \
    "Through-wafer-via width must be at least 6 (3D IC rule #1.1)"
    spacing twv ~(twv),pad 2 touching_illegal \
    "Through-wafer-via spacing from any other material must be 2 (3D IC rule
    #1.2) Through-wafer-via must not overlap with any material (3D IC rule
    #1.3)"
    width m2topc 4 \
    "Metal2_top_contact width must be at least 4 (3D IC rule #2.1)"
    spacing m2topc m3topc,pdc 2 touching_illegal \
    " Metal2_top_contact spacing from pdiffusion contact and Metal3_top_contact
    must be 2 (3D IC rule #2.2)"
    spacing m2topc m3 2 touching_illegal \
    "Metal2_top_contact spacing from metal3 must be 2 (3D IC rule #2.3)"
    edge4way m2topc/m2 ~m2topc/m2 1 ~m2topc/m2 (~m2topc,m2topc)/m2 1 \
    "Metal2 contacts must be rectangular (Magic rules)"
    width m3topc 6 \
    "Metal3_top_contact width must be at least 6 (3D IC rule #3.1)"
    spacing m3topc m2topc 2 touching_illegal \
    "Metal3_top_contact spacing from any other metal3 and 3D contact must be
    2 (3D IC rule #3.2)"
    edge4way m3topc/m3 ~m3topc/m3 1 ~m3topc/m3 (~m3topc,m3topc)/m3 1 \
    "Metal3 contacts must be rectangular (Magic rules)"
    width pbcon 4 \
    "Poly_bottom_contact width must be at least 4 (3D IC rule #4.1)"
    spacing pbcon/w pc 2 touching_illegal \
    "Poly_bottom_contact spacing from Polycontact must be 2 (3D IC rule #4.2)"
    edge4way pbcon/i ~pbcon/i 1 ~pbcon/i (~pbcon,pbcon)/i 1 \
    "Poly contacts must be rectangular (Magic rules)"
```

Excerpt from file: **scmos3D.tech26**

```
    spacing pbcon/w nwell,nsc,nsd,pwell,psc,psd 2 touching_illegal \
    “Poly_bottom_contact spacing from any well or well contacts must be 2 (3D
    IC rule #4.3)”
    spacing pbcon/w poly2,ec/a,cap,capc/a,wcap 2 touching_illegal \
    “Poly_bottom_contact spacing from poly2, poly2 contact or any capacitor must
    be 2 (3D IC rule #4.4)”
    spacing pbcon/w nfet,pfet,enfet,epfet,hnfet,hpfet 2 touching_illegal \
    “Poly_bottom_contact spacing from any fet must be 2 (3D IC rule #4.5)”
    spacing pbcon/w co,em,pb 2 touching_illegal \
    “Poly_bottom_contact spacing from any NPN transistor layer must be 2 (3D
    IC rule #4.6)”
    spacing pbcon/s ndiff,pdiff,hndiff,hpdiff 2 touching_illegal \
    “Poly_bottom_contact spacing from any diffusion must be 2 (3D IC rule #4.7)”
    width m1bcon 4 \
    “Metal1_bottom_contact width must be at least 4 (3D IC rule #5.1)”
    edge4way m1bcon/a ~m1bcon/a 1 ~m1bcon/a (~m1bcon,m1bcon)/a 1 \
    “Metal1 contacts must be rectangular (Magic rules)”
    spacing m1bcon/m1 poly2,ec/a,cap,capc/a,wcap 2 touching_illegal \
    “Metal1_bottom_contact spacing from poly2, poly2 contact or any capacitor
    must be 2 (3D IC rule #5.4)”
    spacing m1bcon/m1 nfet,pfet,enfet,epfet,nffet,pffet,hnfet,hpfet 2 \
    “Metal1_bottom_contact spacing from any fet must be 2 (3D IC rule #5.5)”
    spacing m1bcon/m1 co,em,pb 2 touching_illegal \
    “Metal1_bottom_contact spacing from any NPN transistor layer must be 2 (3D
    IC rule #5.6)”
    spacing m1bcon/m1 ndiff,pdiff,hndiff,hpdiff 2 touching_illegal \
    “Metal1_bottom_contact spacing from diffusion is 2 (3D IC rule #5.7)”
end
```

Excerpt from file: **scmos3D.tech26**

```
extract
        ··· no change in section content ···
end
wiring
        ··· no change in section content ···
end
router
        ··· no change in section content ···
end
plowing
        fixed   nfet,enfet,nffet,pfet,epfet,pffet,glass,pad,,twv,m2topc,m3topc,pbcon
end
plot
      ··· no change in section content ···
end
```

Appendix B

Tutorial on Using 3DMagic in Athena

This appendix provides a tutorial on using 3DMagic in Athena. The tutorial is divided into sections on starting 3DMagic, working with multiple windows, and using the area markers for aligning different layouts. There is a table with different abstract layers for inter-wafer vias and commands for their usage. Some familiarity with the basic usage of MAGIC (MAGIC Tutorial #1: Getting Started) and working with multiple windows (MAGIC Tutorial #5: Multiple Windows) is required for using this tutorial. It is recommended to read chapter 2 of this thesis to understand the layout methodology for 3D circuits.

B.1 Mounting 3DMagic

The first thing to do is to attach the 6.374 locker for running either MAGIC or 3DMagic. Invoke the following commands at the athena prompt:

```
% add 6.374
% set path = (/mit/6.374/ultra-cad/bin $path)
% setenv CAD_HOME /mit/6.374/ultra-cad
```

Once this is done, you are ready to start 3DMagic. cd to the directory where you want to store your magic files and type,

```
% magic -T /afs/athena.mit.edu/user/s/a/salam/www/scmos3D
                    or
% magic -T /afs/athena.mit.edu/user/s/a/salam/www/scmos3D.30
```

The -T option with scmos3D sets *scmos3D.tech27* as the technology file. *scmos3D.tech27* has the extraction rules for 3D integrated circuits laid out in $\lambda = 1\mu m$[1] process.

The -T option with scmos3D.30 sets the technology file to *scmos3D.30.tech27* which has the extraction rules for $\lambda = 0.3\mu m$ process. Both technology files have the same CMOS scalable design rules and abstract layers for inter-wafer vias in a 3D circuit layout. These files are similar to *scmos3D.tech26* (discussed in Appendix A). However, *.tech27* is required for running them with the current version of MAGIC in Athena.

B.2 Working with Multiple Windows

The key of designing 3D circuit in 3DMagic is to layout different wafers in different layout windows with inter-wafer vias embedded in the layout. Therefore, some familiarity with commands and schemes for working with multiple layout windows is required. You should read MAGIC Tutorial #5 for complete reference [28]. The most useful commands for working with multiple windows while doing 3D circuit layouts are as follows:

- To open a new layout window or another layout file in a different window,

  ```
  :openwindow [layoutfile]
  ```

- To close a layout window, position the mouse pointer on the layout window, and type,

[1] λ designates the smallest grid size in a MAGIC layout window.

:closewindow or use macro ‘O’ (capital letter ‘O’)

- To make a particular window an edit window,

    1. select any drawing on that window using the box tool, and then macro ‘s’ or ‘a’
    2. type, :edit

B.3 Painting Inter-wafer Vias

Painting inter-wafer vias are very similar to painting any other mask layers in a layout. After marking a boxed area, the MAGIC command “:paint” with a name of an abstract layer paints the corresponding inter-wafer via. Table B.1 illustrates the commands for different types of inter-wafer vias. Appendix C has tables with design rules for the inter-wafer vias.

Table B.1: *Painting inter-wafer vias for 3D circuit layout in 3DMagic.*

Name\Type	**Command**
Metal3_top_contact or m3topc	:paint m3topc
Metal2_top_contact or m2topc	:paint m2topc
Through-wafer Via or twv	:paint twv
Poly_bottom_contact or pbcon	:paint pbcon
Metal1 to bottom contact	:paint m1bcon and :paint pbcon
Metal1 to top contact	:paint m2c and :paint m2topc

B.4 3D Layout Alignment

Following steps describe the layout alignment from multiple layout windows:

1. Use the mouse pointer to box the area from a layout of one window for its 3D counterpart.
2. :feedback add ‘anyname’ outline (MAGIC command)

- 'anyname' can be mbnd to indicate main boundary for the 3D layout.
- 'outline' type will draw a white outlined box.

3. :feedback add 'inter-wafer_via_name'   (MAGIC command)
   - This will mark the positions for inter-wafer vias with white diagonal lines.
4. Repeat step 3 to mark as many inter-wafer vias as required.
5. After all the inter-wafer vias and main boundary areas are marked,

   ```
   :feedback save afilename
   ```

6. Open the second layout window (the 3D counterpart), and make it editable.
7. To port all the area marks in this layout window,

   ```
   :source afilename
   ```

8. To delete all the feedback, when a chip is fully designed,

   ```
   :feedback clear
   ```

9. The area marks are not saved in a layout file. Therefore, follow step 7 if you open your design and no marks are there.
10. To view all different options of feedback,

   ```
   :feedback help
   ```

The above steps describe the procedure for aligning two layouts from different windows. While designing a 3D circuit with more than two wafers in a stack, the layouts can still be aligned by repeating step 7 for different layout windows.

Appendix C

Layout Design Rules for 3D ICs

The following tables provide the lambda (λ) based design rules for layout of 3D ICs in 3DMagic. These rules are derived using the existing spacing and width rules, and also from the layout methodology described in chapter 2. The "drc" section of the technology file, shown in Appendix A, contains the declarations for the rules. The design rules may differ or some absolute design rules in microns may be required based on which processing technology is used for fabrication and bonding of 3D ICs.

Table C.1: *Preliminary design rules for the layout of 3D ICs using 3DMagic.*

Type	**Connectivity**	**Rule Number**	**Description**
m3topc	metal3 & 3D contact point at the top of the wafer	3D IC rule#3.1	Metal3_top_contact width must be at least 6
		3D IC rule#3.2	Metal3_top_contact spacing from any other metal3 and 3D contact must be 2
		Contact rule	Metal3 contacts must be rectangular
pbcon	poly1 & 3D contact point at the bottom of the wafer	3D IC rule#4.1	Poly_bottom_contact width must be at least 4
		3D IC rule#4.2	Poly_bottom_contact spacing from Polycontact must be 2
		3D IC rule#4.3	Poly_bottom_contact spacing from any well or well contacts must be 2
		3D IC rule#4.4	Poly_bottom_contact spacing from poly2, poly2 contact or any capacitor must be 2
		3D IC rule#4.5	Poly_bottom_contact spacing from any fet must be 2
		3D IC rule#4.6	Poly_bottom_contact spacing from any transistor layer must be 2
		3D IC rule#4.7	Poly_bottom_contact spacing from any diffusion must be 2
		Contact rule	Poly contacts must be rectangular
m1bcon + pbcon	metal1 & 3D contact point at the bottom of the wafer	3D IC rule#5.1	Metal1_bottom_contact width must be at least 4
		3D IC rule#5.4	Metal1_bottom_contact spacing from poly2, poly2 contact or any capacitor must be 2
		3D IC rule#5.5	Metal1_bottom_contact spacing from any fet must be 2
		3D IC rule#5.6	Metal1_bottom_contact spacing from any transistor layer must be 2
		3D IC rule#5.7	Metal1_bottom_contact spacing from any diffusion must be 2
		Contact rule	Metal1 contacts must be rectangular

Table C.2: *Preliminary design rules for the layout of 3D ICs using 3DMagic (continued).*

Type	**Connectivity**	**Rule Number**	**Description**
twv	Through-wafer Via	3D IC rule#1.1	Through-wafer-via width must be at least 6
		3D IC rule#1.2	Through-wafer-via spacing from any other material must be 2
		3D IC rule#1.3	Through-wafer-via must not overlap with any material
m2topc	metal2 & 3D contact point at the top of the wafer	3D IC rule#2.1	Metal2_top_contact width must be at least 4
		3D IC rule#2.2	Metal2_top_contact spacing from pdiffusion contact and Metal3_top_contact must be 2
		3D IC rule#2.3	Metal2_top_contact spacing from metal3 must be 2
		Contact rule	Metal2 contacts must be rectangular

Appendix D

MAJIC Source Code Overview

This appendix contains a complete overview of the code for MAJIC version 1.0 (beta). MAJIC version 1.0 has both ERNI and ERNI-3D as its client application. Using the file-management scheme described in section 2.3.3, MAJIC automatically detects if the provided circuit layout is for a 3D circuit, and then establishes ERNI-3D as the default reliability analysis tool. The complete code of MAJIC is available from Prof. Donald E. Troxel or Prof. Carl V. Thompson at MIT [49]. The code is written in Java 2. This appendix gives an overview of the source code by categorizing the Java classes into functional groups, such as main application classes, graphical interface classes, file parser classes, and so on. The major functionalities and features of all the classes, defined in the java file format, *classname.java*, are briefly described under each category.

D.1 Main Application Classes

The main application classes are responsible for starting the MAJIC application when it is invoked from the command line. These classes also define global variables for use in rest of the application.

D.1.1 MAJICapp.java

MAJICapp is the topmost class of MAJIC. Hence the command line argument that starts the MAJIC application is *java MAJICapp [optional arguments].* The optional argument to invoke MAJIC with ERNI-3D is *-3D or -3d dir*, where *dir* is the directory name in a 3D circuit layout scheme. The primary function of MAJICapp is to process the command line argument to retrieve the layout files and invoke the graphical user interface with either ERNI or ERNI-3D.

D.1.2 Globals.java

This class contains the declaration for global constants, runtime variables, and debugging flags for use in the entire MAJIC application.

D.1.3 HashMap2D.java

The HashMap2D class implements a two-dimensional HashMap[1] where values are stored against two keys. The Java Development Kit (JDK) does not have a predefined class for such a structure. HachMap2D class is useful for computing different paths in an interconnect tree.

D.2 Graphical Interface Classes

The graphical interface classes are developed using the Abstract Window Toolkit (AWT) which is a part of the freely distributed Java Development Kit (JDK) version 1.2 [44, 50]. AWT is defined in the package *java.awt*, and it supports everything from creating menus, dialog boxes, and buttons in a Graphical User Interface (GUI) application.

[1]HashMap is a data-structure for storing values with associated keys.

D.2.1 MAJICframe.java

The MAJICframe class extends or inherits from the AWT Frame class [44]. MAJICframe is a derived container class, and it is responsible for creating the window display for MAJIC. The MAJICapp class creates the instances of MAJICframe. When MAJIC is run with a 3D circuit, multiple instances of MAJICframe are created to support multiple layouts for the different wafers in a 3D stack. MAJICframe adds different menus, such as "File", "View", "ERNI" or "ERNI-3D", and "About", to its window with proper event handlers. It has methods for invoking the parsers to read layout and technology files. MAJICframe also creates an instance of a graphical component class, MAJICcomponent.

D.2.2 MAJICcomponent.java

MAJICcomponent inherits from the AWT Panel class [44]. It is itself a component class, and it adds another AWT component class ScrollPane to display a layout in a scrollable window. MAJICcomponent has its own Graphics[2] object and a paint method to handle all the drawing tasks for the display.

D.3 File Parser Classes

The file parser classes primarily serve two purposes, parsing the MAGIC technology and layout files. While creating a layout window, MAJICframe instantiates the file parser classes.

D.3.1 TechDB.java

The TechDB class implements a recursive parser for reading MAGIC's technology file. The main constructor for the class is called with a string representation of tech-

[2]The Graphics class is the abstract base class of Java for all graphics contexts that allow an application to draw onto components that are realized on various devices, as well as onto off-screen images.

nology file's Uniform Resource Locator (URL[3]). This allows MAJIC to incorporate a technology file from anywhere in a networked system. An InputStream[4] object is created from the URL for reading the ASCII representation of a technology file. Then all the sections in a technology file (discussed in Appendix A) are parsed using the parser methods, and information on different planes, tiles, and contacts are stored in Hashtable[5] data-structures.

D.3.2 ParseMAGFile.java

The ParseMAGFile class implements the recursive parser for reading a MAGIC layout or a *.mag* file, and works in a similar fashion as TechDB. Given an URL, it opens a *.mag* file, and conditionally calls its parser methods, ParseSection(), ParseRect(), or ParseRLabel(), depending on what ParseLine() interprets from the current line in the file. All parsed information is "cached" internally to reduce the amount of time required for parsing large layouts. Following classes implement the cached representation of layout data.

D.3.3 CachedSection.java

This class implements the object representation of a "section" in a *.mag* file. A section in a MAGIC file is defined to start with a "<< foo >>", where "foo" is the name of a mask layer (e.g. polysilicon, metal1, etc.) or some control sections (e.g error_s, checkpaint, end). A CachedSection object contains all the coordinates of rectangular tiles that follow the section header tag.

D.3.4 CachedStmt.java

The CashedStmt class implements the cached representation of any statement that is in *.mag* file's sections. It is an abstract type that must be subclassed to CachedRect,

[3]An URL is a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory.

[4]A superclass of all classes representing an input stream of bytes in Java.

[5]Hashtable is a Java data-storage class that maps distinct keys to stored values.

CachedRLabel, or CachedUse.

D.3.5 CachedRect.java

This class extends the CachedStmt class. Given the parsed coordinates from the "rect ll lr ul ur" statement in a "section" of a *.mag* file, it defines the rectangular tile representing a mask layer.

D.3.6 CachedRlabel.java

The CachedRlabel class extends CachedStmt, and implements the object representation for an "rlabel" statement in a section. The "rlabel" statement assigns user defined text labels to mask layers in a layout file.

D.3.7 CachedUse.java

The CachedUse class also extends CachedStmt, and implements the object representation for an "use" statement in a *.mag* file. The "use" statement allows users to import instances of another *.mag* file into the layout.

D.3.8 CachedCell.java

Finally, the CachedCell class implements the cached representation of the entire *.mag* file. A CachedCell contains multiple instances of CachedSection, CachedStmt, CachedRect, CachedRlabel, and CachedUse to fully define a layout. In a hierarchical layout, where a *.mag* file imports other *.mag* files, the CachedCell object for the parent layout has pointers to other CachedCell objects for the sub-layouts.

D.4 Corner-stitched Data-structure Classes

These classes define the corner-stitched data-structure for the internal representation of a layout file. The corner-stitched data-structure is described in section 4.2.

D.4.1 Tile.java

The Tile class implements the basic representation of a corner-stitched "tile" object in a layout. A "tile" is a rectangle corresponding to either some part of a mask layer or, in case of a "space tile", absence of it. The Tile class contains "pointers" to eight tiles adjacent to the corner-edges. The neighboring tiles are stored in an array to facilitate faster search algorithms.

D.4.2 TileType.java

The TileType class implements different categories of tiles in a particular layout. The different categories are defined according to the input from "types" and "contact" sections in a technology file. For every instance of a Tile class, there is a TileType object to represent its mask layer, such as metal1, poly, and metal2.

D.4.3 Plane.java

The Plane class inherits from Tile, and represents a collection of Tile objects, including "space", that exists in any particular plane of a layout. Initially, a plane is a large "space tile" representing the absence of mask layers. As a layout is parsed, solid tiles are added to this large space plane at the proper positions.

D.4.4 PlaneType.java

Similar to the TileType class, this class implements different categories of planes in a particular layout. The planes are defined according to the input from the "planes" section in a technology file.

D.5 Layout and Tree Representation Classes

The layout and tree representation classes implement the internal representation of a parsed *.mag* file with the corner-stitched data-structures. These classes also store the interconnect trees for future analyses.

D.5.1 Layout.java

The Layout class defines a layout as a stack of multiple Plane objects where an individual Plane object has different types of Tiles to represent the mask layers. The constructor for this class takes CachedCell and TechDB objects as its input parameters for creating such a representation.

D.5.2 ITree.java

This class represents an interconnect routing tree built from corner-stitched layout representation. An interconnect routing tree is a collection of adjacent tiles that would form a continuous electrical path.

D.5.3 ISegment.java

The ISegment class implements a Tile that is a part of an ITree. The ISegment class allows differentiation of Tile objects based on whether they form an interconnect tree or not. An ISegment object is specially tagged if it is a contact tile in a tree.

D.5.4 ISurface.java

This class implements the surface area between adjacent ISegments in an ITree. The ISurface class can be used for generating tree input data for MIT EMSIM [22].

D.5.5 Path.java

The Path class inherits a Vector[6] class and defines a specific path along an ITree object. It consists of a sequence of ISurfaces between starting and ending ISegments. An ITree object can have multiple paths as more than one ending ISegments can exist in an interconnect tree.

[6]The Vector class in Java implements a growable array of objects. Like an array, it contains components that can be accessed using an integer index.

D.6 Reliability Computation Classes

The reliability computation classes are responsible for defining the methods and associated algorithms for reliability analysis. Only these classes need to be extended to add more functionalities for reliability computation.

D.6.1 ERNI.java

The ERNI class implements the methods that operate on ITree objects and apply different filtering algorithms to isolate mortal interconnect trees for further electromigration analysis. Different methods, such as *applyJmaxLfilter()*, *applyRandomBC()*, and *applyExtraction()*, are called when the main application invokes reliability analyses from the "ERNI" menu. However, all the computation are handled in the method, *dispatch(int ruleINDEX, MAJICframe frm)*, where different types of analyses are indexed with "ruleINDEX" using a "switch" statement[7].

D.6.2 ERNIw3D.java

The class ERNIw3D is the extended version of ERNI with the capability of treating multiple ITree objects as a single reliability unit when they form a 3D interconnect tree. In addition, the *dispatch(int ruleINDEX, MAJICframe frm)* method has new indices for identifying 3D trees and calculating a 3D-chip reliability model.

D.7 Circuit Simulator Classes

The circuit simulator classes in MAJIC are directly used from JSim, a Java based switch-level circuit simulator developed by Christopher J. Terman at MIT [51]. Both ERNI and ERNIw3D classes use the JSim classes for extracting circuit netlist from an interconnect tree. First, an instance of the SpiceNetwork class is created with

[7]A switch statement in programming languages (C, C++, and Java) has an integer expression and a body that contains various numbered entry points. A switch statement is a replacement for multiple if else statements.

an ITree object. SpiceNetwork invokes several other classes in JSim for extracting resistance and capacitance from given interconnect geometries. More information on the circuit simulator classes and a comprehensive guide on using JSim can be found in reference [51].

Bibliography

[1] Andy Fan, Arifur Rahman, and Rafael Reif. Copper Wafer Bonding. In *Electrochemical and Solid State Letters*, volume 2, pages 534–536, 1999.

[2] S. J. Souri and K. C. Saraswat. Interconnect Performance Modeling for 3D Integrated Circuits with Multiple Si Layers. In *Proceedings of IITC*, pages 24–26, 1999.

[3] Gerold W. Neudeck. Three-Dimensional CMOS Integration. *IEEE Circuits and Devices Magazine*, 6:32–38, 1990.

[4] K. F. Lee, J. F. Gibbons, K. C. Saraswat, and T. I. Kamins. Thin Film MOSFET Fabricated in Laser-Annealed Polycrystalline Silicon. *Journal of Applied Physics Letters*, 35:173–175, 1979.

[5] A. Rahman, A. Fan, J. Chung, and R. Reif. Wire-length Distribution of Three-Dimensional Integrated Circuits. In *Proceedings of IITC*, pages 233–235, 1999.

[6] Anantha Chandrakasan. The Collaborative Node. Interconnect Focus Center Program Review, August 15 2000.

[7] Stefan P. Hau-Riege. *New Methodologies for Interconnect Reliability Assessments of Integrated Circuits.* PhD Dissertation, Massachusetts Institute of Technology, Department of Materials Science and Engineering, April 2000.

[8] 1999 SIA Roadmap. http://public.itrs.net/files/1999_SIA_Roadmap/Home.htm. Assembly and Packaging.

[9] Comparison of Power Specification from the Crusoe and conventional microprocessor. http://www.transmeta.com/crusoe/lowpower.

[10] J. R. Black. Mass transport of Aluminum by Momentum Exchange with Conducting Electrons. In *Proceedings of 6th Annual International Reliability Physics Symposium*, page 148, 1967.

[11] Seri Lee. Optimum Design and Selection of Heat Sink. *IEEE Transactions on CPMT*, pages 812–817, 1995.

[12] A. P. Chandrakasan and R. W. Brodersen. Minimizing Power Consumption in Digital CMOS Circuits. In *Proceedings of the IEEE*, volume 83, pages 498–523, 1995.

[13] Danny Seth, Syed M. Alam, and Paul R. Herz. Low Power UART Design for Serial Data Communication. In *IEEE International Conference on Electrical and Computer Engineering*, January 2001.

[14] Arifur Rahman. *System-Level Performance Evaluation of Three-Dimensional Integrated Circuits.* PhD Dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, January 2001.

[15] R. Tadepalli, R. Krishnan, R. Reif, M. Spearing, and C. Thompson. Characterization of Wafer Bonding in 3D Integrated Circuits. *MTL Annual Report*, 2000.

[16] J. F. McDonald, R. P. Kraft, J. Q. Lu, A. Kumar, T. Cale, T. M. Lu, P. Belemjian, O. Ergodan, and Y. Kwon. Face to Face Wafer Bonding for 3D Chip Stack Fabrication to Shorten Wire Lengths. In *17th International VMIC Conference*, pages 90–95, 2000.

[17] C. K. Hu. Electromigration Failure Mechanisms in Bamboo-grained Al(Cu) Interconnections. *Thin Solid Films*, 260:124–134, 1995.

[18] J. J. Clement, S. P. Riege, R. Cvijetic, and C. V. Thompson. Methodology for Electromigration Critical Threshold Design Rule Checking. *IEEE Transactions on CAD*, 18:576, 1999.

[19] S. P. Riege, C. V. Thompson, and J. J. Clement. A Hierarchical Reliability Analysis for Circuit Design Evaluation. *IEEE Transactions on ED*, 45:2254, 1998.

[20] Yonald Chery, Stefan Hau-Riege, Syed M. Alam, Donald E. Troxel, and Carl V. Thompson. ERNI: A Tool For Technology-Generic Circuit-Level Reliability Projections. Interconnect Focus Center Annual Review, December 14-15 1999.

[21] Yonald Chery. *ERNI: A Tool For Technology-Generic Circuit-Level Reliability Projections.* PhD Dissertation in progress, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

[22] MIT/EmSim: Electromigration Simulator. http://nirvana.mit.edu/emsim/.

[23] John K. Ousterhout, Gordon T. Hamachi, Rober N. Mayo, Walter S. Scott, and George S. Taylor. The Magic VLSI Layout System. *IEEE Design and Test*, pages 19–30, 1985.

[24] Syed M. Alam, Donald E. Troxel, and Carl V. Thompson. A Novel Layout Methodology for Three-dimensional Integrated Circuits. In *International Conference on Computer Design (ICCD)*, September 2001. (under review).

[25] John K. Ousterhout et al. Magic Tutorial #1: Getting Started. *University of California at Berkeley*, September 1990.

[26] John K. Ousterhout, Walter S. Scott, et al. Magic Maintainer's Manual #2: The Technology File. *University of California at Berkeley*, September 1990.

[27] John K. Ousterhout. Corner-Stitching: A Data Structuring Technique for VLSI Layout Tools. *UC Berkeley Technical report CSD-83-154*, December 1983.

[28] John K. Ousterhout et al. Magic Tutorial #5: Multiple Windows. *University of California at Berkeley*, September 1990.

[29] John K. Ousterhout et al. Magic Tutorial #8: Circuit Extraction. *University of California at Berkeley*, September 1990.

[30] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design, A Systems Perspective*, chapter 8.2, pages 514–536. Addison-Wesley, second edition, 1994.

[31] Digital Integrated Circuits The IRSIM Corner. http://bwrc.eecs.berkeley.edu/classes/icbook/irsim/. Part of the Design Tool Corner.

[32] Using IRSIM. http://www-mtl.mit.edu/ 6.374/. 6.374: Analysis and Design of Digital Integrated Circuits.

[33] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design, A Systems Perspective*, chapter 6, pages 381–413. Addison-Wesley, second edition, 1994.

[34] Stephen M. Trimberger. *Field-Programmable Gate Array Technology*. Kluwer Academic Publishers, 1994.

[35] Stephen D. Brown, Robert J. Francis, Jonathan Rose, and Zvonko G. Vranesic. *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, 1992.

[36] Andre DeHon. *Reconfigurable Architectures for General-Purpose Computing*. PhD Dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1996.

[37] G. Boriello, C. Ebeling, S. Hauck, and S. Burns. The Triptych FPGA Architecture. *IEEE Transaction on VLSI Systems*, 3(4):491–501, 1991.

[38] S. Hauck, G. Borriello, and C. Ebeling. TRIPTYCH: An FPGA Architecture with Integrated Logic and Routing. *Brown/MIT Conference on Advanced Research in VLSI and Parallel Systems*, March 1992.

[39] M. Leeser, W. M. Meleis, M. M. Vai, and P. Zavracky. Rothko: A Three Dimensional FPGA Architecture, its Fabrication, and Design Tools. *Seventh International Workshop on Field Programmable Logic and Applications (FPL97)*, September 1997.

[40] W. M. Meleis and Others. Architectural Design of a Three Dimensional FPGA. *17th Conference on Advanced Research in VLSI*, September 1997.

[41] M. Alexander, J. Cohoon, J. Colflesh, J. Karro, and G. Robins. Three-dimensional Field-programmable Gate Arrays. *Proceedings of IEEE International ASIC Conference*, pages 253–256, September 1995.

[42] Nisha Checka and Charlotte Lau. Performance Comparison of a 2D and 3D FPGA. Design and Analysis of Digital Integrated Circuits, Professor Anantha Chandrakasan, Fall 2000.

[43] Manuel Perez. *Java Remote Microscope for Collaborative Inspection of Integrated Circuits.* Bachelor of Science and Master of Engineering Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, May 1997.

[44] Java Platform 1.2 API Specification. http://web.mit.edu/java_v1.2ref/distrib-/sun4x_56/docs/api/index.html. This is a web document on Java Platform core API provided by Sun Microsystems.

[45] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*, chapter 23.3, pages 477–484. MIT Press, 1997.

[46] R. Kraayeveld, A. Willemsen A. Verbruggen, and S. Radelaar. *Applied Physics Letter*, 67:1226, 1995.

[47] Dimitri P. Berstsekas and John N. Tsitsiklis. Introduction to Probability. 6.041/6.431 Class Notes, Spring 2001, Massachusetts Institute of Technology.

[48] Anantha Chandrakasan, Rafael Reif, and Shamik Das. 3D Design Automation with Physical Design Tools. MARCO Interconnect Focus Center.

[49] Donald E. Troxel, troxel@mtl.mit.edu and Carl V. Thompson, cthomp@mtl.mit.edu. Massachusetts Institute of Technology.

[50] David M. Geary and Alan L. McClellan. *Graphic JAVA Mastering the AWT.* The SunSoft Press, A Prentice Hall Title, 1997.

[51] Christopher J. Terman. http://6004.lcs.mit.edu/courseware/jsim/. JSim Guide for 6.004, Massachusetts Institute of Technology.

4423-51