ANALYSIS OF POWER SUPPLY NETWORKS IN VLSI CIRCUITS

Don Stark

Technical Report: CSL-TR-91-465

March 1991

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 943054055

Abstract

Although the trend toward finer geometries and larger chips has produced faster systems, it has also created larger voltage drops and higher current densities in chip power supply networks. Excessive voltage drops in the power supply lines cause incorrect circuit operation, and high current densities lead to circuit failure via electromigration. Analyzing this power supply noise by hand for large circuits is difficult and error prone; automatic checking tools are needed to make the analysis easier.

This thesis describes Ariel, a CAD tool that helps VLSI designers analyze power supply noise. The system consists of three main components, a resistance extractor, a current estimator, and a linear solver, that are used together to determine the voltage drops and current density along the supply lines. The resistance extractor includes two parts: a fast extractor that calculates resistances quickly using simple heuristics, and a slower, more accurate finite element extractor. Despite its simplicity, the fast extractor obtained nearly the same results as the finite element one and is two orders of magnitude faster. The system also contains two current estimators, one for CMOS designs and one for ECL. The CMOS current estimator is based on the switch level simulator Rsim, and produces a time-varying current distribution that includes the effects of charge sharing, image currents, and slope on the gate's inputs. The ECL estimator does a static analysis of the design, calculating each gate's tail current and tracing through the network to find where it enters the power supplies. Extensions to the estimator allow it to handle more complex circuits, such as shared current lines and diode decoders. Finally, the linear solver applies this current pattern to the resistance network, and efficiently calculates voltages and current densities by taking advantage of topological characteristics peculiar to power supply networks. It removes trees, simple loops, and series sections for separate analysis. These techniques substantially reduce the time required for solution.

This report also includes the results of running the system on several large designs, and points out flaws that Ariel uncovered in their power networks.

Key Words and Phrases: Power Distribution, Noise, Electromigration, Computer Aided Design, VLSI, resistance extraction, switch level simulation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> When at last this little instrument appeared, consisting, as it does, of parts every one of which is familiar to us, and capable of being put together by an amateur, the disappointment arising from its humble appearance was only partially relieved on finding that it was really able to talk.
>
> James Clerk Maxwell
> ***The Telephone (1878)***

Although Maxwell was describing one of the technological marvels of his time, the telephone, rather than one of our time, the integrated circuit, his observation would not be out of place today. From a systems perspective, the operation of an individual transistor or resistor is quite simple, yet, considered in the aggregate, the operation of the entire circuit is quite remarkable. As the number of devices in a design increases, however, the relative simplicity of the individual devices is belied by the complexity of their collective behavior. Insuring that a million transistors are correctly arranged and interconnected is a nontrivial task. Designers have developed an array of tools to handle this increasing complexity, including simulators to check that the circuit implements the logic function desired, design rule checkers to verify that components are arranged in permissible topologies, and circuit extractors to see that the devices on chip are interconnected as the designer intended.

Even a design that has been fully analyzed at all these levels, however, may not work correctly when fabricated. It must also satisfy electrical constraints, for which fewer

analysis tools exist. A set of these constraints surround the design's power distribution system. To distribute power to all the devices on chip, each design includes a network of wires; if this network is not designed properly, the system will not operate as desired. Excessive voltage drops along this network will slow down the circuit, and, if high enough, even cause it to switch incorrectly. High current density in these power connections can also cause circuit failure via electromigration. Metal interconnect in VLSI circuits is designed to withstand an average current density of about $1mA/\mu m^2$ and a peak current density of approximately $10mA/\mu m^2$ [19]. At current densities above these values, the electron wind will rearrange the metal ions, causing the metal to thin in some places and accumulate in others. Eventually, the chip will fail due to either an open or short circuit. As systems are scaled, these voltage and current problems are exacerbated.

The analysis tool described in this thesis, ***Ariel***[1], is designed to fill this gap. Given a design in either a CMOS or a silicon ECL technology, Ariel will extract a a set of resistors to represent the power network, analyze the circuit to calculate when and where currents enter this resistance network, and solve the resulting system of equations. This information allows to the designer to see if the power network has sufficient capacity for the circuits it must supply.

1.1 System Overview

Power supply analysis is conducted in several stages, as shown in Figure 1. The first two steps, layout and extraction, are performed using the Magic layout editor[44]. These produce a mask level description of the design and its corresponding netlist, including parasitic capacitances. The remaining steps (inside the dotted box) are performed by Ariel, and are the scope of this thesis. These steps fall into three categories, corresponding to the three parts of Ohm's law: resistance extraction, current estimation, and voltage calculation.

In the next chapter, I describe techniques for efficiently extracting resistances, paying special attention to problems inherent in power supply networks. Included in the system

[1]The name has two meanings. Ariel is the spirit who carries out commands for the magician Prospero in *The Tempest*. It is also an acronym of "Analyzer for Resistance and current (I) ELements".

are two extractors: a fast, simple one that assumes uniform current flow, and a slower, more accurate one based on finite elements.

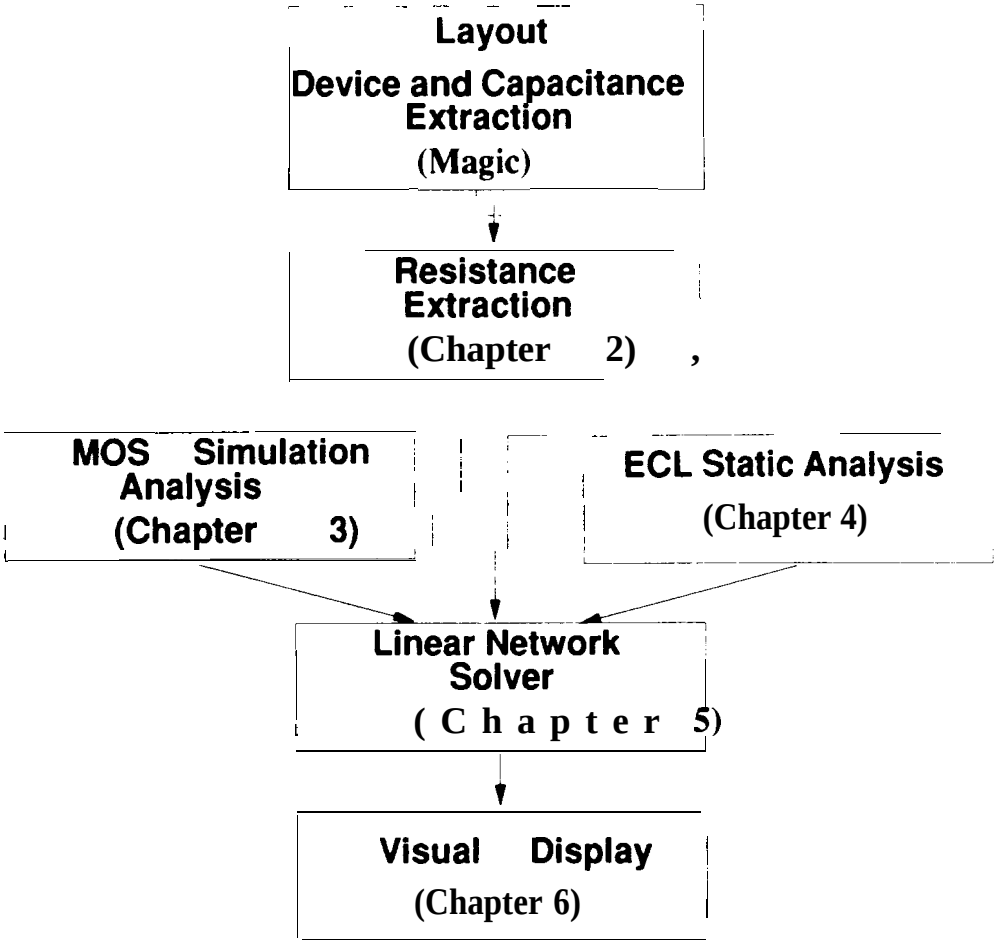


Figure 1: System Overview

Chapter Three investigates current estimation for CMOS circuits. Currents in CMOS are dynamic and pattern dependent; an accurate current estimator must take both these factors into account. After discussing algorithms adopted by other researchers, I describe my approach, which is based on timing simulation.

Chapter Four describes current estimation for ECL designs. Here, the magnitude of current in the design is relatively constant; only its distribution varies. I describe techniques for calculating the current magnitude, tracing currents through the circuit, and

arranging these in a conservative distribution.

Chapter Five examines techniques for solving the resulting large system of equations. I investigate some configurations peculiar to power networks, and develop techniques for partitioning the network into smaller, more easily solved sections based on these configurations. Methods for efficiently solving the remaining portions of a network are also investigated.

In Chapter Six, I analyze several fairly large designs using the system. This analysis uncovered several mistakes made by the chips' designers, which are visible on plots of the voltage and current density distributions. Possible improvements in the voltage and current distributions for the designs are also discussed.

1.2 Test Circuits

Throughout the thesis, several chips are used as test cases for the system. Analyzing large designs helps insure that the algorithms developed are practical for use on real systems. There are six chips in the test set: three CMOS designs from Stanford and Digital Equipment Corporation, and three ECL designs from MIPS Computer Systems. Table 1 summarizes their sizes, speeds, and technologies. The features of these systems are:

1. MIPS-X[24] is a 32-bit RISC microprocessor designed at the Computer Systems Lab of Stanford University by Mark Horowitz and a team of students. It is designed in a 2μm, two-level-metal, n-well CMOS technology, and runs at a clock speed of 20MHz.[2]

2. μTitan is also a 32-bit RISC Microprocessor, designed at the Digital Equipment Corporation Western Research Laboratory by Norman Jouppi[28]. It is designed in a 1.5μm, two-level-metal process, and runs at a speed of 25Mhz.

3. SPIM is a 64 by 64 iterating array multiplier designed by Mark Santoro of Stanford University[51]. It is designed in a 1.6μm two-level-metal, CMOS technology, and

[2] The on-chip instruction cache was not included in any of the analyses; the device count listed in the table also excludes the cache.

runs at 85MHz.

4. The R6000, R6010, and R6020 form an ECL chipset designed by David Roberts, Tim Layman, and George Taylor at MIPS Computer Systems[48]. All are designed in Bipolar Integrated Technology's 2μm, triple diffused ECL, three-level-metal process, and run at 66.7MHz. The R6000 is a 32-bit microprocessor with on-chip TLB, the R6010 a 64-bit floating point controller, and the R6020 a system bus controller chip.

Circuit	Devices	Speed	Technology
MIPS-X	47130	20MHz	2.0uM CMOS
uTitan	179390	25MHz	1.5uM CMOS
SPIM	41804	85MHz	1.6uM CMOS
R6000 CPU	149619	66.7MHz	2.0uM ECL
R6010 FPC	148745	66.7MHz	2.0uM ECL
R6020 SBC	163925	66.7MHz	2.0uM ECL

Table 1: Summary of Test Circuits

Chapter 2

Resistance Extraction

For out of olde feldes, as men seyth
Cometh al this newe corn fro yer to yet-e,
And out of olde bokes, in good feyth,
Cometh al this newe science that men lere.

Geoffrey Chaucer
The Parliament of Fowls

Calculating the voltage and current distributions for a power network requires a conductance matrix **G** relating the voltage and current distributions through Ohm's law, $G\vec{v} = \vec{i}$. The resistance extractor's job is to produce this matrix from a mask level description of the design.

As will be seen in the descriptions of previous work contained in subsequent sections, resistance extraction is a fairly mature field. What "newe science" will yet another implementation yield? The first and most pragmatic reason for writing my own extractor was that one was not available at the outset of the project. Had such an extractor been available, however, it probably would not have satisfactorily processed power buses; most extractors are designed to operate on signal lines, which are topologically quite different. Power buses are much larger and have greater variations in width; extractors geared for regions of modest size and fairly uniform features have problems in this new environment. A second goal was to determine what modifications are necessary to allow existing extraction algorithms to operate in this new domain. Finally, writing an extractor

presented an opportunity to see how Magic's tiled, and comer-stitched database could be advantageously used to implement these algorithms.

The next six sections review what resistance extraction entails, discuss the algorithms commonly used, and describe the two implementations used in Ariel. The first section gives a brief review of the underlying field theory. Following this is a description of the one-dimensional approximation to Laplace's equation that underlies the fastest algorithms, and a description of the implementation of this algorithm. Next are descriptions of two slower but more accurate approaches, finite differences and finite elements, and a description of Ariel's finite element implementation. Finally, there is a comparison of the two extractors, which shows that the simple polygon method is nearly as accurate and two orders of magnitude faster than using finite elements.

2.1 Underlying Field Theory

From Ohm's law, the current flowing into a surface S can be calculated as the surface integral of the normal component of the electric field:

$$J = \sigma \vec{E} \tag{1}$$

$$I = \int_s \vec{J} \cdot \vec{n} ds = \int_s \sigma \vec{E} \cdot \vec{n} ds \tag{2}$$

Combining Gauss's law for charge free regions, $\nabla \cdot \sigma \vec{E} = 0$, with the definition for the scalar potential, $\vec{E} = -\nabla V$, gives a partial differential equation for the potential:

$$\nabla \cdot \sigma \nabla V = 0 \tag{3}$$

For regions of constant conductivity, this reduces to Laplace's Equation.

$$\nabla^2 V = 0 \tag{4}$$

To calculate the resistance of a region, the extractor must find a solution to Laplace's equation that satisfies the region's boundary conditions. Resistive regions in integrated circuits are generally modelled as planar regions of constant conductivity bounded by

conducting and insulating edges (Figure 2a). Since the region is flat, the extractor will assume that the potential is a function of two dimensions only; unless explicitly noted otherwise, this approximation will be used throughout the chapter. The edges represent the two possible types of boundary conditions. On a conducting boundary, the potential is constant along the entire edge; such an edge satisfies ***an essential*** or ***Dirichlet*** boundary condition. These edges represent sources or sinks of current in the region. On an insulating boundary, the current normal to the edge is zero; these edges satisfy a ***normal*** or ***Neumann*** boundary condition. In this model, they represent the edge between conducting and nonconducting materials.

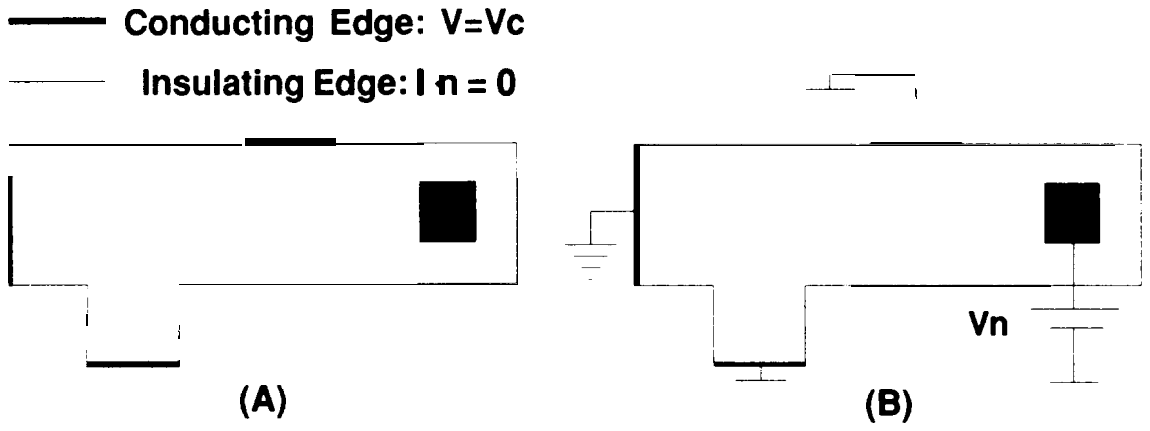


Figure 2: Resistive Region Model

To calculate the resistance for a region, a test voltage V_n is applied to one of the conducting edges, and all the other edges are grounded. The extractor finds an approximate solution to Laplace's equation subject to these boundary conditions, then finds the current entering each of the grounded terminals using Equation 2. The resistance between each grounded terminal t and the excited one is V_n/I_t. For regions with more than two conducting edges, the extractor repeats this operation with V_n applied to different edges until the resistance between each pair of terminals has been calculated.

The following three sections contain different approaches to the solution of this general problem.

2.2 One Dimensional Current Flow

The fastest resistance extraction algorithms rely on the observation that the current flow in interconnect is usually one-dimensional. The field lines in Figure 3 demonstrate this property; near disturbances such as corners, junctions, and contacts, the current density distribution is complex, but in the long regions that form most of the pattern, it is uniform. In these straight sections, the Y and Z partial derivatives are 0, and Laplace's equation is reduced to a l-dimensional case. For the region depicted in Figure 4, the y and z components of $\vec{E}$ are 0, as are those of the current density:

$$\vec{J} = \sigma E_x = -\sigma \frac{dV}{dx} \tag{5}$$

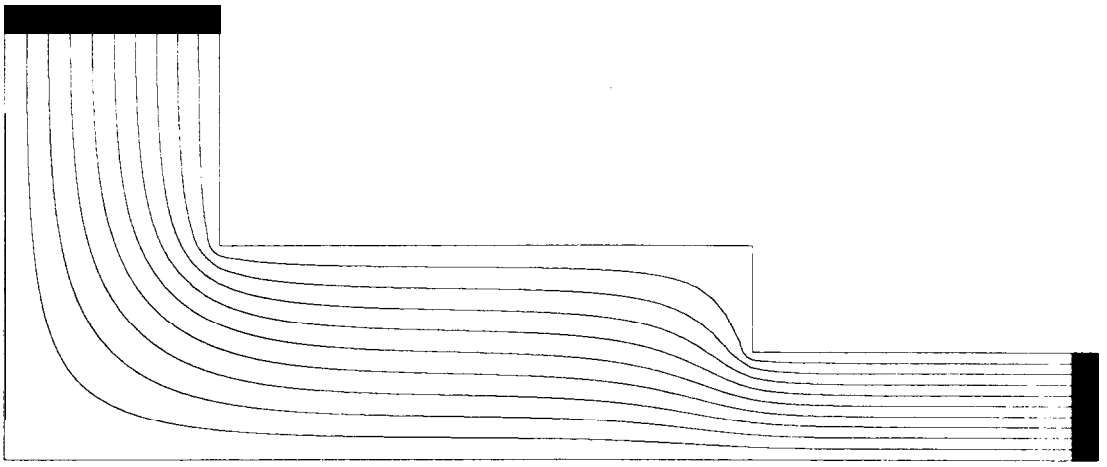


Figure 3: Current Distribution Near Disturbances

If we integrate this equation from vi to 0 and from 0 to x_1, we can derive a relation between the current density and the voltage:

$$\int_{v_1}^{0} \sigma dV = -\int_{0}^{x_1} J_x dx \tag{6}$$

$$\text{vl} = \int_{0}^{x_1} \frac{1}{\sigma} J_x dx \tag{7}$$

The resistance can then be calculated by dividing the voltage by the current in the region. Because the electric field only has an x component, the surface integral of

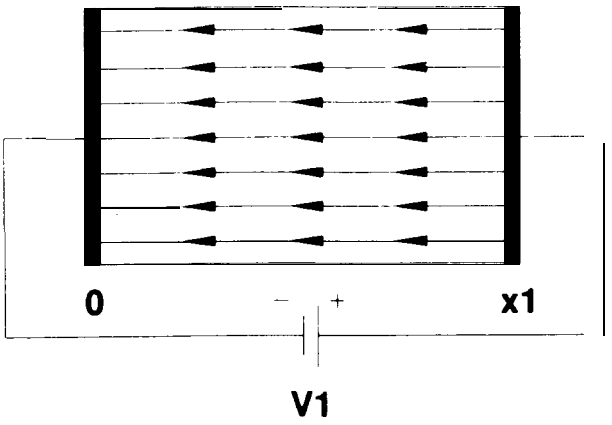


Figure 4: Uniform Current Region

Equation 2 is equal to this x component times the cross-sectional area, $y_1 z_1$. Since current is conserved, J_x is a constant:

$$R = \frac{v}{i} = \frac{\int_0^{x_1} \frac{1}{\sigma} J_x dx}{\int_S J_x dS} = \frac{x_1}{\sigma y_1 z_1} \tag{8}$$

If a sheet resistance $R_{sh} = 1/(\sigma z_1)$ is defined, then Equation 8 becomes the familiar expression $R = R_{sh} L/W$, where R_{sh} has units of $\Omega/\square$. Programs based on this approximation break a region into constituent rectangles, each of which contains connecting nodes formed by transistors, contacts and adjoining rectangles. The extractor then determines the dominant direction of current flow, sorts the nodes in this direction, and adds resistors between adjacent nodes in the list. Each resistor has a value $R = R_{sh}(y2 - y1)/W$, where $(y2 - y1)$ is the distance between the points in the direction of current flow and W is the width of the rectangle.

This algorithm is extremely fast because it has reduced the system of linear equations produced by most methods to a single equation. Since matrix decomposition is not required, its running time is linear in the number of regions. It also tends to produce more manageable networks; connections are generally only made between nodes in a given rectangle instead of between each pair of nodes in the entire system. Its accuracy depends strongly on the topology of the net being extracted; if the net is dominated by long, straight sections, as many integrated circuit wires are, the values it produces should

be fairly accurate. If the net is highly irregular, or has an aspect ratio near unity (such as the well resistance in CMOS), the approximation will be fairly poor.

Most extractors designed for use on large designs are implementations of this algorithm. The simplest [37, 47, 53, 55, 61] simply calculate the resistance for each rectangle and assume that the overall result will be fairly accurate because these resistances are dominant. More sophisticated methods [3, 25] have empirically developed correction factors to compensate for comers, contacts, and other sources of field disturbance. The most sophisticated program using this method is McCormick's EXCL[35], which only assumes one-dimensional flow in regions where it is certain to be valid. The values for the remaining sections of a net are either looked up in a library or solved using finite differences.

Both extractors described later in this chapter use the one-dimensional approximation. The fast extractor described in the next section uses it exclusively, while the finite element extractor (Section 2.6) uses it selectively for sections where it is an accurate approximation.

2.3 Polygonal Decomposition Implementation

This section describes how the fast extractor, which is based on the one-dimensional approximation of the last section, is implemented. The resistance extractor is a component of the layout editor Magic. The next subsection provides a brief overview of Magic's underlying database, including discussion of the opportunities that the database affords and some of the challenges that it presents to resistance extraction. Following this is a description of the modifications the extractor makes to the layout representation and a description of the algorithm's core.

2.3.1 An Overview of Magic's Database

Magic is a layout editor for integrated circuits developed at the University of California at Berkeley by John Ousterhout, Gordon Hamachi, Bob Mayo, Walter Scott, and George Taylor. It introduced many new features, including continuous background design

rule checking, hierarchical circuit extraction, and plowing. A more detailed description of these and other features can be found in the 1984 Design Automation Conference Proceedings[44]. This section will concentrate on another Magic innovation: its novel database design.

The basic Magic data structure is the *tile. The* entire design area (Figure 5), extending to infinity, is covered by a mosaic of these tiles. Each point in the plane is covered by exactly one tile. Tiles may represent part of the design, as do the shaded ones in the example, or they may represent space. There are many possible configurations of rectangles that could be used to cover a region; Magic represents areas composed of a single material as a set of horizontal strips. In the example, the shaded region is broken into four tiles, *t2*, *t4*, *t7*, and *t9*. This representation prevents fragmentation of the region into many small rectangles and provides a canonical form for the design.

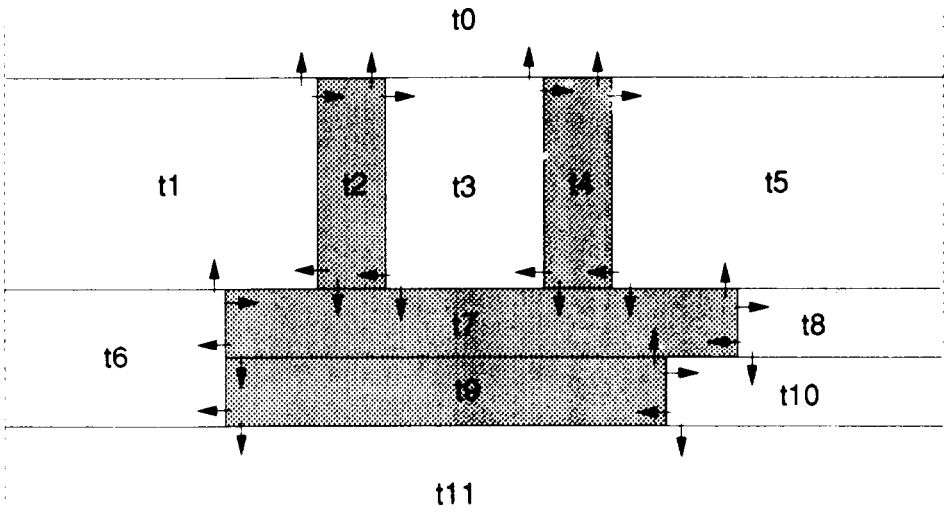


Figure 5: A Plane of Magic Tiles

Corner stitches are used to represent the interrelation between tiles. A stitch is a pointer to another tile in the plane. Each tile has four stitches; one pointing to the rightmost tile along its top edge, one to the top tile along the right edge, one to the

bottom tile along the left edge, and one to the leftmost tile along the bottom edge. Each tile's corner stitches and the neighbors to which they point are shown as arrows in the example. These four stitches make local searching very fast. For example, all the neighbors of a given tile can be found by following stitches; in the figure, the top neighbors of tile *t7* can be found by first following its right top pointer, then by following the bottom left pointers of the neighbors until the right edge of a neighbor is less than the left edge of the original tile. Similar algorithms exist for locating a point on the plane, searching for tiles in a given area, and visiting each tile in a region[45].

The remaining problem is developing a correspondence between the tile types of the database and the physical mask layers of a fabrication process. This problem is technology dependent; the designer must set up this correspondence for each fabrication technology used. One approach would be to use a separate tile type for each possible combination of overlapping layers, but this mapping would require an exponential number of types and would fragment the database into many small pieces. Another possible solution would be to use a separate tile plane for each mask layer. This solution requires fewer types (only one per mask layer), but is still memory inefficient because there are many more space tiles. This arrangement also lessens the advantages of comer stitching. Many layout operations involve more than one layer; calculating the interactions between such tiles is more difficult when they do not lie in the same plane.

Most technology mappings adopt an approach somewhere between the two described above. All layers that commonly interact with one another are placed in the same plane, while layers that do not are placed in separate ones. For example, the standard MOSIS SCMOS technology uses five planes: *well, active, metal1*, *metal2*, and *oxide.* As can be. guessed from their names, the well, metal1, and metal2 planes contain types representing the wells and the two layers of metal used in the design. The oxide plane contains the locations of cuts in the chip's passivation layer. The active plane contains the diffusion and polysilicon masks, plus combinations of these layers, such as transistors. These layers are put in the same plane because they closely interact. Interactions between layers on different planes is much rarer; for example, few operations need to know the relative spacings of polysilicon and metal.

The remaining problem is representing mask layers that interact with types on more

than one of the above planes, such as contacts. This is done using special abstract tiles that have separate copies on both planes. For example, in Figure 6, the polysilicon, contact cut, and metal1 masks are combined to form the single abstract type *pc*. Duplicate copies of the *pc* tiles are kept on both the active and metal 1 planes. These **abstract** types facilitate analysis of the layout because mask interactions need not be calculated explicitly; they are implied by the composite type.

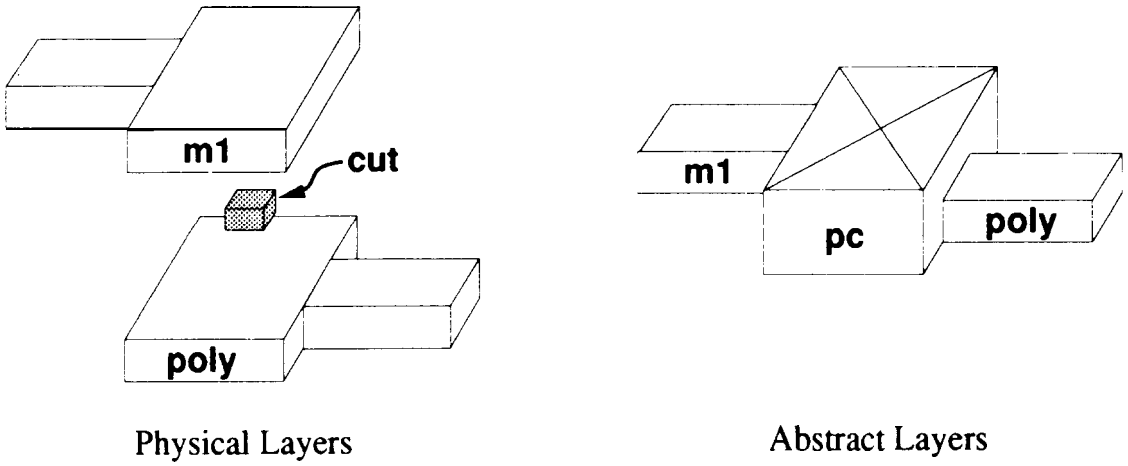


Figure 6: Abstract Types

A cell is thus represented as a set of planes, each composed of tiles of varying types. Each cell can also contain subcells; interactions between these subcells present special problems. Magic allows nearly arbitrary overlap between cells; the only limitations are that cells must be individually design-rule correct and that overlap must not create or destroy devices. Any tools developed for the system must operate correctly regardless of the cell topology. Magic's regular circuit extractor [52] is both hierarchical and incremental. To handle overlap, it extracts each cell individually, then flattens areas of cell interaction, extracts them and adjusts the connectivity and capacitance information accordingly. Because each cell is extracted independently of its context, only a cell and all its ancestors must be re-extracted when it is modified.

Extracting parasitic resistances in the same manner would be extremely difficult. Because of nearly arbitrary overlap, any point in a polygon may be used as a terminal.

In the example of Figure 7a, the extraction of net1 in cellA produces a single resistor connecting the two transistors. When a second cell, cellB, is added over cellA, its metal line is shorted to the middle of netl. The initial version of net1 has no node at this point; the network for cellA would have to be split to create the required connection point. Allowing arbitrary overlap thus precludes context free extraction of cells.

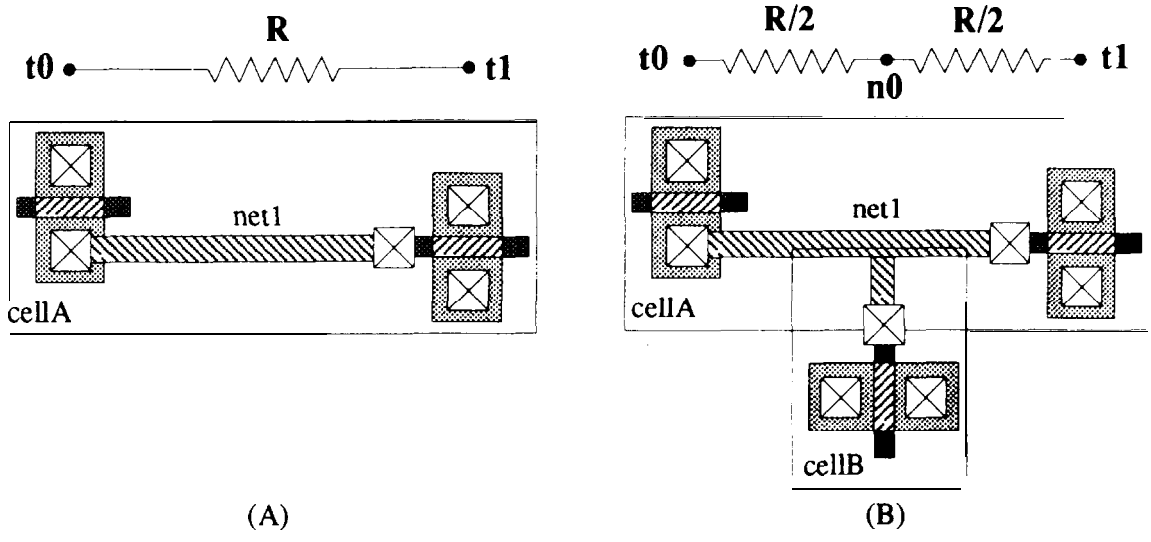


Figure 7: Cell Overlap

It might be possible to devise a modified hierarchical system that allows network modification and back annotation to handle cases where cell overlap changes a network's topology, but doing so would eliminate much of the advantage yielded by hierarchical extraction and would make the extractor much more complex. Instead, the resistance extractor copies and flattens all the electrically connected rectangles into a dummy cell. Despite the overhead of this approach, the extractor is still fast enough to run on an entire design, as will be seen in Section 2.7.

2.3.2 Database Preprocessing

Once a hierarchical net has been flattened into a single dummy cell, the extractor modifies Magic's standard layer representation into one more conducive to resistance extraction. This is done in two steps: dissolution of contacts and coalescence of regions.

Contact Removal

Magic's contact types present problems for the extractor. Resistance extraction operates primarily on regions composed of a single mask layer. Abstract **contact** types tend to fracture this single layer into multiple pieces, as shown in Figure 8a. This artificial fragmentation makes extraction more difficult because it hides a region's true topology. To avoid this, the extractor notes the position of each contact and then replaces it with its constituent mask layers (Figure 8b). This reduces the number of tiles and makes the inherent structure of the region more explicit.

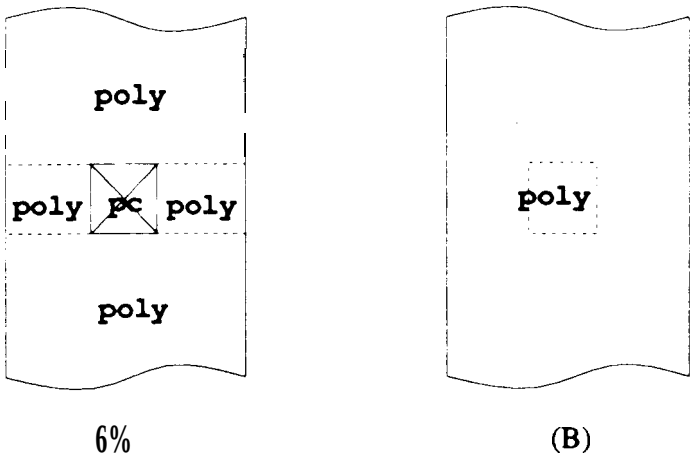


Figure 8: Dissolving Contacts

Region Coalescence

Another source of artificial region fragmentation is Magic's use of of maximum horizontal strips (Figure 9a). This representation splits long horizontal regions between several rectangles. Since the one dimensional approximation is most accurate when the conducting **edges** of the region are perpendicular to the current flow, these areas need to be reshaped.

A modified version of Horowitz's fracturing algorithm [25] is used to fix them. At each concave comer, the extractor checks to see if the width of the region measured from

the comer is greater than its height. If it is (Figure 9b), then each tile in the region is split vertically at the comer. Once the tiles have been split, the extractor checks to see if they can be combined with their vertical neighbors (Figure 9c). Comers whose region height is greater than their width are likewise split and merged horizontally. Performing this operation at all concave comers produces the region of merged rectangles shown in Figure 9d.

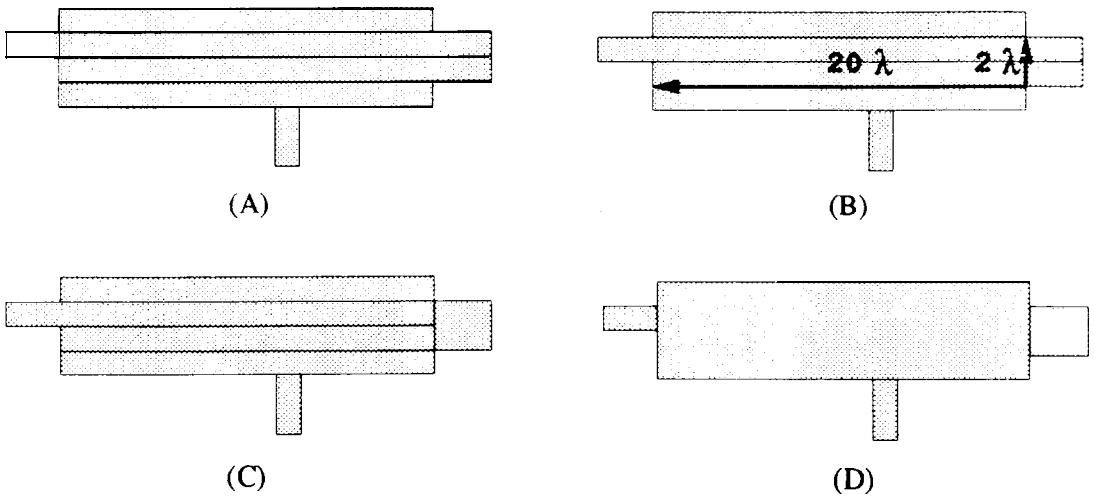


Figure 9: Modifying Horizontal Strips

Configurations where two concave comers interact, like those in Example 10, must be handled carefully. This configuration is not uncommon in power networks; a designer will sometimes nick a comer out of the power bus to avoid a spacing design rule violation. Once the region has been fractured at both comers, there are two pairs of rectangles that share a common edge; the extractor must decide which pair to merge. In the example, removing the horizontal edge (marked *bad)* leaves two tiles with current flow parallel to their common border, while removing the vertical edge (marked good) leaves two tiles with current flow perpendicular to their common edge. The general rule is that the longer of the two edges is removed; in the example, this is the vertical edge.

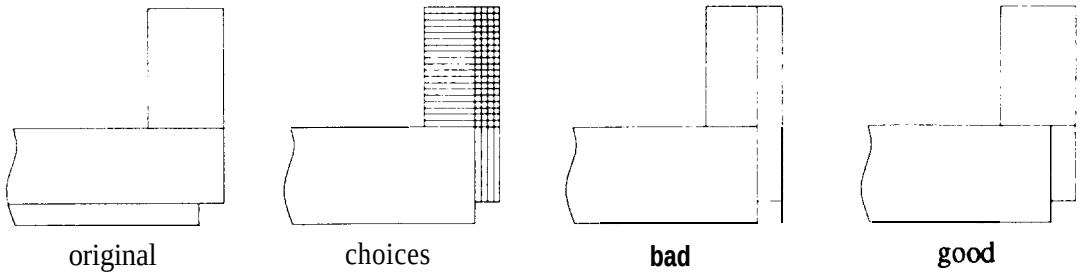


Figure 10: Interacting Concave Comers

2.3.3 Resistance Calculation

Once the database has been preprocessed, a resistance network is formed tile by tile. The user specifies a set of initial tile(s) that form the root of the power distribution tree, generally the power pads, which are put into a pending tile list. Tiles are processed one by one until none remain in the list. Each tile is processed in eight steps:

1. Check to see if this tile forms the gate or emitter of a transistor. If it does, add a node at the center of the tile, and set the corresponding device terminal equal to it.

2. Walk along the tile's edges looking for electrically connected materials. For each connecting tile found, add a node at the center of the junction between tile edges. If the other tile has not already been processed, add it to the pending list.

3. If this tile type can form the source/drain or base terminal of a device, search the tile edges for transistor tiles. For each one found, add a node at the center of the common edge and set the correct terminal equal to it.

4. If this tile type forms the collector of a bipolar device, search under the tile on the emitter's home plane for transistors. For each one found, add a node in the center of the emitter tile and set the collector terminal equal to it.

5. Check to see if the tile originally contained any contacts. If so, add a node for each one. If the other tiles that formed the contact have not been processed, add them to the pending list.

6. Calculate the minimum and maximum X and Y coordinates of all the nodes found in the previous steps. If max(X) − min(X) > max(Y) − min(Y), assume that current flows horizontally. If not, assume current flows vertically.

7. Sort the nodes from minimum to maximum in the direction of current flow. Merge nodes with the same coordinate.

8. Add a resistor between each adjacent pair of rectangles.

$$R = \begin{cases} R_{sh}\,\Delta Y/\text{width(tile)} & \text{if current flow is vertical} \\ R_{sh}\,\Delta X/\text{height(tile)} & \text{if current flow is horizontal} \end{cases}$$

A simple example is shown in Figure 11. The extractor creates node N_0 when it finds the adjoining tile during the perimeter walk of Step 2. Nodes N_1 and N_2 are created during Step 3 because the tile forms the source terminal of two transistors. Node N_3 is created during Step 5 for the contact contained within the rectangle. Since the greatest horizontal separation (between nodes N_1 and N_2) is greater than the maximum vertical separation (between nodes N_0 and N_3), the extractor assumes that current flows horizontally. The nodes are sorted by x coordinate; since Nodes N_0 and N_3 have the same value, they are merged. Two resistors, R1 and R2, are created between the node pairs, $N_1 - N_0$ and $N_0 - N_2$, and are added to the overall network description. The extractor marks this tile as processed and goes on to the next one in the pending list.

When all the tiles associated with a node have been processed, the resistors and transistors connecting to the node are examined. Resistors with both terminals connected to the node are eliminated. The extractor tries to combine any resistors in parallel connecting to the node. If there are no transistors and only one resistor connected to the node, the node and its connecting resistor are eliminated. Nodes with two connecting resistors and no transistors are also removed, and their resistors are combined.

The extractor continues in this manner until all the tiles are visited and all the nodes have been processed. The resulting network of nodes, resistors, and transistors is then saved in a file for processing by the linear solver.

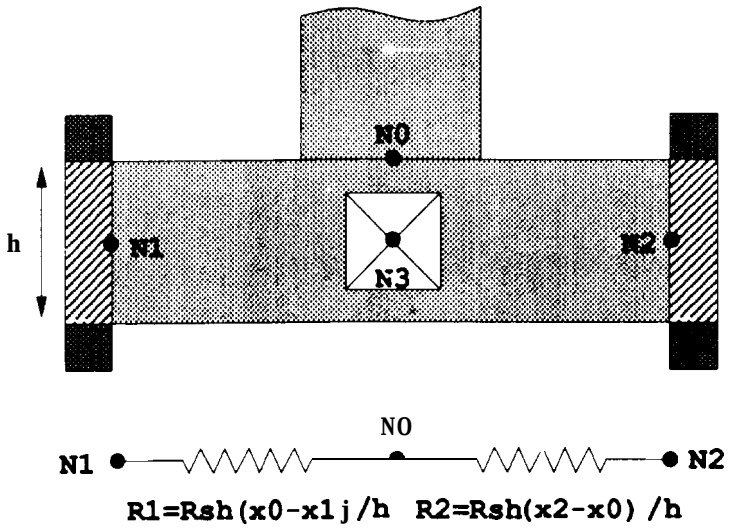


Figure 11: Extraction Example

2.4 Finite Differences

The one dimensional approximation is extremely efficient, since is is basically a scan through the list of tiles. For irregular shapes, however, it can produce a poor estimate of the resistance. Two other approaches, finite differences and finite elements, are often used when greater accuracy is needed. This section describes finite differences, which are simple to implement and adequate for some problems, while the next section describes the more powerful (and complicated) finite element method.

If the Taylor Series for the voltage is expanded around the point (x. y), it can be used to estimate the values at (x + Δx. y) and (x – Δx, y).

$$V(x+\Delta x,y) = V(x,y) + \Delta x\frac{\partial V}{\partial x} + \Delta x^2\frac{\partial^2 V}{\partial x^2} + \Delta x^3\frac{\partial^3 V}{\partial x^3} + \cdots \qquad (9)$$

$$V(x-\Delta x,y) = V(x,y) - \Delta x\frac{\partial V}{\partial x} + \Delta x^2\frac{\partial^2 V}{\partial x^2} - \Delta x^3\frac{\partial^3 V}{\partial x^3} + \cdots \qquad (10)$$

If these two equations are added, all the odd powers of Ax cancel out. By rearranging this sum and neglecting all even terms higher than second order, we get an equation for the second partial derivative with respect to x:

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(x+\Delta x, y) + V(x-\Delta x, y) - 2V(x,y)}{\Delta x^2} \tag{11}$$

An analogous equation for $\partial^2 V/\partial y^2$ can be derived in the same way. When these two equations are added, the result is an approximation for Laplace's equation in two dimensions:

$$\frac{\partial^2 V}{8x2} + \frac{\partial^2 V}{\partial y^2} \approx \frac{V(x+\Delta x, y) + V(x-\Delta x, y) - 2V(x,y)}{\Delta x^2} + \frac{V(x.\,y+\Delta y) + V(x,\,y-\Delta y) - 2V(x,y)}{\Delta y^2} = 0 \tag{12}$$

Rearranging Equation 12 gives an approximation for V(x. y) in terms of its four neighbors, $\mathbf{V}(\mathbf{x} - 4\mathrm{x}, \mathbf{y})$,$\mathbf{V}(\mathbf{x} + 4\mathrm{x}.\ \mathrm{y})$, $\mathbf{V}(\mathbf{x}, \mathrm{y} - \Delta\mathbf{y})$, and $\mathbf{V}(\mathbf{x}, \mathrm{y} + \Delta\mathbf{y})$.

$$V(x,y) \approx \frac{\Delta y^2 (V(x+\Delta x, y) + V(x-\Delta x, y))}{2\Delta x^2 + 2\Delta y^2} + \frac{\Delta x^2 (V(x, y+\Delta y) + V(x,\,y-\Delta y))}{24\mathrm{x}2 + 2\Delta y^2} \tag{13}$$

If the region is covered with a rectilinear mesh, as shown in Figure 12, then the voltage distribution can be calculated by solving the system of linear equations relating the mesh node potentials to one another. In this derivation, the distances between mesh points in a given direction (4x and $\Delta\mathbf{y}$ are constants), but an equivalent expression for a nonuniform node distribution may be derived in the same manner, the only difference is that Equation 10 must be scaled by the ratio of the mesh spacings so that the first derivative terms will still cancel one another[22].

While Equation 13 is clearly true for points in the interior of the mesh, it must be modified for points along the boundary. For points on a conducting edge, the potential is fixed at V_c, the edge's potential. For points along an insulating edge, the boundary condition requires than the normal current be 0. This can be achieved by mirroring the voltage about the edge:

$$V(x,y) = \frac{2\Delta y^2 V(x+\Delta x, y) + \Delta x^2 (V(x, y+\Delta y) + V(x, y-\Delta y))}{2\Delta x^2 + 2\Delta y^2} \quad \text{leftedge} \tag{14}$$

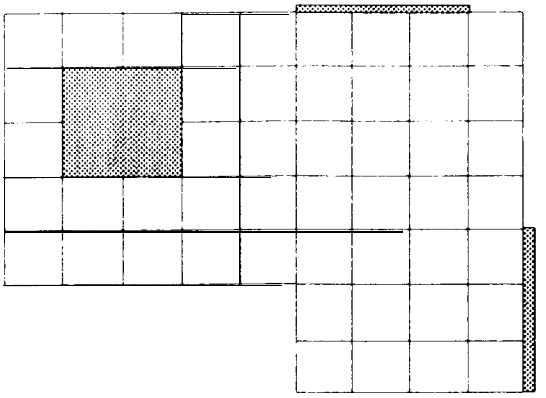


Figure 12: Finite Difference Mesh

$$V(x,y) = \frac{2\Delta_y^2 V(x-\Delta_x,y)+\Delta_x^2(V(x,y+\Delta_y)+V(x,y-\Delta_y))}{2\Delta_x^2+2\Delta_y^2} \quad \text{rightedge} \tag{15}$$

$$V(x\ y) = \frac{\Delta_y^2(V(x+\Delta_x,y)+V(x-\Delta_x,y))+2\Delta_x^2 V(x,y+\Delta_y)}{2\Delta_x^2+2\Delta_y^2} \quad \text{bottomedge} \tag{16}$$

$$V(x\ y) = \frac{\Delta_y^2(V(x+\Delta_x,y)+V(x-\Delta_x,y))+2\Delta_x^2 V(x,y-\Delta_y)}{2\Delta_x^2+2\ \ y^2} \quad \text{topedge} \tag{17}$$

For convex comers, the voltage is mirrored in both directions.

These equations give the potential for discrete points in the region; the next step is to convert this system of equations into a corresponding resistance network. Surprisingly, a resistance network can be produced without explicitly calculating all the voltages interior to the region. The next two sections describe an efficient method for performing the conversion: the finite difference grid is represented as a mesh of resistors, which can be transformed directly into the desired resistor network.

2.4.1 Physical Analogs of Finite Differences

Just as the discrete finite difference equation was formulated to approximate to the continuous Laplace Equation, a discrete resistor network can be formulated to approximate the continuous resistive region. Consider the resistor network of Figure 13. Applying Kirchoff's Current Law at node $\mathbf{V(x,y)}$ gives:

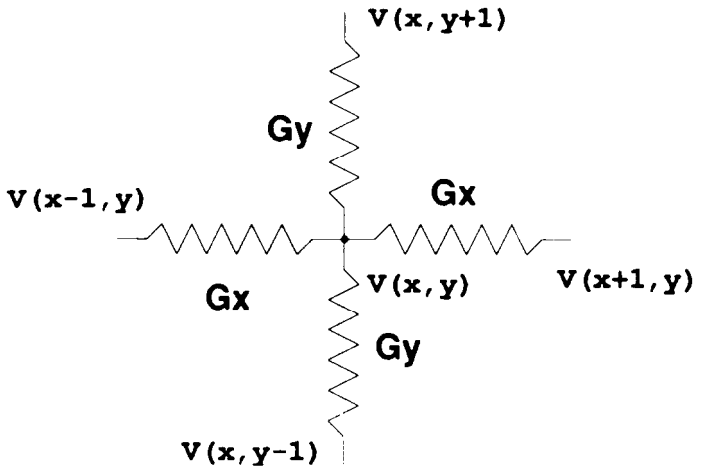


Figure 13: Lumped Analog of Finite Difference Equations

$$V(x,y) = \frac{G_X V(x+\Delta x,y) + G_X V(x-\Delta x,y) + G_Y V(x,y+\Delta y) + G_Y V(x,y-\Delta y)}{2G_X + 2G_Y} \tag{18}$$

This equation is very similar to Equation 13; if $G_x \equiv \Delta y^2$ and $G_y \equiv \Delta x^2$, then the two are identical. The solution is not unique, however; any values of G_x and G_y that have the same ratio $G_y/G_x = \Delta x^2/\Delta y^2$ will produce the same voltage distribution. With this in mind, we can pick conductance values that also produce the correct current distribution. Equations 19 and 20 give the discrete approximations for the current density.

$$J_x = \sigma E_x \approx \sigma \frac{V(x+\Delta x,y) - V(x,y)}{4x} \tag{19}$$

$$J_y = \sigma E_x \approx \sigma \frac{V(x,y+\Delta y) - V(x,y)}{\Delta y} \tag{20}$$

If Δx and Δy are small, then J_x and J_y will be essentially constant across the rectangle and the integral of Equation 2 will just be the current density times the cross-sectional area. For a region of thickness t:

$$I_x \approx t(\Delta y)J_x = \frac{\sigma t \Delta y}{\Delta x}(V(x+\Delta x,y) - V(x,y)) \tag{21}$$

$$G_x = t\sigma \frac{\Delta y}{\Delta x} \tag{22}$$

$$I_y \approx t(\Delta x)J_y = \frac{\sigma t \Delta x}{\Delta y}(V(x, y + \Delta y) - V(x, y)) \tag{23}$$

$$G_y = t\sigma \frac{\Delta x}{\Delta y} \tag{24}$$

The values of G_x and G_y have the correct ratio, $\Delta x^2/\Delta y^2$. These analogs provide an intuitive feeling understanding of finite difference analysis. A region is broken into a set of small rectangles, each of which is replaced by a simplified resistor network. This network can then be solved and replaced by an equivalent network that does not contain the interior portion of the mesh.

2.4.2 Solving the Equations

Once the equations have been formulated and modified to account for the various boundary conditions, the resulting system must be solved. Any algorithm for solving sparse, positive definite matrices may be used here, including Successive Overrelaxation (Section 5.5.2) and Cholesky Decomposition (Section 5.5.1), but significant performance advantages can be obtained by using the node elimination approach of Harbour and Drake[22]. As noted in the previous section, a finite difference formulation produces a mesh of lumped resistors, as does the entire extractor; instead of applying a test voltage to each boundary in succession, node elimination simply transforms the finite difference network into the desired network. To do this, the conductance matrix is first partitioned into two sections: one containing the set of nodes ϵ that are to be retained and the other the set of nodes i that are to be removed

$$G = \begin{bmatrix} G_{\epsilon\epsilon} & G_{\epsilon i} \\ G_{i\epsilon} & G_{ii} \end{bmatrix} \begin{bmatrix} V_\epsilon \\ V_i \end{bmatrix} = \begin{bmatrix} I_\epsilon \\ I_i \end{bmatrix} \tag{25}$$

Since there is no current injected into the internal nodes, I, = 0, and the equations can be rewritten and solved for V_e:

$$G_{ee}V_e + G_{ei}V_i = I_e \tag{26}$$

$$G_{ie}V_e + G_{ii}V_i = \mathbf{CI} \tag{27}$$

Rearranging Equation 27 and substituting it into 26 gives an equation for V_e alone:

$$V_i = -G_{ii}^{-1} G_{ie} V_e \tag{28}$$

$$(G_{ee} - G_{ei} G_{ii}^{-1} G_{ie}) V_e = I_e \tag{29}$$

The term in parenthesis in Equation 29 is an equivalent conductance matrix G'_{ee} that relates the boundary nodes and voltages without calculating the internal values; it is the same matrix that would be produced by applying a test voltage to each edge in turn and measuring the currents flowing to all the other edges. The conductance matrix for the entire system could be constructed using this equation, but inverting G,, would be very expensive because it is nearly as large as G. Instead, Equation 29 can be applied to individual nodes as they are created.

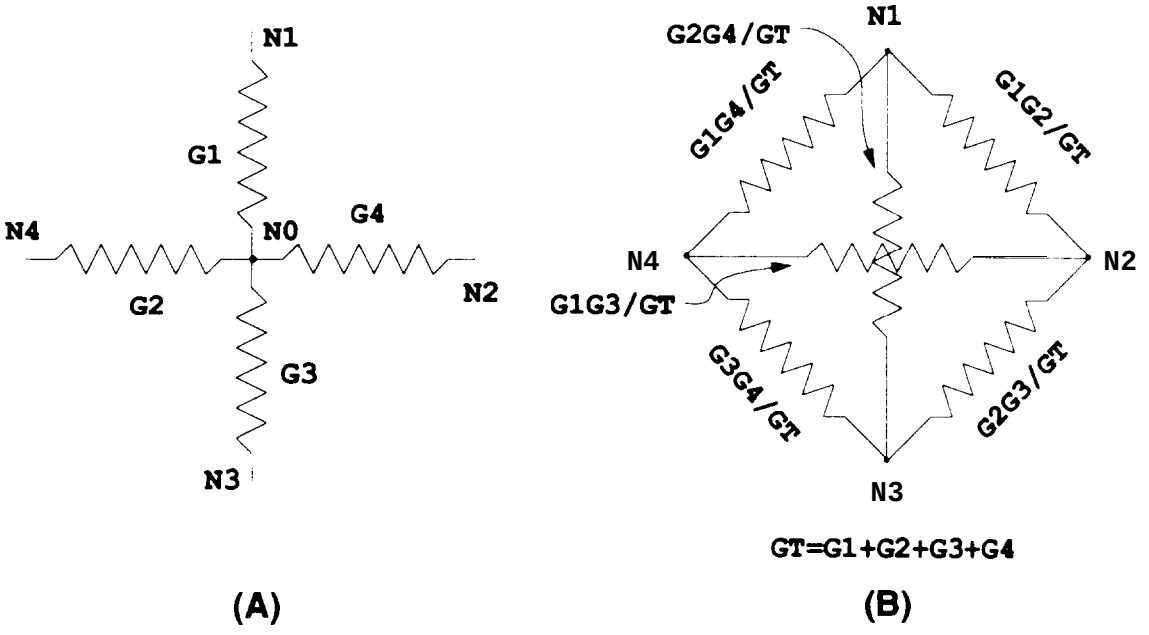


Figure 14: Node Elimination

Consider a node N_0 that connects to n other nodes, shown in Figure 14a for n = 4. This node can be eliminated by applying Equation 29. The inverse matrix G_{ii}^{-1} is the inverse of the sum of all the conductors connecting to the node, and G_{ei} and G_{ie} are just

the row and column vectors containing the values of G_1 . . G,. Multiplying these terms gives the value for a new resistor G_{ij} in terms of the old ones.

$$G_{ij}(new) = G_{ij}(old) + \frac{G_{0i}G_{0j}}{G_T} \tag{30}$$

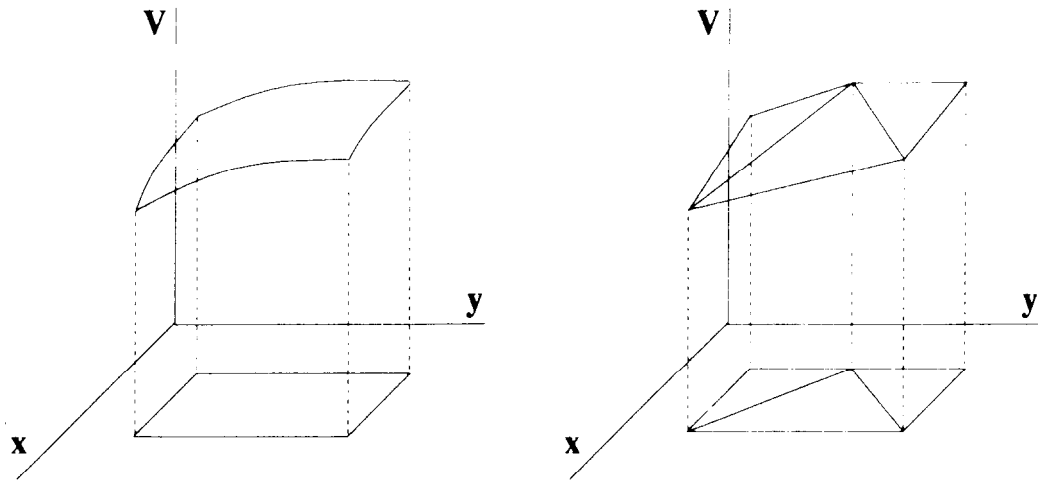
$$G_T = \sum_{k=1}^{n} G_k \tag{31}$$

Once all the resistor analog elements connecting to an internal point have been calculated, this formula can be applied to eliminate the node. Internal nodes can thus be eliminated as they are created; if the order in which nodes are processed is chosen carefully, this algorithm will be considerably faster and will use less memory than would creating the entire matrix and then solving it. Section 2.4 describes an implementation of this algorithm.

2.5 Finite Elements

The finite difference approximation is adequate for regions without much variation in width and current density. When the current density does change considerably, maintaining acceptable accuracy is difficult due to the rectilinear grid; using small enough elements to provide sufficient accuracy in complicated areas requires use of too many elements in simpler sections. Since power supply networks often contain large variations in width and current density, solving them using finite differences would be very expensive. A more general approach, the finite element method, can be used to circumvent this problem.

A two dimensional voltage distribution forms a 3-dimensional surface. The finite element method approximates this curved surface as a set of triangular patches called elements; the voltage in each element is a linear interpolation of the voltages at the vertices. Since the interpolation is linear, the gradient of the potential is constant throughout the patch, and the divergence of the gradient is zero. A constant gradient makes the current density constant, and zero divergence makes Laplace's equation valid inside each element.



Original surface Finite Element Approximation

Figure 15: Approximation of a Potential Surface

Since the potential in each element is fixed by the potentials at its vertices, the key is finding a set of equations that relate an element's vertex potentials to one another and to those of neighboring elements. This is done by requiring the current flow between elements to be continuous; the current flowing out across each edge of one element must equal that flowing in across the same edge of its neighbor. For the center element in Example 16, this gives three equations:

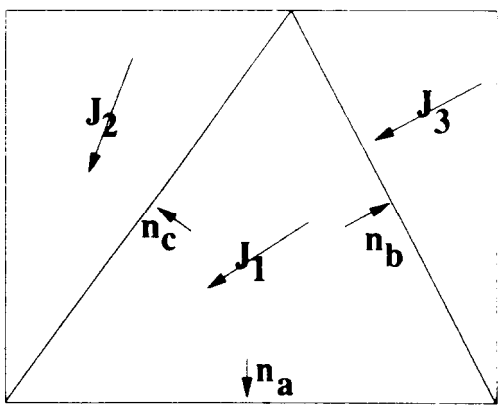


Figure 16: Matching Current Flow Across Boundaries

$$J_1 \cdot n_a = 0$$

$$J_1 \cdot n_b = J_3 \cdot n_b$$

$$J_1 \cdot n_c = J_2 \cdot n_c$$

By repeating this for all the patches adjoining a given vertex, the finite element approximation produces an equation defining the vertex's potential in terms the values at adjoining vertices. When all the elements covering the region are processed, the result is a system relating the node potentials to one another; with the correct potentials applied at the conducting boundaries, the system will give an approximate solution to Laplace's equation for the region. The accuracy will depend on how closely each patch lies to the actual surface. As patches get smaller, the surface regions they represent become more planar, and the solution accuracy improves. The same conclusion can be reached by considering the current densities; as patches get smaller, the current density of the surface region that they represent becomes more and more constant and approaches that of the patch. This suggests that the ideal mesh would have many small elements in places where the current density changes rapidly, and fewer, larger ones where the density is more uniform. Section 2.6 explores this problem in detail.

The finite element method can also be considered a generalization of the finite difference method. The finite difference approximation requires that the first derivatives of the potential be continuous in both the X and Y directions. Using finite elements, the first derivative (in this case the gradient) must again be continuous, but the mesh does not have to be rectilinear and the derivative need not be independently continuous in both the X and Y directions.

Like the finite difference method, relations between the vertex node potentials are often expressed in terms of a physically analogous resistor network to allow use of the node elimination technique described in the last section. Appendix A contains a detailed derivation of the analog for a triangular finite element; only the result is included here. The relation between the three vertices is represented as three conductors, with values given below. A is the element's area. A network for the entire system can be constructed by calculating the conductances for each patch and adding together the two elements that

adjoin each edge.

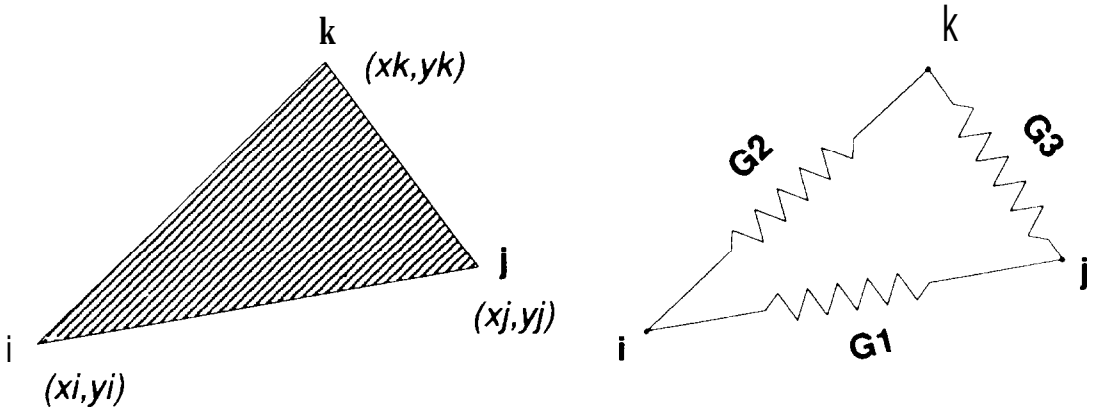


Figure 17: Triangular Finite Element

$$
\begin{aligned}
G_1 &= \frac{\sigma}{4A}(x_j x_k + x_i x_k - x_k^2 - x_i x_j + y_j y_k + y_i y_k - y_k^2 - y_i y_j) \\
G_2 &= \frac{\sigma}{4A}(x_i x_j + x_j x_k - x_j^2 - x_i x_k + y_i y_j + y_j y_k - y_j^2 - y_i y_k) \\
G_3 &= \frac{\sigma}{4A}(x_i x_j + x_i x_k - x_i^2 - x_j x_k + \quad y_i y_j + \quad y_i y_k \; y_i^2 - y_j y_k)
\end{aligned}
\tag{32}
$$

2.5.1 Rectangular Elements

Another commonly used element shape is the rectangle. The discrete conductors for a rectangular element (Figure 18) can be derived by considering it as two triangles. Solving the two elements ijk and $ij'k$ and summing the two diagonal resistors gives values for the five conductors.

$$
\begin{aligned}
G_j &= (\sigma(y_1 - y_0))/(2(x_1 - x_0)) \\
G_k &= (\sigma(y_1 - y_0))/(2(x_1 - x_0)) \\
G_l &= (\sigma(x_1 - x_0))/(2(y_1 - y_0)) \\
G_m &= (\sigma(x_1 - x_0))/(2(y_1 - y_0)) \\
G_n &= 0
\end{aligned}
\tag{33}
$$

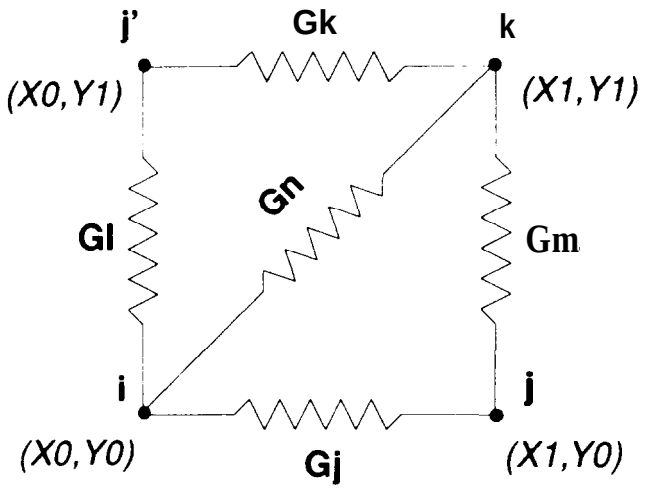


Figure 18: Rectangular Finite Element

As expected, these equations are similar to those of the finite difference analog in Section 2.4.1; the total conductance is the same, but in the finite element case, it is split between the two node-pairs along each edge instead of being assigned to a single one. For a uniform grid, each finite element node-pair not on a boundary will receive half an element of conductance, making the finite element and finite difference physical analogs identical. Finite difference analysis is thus a special case of finite element analysis with rectilinear elements.

2.5.2 Boundary Conditions

Satisfying the boundary conditions for finite element systems is simple. Insulating boundaries conceptually can be treated as any other region, except that the conductance σ is 0. Because of this, the discrete conductors all have 0 value and can be ignored. Perfectly conducting boundaries are regions for which $\sigma = \infty$, so the discrete conductors have infinite value; all nodes adjoining such an element are shorted together and are represented by a single node in the system matrix.

2.6 Finite Element Implementation

To check the accuracy of the simple one-dimensional extractor, Ariel also includes a finite element extractor. Since finite element analysis is slow, this second extractor will only perform it on subregions too complicated to extract by other means. Techniques described in the next two sections subdivide a region and identify the parts where detailed analysis is either unnecessary or redundant. These methods are based on similar components of the extractor EXCL[35]. Following this is a description of the finite element mesh generation and solution techniques used.

2.6.1 Region Subdivision

Finite difference and finite element analysis both produce resistors between each pair of nodes in a net, or $(N^2 - N)/2$ elements in all. For a power bus, the number of nodes is quite large because the region itself is large and has many connecting transistors. Producing a network for the entire region at once is not feasible; it would take too long to compute and would be too large to use. The region needs to be subdivided into smaller sections which can be extracted independently of one another.

The best way to do this is using the idea of *breakfines* developed by Horowitz [25] and extended by McCormick [35]. Breaklines are subdivisions added in long, straight parts of the region in such a way that the current distribution is not significantly disturbed. An example is shown in Figure 19. At the region's comer, the lines of constant potential are unevenly spaced, but they become quite uniform a relatively short distance away. If the region is split parallel to these field lines, and the newly created regions are modelled as perfectly conducting boundaries, then the region's field is virtually unchanged. McCormick calculated that splitting the region one square away from a source of disturbance only adds an error of about 0.1% in the calculated resistance.

Although adding a breakline increases the total number of nodes by 2, it makes two smaller problems out of the original large one. When breaklines are added next to all long, straight sections, the large region is divided into many small ones.

Implementing this algorithm in a comer stitched database is straightforward. Once the region coalescence of Section 2.3.2 has been performed, the extractor makes a second

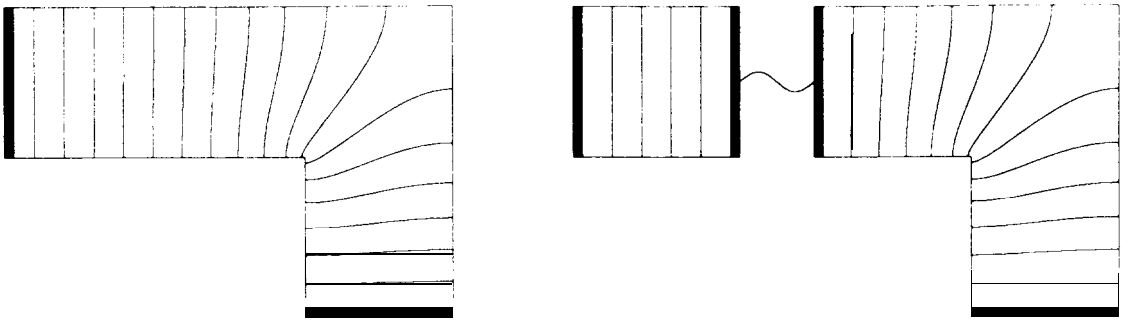


Figure 19: Adding Breaklines to Regions

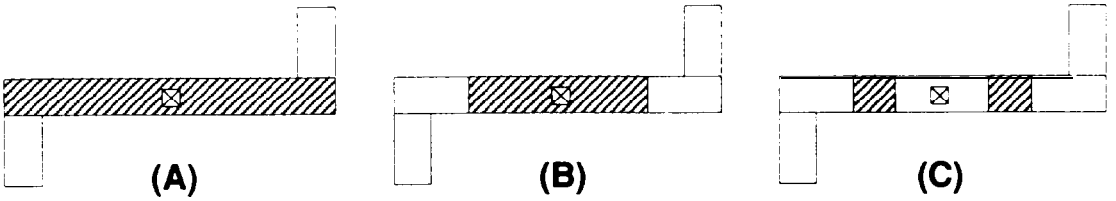


Figure 20: Implementing Region Subdivision

pass through the database looking for rectangles with an aspect ratio greater than 2 or less than 1/2, like the one shaded in the example of Figure 20a. Each of these rectangles is copied into a dummy cell. The four edges of the original rectangle are checked for adjoining material; each adjoining rectangle found is bloated by its width, and any material in the copied rectangle is erased, leaving the truncated rectangle shown shaded in Figure 20b. An analogous operation is performed for any contacts that overlap the rectangle; they are bloated by the height of the original rectangle, then erased in the copy. In the example, this leaves two shaded regions (Figure 20c), which represent areas where the current flow and potential are uniform. The rectangle is split into parts along the left and right edges of the shaded regions. The new rectangles corresponding to the shaded parts of the original are marked as having uniform current flow.

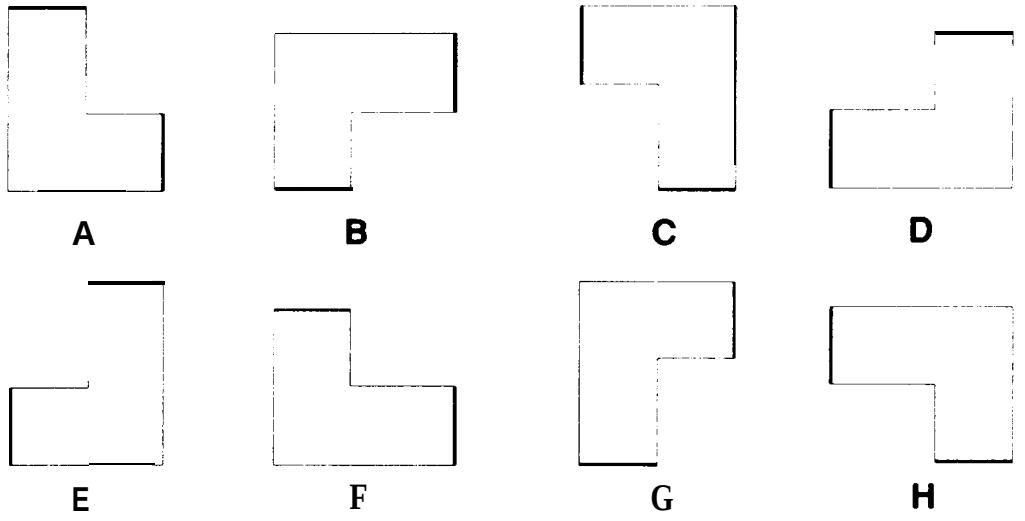


Figure 21: Possible Rotations of a Region

2.6.2 Subregion Library

The extra region fracturing performed in the previous section produced two classes of rectangles: those with uniform current flow and those with nonuniform flow. For the former, resistance calculation is trivial; from Section 2.2, R = $R_{sh}L/W$. The latter require further calculation. The resistance for each nonuniform rectangle cannot be calculated by itself; it must be considered along with its neighbors. Each group of adjoining non-uniform tiles form a *cluster,* which is bounded by space tiles, modelled as insulating edges, and by transistors, contacts, and uniform-flow rectangles, which are modelled as perfectly conducting edges.

Due to the repetitive nature of VLSI designs, many of these clusters are either identical or mirrored/flipped copies of one another. Each such group need only be extracted once. This is done using a dynamic library. Before a cluster is extracted, the relative positions of its constituent rectangles and conducting edges are used as the key to a hash table. The entry corresponding to each key is the resistance network extracted for the cluster. If an entry is found for a given cluster configuration, then a copy of the entry is appended to the resistor network. If no entry is found, then a new one is added for the cluster after it is extracted

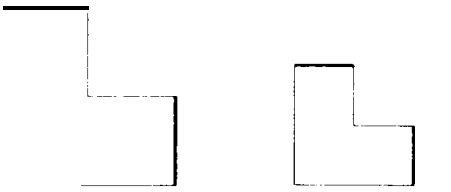


Figure 22: Region Scale Invariance

Since cells are often rotated or flipped, it is important that the clusters match regardless of rotation. Figure 21 shows the 8 possible rotations of an asymmetric cluster. There are a number of ways to make entries match regardless of rotation. One approach would be to make an additional key for each unique rotation of the cluster. The advantage of this is that table accesses are fast; no additional processing must be done to a cluster before it is compared against the library keys. The primary disadvantage is that it uses additional memory; since the key for an entry is often as large as the entry itself, and most large clusters are asymmetric, such a library would be prohibitively large. Instead, a single canonical key is used The extractor first calculates the centroid of each region, then rotates the region so that the center of gravity is as low and as far to the left as possible. In the example, rotation F would be chosen. This scheme usually only requires one key, making it memory efficient. It has two disadvantages: the library access time is greater due to the cost of calculating the centroid, and the rotation may not be unique if the centroid lies on the line x = y. In practice, the increased access time is unimportant because extraction time is dominated by the finite element calculation itself. To ameliorate the second problem, the extractor calculates the centroid of the conducting edges if the correct rotation is ambiguous. (Since the edges are actually lines, they are first assigned a small finite width.) Sometimes, this will also fail to give a unique rotation. In this case, the region will actually get extracted more than once. The loss of efficiency due to unnecessary duplication of extraction is negligible, however, in practice, the large clusters whose processing dominates the extraction time have too many rectangles and edges to be ambiguous.

One other possible enhancement is scale invariance. Although they are of different

sizes, the two regions of Figure 22 have the same resistance. In his library implementation, McCormick normalizes all rectangles to 1/256th of the first rectangle's height. Such an implementation would not work for a power bus extractor due to the great variation in width. It is not uncommon for minimum width wires to connect to an extremely wide main power bus. Any rounding in the width of these minimum width wires due to quantization could introduce substantial error in the overall resistance.

Another scaling implementation might be possible, but its utility in the power network domain is questionable. The large, multirectangle regions that dominate the extraction time are not scaled versions of other regions, and any time spent providing scale independence is wasted for them. Since the potential return for scaling is problematical, the extractor does not use it.

2.6.3 Mesh Generation

The extractor must produce a network for any cluster that does not match an entry in the library. McCormick and Horowitz both use finite difference analysis when they need to accurately extract a region; unfortunately, this approach proves inadequate for power bus extraction. Power buses have a wide variations in width; a rectilinear grid that provides adequate accuracy near small features will run too slowly in the large, coarse sections of the design.

To accommodate large feature size variation, some sort of nonuniform finite element mesh generation is needed. One possible approach is the adaptive mesh generation algorithm devised by Machek and Selberherr[34]. This method has two drawbacks for resistance extraction. First, it requires that the internal node voltages be calculated; the fastest solution algorithm for resistive meshes, node elimination (Section 2.4.2), does not calculate these intermediate values. Use of adaptive mesh generation would require the one of the slower solution techniques be used. The other problem is that the mesh would have to be regenerated for each edge; since the potential distribution varies depending on the boundary to which the voltage is applied, the mesh will also vary.

Because of these drawbacks, a modified version of Kemp's heuristic mesh generation

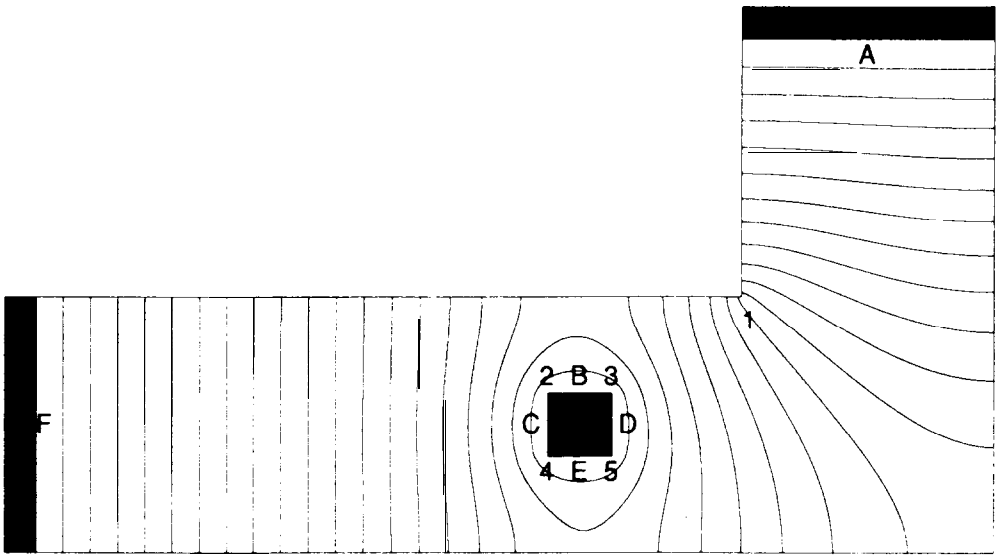


Figure 23: Sources of Potential Disturbance

[31] is instead used. The basic idea of this algorithm is to produce a mesh with many elements in places where the current density is changing rapidly and fewer in places where it is changing slowly. To do this, the region is first represented as a set of rectangular elements' containing regions of homogeneous resistivity. Each of these elements adjoining a point of voltage disturbance needs to be subdivided. The disturbances are caused by concave comers in the regions. In the example of Figure 23, the disturbances are the bend in the region (labelled 1) and the comers around the contact (labelled 2-5). Kemp also uses the entire edge between two regions of differing resistivity (labelled A-F), but this seems unnecessary; as can be seen from the potential lines, the current does not change rapidly except at the comers.

Each element adjoining a disturbance is recursively split in two until the elements nearest the comer are below some minimum size. The basic idea in splitting is to subdivide the element in such a way that the resulting children are well ratioed,[2] and to

'The initial elements are rectangles because Magic's database only supports orthogonal shapes. Although Kemp's algorithm can handle arbitrary shapes, the current discussion is limited to the Manhattan case.

[2]The aspect ratio of an element is the length of the short side divided by the length of the long one;

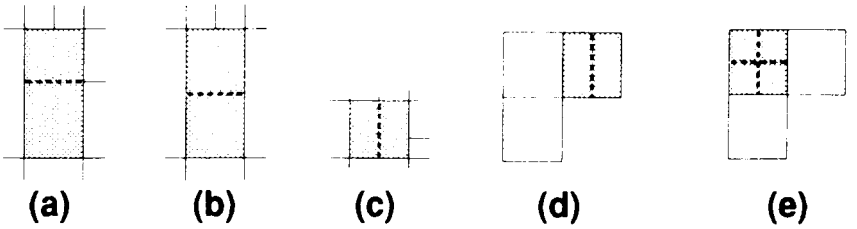


Figure 24: Subdivision of Elements

minimize the number of nodes required by making the split points of adjacent elements align. The following rules (illustrated in Figure 24) are applied:

- Ill-ratioed rectangles are split in two along their long side.

  a. If there is an edge between two elements along one side, and subdividing there does not give children more ill-ratioed than the parent, split the element at the edge. If there is more than one such edge, use the one that gives the best ratioed children.

  b. If there is no such edge, split the element in the middle.

- Well-ratioed rectangles can be split in either direction.

  c. Split at the adjacent edge that gives the best ratioed children,

  d. If no such edges exist, split along a line perpendicular to the expected direction of current flow.

  e. For a comer element with no adjacent edges, split in both directions.

Once the adjacent element is subdivided, the same procedure is applied to all of its children that are next to the disturbance. This continues until all of the adjacent elements are smaller than some predefined size. Figure 25 shows how this algorithm operates on a concave comer. The edges are numbered in the order in which they were added, with the letter (a-e) in parentheses showing which rule was applied

the closer this ratio is to 1, the more 'well-ratioed' the element is.

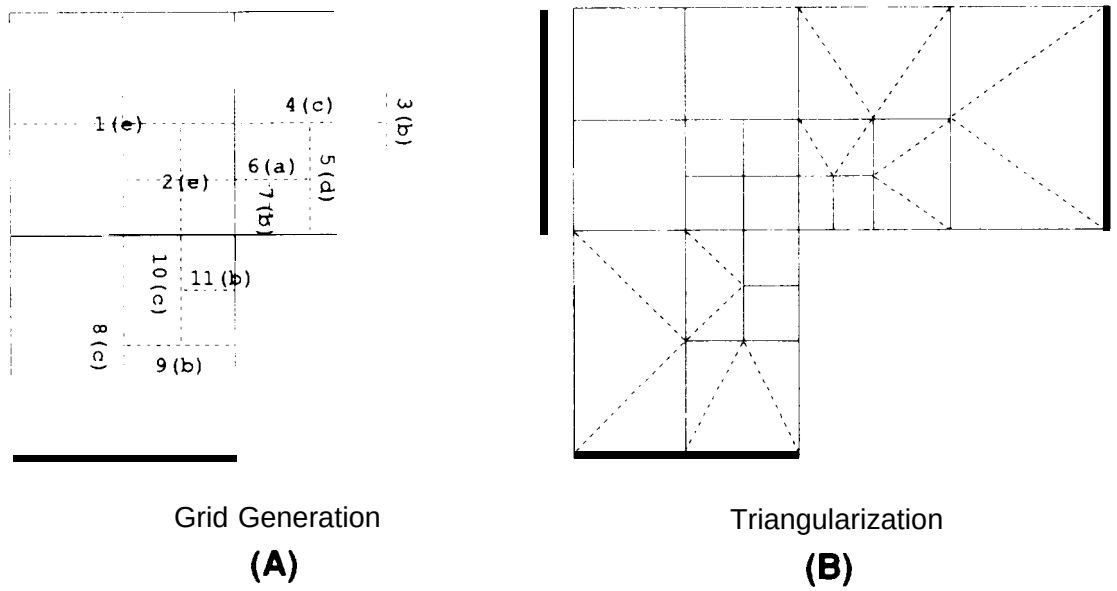


Figure 25: A Mesh Generation Example

This algorithm works reasonably well. Elements end up being fairly square; such regions generally have less current density variation than does a long, thin one. Requiring mesh lines to match up reduces the total number of elements. Splitting in two makes the element density spread out in about a 45 degree line from the disturbance, just as the change in current density does. The mesh generator thus puts many elements where they are necessary and fewer where they are not.

Once the elements have reached the desired size, they are replaced by a finite element mesh. Element vertices and edges form the nodes and edges of the mesh, respectively. The mesh is composed of two element types: rectangles and triangles (Figure 26). The triangles always occur in groups of three that form a rectangle. The extractor decides which element to use depending on the number of neighbors the element has. Elements with four neighbors use the rectangle, while those with five use the three triangle set. Some of the elements may have more than five neighbors; this is fixed by splitting them along one of the edges to form two elements. Figure 25 shows the mesh elements added during triangularization.

Implementing the above algorithm in Magic is straightforward. All of the tiles composing a cluster, along with the conducting edges (which are assigned some finite width

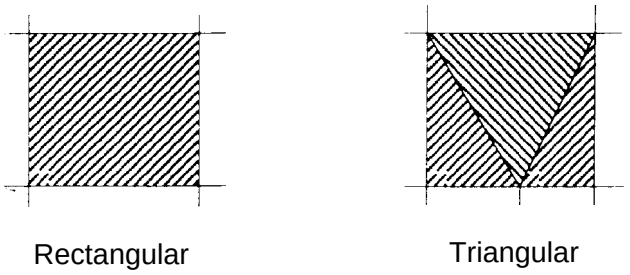


Figure 26: Finite Elements Used in Generation

so that they can be represented as a tile) are copied into a dummy cell. Solution accuracy can be controlled using an optional scale factor by which all rectangles are magnified as they are copied. A large scale factor results in smaller final elements near a disturbance. The cluster is then searched for concave comers. At each such comer, the horizontal tile is split along the y-axis at the comer to produce the three initial rectangles of Figure 25, each of which is then processed by the above algorithm. The algorithm is done when the comer's adjacent elements are all 1 by 1 squares. The extractor then makes a second pass through the entire cluster, looking for elements with more than five neighbors. Each such element is subdivided again, and the adjacent elements are rechecked to insure that the additional edge does not leave them with too many neighbors. The final triangularization is not explicitly performed because Magic can only represent rectangles. Instead, the extractor adds the elements for all three triangles at once whenever a five-neighbor rectangle is encountered.

2.6.4 System Solution

Once the mesh has been generated, it must be solved for the node to node resistances. The extractor uses Harbour and Drake's node elimination algorithm (Section 2.4.2) to remove the unwanted interior nodes. In implementing this algorithm, the extractor has to decide the order in which the internal nodes will be processed. If the mesh is rectilinear, as Harbour and Drake's is, then one good ordering is to process the nodes row by row. The extractor walks down the long side of a rectangle, adding rows of new nodes. As the mesh is extended for each element in a row, any nodes sandwiched between this element and previously processed ones have all their connections in place and may be eliminated.

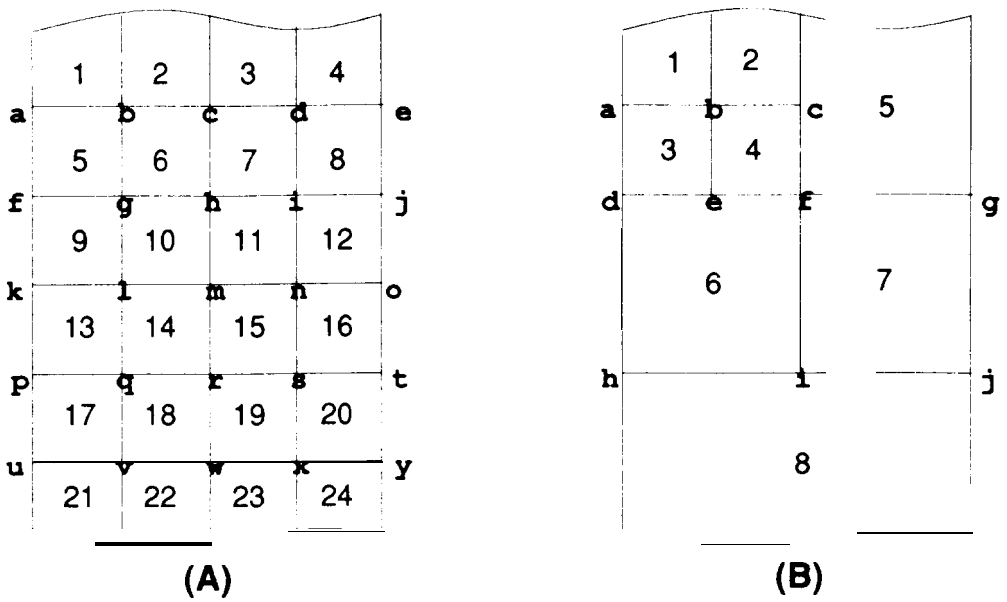


Figure 27: Order of Node Elimination

By processing the elements in short rows, the number of outstanding nodes (nodes that have some but not all of their connecting resistors in place) is kept low. The numbers in Figure 27a show the order in which elements are processed, while the letters indicate the order in which nodes are eliminated. For rectangles that are wider than they are tall, elements are processed by column instead of by row.

Implementation of this algorithm is trickier when is grid is not rectilinear. There are two interrelated problems: choosing an order of element processing that will minimize outstanding nodes and determining when all the resistors connecting to a node have been processed For regions that are taller than they are wide, the extractor uses Magic's tile enumeration algorithm. This algorithm visits tiles in the top to bottom, left to right order of their lower left comers. The example of Figure 27b shows this ordering. At each tile, the extractor generates the correct mesh elements, then marks the tile as processed. It then checks the nodes along the sides of the tile to see if all their adjoining tiles have also been processed.[3] If they have, then the node is eliminated For rectangles that are

[3] It is not really necessary to check all four edges; another method would be to automatically process all node-s along the top and left sides, except those at the bottom left and top right corners. Due to the tile enumeration order, these nodes will only have processed neighbors. Assuming the space tiles are also visited, then all the nodes will eventually be processed. The problem with this approach is that all the nodes along the right edge will get processed last because the bottom right space tile is the last to be

wider than they are tall, the tiles are instead visited left to right, top to bottom.

This enumeration works fairly well for regions that have an aspect ratio that is not near one, but is not as useful for more complex areas that are more or less square. A better algorithm might try and follow the twists and turns of individual rectangles within the cluster. For complicated structures, however, doing this while still ensuring that all elements are visited is tricky. For simplicity, the straightforward enumeration is used even for more complicated structures.

2.7 Results

To compare the speed and accuracy of the two algorithms, I extracted the ground buses from two midsized circuits: the register file from the MIPS-X microprocessor[24], and a self-timed divider[60]. Table 2 gives the circuit sizes and running times on a Titan, a 15-MIP RISC machine. The finite element grid size chosen gave solutions accurate to about 2%[35]. Even with the library of stored shapes, subdivision of regions, and heterogenous grid, the finite element extractor is over two orders of magnitude slower than the simple polygon extractor.

	Size		Running Time (seconds)	
Circuit	Transistors	Rectangles	Polygonal Decomposition	Finite Element
Register File	10159	12918	68	13245
Self-Timed Divider	5777	7173	39	10393

Table 2: Extraction Times for Example Circuits

The summary of library effectiveness in Table 3 suggests why the finite element method is so much slower. The first two columns show the number of shapes that missed and matched in the library. The third column gives the time required to solve the shapes

visited. These extra outstanding nodes add overhead to the node elimination. It is therefore faster, though less elegant, to check all the edges.

which were not present, while the final column gives the time that would have been required to solve all the matching shapes had the library not been used. Although over 90% of the shapes matched for both test cases, the time saved by using the library was relatively small: less than 30% for both cases. This suggests that the finite element solver is spending most of its time on a few large, complicated shapes that occur only once in each design. In retrospect, this is not particularly surprising. A power bus is generally fairly wide near its root; this extra width limits the number of places where long, straight sections free of current disturbances can be found. Each bus will thus have at least one large region with multiple connection points that will need to be solved. Conversely, the shapes that match will be the simple bends and junctions near the leaves of the network, and the time saved by each match will be small.

Circuit	Miss Count	Match Count	Miss Solve Time	Match Saved Time
Self-Timed Divider	260	3535	10337	3085
Register File	154	4195	13150	5164

Table 3. Previous Solution Library Efficacy

Despite the disparity in running times, the two extractors produce nearly identical results. Figure 28 shows the accuracy of the methods on the two test circuits. For each circuit, I applied the same input current distribution to both networks and compared the resulting voltages for nodes in the metal sections of the bus. The correlation between the methods is fairly good in both cases.

To see how the simple polygon extractor performed on larger designs, I extracted power networks for the three large CMOS designs described in Section 1.2. The results are given in Table 4. In all cases, the extraction time was dominated by flattening the power network layout; the remaining preprocessing steps and the actual extraction were less than half the total. This implies that substantial speed improvements will be difficult unless the design methodology restricts cell overlaps sufficiently to permit hierarchical

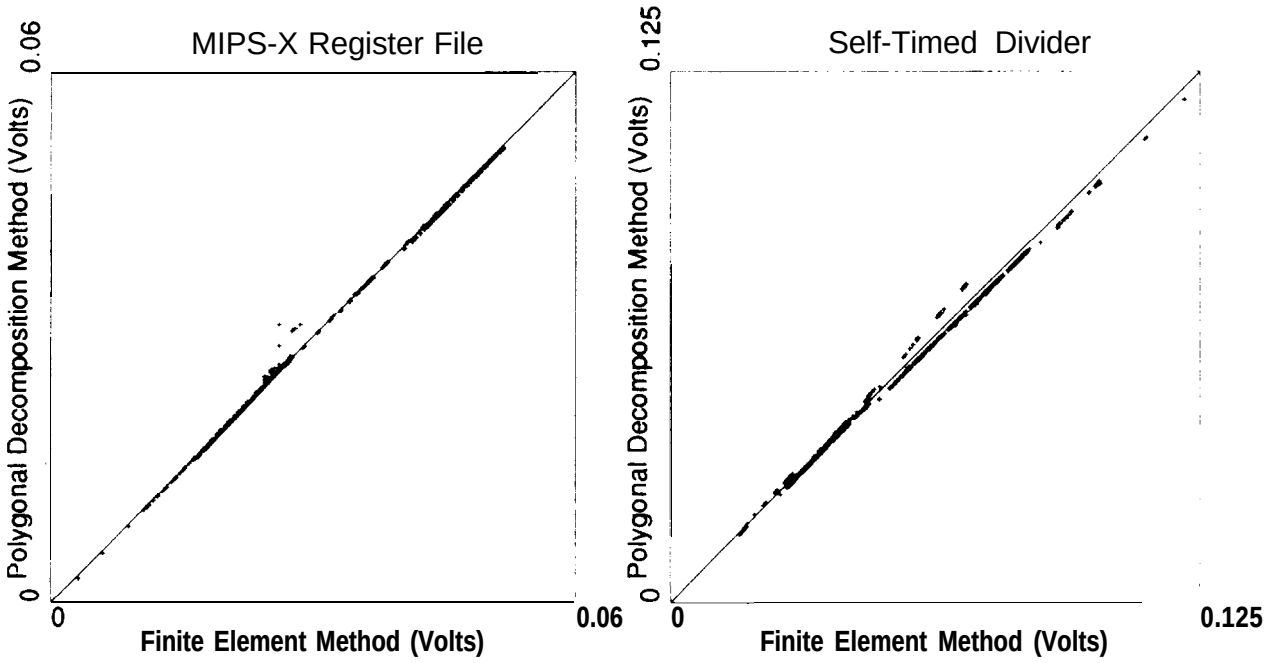


Figure 28: Accuracy of Resistance Extraction

extraction. Despite the cost of flattening the bus, however, the extractions still generally took ten minutes or less. Since the polygonal extractor gives nearly the same result in a fraction of the time required by the finite element method, I chose it as my primary extractor for large designs.

Circuit	'lime (seconds) Flattening	Fracturing	Extraction	Total	Memory Usage (Mb)
MIPS-X Ground	418	54	139	611	50.3
MIPS-X Vdd	299	39	50	388	57.0
SPIM Ground	244	56	78	378	27.3
SPIM Vdd	289	53	118	460	33.9
μTitan Ground	238	35	115	388	43.6
μTitan Vdd	413	79	130	622	66.8

Table 4: Example Circuit Extraction Times

The rightmost column in the table gives the memory usage for the various designs. The memory usage is fairly large, primarily due to flattening the Rower bus. Extraction

without flattening is not an easy problem, however. Section 2.3 described why hierarchical extraction is difficult when arbitrary overlaps are permitted: overlapping sections of layout may connect to one another at any location. Flattening one section of the layout at a time, which Magic's capacitance extractor does for regions of cell overlap, would also be tricky to implement for resistance extraction. A tile's resistance cannot be easily calculated if it has been bisected by the section boundary, and determining which tiles are part of the power bus is also difficult. There is no obvious approach that I can see for reducing the extractor's memory usage; this is one of the areas in which I plan future work.

Chapter 3

Current Estimation for CMOS

> So wonderfully are these two states of Electricity, the ***plus*** and ***minus,*** combined and balanced in this miraculous bottle! situated and related to each other in a manner that I can by no means comprehend! If it were possible that a bottle should in one part contain a quantity of air strongly comprest, and in the other part a perfect vacuum, we know the equilibrium would be instantly restored ***within.*** But here we have a bottle containing at the same time a ***plenum*** of electrical fire, and a ***vacuum*** of the same fire; and yet the equilibrium cannot be restored but by a communication ***without!***
>
> Benjamin Franklin
> ***Experiments on Electricity (I 774)***

The last chapter described a resistance extractor that transforms a mask-level description of the power supply network into a lumped resistor network. This resistor network can be combined with the current profile generated using the techniques of the next two chapters to calculate the power supply's voltage-current distribution. This chapter examines techniques for deriving the power supply current distribution for CMOS circuits; the following chapter does the same for ECL designs.

This chapter is divided into four main sections. The first examines the underlying sources of current and their relative importance in different logic families. Following this is a description of two approaches that others have tried and discussion of their strengths

and weaknesses. Next is a description of the switch level simulation approach that I have adopted, together with analysis of some of the more subtle problems that had to be solved. The final section discusses the estimator's performance on some real designs.

3.1 Introduction

The currents that circuits draw from their power supply buses can be divided into two categories: ***capacitive*** and ***short-circuit.*** The CMOS inverter in Figure 29 dissipates both currents. When the inverter input **In** falls, Transistor Ml turns on and begins to charge the capacitor at Node Out. This charging requires the capacitive current component I,,.

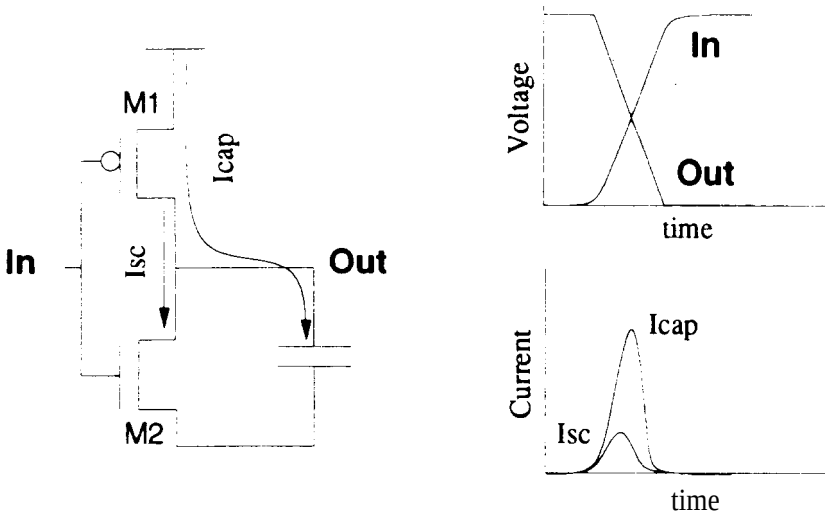


Figure 29: Simple Current Example

As Ml is turning on, the n-channel transistor M2 is turning off. While In is in the range $V_{TN} < V_I < V_{dd} - V_{TP}$, both transistors are conducting, and the short-circuit current I,, flows directly from one power bus to the other through the two devices.

The relative importance of the two components varies with the technology. CMOS circuits are usually capacitive current dominated; the short-circuit current is only significant for overdriven gates, where the input rises much slower than the output. For normal gates, where the input rise time is equal or less than the output time, Veendrick [59]

has calculated that the capacitive current accounts for at least 80% of the total. In ECL, the situation is usually reversed; the short-circuit component dominates. Unlike CMOS, ECL circuits dissipate static current; the extra current required to change circuit state is relatively insignificant, except as a source of inductive noise. Most other technologies that dissipate static power are likewise short-circuit dominated; in NMOS, for example, the main source of power dissipation is the pulldown current of gates with low outputs.

The actual capacitive currents arising from a node transition are more complex than Figure 29 might suggest. A more complete circuit is shown in Figure 30a. The capacitance at Node N has several components, some of whose bottom plates are connected to Vdd and some to ground. These capacitors are divided into two groups, Cvdd and Cgnd.

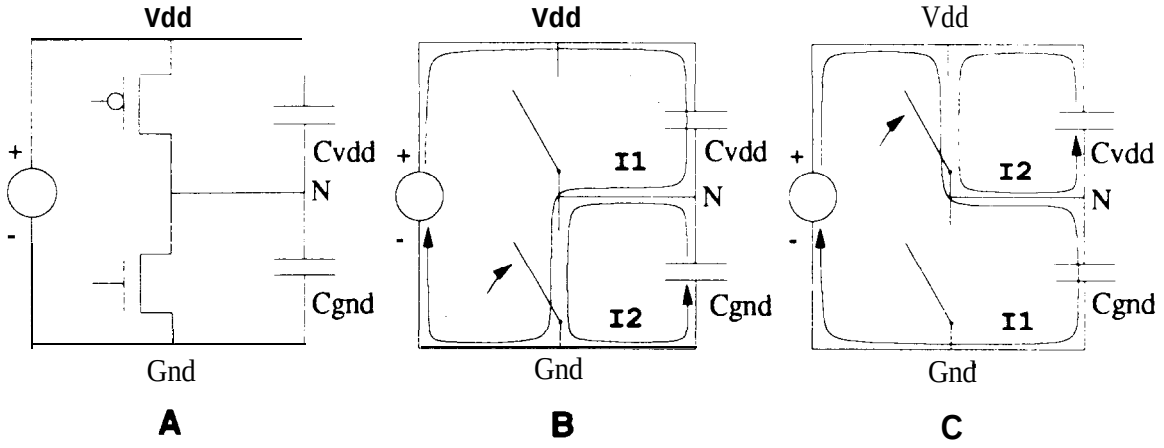


Figure 30: Effects of Capacitance on Currents

Figure 30b shows the currents generated by a transition from 1 to 0. The charging current for the Vdd capacitors (11) comes from the power supply, through a well contact to the Vdd capacitor and back to the power supply via the discharging transistor. The discharging currents for ground capacitors (12) flow from the top plate to the bottom plate through the discharging transistor and a substrate contact. The corresponding currents for a 0 to 1 transition are shown in Figure 3Oc.

The image currents that charge and discharge the capacitors' bottom plates have two interesting effects. First, every node transition produces transient currents in both the Vdd and ground distribution networks. Second, the current to discharge a capacitor does

not go through the power supply; it makes a loop that is internal to the chip. A power analysis system that either ignores the bottom plate connections or assumes they all are connected to the same terminal will miss these two important effects.

The previous example outlines how to calculate the current drawn by a single gate; extending the estimation for an entire design is much more complicated. The fundamental problem is pattern dependence; state changes in the design are a function of its inputs. The entire circuit does not change state at the same time, nor do all nodes change state at the same rate. To extract an accurate current profile for the circuit, some pattern and timing information must be included.

3.2 Previous Work

Besides the simple (and slow) expedient of running Spice on the circuit[21], other researchers have investigated a couple of approaches to current estimation. The first, described in the next section, uses a pattern independent approach adopted from timing verification. The second, based on probabilistic simulation, modifies techniques originally pioneered for test-pattern generation. These two methods and the timing simulation approach that I use were developed at approximately the same time, providing an interesting contrast on how competing teams can reach different solutions to the same problem.

3.2.1 Timing Analysis

A timing analyzer such as *TV* [27] or *Crystal*[46] might be used to derive a current distribution. Tyagi's estimator ***HERCULES*** [58] uses this approach. Its primary advantage is pattern independence; instead of propagating individual real or boolean signals, it applies all possible values to the circuit at once.

A simple example is shown in Figure 31. Nodes **In0** and **In1** are assumed to be stable when the clock ϕ_1 rises. Depending on the values of these signals, either a 1 or a 0 will be propagated to the inputs of the inverters. The timing analyzer considers both cases. Possible waveforms at the circuit's nodes are shown on the right. (Two other possible waveforms, constant 0 and constant 1, are not shown.) The waveforms for the

two inverters are simple; the input signal is simply inverted and delayed at the output. For node N2, the potential waveforms are more complicated. There are three possible cases for each transition, depending on whether nodes NO and **N1** change value. The transition starting time, ending time, and slope all vary depending on what values **In1** and In2 initially held; the timing analyzer simply reports that a transition did not start before time t2 and was definitely over by time t5.

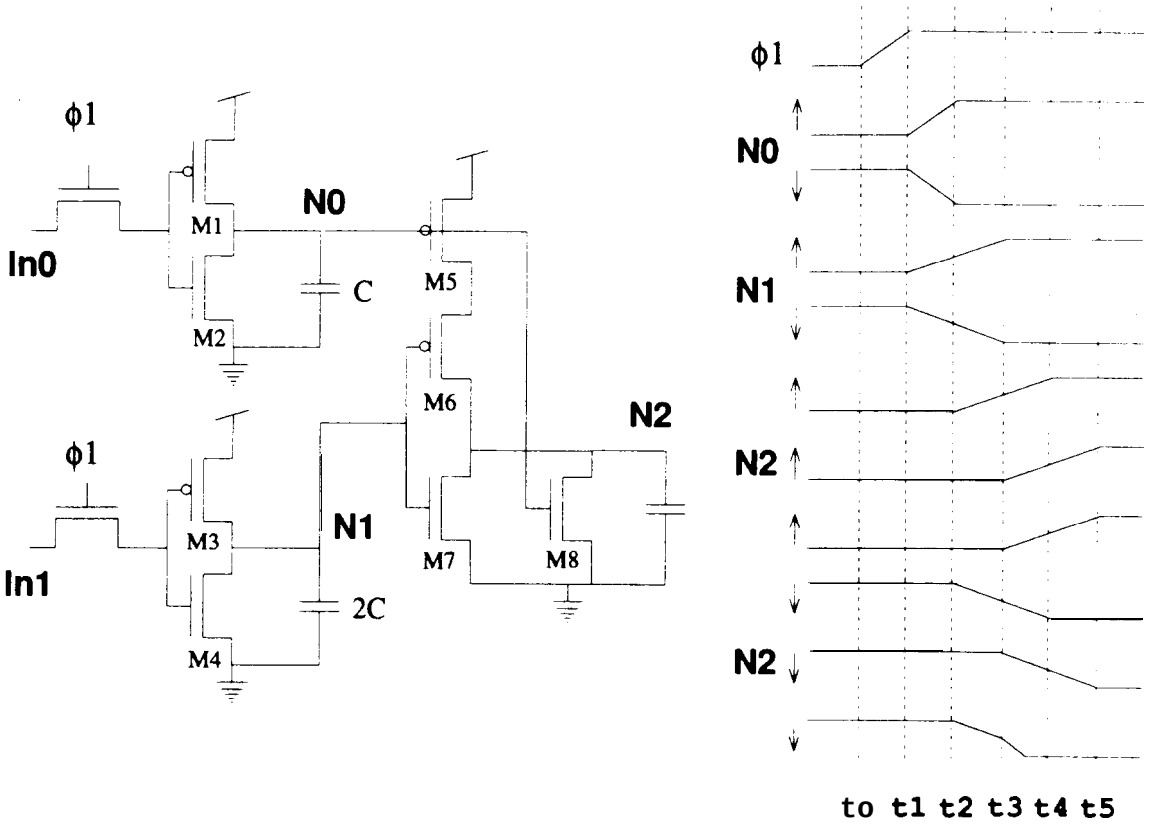


Figure 31: Timing Analysis Example

The next step is to convert this timing information into a current profile for the circuit. For Nodes NO and **N1**, this is fairly simple; the starting and ending times for the transitions are known and the average current is simply (Vdd)(C)/($t_f - t_i$). If desired, a more accurate current pulse shaping algorithm, such as Ousterhout's tabular **method[46]**, may be applied These current pulses are applied to the power buses at the source connection of transistors M 1 -**M4**.

Calculating a pulse for node N2 is more complicated. Both the current pulse's magnitude and its timing depend on the inputs. For the falling edge, the location of the pulse (either through M7 or M8, or through both) is also input dependent. The most conservative approach is to assume that the maximum current, which occurs when both M7 and M8 are on, flows for the entire interval from t2 to t5.

In theory, this approach seems attractive because it is pattern independent, giving the designer a worst-case approximation. In practice, it does not work particularly well. When I modified Jouppi's timing analyzer, TV, to estimate currents, the voltage drops calculated were greater than the supply voltage of the chip. This was due to a number of causes. the most important of which was overestimation of decoder currents. In the 2-bit decoder of Figure 32, there are eight possible transitions for the four output signals; each may go from high to low or low to high. TV assumes they all occur and produces eight current pulses. In practice, at most two of the transitions will occur in a given cycle; one line will rise and one will fall. For the example, TV will overestimate the current by a factor of 4. Circuits with a 32-bit datapaths often have 5-bit decoders; for these, the current estimator will be off by a factor of 32. A decoder's load capacitance is usually fairly large, making the error from just this one source substantial. In the layout, these decoders are placed next to one another and share common power and ground lines, which are generally sized to support only the two decoders active in a cycle. Putting an order of magnitude more than the actual current through these wires produces some horrendous voltage spikes in the supplies.

There are some other sources of error in this approach. As shown in the example, any gate with a fan-in greater than one will have some uncertainty about the timing, location, and magnitude of its current pulses. To be conservative, the timing analyzer must assume that the maximum current occurs for the entire interval. In a synchronous CMOS design, power usage is spread out across the clock cycle; extending the intervals during which a gate draws current leads to greater overlap between current pulses and an overly conservative current estimate.

Remedying these problems in a pattern independent manner is difficult. For decoders, the designer could label the outputs as having mutually exclusive transitions; the analyzer would then select which transition produces the highest voltage drop. The success of

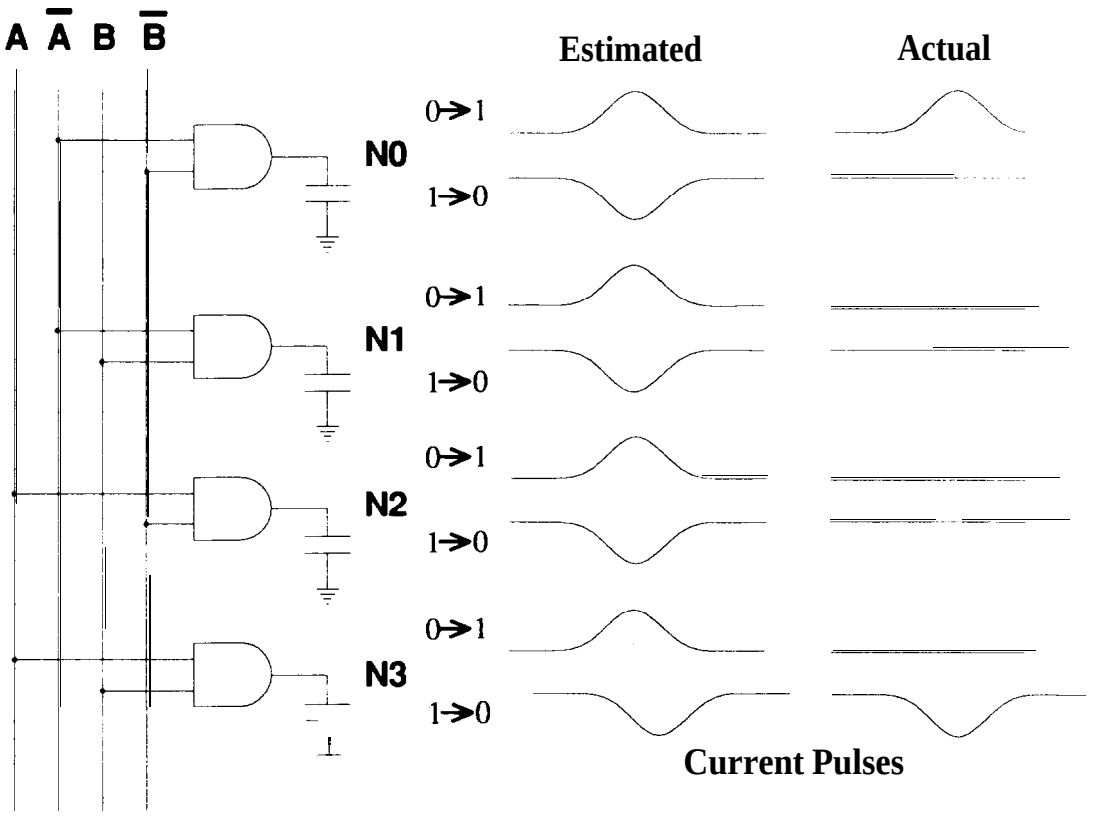


Figure 32: Decoder Current Estimation

this depends on the analyzer's ability to pick the worst case; in a system with many such possible circuits, picking the correct combination for all of them is somewhat problematical. Removing the uncertainty of high fan-in gates is more difficult; specifying the current pulse more precisely reintroduces pattern dependence into the system. A workable current estimator based on timing analysis may be possible, but devising a system that is not overly conservative while preserving pattern independence and ease of use is a difficult, and unsolved, problem.

3.2.2 Probabilistic Analysis

A timing analyzer overestimates the amount of activity in a system because it assumes all possible transitions actually do occur. The probabilistic estimator *CREST*[40, 41] avoids this overestimation by attaching an event probability to each transition in the system.

A simple example of this method is shown in Figure 33. Each node in the circuit is represented by four probabilistic waveforms:

- P_L - The probability that the node has a low value.
- P_H - The probability that the node has a high value.
- P_{HL} - The probability that the node value is changing from high to low.
- P_{LH} - The probability that the node value is changing from low to high.

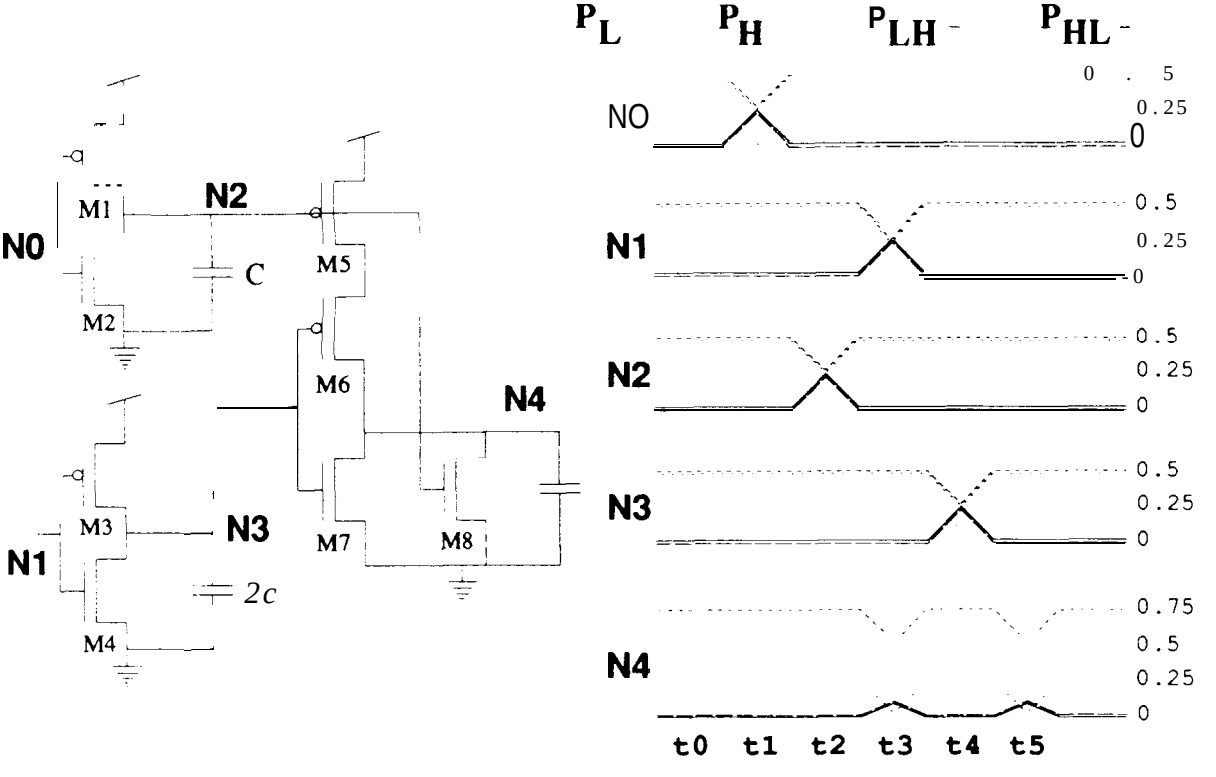


Figure 33: Probabilistic Analysis Example

For each node, the sum of these waveforms is one. Given the probability waveforms for a circuit's inputs, CREST derives the waveforms for the internal nodes. For simple gates, this process is straightforward. An inverter simply swaps the sense of the waveforms:

- $P_{H_o} = P_{L_i}$
- $P_{L_o} = P_{H_i}$
- $P_{HL_o} = P_{LH_i}$
- $P_{LH_o} = P_{HL_i}$

For the nor gate, the output probability waveforms are slightly more complicated because the value of one input may mask a transition on the other. A high to low transition is only propagated if the other input is already low or if it is also changing from high to low. A low to high event is only propagated if the other input is low or it is also changing from low to high. The high and low probabilities cover the other cases. The probabilites for a nor gate are:

- $P_{H_o} = P_{L_{i0}} \times P_{L_{i1}}$
- $a\ P_{L_o} = P_{H_{i0}} + P_{H_{i1}} - P_{H_{i0}} \times P_{H_{i1}} + P_{LH_{i0}} \times P_{HL_{i1}} + P_{HL_{i0}} \times P_{LH_{i1}}$
- $P_{HL_o} = P_{LH_{i0}} \times P_{L_{i1}} + P_{LH_{i1}} \times P_{L_{i0}} + P_{LH_{i0}} \times P_{LH_{i1}}$
- $P_{LH_o} = P_{HL_{i0}} \times P_{L_{i1}} + P_{HL_{i1}} \times P_{L_{i0}} + P_{HL_{i0}} \times P_{HL_{i1}}$

Given the probability waveforms for the nodes, the next step is to calculate the expected current waveform that a node transition draws from the power buses [1]. CREST models current waveforms as triangular pulses with a maximum value i_{max} and duration τ, as shown in Figure 34.

CREST estimates the maximum expected current as the supply voltage times the expected peak conductance. This conductance is the sum of the conductances of a given set of 'on' transistors between the supply and the output node times the probability that this configuration occurs. For the inverters, the current is easy to calculate; it is the maximum possible current times the probability that a transition occurs:

[1] This discussion of CREST is considerably simplified in order to give the algorithm's flavor without becoming mired in detail. Among the points omitted from this discussion are the different effects of ground capacitance and power capacitance described in Section 3.1 and a mathematically rigorous derivation of the expected current pulse.

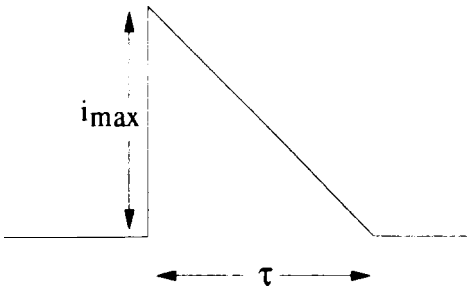


Figure 34: CREST Current Waveform

$$i_{max_P} = Vdd \; x \; g_p \; x \; P_{hl_1} \tag{34}$$

$$i_{max_N} = Vdd \; x \; g_n \; x \; P_{lh_1} \tag{35}$$

g_n and g_p are the conductances of saturated n-channel and p-channel devices. For the nor gate, the currents are different for the pullup and pulldown trees.

$$i_{max_P} = Vdd \times g_p/2 \times (P_{HL_{10}} \times P_{L_{11}} + P_{HL_{11}} \times P_{L_{10}} + P_{HL_{10}} \times P_{HL_{11}}) \tag{36}$$

$$i_{max_N} = Vdd \times (g_n \times P_{LH_{10}} \times P_{L_{11}} + g_n \times P_{LH_{11}} \times P_{L_{10}} + 2g_n \times P_{LH_{10}} \times P_{LH_{11}}) \tag{37}$$

In the pullup tree, the conductance is fixed at $g_p/2$ by the two devices in series, so the current is simply this value times the probability that the output rises. In the pulldown tree, the conductance depends on whether one or both devices are on. If only one is on, the conductance is g_n; if both are on, the conductance is $2g_n$. CREST derives the expected current by summing over all the cases.

The event duration, τ, is chosen so that the area of current pulse equals the expected charge delivered to the capacitor. Since the expected charge is the total charge on the node times the probability the node changes state, and the current pulse is triangular, τ must be:

$$\tau_p = 2 \times \frac{Vdd \; x \; C \; x \; P_{HL_i}}{i_{max_P}} \tag{38}$$

$$\tau_n = 2 \times \frac{Vdd \; x \; C \; x \; P_{LH_i}}{i_{max_N}} \tag{39}$$

For an inverter, τ is $2C/g_p$ and $2C/g_n$, which are the same time constants that would occur in a non-probabilistic simulation, as is the value of τ_p for a nor gate, $4C/g_p$. A nor gate's τ_n is more interesting. If only one of the devices is on, its value should be $2C/g_n$, while if both are on, the value should be C/g,. The actual value chosen will be somewhere in this range, depending on the relative probabilities of the two events:

$$\tau_n = \frac{2c}{g_n} \times \frac{P_{LH_{i0}} \times P_{L_{i1}} + P_{LH_{i1}} \times P_{L_{i0}} + P_{LH_{i0}} \times P_{LH_{i1}}}{P_{LH_{i0}} \times P_{L_{i1}} + P_{LH_{i1}} \times P_{L_{i0}} + 2 \times P_{LH_{i0}} \times P_{LH_{i1}}} \tag{40}$$

This simple example ignores some of the trickier problems involved in a probabilistic approach. Propagating the probability waveforms through the circuit can be done in linear time only if a gate's inputs are independent random variables; when there is reconvergent fanout or feedback, the output probabilities can only be calculated by exhaustively enumerating the inputs. To avoid this operation, CREST includes some heuristics to propagate probabilities in dependent sections of the design. CREST can also handle designs that contain circuits more complicated than static gates, such as pass transistor networks.

Probability waveform analysis has both advantages and disadvantages. It avoids the overestimation problems of timing analysis, and for phenomena that are substantially dependent on the expected current, such as electromigration, it provides a useful current estimate. For circuit failures dependent on the peak current, however, this approach fails. For the decoder of Figure 32, CREST will equally divide the current peak caused by a single decoder output changing value among all the decoders, assuming that all the outputs are equally likely to be high. Any problems associated with this large peak current will not be detected.

3.3 Switch Level Simulation

The approach I adopted is based on switch level simulation. A switch level simulator represents a circuit as a set of nodes interconnected by transistors and resistors, with capacitance only to ground. For MOS circuits, the transistors act as voltage-controlled resistors; a conducting transistor connects its source and drain via a nonlinear resistor. MOS simulators typically model transistors as bidirectional switches; "closing" the switch connects the two nodes via a resistor. Some examples of this type of simulator are Bryant's *MOSSIM II*[6] and *COSMOS*[7] and Terman's *Rsim*[57].

Deriving current pulses from such a simulator, described in detail below, is fairly simple; whenever a node in the circuit changes value, a pulse is added to the circuit current profile to represent the current required to charge or discharge the node capacitance. This approach has several advantages. It avoids the overestimation inherent in timing analysis; in a given cycle, only one of the mutually exclusive paths shown in Figure 32 will be exercised in a given cycle. It also gives a better estimate of the peak current than a probabilistic simulator because a pulse's magnitude is not reduced by its probability of occurrence.

A switch level current estimator also has a significant disadvantage: pattern dependence. The current profile generated depends strongly on the input vectors applied to the circuit. Figuring out which pattern gives the worst voltage drop would require trying all possible input combinations, which is infeasible for all but trivial circuits. Despite this drawback, a pattern dependent current estimate is useful; it gives the designers some idea of a circuit's behavior under real operating conditions, and allows them to experiment with patterns that they feel may cause noise or electromigration problems.

I use Rsim as the basis for my current estimator. It has several advantages: since it is the most commonly used switch level simulator in the Stanford design environment, designers are familiar with it. Before doing power analysis, the user will already have a design simulating correctly under Rsim. The individual quirks inherent in any simulator will already have been surmounted, and little additional effon will be required to generate a current profile from the design. Rsim also correctly models some of the more subtle MOS timing effects: Horowitz[25] has modified it to include the effects of input slope

and RC trees, and Chu[12] has included improved charge sharing models.

3.3.1 Implementation

This section outlines the steps that Rsim performs when a node changes value, together with the modifications made to support current estimation. Following this is the operation of the algorithm on an example circuit, and a discussion of some of the estimator's limitations and some solutions for them.

The basic current estimation algorithm is:

1. When a node changes value, check all the devices with attached gates. At the source and drain of each such device, trace out *channel-connected clusters*, which are sets of nodes connected together by conducting transistors.

2. In each cluster, walk through the graph of conducting transistors, calculating four things:

   (a) The equivalent resistance between each node and the two power supplies.

   (b) The delay should the node change value. This delay is estimated using the Penfield-Rubenstein-Horowitz algorithm[50].

   (c) The total charge in the cluster. When a transistor turns on, its source and drain nodes redistribute their charge so that they are at the same potential. This charge sharing can change the values of the nodes.

   (d) The locations of conducting transistors that connect the cluster nodes to the supplies. These points will later be used to place the current pulses.

3. Check to see if charge sharing between nodes in the cluster will change any of their values. If it will, schedule an event, which is a record that the node will change value at some point in the future.

4. Calculate the final value for each node using the equivalent resistances to power and ground. These two values form a resistive divider; if the value between them is below a low threshold, the final value will be low, if it is above a high threshold, it will be high, and if it is between the two, it will be undetermined (X).

5. If the node's final value differs from its present one, or from the one it will assume after the charge sharing event is processed, schedule another event. The timing of this event depends on the delay calculated in Step 2b.

6. When the simulation reaches the event time, the node is updated and a pulse is added to the circuit's current profile. This entire procedure is then repeated with the devices whose gates connect to the event node.

For the example circuit of Figure 35, one cluster, consisting of nodes Out and n2, is created when Input A rises. Before the input changed, n2 was at ground potential and **Out** at **Vdd.** After M_4 switched on, **n2** will rise slightly, but not enough to change state because it is pulled down by devices M_5 and M_6. This slight rise is ignored by Rsim. The equivalent resistances to ground that Rsim calculates for the two nodes are:

$$R_{equiv_{Out}} = R_4 + R_5 R_6/(R_5 + R_6) \tag{41}$$

$$R_{equiv_{n2}} = R_5 R_6/(R_5 + R_6) \tag{42}$$

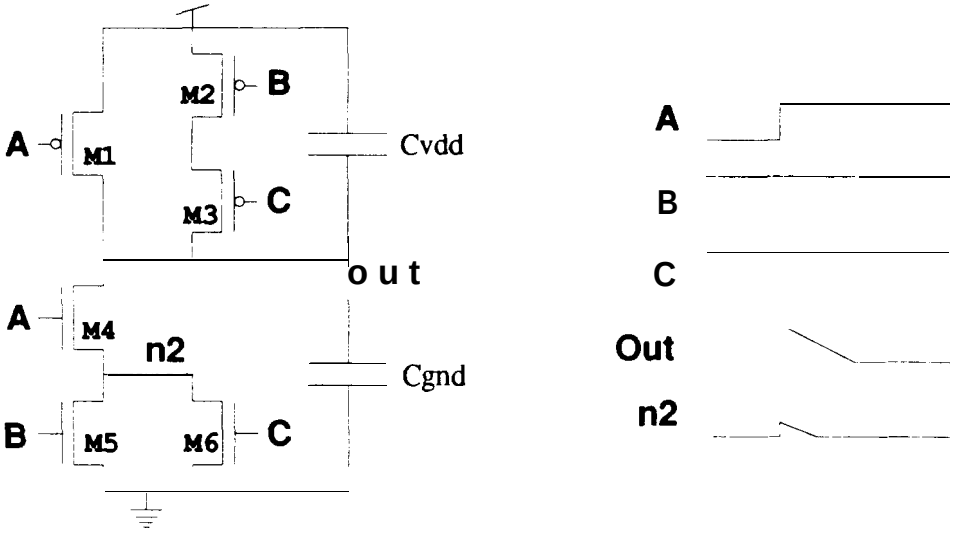


Figure 35: An Example And-Or-Invert Gate

With the p-channel **pullup** network off, the resistance to **Vdd** is infinite and the final values for both nodes will be low. Since **n2** is already low, no event is scheduled for it.

Because **Out** is currently high, Rsim adds an event, scheduling it to fall τ_f time units in the future.

$$\tau_{f_{Out}} = C_{out}(R_4 + \frac{R_5 R_6}{R_5 + R_6})\ln(2) \tag{43}$$

Assuming there is no other activity in the system that will cause the events to be aborted, at time $t_{current} + \tau_f$, the value of **Out** will be changed, and the event will be converted into a current pulse. As noted in Section 3.1, different pulses arise depending on the supply to which the bottom plate of the capacitor is connected. For the falling transition of node **Out,** the currents are shown in Figure 36: the fraction of the capacitance whose bottom plate is connected to Vdd is charged from the power supply, and the capacitance to ground is discharged locally. The current pulse injected into the lower supply via the pulldown network has a duration τ (derived in the next section), an area $(C_{vdd} + C_{gnd})V_{out}$, and network entry points P1 and P2, which are the sources of *M5* and *M6*. Estimating the currents flowing into the bottom plates is much more difficult; the lower terminals for the two capacitors, nodes P3 and P4, are distributed regions, not single points. These currents flow through substrate and well plugs and across the surface of the chip to the areas adjoining the wires of node **Out.** To calculate this image current exactly, the well/substrate resistance from each plug to the capacitor structure would have to be extracted, and the resulting system solved to see how much current goes to each plug. Because this would be prohibitively slow, the current estimator uses a simpler method; it divides the current evenly between all plugs in the vicinity of the node. Section 3.3.3 gives a more detailed description of this approximation, together with an investigation of its accuracy. For the event in the example, the currents generated are shown in Figure 37.

This example glossed over many of the details and limitations of the current estimator. The next sections describe some important aspects in more detail: the first describes how a current pulse is fashioned from an event, the second investigates the effects of image current, and the third discusses some limitations of this approach.

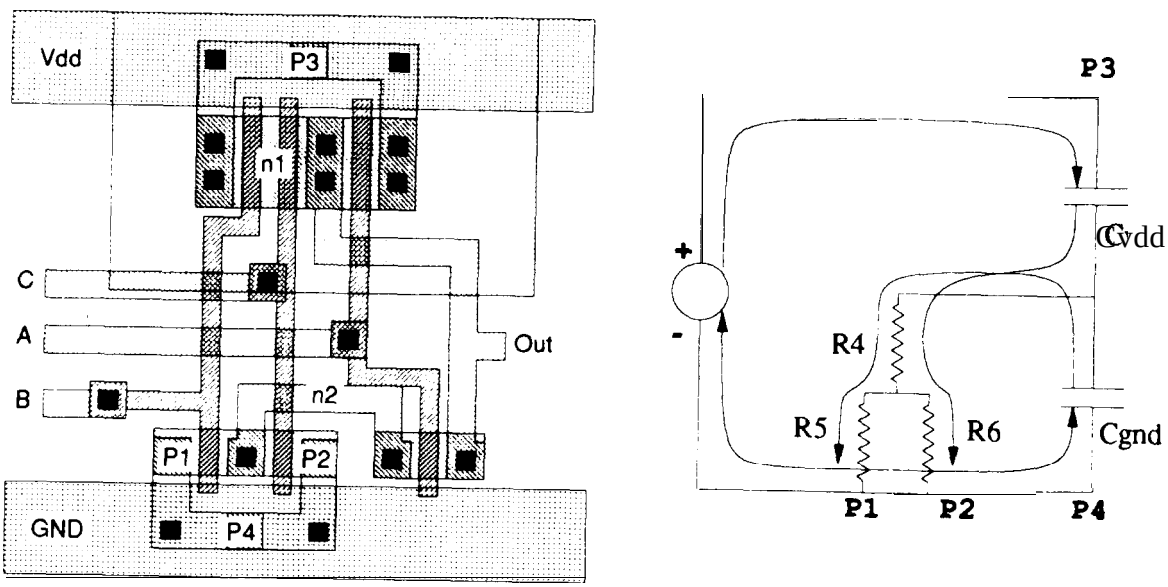


Figure 36: Charging Paths for And-Or-Invert Gate

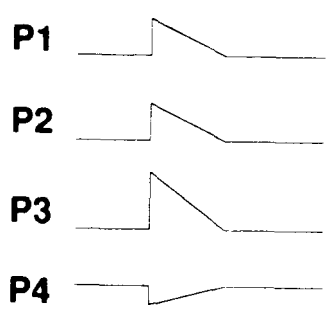


Figure 37: Current for And-Or-Invert Gate

3.3.2 Current Waveform Generation

The current pulses generated by timing verification and probabilistic simulation are of necessity somewhat approximate because the shapes of the underlying voltage waveforms are not well known. Rsim can do a better job shaping the current waveforms because it has more information, particularly about the slopes of inputs. The shape of the current waveform generated by a node transition is affected both by the slope of the node's voltage waveform and by the slope of the input signal. The current pulse that Rsim produces for an event should reflect this dependency.

Horowitz[25] has modified Rsim to include the effects of input slope. He models the transition of a gate's output in two parts. Initially, the gate current is a linear function of the input voltage; this produces a quadratic output voltage waveform:

$$Vout(t) = 1 - \frac{t^2}{2(\tau_{in}\tau_f)/(g_m R_L)} \tag{44}$$

In Equation 44, g_m is the input transconductance and R_L is the output resistance. τ_{in} is the time required for the input to rise from 0 to 1, while $\tau_f = R_L C_L$ is the intrinsic delay of the gate.

Once the output voltage reaches the voltage drop across the output resistance, $Vout = i_{out} R_f$, the current becomes independent of the input voltage, and is dependent only on the output resistance and capacitance. This model produces a decaying exponential for the second part of the waveform.

$$Vout(t) = (1 - \frac{t_s^2}{2(\tau_{in}\tau_f)/(g_m R_L)})\epsilon^{(t_s - t)/t_f} \tag{45}$$

The transition between these two regions occurs at τ_s:

$$\tau_s = \tau_f(\sqrt{1 + \frac{2\tau_{in}}{\tau_f g_m R_L}} - 1) \tag{46}$$

Since the output current is just $I_{out} = C_{out} dV_{out}/dt$, one possible approximation for the output current would be to take the derivatives of Equations 44 and 45. The disadvantage of this method is that the waveform is somewhat complex to manipulate; the rising edge is a ramp and the falling is an exponential. To avoid this, the current

estimator approximates the current waveform as a triangular pulse of equivalent duration and equal area (Figure 38). The rising portion of the pulse occurs while the current is a function of the input voltage, which occurs from $t = 0$ to $t = t_s$. After t_s, the current is a decaying exponential. The integral of an exponential is $\int_0^\infty A\epsilon^{-t/\tau_f} dt = A\tau_f$; for a triangular pulse of the same height to have the same integral, its duration must be $2\tau_f$. With the pulse's base width fixed at $(t_s + 2\tau_f)$, the height must be set to make the integral equal to the charge which changed state:

$$I_{peak} = 2C_{load}V_{dd}/(t_s + 2\tau_f) \quad (47)$$

The only remaining consideration is situating the pulse correctly in time. Assuming the event occurs when the output has reached the halfway point, the pulse is is placed so that its integral equals $C_{load}V_{dd}/2$ at the event time. This gives equal area to the two stippled regions in Figure 38.

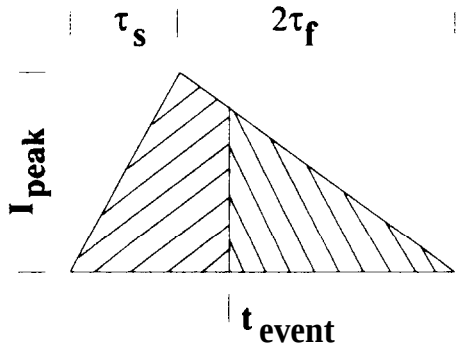


Figure 38: Current Pulse Generated for an Event

The estimated and actual cut-tents for an inverter under some representative conditions are shown in Figure 39 and detailed in Table 5. The current estimator's performance is fairly good for the first three cases, where the output delay is equal to or greater than the input delay; the root-mean-square error is between 15 and 20 percent. The poor performance in the final example, where the input is much slower than the output, is primarily due to the short-circuit current, which is not included in the pulse generation. Fortunately, of the four cases, the last is of the least interest; such gates are rare because they are severely oversized for their loads. Figure 40 shows the relative importance of

various input/output slope pairs for a typical MIPS-X simulation run. The graph on the left shows the number of events occuring for each slope ratio; the graph on the right weights these events by the capacitance of the node changing state. Unsurprisingly, the most common occurance is for the input and output slopes to be approximately equal. These events, along with the transitions of large, heavily-loaded drivers which appear in the rightmost column, dominate the power dissipation.

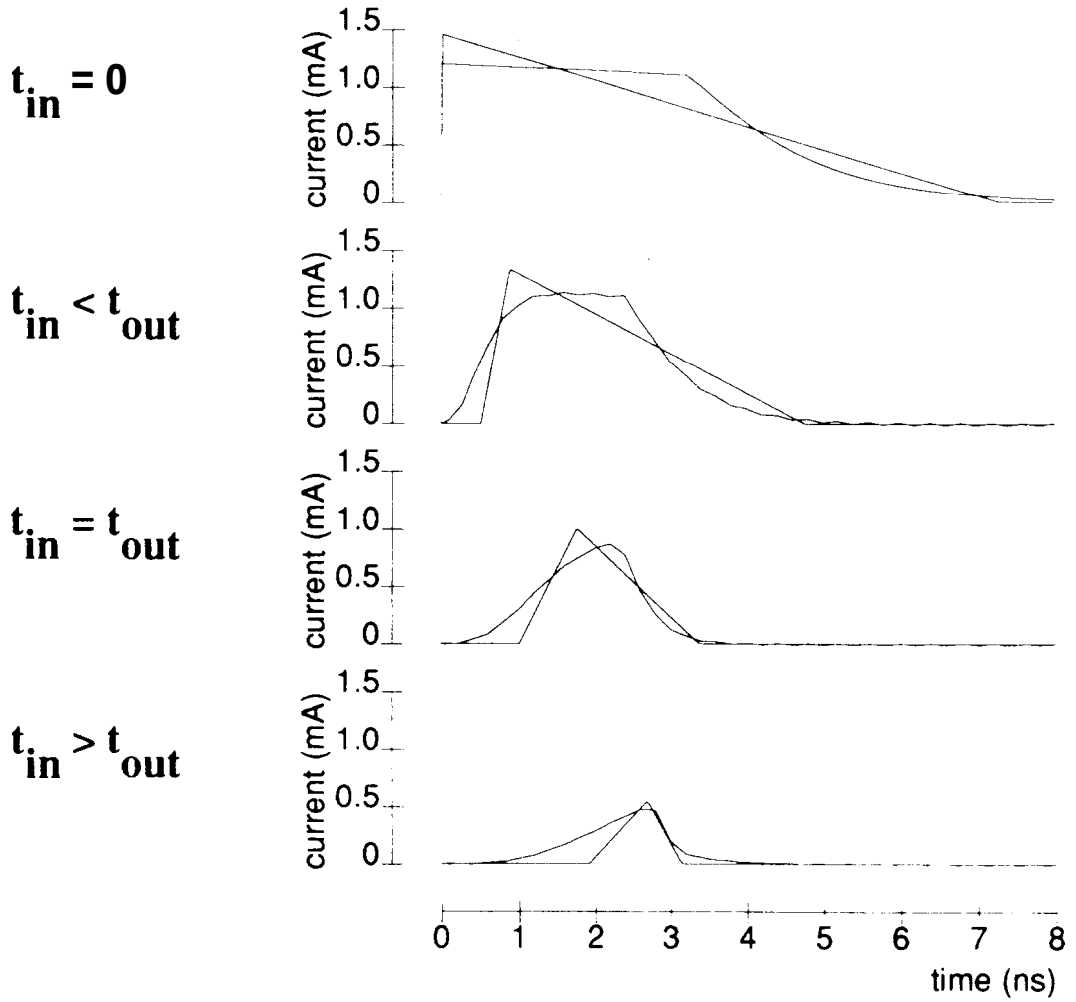


Figure 39: Pulses for Various Input and Output Slopes

Waveform	Input Risetime	Output Delay	Spice Peak Current	Spice Time of Peak	Rsim Peak Current	Rsim Time Peak	of rms error
A ($t_{in} = 0$)	0	2.3ns	1.2mA	Ons	1.5mA	0	0.16mA
B ($t_{in} \ll t_{out}$)	0.4ns	1.5ns	1.1mA	1.6ns	1.3mA	0.8ns	0.18mA
C ($t_{in} \approx t_{out}$)	1.Ons	1 .Ons	0.9mA	2.2ns	1.0mA	1.8ns	0.15mA
D ($t_{in} \gg$ tout)	2.0ns	0.5ns	0.5mA	2.6ns	0.5mA	2.7ns	O.11mA

Table 5: Comparison of Current Pulses for Various Input and Output Slopes

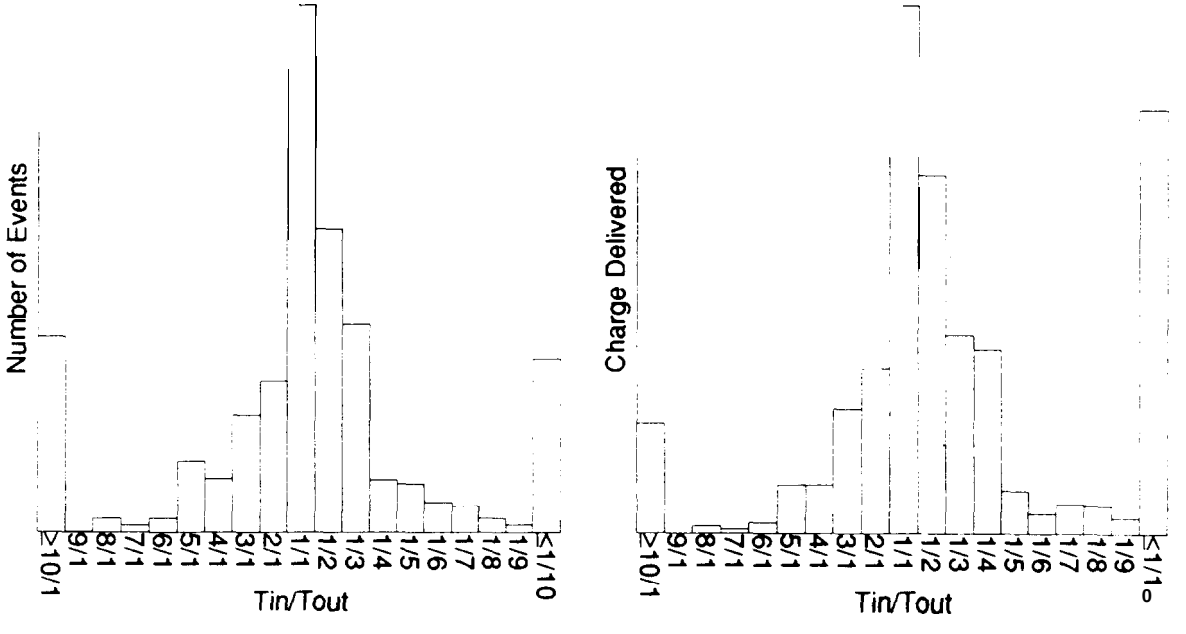


Figure 40: Typical Input/Output Slope Distribution

3.3.3 Image Currents

For the current profile to be accurate, the image currents, which enter the power network through well and substrate contacts, must be included. In general, this current distribution is quite complex; for each element of capacitance, the image current will split between various well and substrate plugs depending on the locations of the plugs and the topology of the wells. Information about this distribution is lost by the circuit extractor, which reduces the capacitor's distributed bottom plate to a single node.

Despite the apparent difficulty, it is important to at least estimate these currents. Figure 41 shows the results of ignoring these currents for two examples: the register file from MIPS-X [24] and a self-timed divider [60]. These scatter plots compare the actual node voltages (as calculated by the second algorithm below) against the values obtained if image currents are ignored. Ignoring the image currents gave errors of up to 35% in the examples tested.

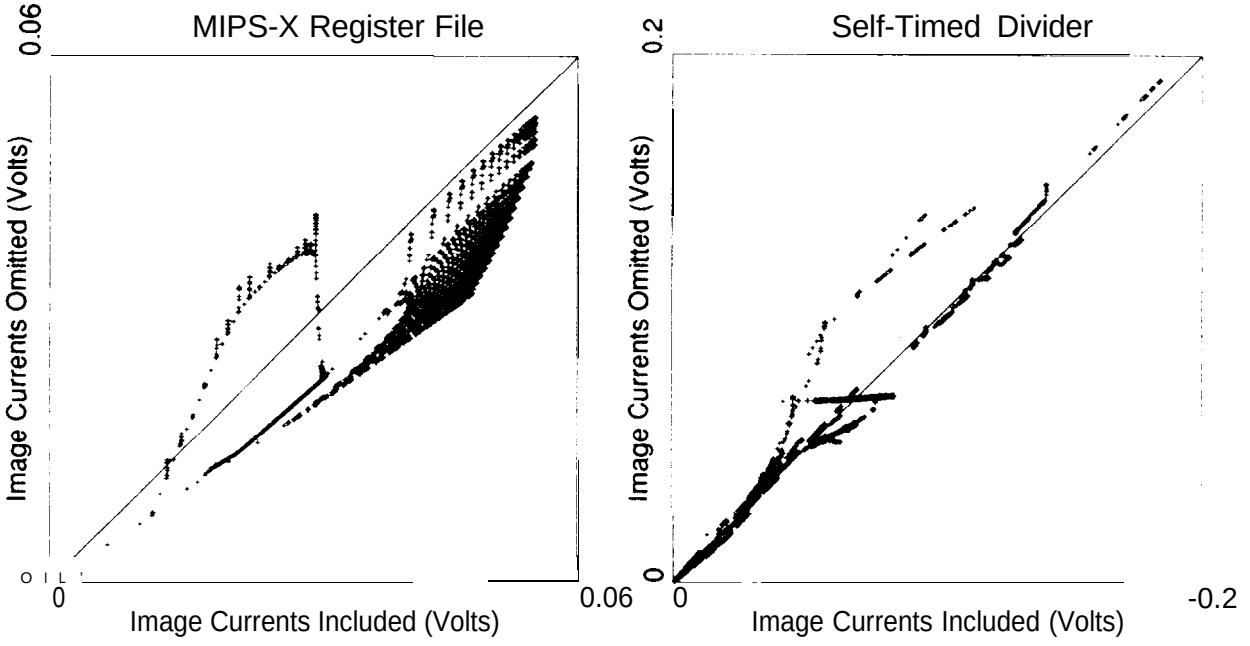


Figure 41: Effects of Ignoring Image Current

The current estimator uses a simple bounding box algorithm to decide how the nodes' image currents divide. For each node in the circuit, the estimator forms a bounding box around all the rectangles that compose the node, grows this box by a small, fixed

distance, and searches this area for substrate contacts of the correct type. Whenever the node changes value, the estimator evenly divides the substrate current between all the plugs in the box. If no plugs are found initially, the estimator doubles the box size and repeats the search.

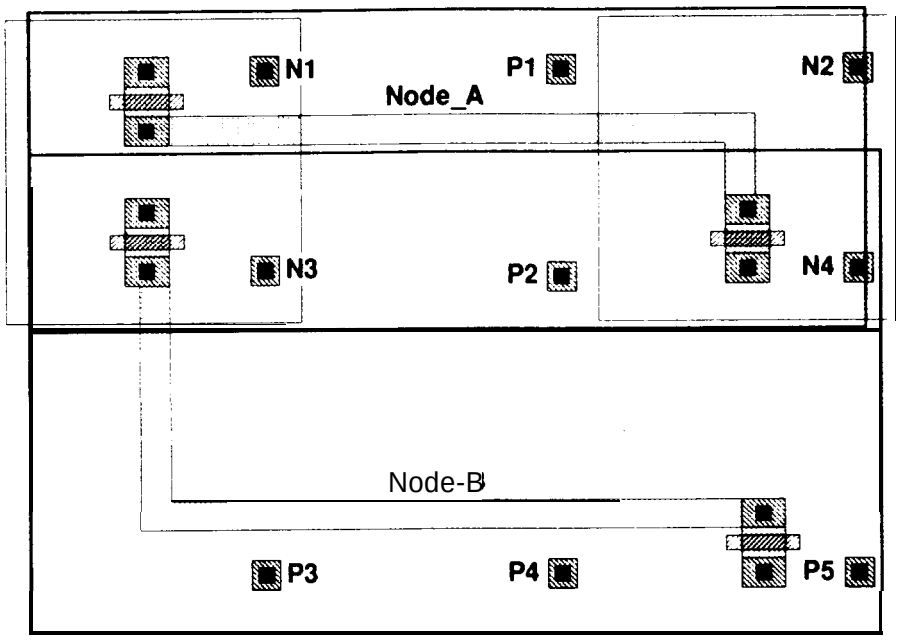


Figure 42: Bounding Box Current Estimation

This algorithm is easy to implement in Magic's hierarchical, comer-stitched database. The current estimator creates a special plane, into which it copies all the plugs encountered during resistance extraction. When the regular extractor, which hierarchically calculates connectivity and capacitance, processes a cell, it forms a bounding box for each node. During the conversion of this hierarchical extracted description into a flat netlist suitable for simulation, the bounding boxes are also combined to form a single box for each node. Using Magic's standard area searching routines, the estimator can find each node's plugs by searching the plug plane using this box.[2]

*This special plane is actually redundant; the temporary cell created to hold the flattened bus already has copies of all the plugs. However, the time to search a given area in a comer-stitched database is basically proportional to the numbers of items that it contains; this regular plane that contains plugs also contains transistors, polysiticon, and diffusion. Using a separate plane speeds up searching by drastically reducing the number of tiles that the current estimator has to traverse.

For some nodes, like Node A in Figure 42, this approximation is a fairly reasonable one. The estimator spreads the current between plugs near the node, in this case **Nl-N4** and **P1-P2**, which mimics what actually happens. The approximation is less accurate for nodes containing 'doglegs', such as node **B.** Although the bounding box encloses node N4, no current from **B** will enter the bus there because the node does not extend over **N4**'s well.

To test the validity of the bounding box algorithm, a more accurate estimate of the image currents is needed. One method would be to calculate for each point in the substrate how current injected there would divide between the various plugs. The fraction of a node's current that will flow to a given plug could then be set by integrating the capacitance of each element of the node times the fraction that flows to the plug from the points under that element. By repeating this for all plugs and all nodes in the circuit, the estimator can get an accurate value for the entire current.

Calculating how an injected current would split from each point in the substrate could be done using finite element techniques, but this would be extremely slow and memory-intensive, and would preclude comparison on any but the simplest of examples. Instead, the extractor assigns each point in a nwell or pwell region to the plug nearest it, as shown in Figure 43. The estimator then re-extracts each node in the circuit and subdivides the image current based on the capacitance to each plug's subregion.

Even with the simplification on substrate subdivision, this method is still too memory-intensive to run on an entire design- Instead, I ran it on some smaller circuits: MIPS-X's register file (10159 transistors), and the self-timed divider (5777 transistors). A comparison of the bounding box and t-e-extraction algorithms is shown in Figure 44. In both cases, the agreement between the two methods is quite good; it is better in the register file because it contains fewer 'dogleg' wires.

3.3.4 Coupling Capacitance

Coupling capacitance is tricky to handle correctly because it is difficult to keep track of where the capacitor's bottom plate is connected. In Figure 45, a falling transition on **Node out1** will give two different current distributions depending on the value of Node

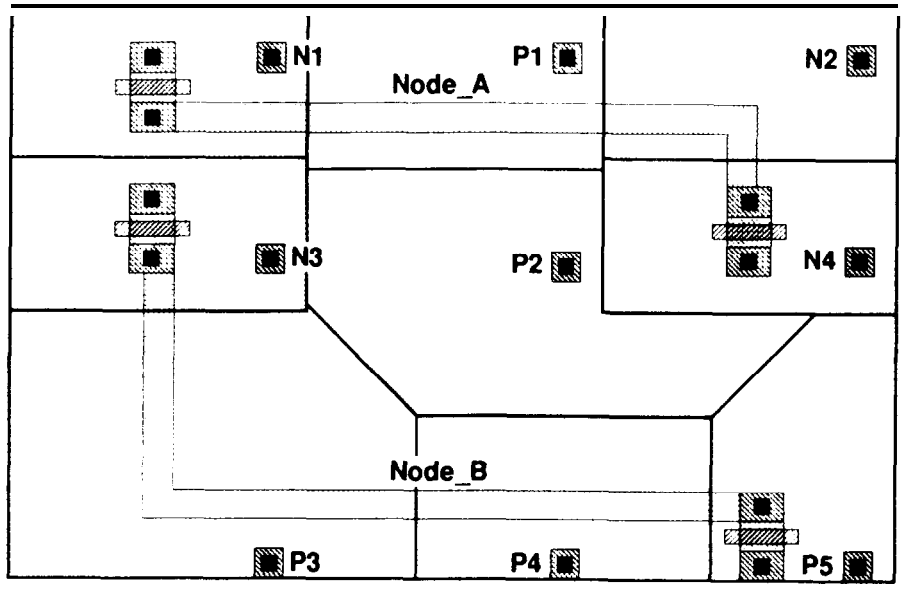


Figure 43: Re-Extraction Current Estimation

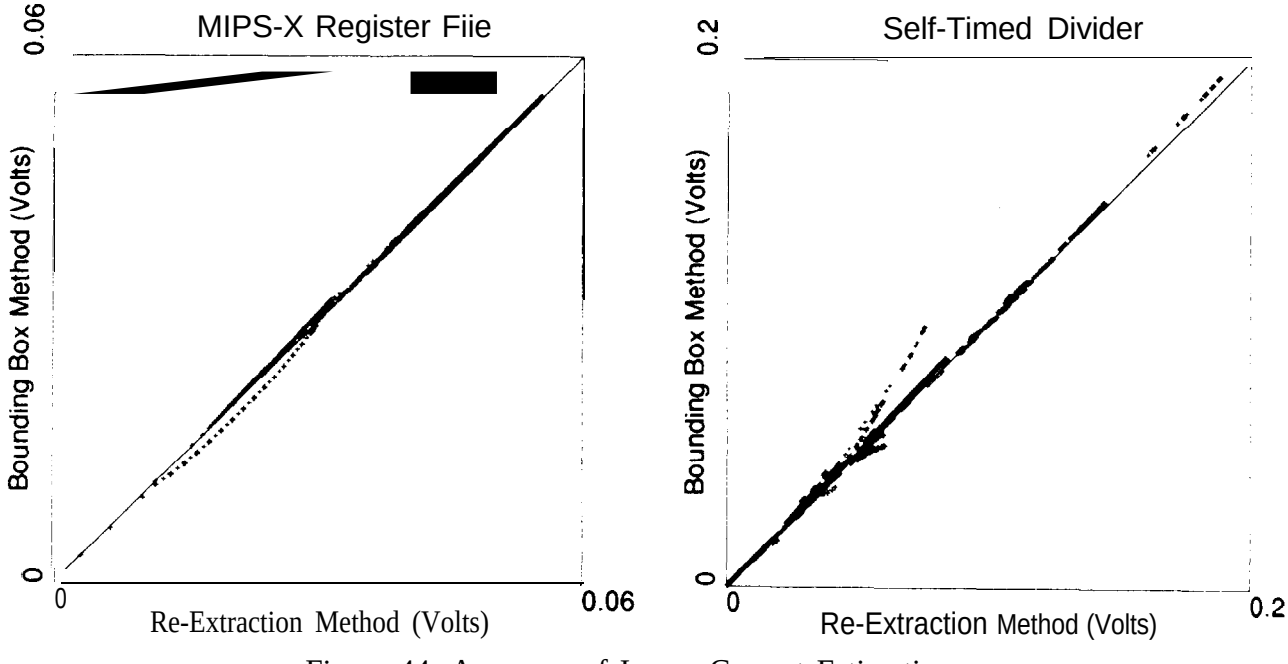


Figure 44: Accuracy of Image Current Estimation

out2. If **out2** is low, then the capacitor is discharged in a local loop; if **out2** is high, then the capacitor is charged through the power supply.

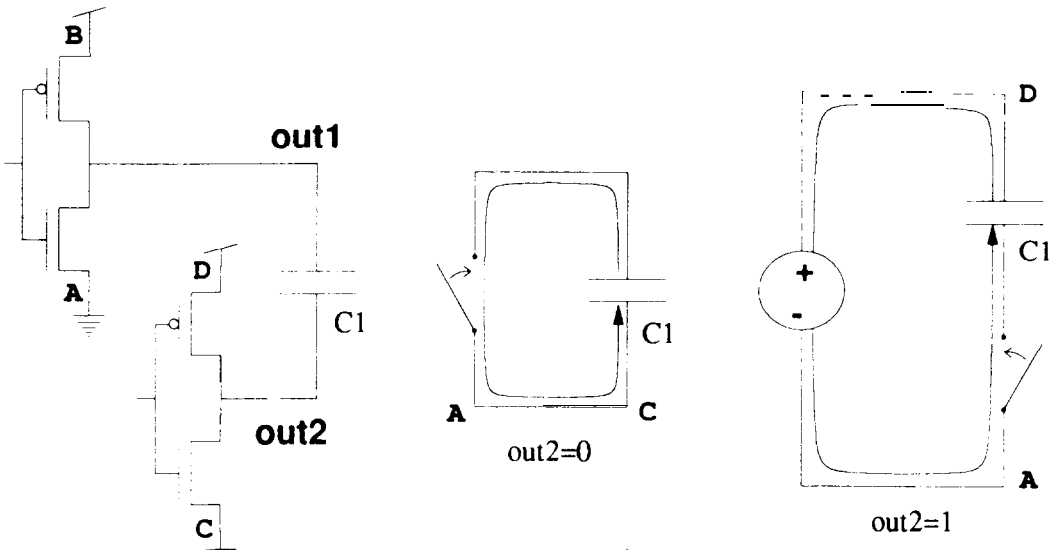


Figure 45: Effects of Coupling Capacitance

The relative importance of coupling capacitance varies with the design being analyzed. For the three test designs, Figure 46 shows the division of the bottom plate capacitance for all the chip's signal capacitance. C_{GND} and C_{Vdd} are the portions of the capacitance whose bottom plate is connected to one of the power supplies, C_{pad} is the (estimated) off-chip capacitance that the chip outputs must drive, C_{couple} is the coupling capacitance between on-chip signals, and C_{gate} is the gate to channel capacitance.

I have listed C_{gate} separately because its behavior is fairly complex. It is a substrate capacitance when the device is off, a coupling capacitance to the source terminal when the device is saturated, and a coupling capacitance to both the source and drain when the transistor is in the linear region. The channel capacitance only couples the gate to a diffusion terminal when the gate-source or gate-drain voltage is greater than the threshhold voltage, which generally occurs when the source is being pulled low. This requirement virtually eliminates the Miller effect; the majority of the capacitance is to the transistor source, which is not changing value. Since the channel capacitance is generally either to a signal line that is being pulled to a supply voltage or to the substrate, the system treats it as ground capacitance for n-channel devices and Vdd capacitance for p-channel

devices. This misplaces the current pulse slightly; it will be injected through a substrate contact instead of a device source even when the device is conducting, but the resulting change in voltage distribution is typically a local perturbation.

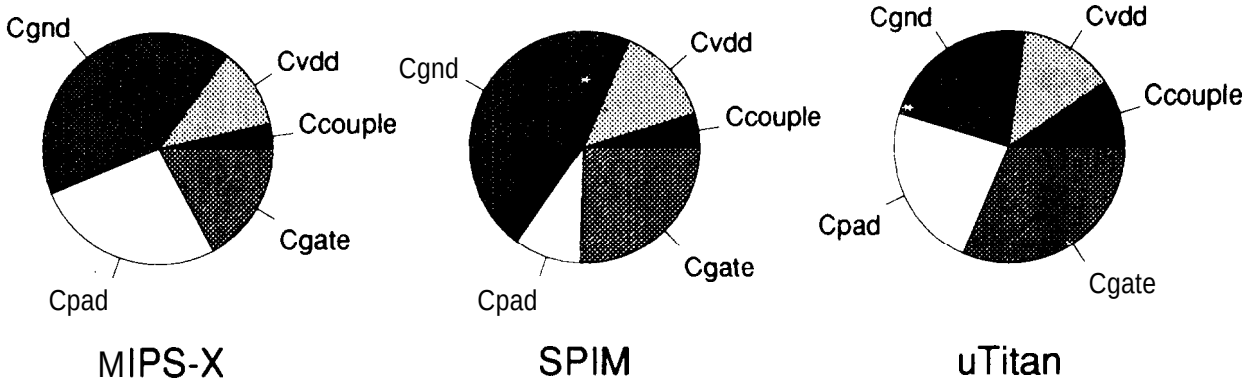


Figure 46: Distribution of Capacitance Bottom Plate

The non-channel coupling component varied from 3% to 10%, depending on the design. Estimating currents for these capacitors inside Rsim is possible, but very expensive. Rsim keeps track of the transistor pulling down a node only when an event is pending for the node; when the node reaches its final value, the simulator throws away the event, and with it the information about the driving transistor. To estimate the bottom plate current, Rsim would have to remember which transistors are driving each node in the circuit.

Even were the driving transistors for all the nodes known, calculating current pulses for all the coupling capacitors in the circuit would be extremely expensive. Table 6 shows the total number of coupling capacitors in the test circuits (the number in parentheses shows the fraction of the total capacitance that is of this type). Each transistor has a coupling capacitance between its gate and drain, and all transistors with source nodes not connected to a power supply have coupling capacitance between their gates and sources. When the large number of parasitic capacitors arising from interactions between wires are added to this total, the system of coupling capacitors is larger than the original circuit. Manipulating this bloated system would be extremely cumbersome, and the resulting current profile would be huge; in addition to the single top-plate pulse,[3] Rsim would

[3] There is actually more than one pulse per event because each node transition may give rise to multiple image pulses. However, these image pulses are added inside Magic, and no additional computation from the simulator is required.

have to add an additional pulse for each coupling capacitor. Since the total coupling capacitance is not that large, the estimator instead makes two simple approximations. First, the gate-drain parastic capacitance is doubled to account for the Miller effect, as is the gate-source capacitance for devices not collected to a supply. Coupling capacitances for the wire parasitics are not doubled because transitions for their two terminals are much less likely to be correlated. Second, the estimator assumes that the capacitor bottom terminal is always at the opposite potential from the top terminal, so that a charging current is generated. This produces larger voltage drops than would a discharging current because charging currents go through the power supply.

Capacitor Origin	MIPS-X	SPIM	uTitan
Transistor Gate-Drain	47136 (1%)	41804 (1.5%)	179390 (4%)
Transistor Gate-Source	15333 (0.3%)	10364 (0.3%)	62433 (1%)
wire Parasitics	76538 (2%)	72585 (3%)	260311 (4.5%)
Total	139007	124753	502134
Nodes	17430	16614	61468

Table 6: Comparison of Nodes and Coupling Capacitors in Test Circuits

3.3.5 Charge Sharing

In a CMOS circuit, not all node transitions are caused by the connection of nodes to the power supplies; sometimes the charge necessary to change a node's value comes from another node, as shown in Figure 47. Although the top plate current comes from another node, the substrate current still comes though the supply network. These *charge sharing* events present different problems for the current estimator than do regular driven events.

To estimate how important estimating charge sharing currents is to power estimation, I modified Rsim to record the type of each event when it is scheduled. In Chu's modified version of Rsim[12], there are three types of events: normal driven events, pure charge

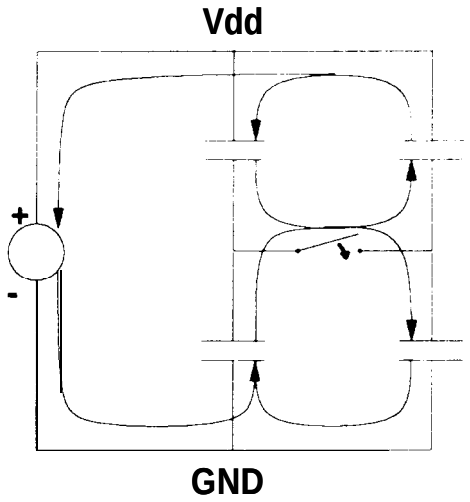


Figure 47: Currents in a Pure Charge Sharing Event

sharing events, and charge sharing with a driven path. The first two types have been discussed previously; the third (Figure 48) is a hybrid of the first two. In this case, the network is composed of two components: a *driving free* and a *charging tree.* When the two trees are connected, nodes in the driving tree will experience a momentary glitch as their charge is shared with charging tree nodes.

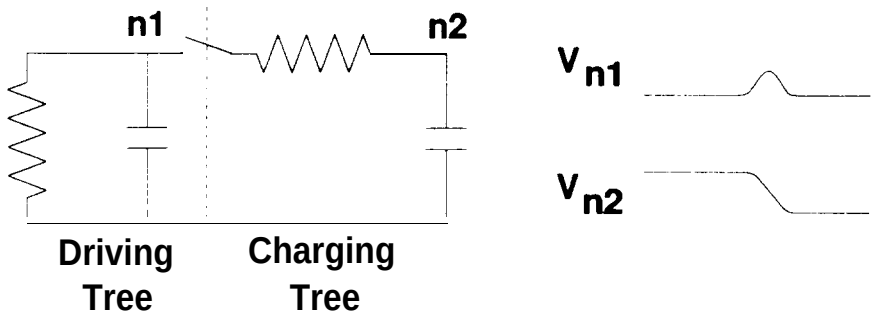


Figure 48: A Driven Charge Sharing Event

The relative importance of the three event types is summarized in Figure 49 for the three test cases. Each section represents the magnitude of the charge transferred during events of the given type. There is a large variation in the importance of charge sharing events; in the multiplier SPIM, the total is small because the design uses static logic throughout. The total in MIPS-X is considerably larger, primarily due to a single

circuit.[4] Although charge sharing does not generally cause a significant amount of the total current, in certain special cases it can be reasonably large.

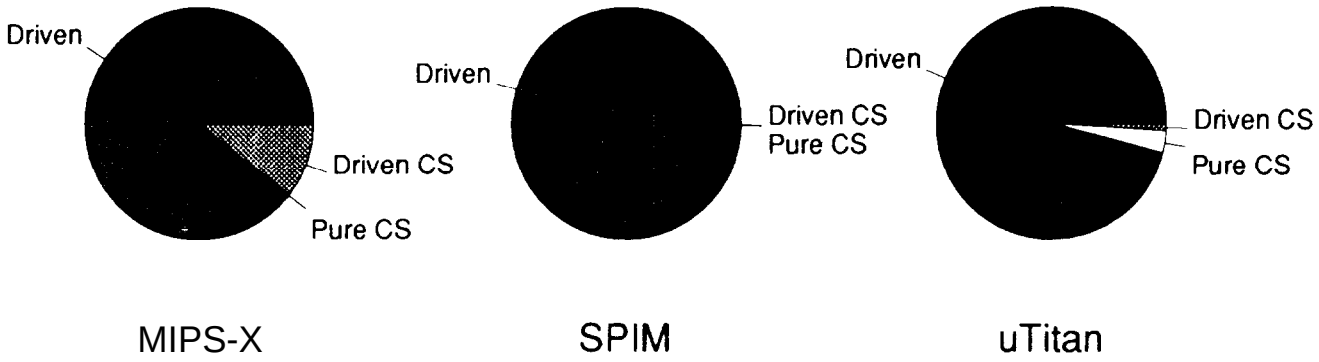


Figure 49: Relative Importance of Charge Sharing Events

Estimating charge sharing currents inside Rsim is easy to do because Chu already calculates the delay for these events. For pure charge sharing events, the node voltage waveform is approximated by a single exponential derived from the circuit topology and the initial charge distribution. A node with time constant τ_e and amplitude V_{peak}, has a corresponding triangular[5] current pulse of duration $2 * \tau_e$ and height $V_{peak} * C_L$.

Driven path charge sharing is more complicated; because the node voltage begins and ends at the same value, it cannot be modeled by a single exponential. Instead, node voltages are represented as the difference of two exponentials. The fast component is set by the time required to share the charge between the driving and charging trees, while the slow component is set by the time required to return the driving node to its original value. The current estimator adds two pulses of equal area for these glitches, governed by the fast and slow time constants in the voltage pulse. The voltage magnitude is a function of these two constants, and derived by table lookup; the peak of the current pulse is just this voltage times the node capacitance.

[4] The address tags contain an exclusive-or circuit that produces a driven path charge sharing glitch; since this circuit is repeated 800 times, the total glitch current can be fairly large[11].

[5] Since Chu's charge sharing model does not include the effects of input slope, the pulse is always a right triangle.

3.3.6 Glitches

Glitches, changes in a signal that do not extend from one supply to the other, are difficult to estimate in a Boolean timing simulator. The nor gate of Figure 50 shows one such example. When Input A falls, Node **out** begins to rise. Before it reaches its final value, however, Input B rises, causing **out** to fall again. Inside Rsim, an event is scheduled for **out** after A falls; before the event time reaches the simulation time, B rises and the event is deleted from the queue. The simulated waveform for node **out** never changes value, and no current pulse is generated for the event.

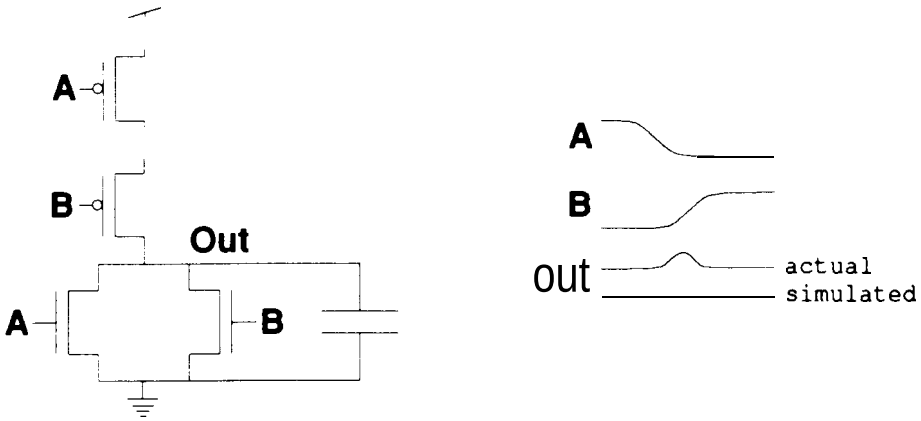


Figure 50: Effects of a Node Glitch

To estimate the importance of this effect, I modified Rsim to keep track of the number of events deleted from the queue and to estimate the amount of charge that these deleted events contain. The charge estimate was produced by integrating each current waveform from its beginning up to the time that the event was deleted. The results are shown in Table 7. Although the number of events deleted can be fairly high, the amount of current that they represent is quite small, indicating that most are removed soon after scheduling. Since they represent such a small fraction of the total current, the system can safely ignore them.

Design	Percentage of Events Deleted	Percentage of Charge Deleted
MIPS-X	14%	1.2%
SPIM	7.6%	
uTitan	7.0%	0.25%

Table 7: Importance of Glitch Currents

3.4 Performance

Current profiles for the three examples were obtained by applying test scripts supplied by the designers to the circuits. Each script initialized a circuit's state, enabled current profiling, then exercised the design using some representative patterns. Table 8 compares the elapsed time in seconds for these runs with those of an Rsim binary that does not contain logging.[6] The degree to which logging slowed down execution varied considerably with the test patterns. The most elaborate script, used on the uTitan design, spent a much larger fraction of its time setting up the state of the machine prior to logging than did the other two. Since the logging module adds little overhead when it is not enabled, the relative cost of logging is lower.

Circuit	MIPS-X	uTitan	SPIM
Without Logging	198s	841s	71s
With Logging	287s	899s	100s
Logging Cost	45%	6.9%	41%

Table 8: Rsim Running Times for Test Circuits

Table 9 shows where Rsim spends the extra running time of logging. For all three designs, the extra time is dominated by simply writing the current profile to disk. The row marked "Other" represents degradation in performance due to memory effects; the logging version of Rsim has a larger working set.

[6] The tests were run on a Decstation 5000/200, which contains a 25Mhz R3000 CPU, 64Kb instruction and data caches, and 128Mb of memory. Both binaries were compiled with optimization level "-04".

Circuit	MIPS-X	uTitan	SPIM
Tracing of Currents	4.5%	2.0%	3.0%
Pulse Shape Calculation	13.5%	10%	16.5%
Writing Log File	71.5%	57%	51.0%
Other	10.5%	31%	29.5%

Table 9: Time Spent in Various Operations During Logging

Although logging slows down Rsim markedly, the degradation should not preclude producing a current profile for any circuit which Rsim can handle. Since the standard methodology is to run Rsim without logging until the circuit is functionally correct, then to rerun the same script with logging enabled, adding current estimation does not lengthen the crucial modify-simulate-debug cycle.[7]

The raw events produced by Rsim need considerable massaging before they are fed to the linear solver. This is done inside Magic during resistance extraction. Each current pulse is matched up with the correct transistor and plug nodes in the resistor network and scaled by the fraction of capacitance to each supply. Once the pulses are matched up and subdivided, their current is summed for all the transistors and plugs, and written to a file for each user-specified time interval. The fraction of time required to perform these operations is shown in Table 10. The variation in time between designs is dominated primarily by the length of the pulse log. The one exception is the ground bus for μTitan; since the design does not have front side substrate contacts, the system does not have to create separate image pulses for each event.

Despite the complications presented by pattern dependence and image currents, a switch-level based estimator does a reasonable job producing current profiles for the system. Rsim is not unacceptably slower during current logging, and the log produced can be converted into a profile which correctly matches the resistance network in a modest amount of processing time.

[7] This is not quite true because an Rsim binary containing logging runs about 5% slower even with logging disabled due to slightly larger data structures. If this 5% is important, however, two binaries can be used, one with the logging module and one without it

Circuit	MIPS-X gnd	MIPS-X vdd	SPIM gnd	SPIM vdd	uTitan gnd	uTitan vdd
Total Time	1761s	1788s	1315s	1787s	741s	3712s
Resistance Extraction	53.1%	32.8%	36.1%	31.7%	55.1%	22.7%
Image Plug Search	1.0%	0.4%	1.1%	1.3%	0.0%	0.3%
Reading Events	7.3%	6.4%	14.5%	10.6%	38.7%	7.4%
Sorting Events	1.9%	1.6%	2.7%	2.7%	1.7%	2.7%
Adding Pulses	22.3%	37.3%	36.4%	40.8%	1.4%	54.3%
Writing Pulses	14.4%	20.5%	9.4%	12.0%	2.2%	13.6%

Table 10: Current Pulse Processing Times

Chapter 4

Current Estimation for ECL

All streams run to the sea,
but the sea is not full;
to the place where the streams flow,
there they flow again.

Ecclesiastes I : 7

In the estimation of current for CMOS circuits described in the last chapter, the current distribution's time dependency was of paramount importance. Information about state changes in the circuit, distribution of nodal capacitance, and substrate topology had to be considered to produce a reasonable current pattern. Fortunately, the process is considerably less convoluted for ECL designs. In ECL, the magnitude of the current is relatively fixed; all the logic does is steer current through different branches of the circuit. As will be seen, this mode of operation simplifies current estimation considerably.

This chapter is divided into five sections. The first discusses the effects of supply noise on ECL circuits and contrasts them to the corresponding effects for CMOS. Next is a description of a simple current tracing algorithm suitable for estimating currents in most gates, followed by extensions necessary to handle mote specialized configurations. Following this is a section describing techniques for handling the pattern dependence of ECL circuits, and a discussion of the current estimator's performance on some large designs.

4.1 Introduction

Noise affects ECL circuits differently than it does MOS designs. Figure 51 shows the effects of voltage drops along both supplies. Drops along the top rail, Vcc, shift a gate's transfer curve down, thereby reducing the circuit's high noise margin. Drops along the bottom rail, Vee, between a bias generator a current source reduce the gate's signal swing, reducing the gate's low noise margin. The signal swing in ECL gates is set just large enough to accommodate these and other sources of noise; the designer would like to insure that the actual noise present is less that the margin allotted.

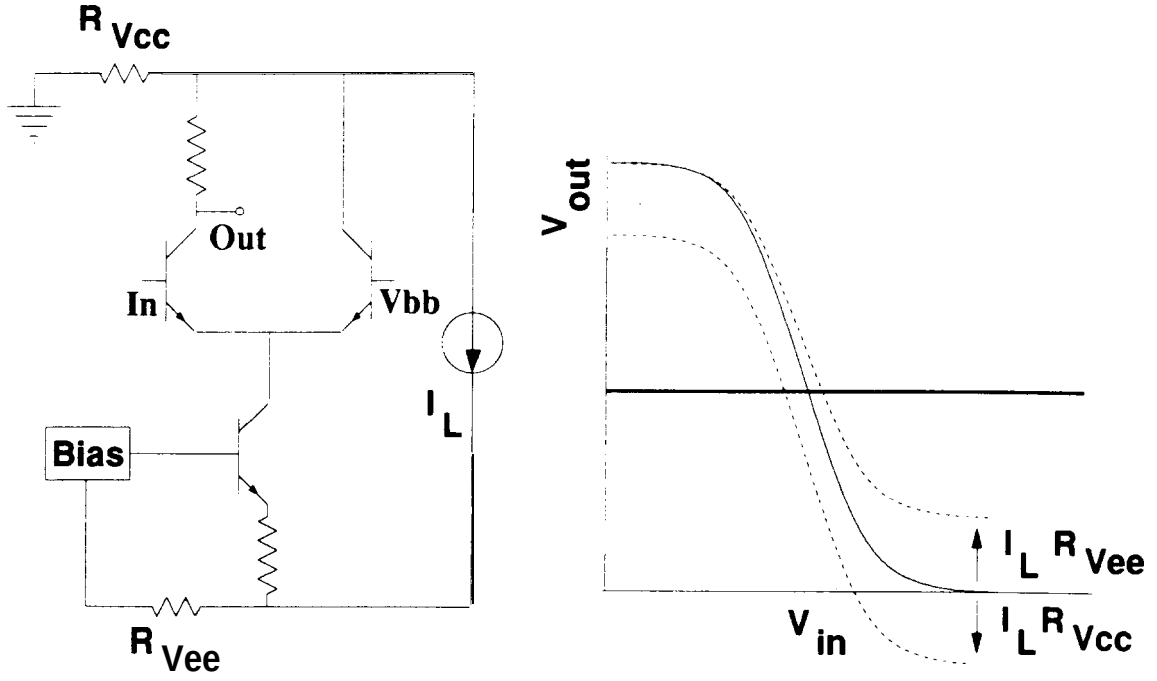


Figure 5 1: Effects of Noise on ECL Circuits

Figure 52 shows the current for a single gate. There are two components: the tail current I_{static} and the capacitive current I_{dyn}. In contrast to CMOS circuits, the capacitive component does not dominate in ECL designs. Equations 48 and 49 give the relation between the static and dynamic currents for an emitter follower during transitions. The static current controls the follower's fall time, while the dynamic current controls the rise time. In general, the designer will want the rise and fall times for the gate to be more or less equal, so the peak magnitude of the dynamic current will be roughly equal to that of

the staticcurrent. The follower output is not always changing state, however. Equation 50. gives the total dynamic output current. The first term in the equation describes the magnitude of the current when the output rises[1], while the second two terms give the number of transitions that actually occur.

$$I_{static} = \frac{C_L V_{swing}}{t_{fall}} \tag{48}$$

$$I_{dyn} = \frac{C_L V_{swing}}{t_{rise}} = \frac{t_{fall}}{t_{rise}} I_{static} \tag{49}$$

$$I_{dyn} = \left(\frac{t_{fall}}{t_{rise}} I_{static}\right)\left(\frac{\text{Cycles changing state}}{\text{Total cycles}}\right)\left(\frac{t_{rise}}{t_{cycle}}\right) \tag{50}$$

Although the second term in Equation 50, which is data dependent, may be unity for some perverse selection of inputs, the final term will be considerably less than one; an individual gate delay is substantially less than the cycle time of the system. This final term, combined with the fact that the dynamic and static currents from a gate changing state are roughly equal, explains why the dynamic current is less important than the static current for the system as a whole.

Although the magnitude of the current is relatively fixed, the current pattern for each gate still varies. In Figure 53, the current will enter the Vcc bus at either point P_1 or P_2, depending on the voltage at node In. In the example, these points are not very far apart, but in some circuits, such as a wired-or configuration or a decoder with shared current line outputs, they may be. For ECL circuits, producing a current pattern includes several tasks: calculating of the magnitude of the current, tracing the possible locations in the circuit where the current may enter the power network, and choosing between these locations.

4.2 Basic Current Tracing

The estimator's first tasks are finding all the current sources in the design, marking the locations where each current source connects to Vee, then tracing up through the network

[1] The formulation for a falling transition is similar, except that the total current through Vcc is reduced by the dynamic current, and its magnitude is always equal or less than the static current.

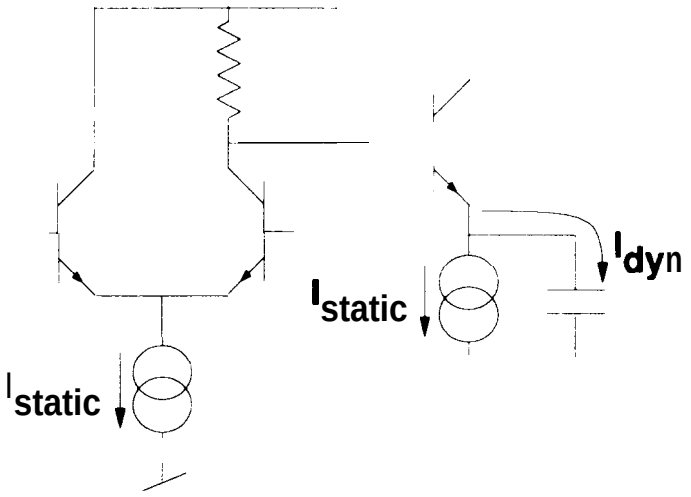


Figure 52: Currents for an ECL Gate

to find where these currents may enter Vcc. This section describes the basic current tracing algorithm; the next extends it to handle more complicated structures.

For simplicity, the current estimator makes two assumptions: it assumes that the diode drop V_{BE} is fixed for all transistors, and that the base current each transistor draws is zero. Since designers typically run all transistors at near the same current density to preserve the circuit's noise margins, and the base current for unsaturated transistors is a small fraction of the total, these approximations should not appreciably affect the estimator's accuracy.

Prior to tracing the currents, the estimator makes an initial pass over the network to assign the direction of current flow for resistors. The tracing algorithm assumes that current always flows through resistors in the same direction; from terminal 0 to terminal 1 in Figure 54. The estimator assigns a direction for each resistor using a set of rules similar to those Jouppi uses to assign signal flow direction in MOS circuits[26]:

1. The high voltage supply Vcc is always terminal 0.

2. The low voltage supply Vee is always terminal 1.

3. For all the resistors not connected to a supply, the estimator checks the other devices connected to each of the resistor's nodes. If one end has only emitter connections

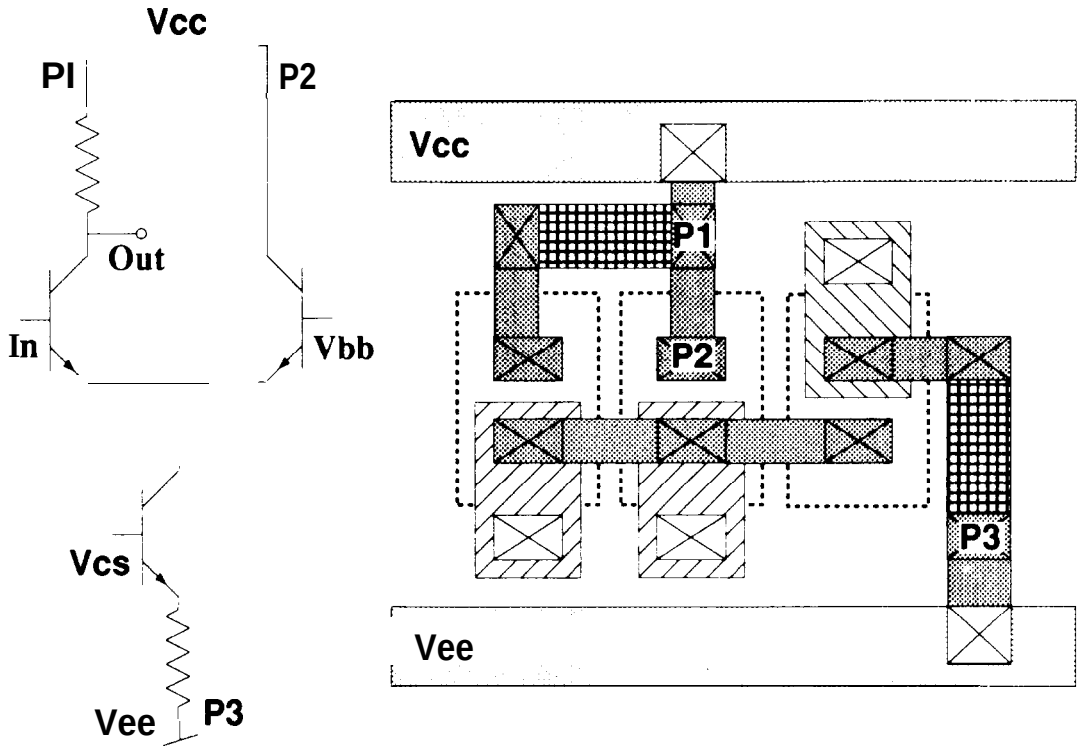


Figure 53: Locations of Currents for a Single Gate

and the other only collector connections, then the emitter end is terminal 0 and the collector terminal 1.

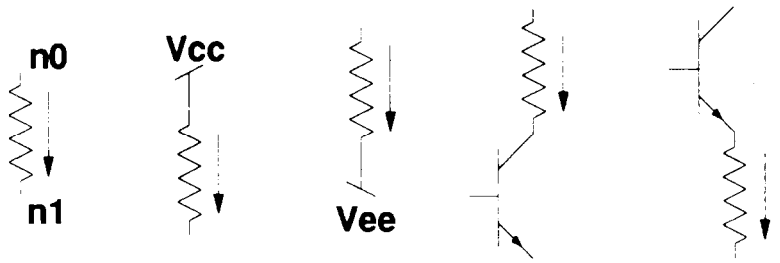


Figure 54: Direction of Current Flow in Resistors

Not all devices can be assigned a direction with these simple rules. An example of this is the temperature compensation circuit shown in Figure 55. The current estimator cannot assign a direction to resistor R_2 because both its terminals are connected to transistor emitters. When a resistor cannot be assigned current flow direction, the estimator removes it from the circuit and optionally issues a warning to the designer.

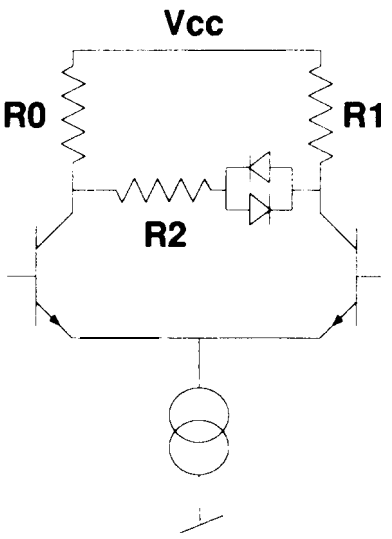


Figure 55: Temperature Compensation Circuit

Once all resistors have been assigned a direction, the estimator calculates the voltage and current ranges that each node and element can assume. Starting at the design's reference voltages, the estimator uses four basic rules:

1. Propagate voltages from base to emitter of a transistor and from terminal 0 to terminal 1 of a resistor. The estimator sets the voltage at a node to the maximum of the voltages at each base node minus a diode drop and the voltages at the top nodes of each resistor minus the resistor voltage drop.

2. Propagate currents from emitter to collector of a transistor and from terminal 1 to terminal 0 of a resistor. For each device, keep track of whether the current flowing though it is *rwnswitched* or *switched;* a nonswitched current always flows though the device, while a switched current may flow through the device depending on the state of the circuit.

3. Resistors connected to Vee form current sources. When the voltage at terminal 0 is set, calculate the current in the resistor and propagate it up the tree.

4. Resistors connected to Vcc form loads. When the current in the resistor is known, set the voltage at terminal 1 and propagate it to all connected bases.

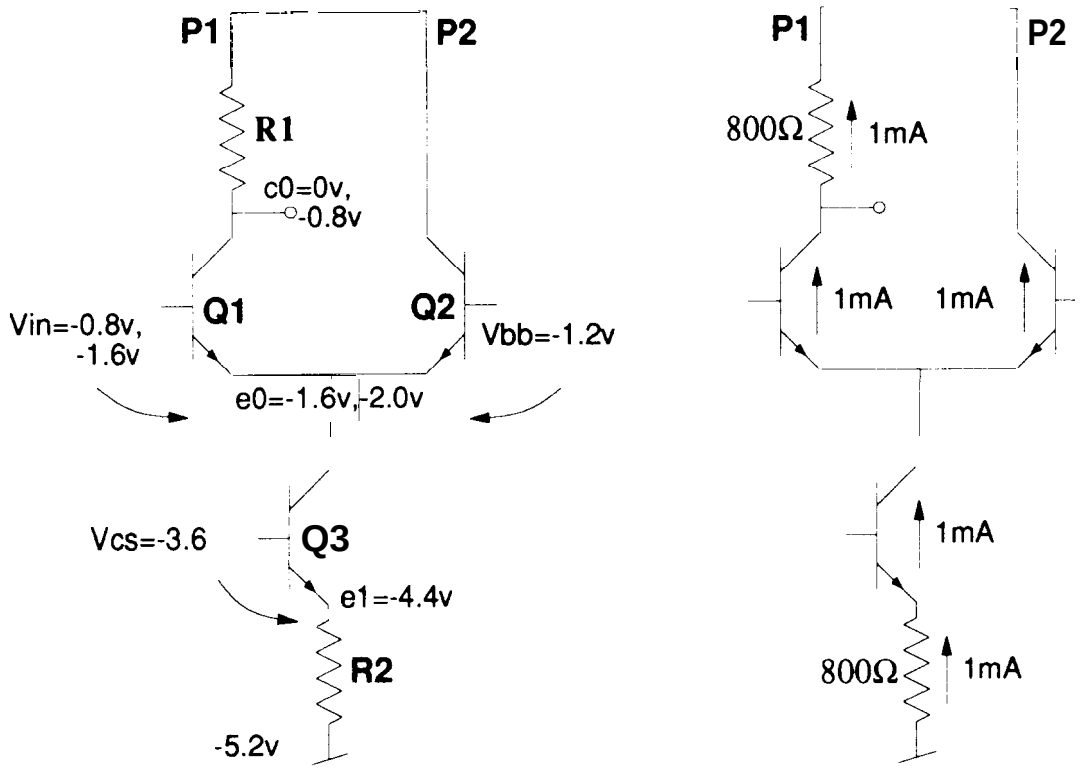


Figure 56: Tracing of Currents

Example 56 shows these rules in operation. Initially, the voltages ranges are known for nodes Vcs, Vin, and Vbb. Starting at these nodes, the estimator sets the voltages at the emitters of transistors Q_1, Q_2, and Q_3. For node **e0**, the maximum voltage is set one diode drop below the maximum voltage at Vin, and the minimum voltage is set one diode drop below the voltage at Vbb. For node el, the voltage is similarly fixed by node Vcs. Once the voltage at **e1** is known, the estimator recognizes that R_2 is a current source, and sets its value to 1 mA. This current is then propagated up the tree, to transistors Q_1, Q_2, and Q_3, and resistor R_1. Once the current in R_1 is set, the estimator recognizes that it is a load, and sets the voltage range at **c0**. Since the current through R_1 is switched, the voltage at **c0** is set to $\mathbf{Vcc, Vcc+I_1 R_1}$. With this gate finished, the estimator can continue the process for other gates with inputs connected to node **c0**.

In the example, the estimator does not really need to calculate the voltages at any node except el in order to trace the cut-tents properly; setting voltage ranges for the rest of the circuit seems extraneous. A simpler scheme might be to identify one signal as the

current source reference and assume that all transistors with bases connecting to it form current sources. The current estimator uses the more general method for two reasons. First, in some design methodologies, emitter followers have a resistor instead of a current source connecting them to **Vee**. Without knowing the voltage range at the base of the emitter follower transistor, the current estimator cannot accurately calculate the emitter follower current. Calculating voltage ranges for all the nodes in the system also allows the current estimator to double as a 'syntax' checker for ECL circuits; it can identify inputs connected at the wrong level, transistors pushed into saturation, and similar simple errors, saving the designer time in debugging the circuit.

4.3 Advanced Structures

Unfortunately, not all circuits can be solved using the basic algorithm. The range of circuits that designers use include some configurations more complicated than the basic ECL current tree. These include resistors used to split currents, current trees that cannot be correctly traced without some knowledge of the underlying logic, and diode structures. This section describes extensions that allow analysis of these more complex circuits.

4.3.1 Switched and Split Currents

The algorithm description glossed over how the estimator determines whether a current is switched or nonswitched. This is actually a fairly complicated problem; while being steered through a gate, currents may split and reconverge in several places. One example of this is shown in Figure 57. Current is switched between resistors R_2 and R_3 by the differential pair of Q_1 and Q_2. The current reconverges at R_1, however, regardless of the circuit state. In order to correctly set the node voltages, the estimator has to determine that the current from R_4 is switched through R_2 and R_3 but not through R_1.

This is done in two steps. The estimator first does a breadth first search of the circuit, starting at Vcc. Each device is assigned a number corresponding to the level at which it was first encountered. In the example, R_1 is a level 0 device, R_2 and R_3 are level 1 devices, Q_1 and Q_2 are level 2 devices, Q_3 is a level 3 device and R_4 is at level 4. Once

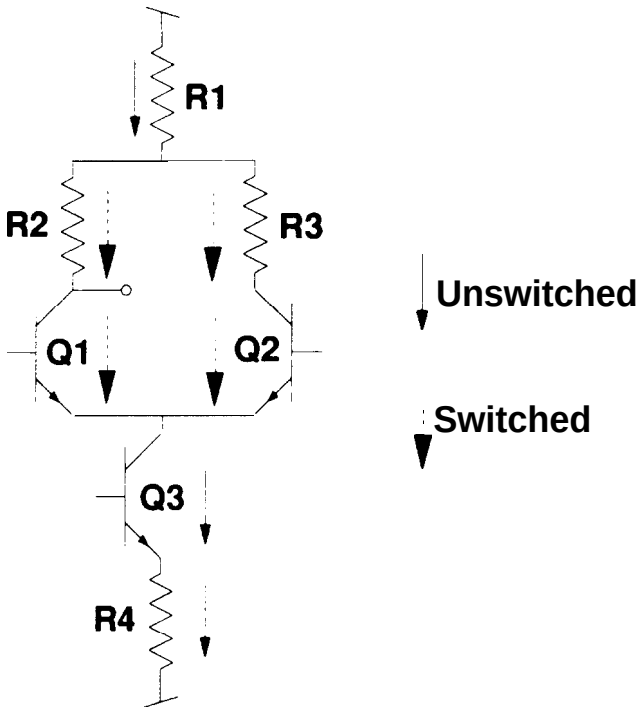


Figure 57: Switched and Unswitched Currents

each device has been assigned a level, the estimator traces out the currents as described in the previous step. In tracing up the tree, the current is considered unswitched until a differential pair is encountered. For the devices in the differential pair and above, the current is labelled switched. Once all the currents have been traced through the tree, the estimator counts the number of devices at level 0 through which each switched current passes. If there is only one level 0 device, the current for this device is changed from switched to unswitched, and any node voltages that are set by the element are updated. This is repeated at each successive level until one is found where the current passes through more than one device.

Implicit in the previous discussion of switched currents is the assumption that the entire current is steered to one branch of the tree or the other; some configurations violate this assumption. An example is shown in Figure 58. All the memory cells are connected to a single current source, with the resistors at the bottoms of cells used to subdivide the current between them.

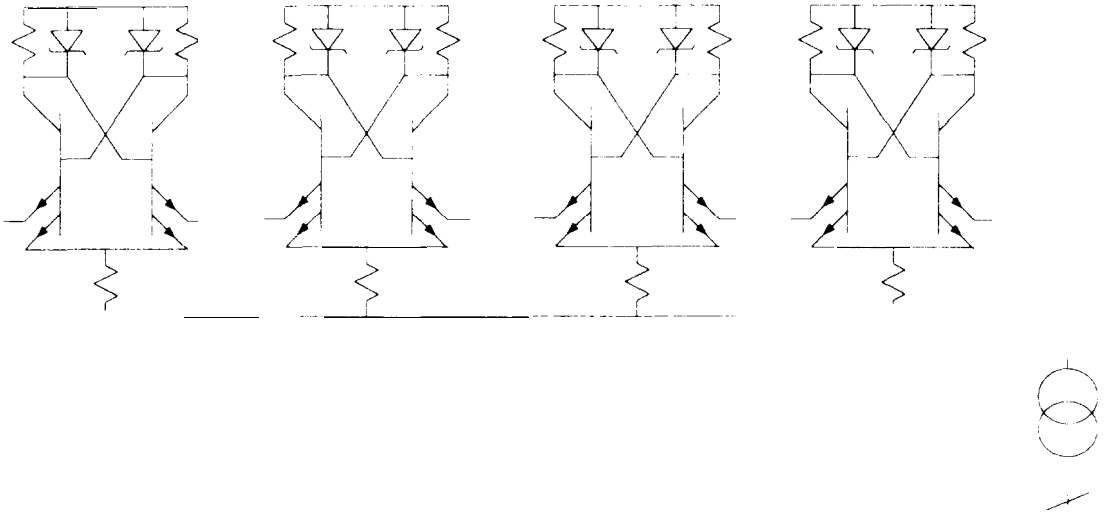


Figure 58: Resistor Divided Currents

Calculating precisely how the currents split is difficult. Since each node is represented by a voltage range rather than a single value, the diode-resistor current divider model employed by Bisim[29] cannot be used. Instead, the estimator assumes that the current divides evenly between all the resistors. When the current is propagated further up the tree, it is tagged with the amount of current that is actually passing through the particular branch. Thus, as the estimator traces each current up the tree, it attaches to each device encountered a pointer to the resistor forming the current source, a flag describing whether the current is switched or unswitched, and a label giving the fraction of the current that passes through this branch of the tree.

4.3.2 Logic Dependent Circuits

Not all circuits can be correctly traced knowing only the voltage ranges at each node; for some, the estimator must also understand properties of the underlying logic function. The Q-bit barrel shifter of Figure 59 is one example. Although any of the four currents I_A-I_D can be steered to any of the four loads, R_1-R_4, each load will receive at most one current at any given time because only one of select lines SO-S3 will be active. If the estimator does not understand this logic dependency, it will set the swings at the outputs to four times their actual values.

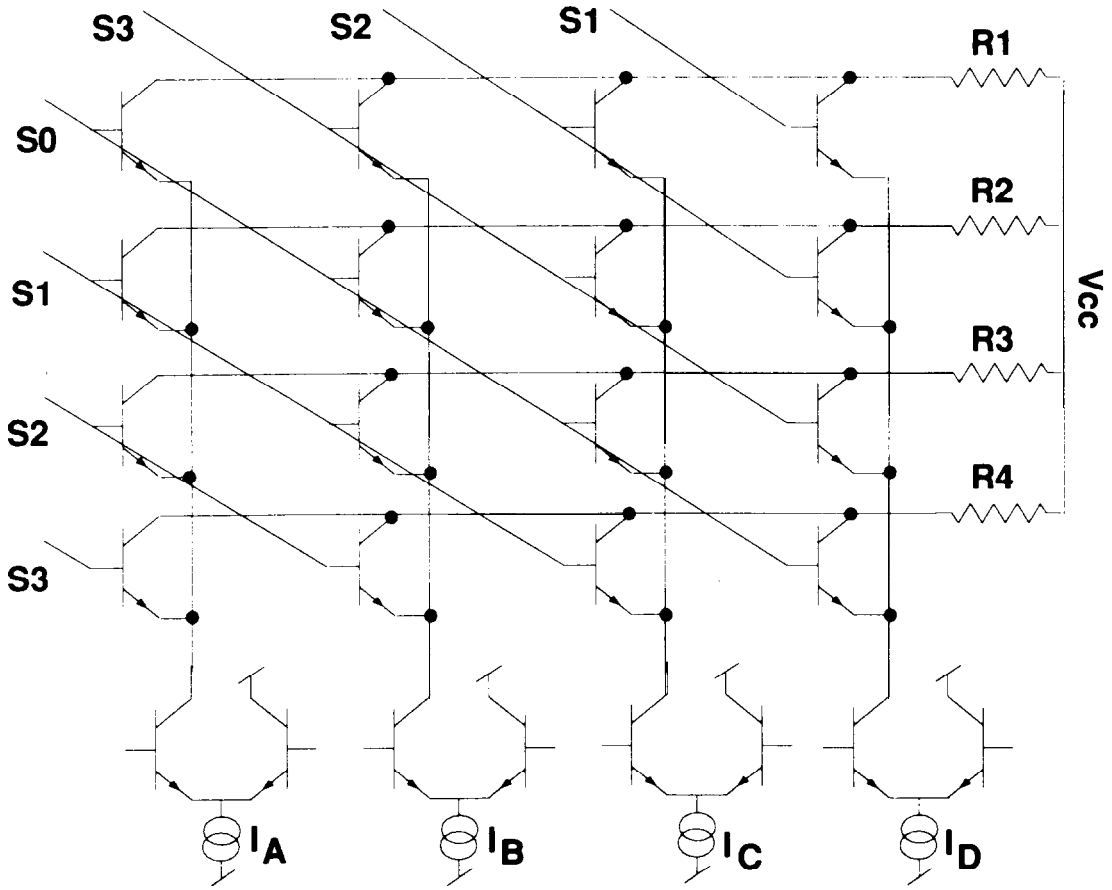


Figure 59: Barrel Shifter

Another state dependent circuit is the current source shared between many decoder outputs shown in Figure 60. Without knowing that only one of the decoder outputs will be high, the estimator will divide the current evenly between all the resistors. Although the voltage ranges for the outputs will be correct, the current will be smeared over all the outputs; any voltage drop problems associated with the large peak currents will not be caught.

In the general case, determining that a set of nodes have this mutual exclusivity property, where only one is high at any given time, is too expensive to calculate. Instead, the estimator provides two simpler mechanisms for incorporating logic dependencies. It recognizes decoder circuits (described in the next section) and marks their outputs as

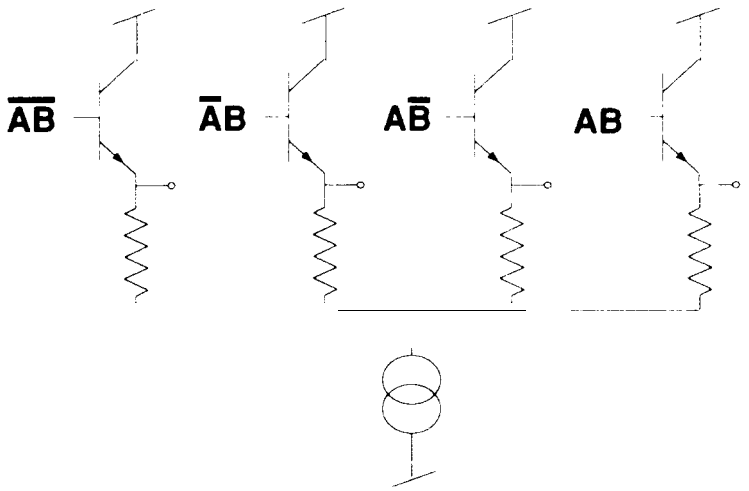


Figure 60: Decoder Output Shared Current Line

being mutually exclusive. For more complicated circuits, the designer may also label any group of nodes as being mutually exclusive. Given this information, the current estimator correctly apportions the 4 barrel shifter currents among the four loads and assigns all the current in the emitter follower circuit to a single device.

4.3.3 Diode Decoders

Another commonly used circuit that the above algorithm cannot handle correctly is the diode decoder[30]. A 2-bit version is shown in Figure 61. One of the four outputs will be high because current is not pulled through either of its diodes. The previous algorithm cannot correctly calculate the voltage at the base-collector (anode) terminal of a diode because it depends on the diode current, and it cannot calculate the diode current because it depends on all the diode voltages.

The estimator handles diode decoders as a special case. It searches for transistors where the base voltage depends on the current through the device. Each node in the circuit contains a list of the sources that can supply current to it. If a current source appears in the lists of both the base and collector[2], then the estimator checks to see what

[2]This algorithm may seem a bit baroque, since in the example, the base and collector are the same node! In some cases, however, the designer will split the pullup resistor into two parts: one section between Vcc

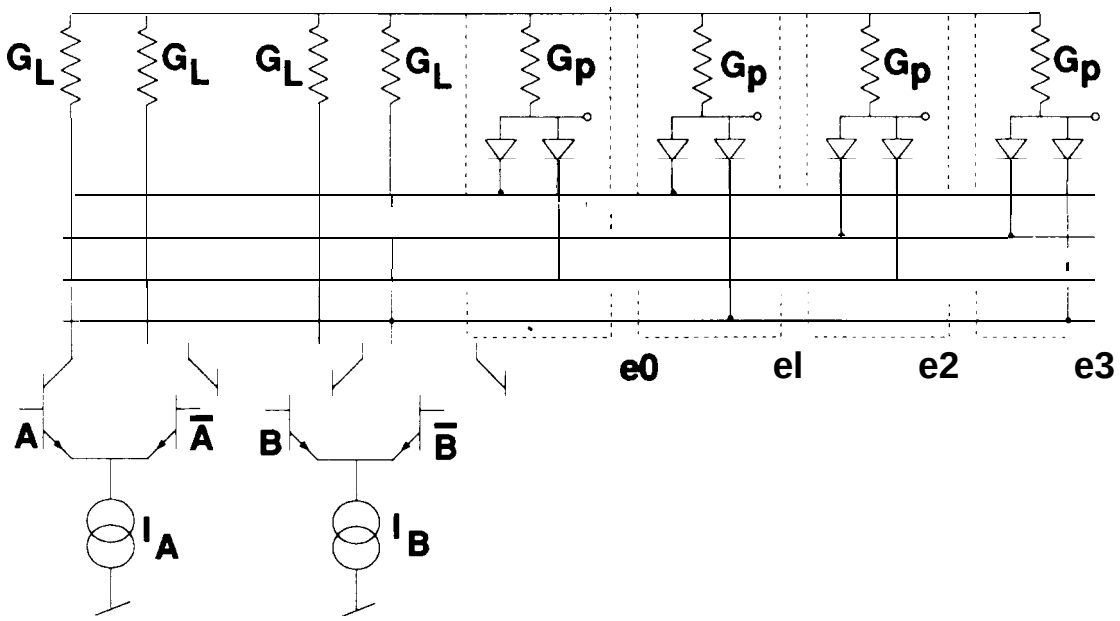


Figure 6 1: Diode Decoder Circuit

else is connected to the base node. If the base node is connected to an emitter, then the diode is being used as a level shifter and will be correctly handled by the original algorithm. If it is connected to a resistor, then its voltage will depend on the diode's current.

Once the current-dependent diodes have been located, they are processed in the following manner:

1. Group the diodes into output elements, which are sets of diodes that have a common base node. These are marked e_0-e_3 in Figure 61.

2. For each output element, find the conductance between the element's base and Vcc,

and the base and one section between the base and collector. This configuration reduces the amount that the base must swing and speeds up the circuit.

G_p.

3. Group the output elements into decoders, which are elements whose diodes have common emitter nodes.

4. Sum all currents that enter the decoder via the common emitter nodes. These are currents I_A and I_B in the example.

5. Check to see what type of structure is used to pull up the common emitter nodes. This is typically either a resistor or a transistor.

6. Calculate the low voltage at the output nodes. The equivalent circuits for this operation are shown in Figure 62. Which circuit is used depends on the type of pullup on the common emitter node. The diodes are assumed to be ideal for this calculation.

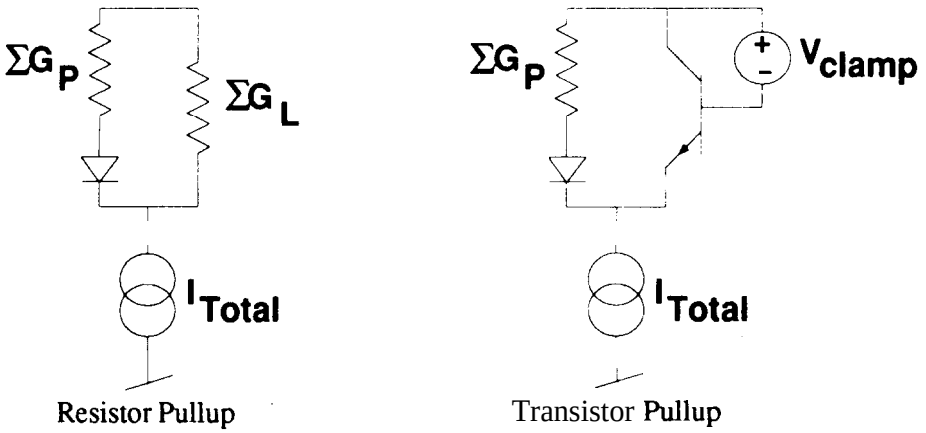


Figure 62: Equivalent Circuits for Diode Outputs

4.3.4 Other Circuits

There are other circuits that the estimator cannot handle properly; chief among these are bandgap generators and 10K/100K interface circuits. Currently, the designer must manually set the voltages at nodes in these circuits if they are to be included in the circuit checking and current estimation. A better approach might be to do a simple operating

point calculation for these circuits, similar to what Spice does. In general, each circuit is not particularly large, and the number of different configurations is also fairly small. By solving each different bias generator once, and using this solution for each instance of the circuit, the estimator could probably solve the remaining circuits in a reasonable amount of time and save the designer some trouble.

4.4 Pattern Selection

Once tracing is complete, the estimator has produced a list of currents together with a set of locations where they can enter the power network. The next step is to choose the node or nodes[3] from each set where the current will be injected. This section explores possible current distributions, and shows that the current pattern dependance in ECL is fairly minimal.

An ideal pattern would produce the greatest drop at each node in the circuit. Unfortunately, no single pattern or small group of patterns is going to give the worst drop for most circuits. Since processing a large number of patterns would be very expensive, the estimator instead tries to choose the node that will give the highest voltage drop for each individual current. The voltage produced by applying current at a single node can be calculated by examining the inverse of the network's conductance matrix.

$$\begin{bmatrix} V_0 \\ \vdots \\ V_j \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} G_{00}^{-1} & & G_{0j}^{-1} & & \\ & \ddots & \vdots & & \\ & & G_{jj}^{-1} & & \\ & & \vdots & \ddots & \\ & & G_{nj}^{-1} & & G_{nn}^{-1} \end{bmatrix} \begin{bmatrix} I_0 \\ \vdots \\ I_j \\ \vdots \\ I_n \end{bmatrix} \tag{51}$$

When a single current I_j is applied, all the nodes which have a nonzero term in column j of the inverse conductance matrix will have a nonzero output voltage. The highest voltage will be produced at the node where the current is injected, which corresponds to the entry on the diagonal, G_{jj}^{-1}. Given some set of nodes where a current can enter the

[3] For circuits like decoders, where all unselected outputs draw current, it will have to choose more than one node.

power network, the location that will give the highest individual voltage drop is the node with the largest diagonal term in the inverse matrix.

Unfortunately, calculating the diagonal of the inverse matrix is fairly expensive. The solution techniques described in the next chapter decompose the matrix into upper and lower triangular parts, but never calculate the inverse directly. Producing the inverse from the decomposed matrix requires forward and back substitution for each node in the circuit. This is much slower than doing the decomposition itself. A faster method for choosing the current pattern is needed.

The diagonal of the inverse matrix has physical meaning; each entry G_{jj}^{-1} is the equivalent resistance between Node j and ground. For networks that form trees, the equivalent resistance can be calculated in linear time by walking down the tree starting at ground; the equivalent resistance for each node is just the resistance of the node above it in the tree plus the value of the intervening resistor. Since distribution networks are generally tree-like, this suggests a method for estimating each node's path resistance: convert the initial resistance graph into a tree by removing resistors that form loops, then calculate the equivalent resistances for the tree. Figure 63 shows how this works for a simple bridge circuit. The circuit is converted to a tree by removing resistors R_4 and R_6, then the equivalent resistances are calculated in a single pass over the tree. Since forming the spanning tree itself can be done in n log(n) time[1], this estimate can be calculated quickly even for large designs.

The accuracy of the estimate, however, is rather poor. Figure 64 shows the actual path resistance versus its spanning tree estimate for the Vcc bus in the R6020 bus controller chip. The spanning tree estimate is considerably more conservative than the actual value for many of the nodes. To calculate the current pattern, however, the estimator does not care about the absolute value of each node's resistance; it is only interested in the resistance of each node with respect to its neighbors. As long as the spanning tree node resistances give roughly the same ordering for the circuit's nodes, the approximation is valid.

To see how well the approximate resistances preserve the node ordering, I compared the current distributions resulting from the exact path resistance and from the tree estimate for the three ECL chips described in Section 1.2. The results are summarized in Table

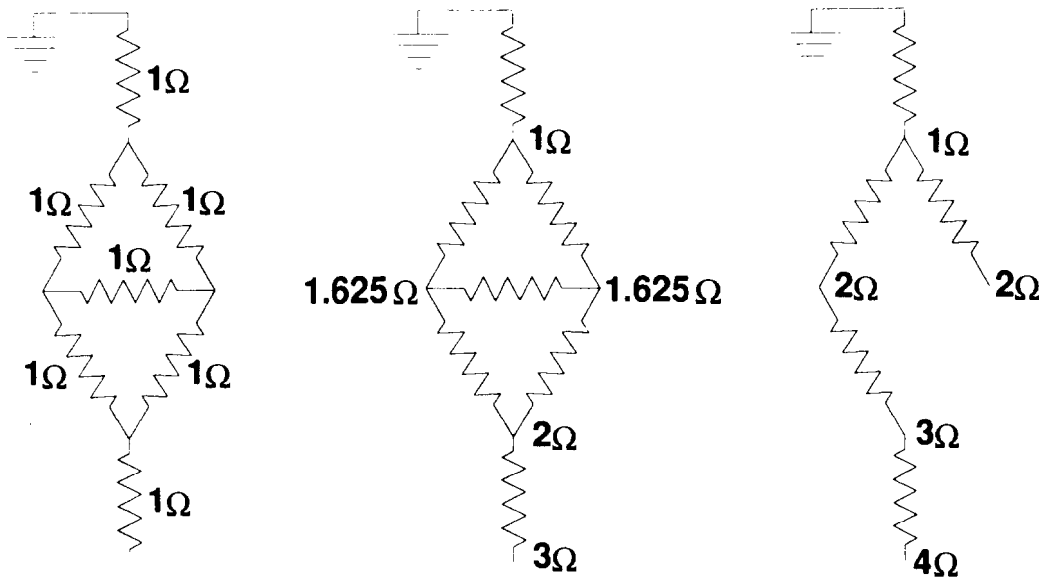


Figure 63: Tree Path Resistance Example

1 1[4]. The two current distributions are nearly identical; in the worst of the circuits, less than 3% of the current pattern changed.

Figure 65 shows the effect of the current pattern change for the R6020. Using the path resistance approximation made essentially no difference in the final voltage distribution in any of the chips tested Since the calculation of tree path resistances is two orders of magnitude faster, the pattern selector uses it to assign current locations.

4.5 Performance

Current profiles for the three designs were obtained by running the ECL current estimator on netlists extracted from the chip layouts. The results are summarized in Table 12. Analysis of fairly reasonably sized designs is thus possible in a modest amount of time.

Given that the current estimator can produce patterns for fairly large designs, the other

[4]The tests were run on a MipsCo RC3260, which contains a 25Mhz R3000 CPU, 64Kb instruction and data caches, and 128Mb of memory. The program was compiled with optimization level "-04" and run under RISC/os 4.50.

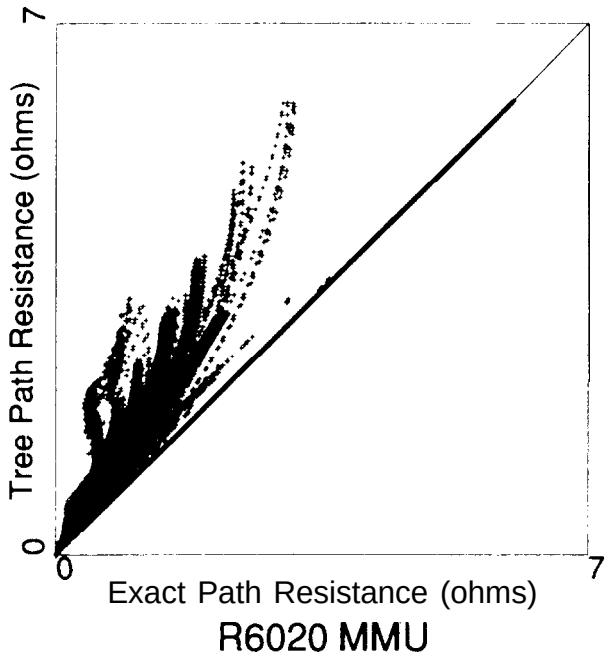


Figure 64: Spanning Tree Estimate of Path Resistance

consideration is the quality of the highest marginal cost current pattern. Underlying this question is a more fundamental one: how much effect does the state of the circuit really have on the current pattern? In most ECL gates, the location where current enters the power bus will not vary greatly; like the example in Figure 53, the various Vcc connections are close together. Only in a few classes of circuits, such as wired-or configurations and decoders with a shared current output, are the points likely to be very far apart. To test how much variation there is in the voltage distribution, I applied two different current distributions to each circuit in the R6000 chipset. The first distribution was calculated using the highest marginal cost criterion described in this section. The second was produced by using a lowest marginal cost function, which produces the smallest possible drop for each individual current. Figure 66 contains scatter plots comparing the two. In general, the correlation is fairly good, indicating that the design is not particularly pattern dependent. Variations from perfect correlation seem to fall into two categories: nodes either form a line that diverges upward from perfect correlation (as seen in the center of the R6000 plot) or they come in pairs, one above and one below the center diagonal.

Chip	R6000 CPU	R6010 FPC	R6020 SBC
Total nodes in Vcc bus	71947	72052	59495
Nodes with current applied	16610	15992	16075
Nodes where pattern varied	3	19	477
Exact method Running time	6.1 hours	5.8 hours	6.1 hours
Tree method Running time	1.8 minutes	1.8 minutes	1.5 minutes

Table 11: Comparison of Current Pattern Selection Methods

Circuit	R6000	R6010	R6020
Running Time	552s	354s	803s
Memory Usage	15.5Mb	19.9Mb	17.2Mb

Table 12: Running Times for ECL Current Estimation

The first pattern corresponds to a circuit where all the possible locations for a current fall in the same branch of the network; by choosing the node farthest from the pad, the highest marginal cost algorithm shifts up the voltage for all the intervening points in the branch. The second pattern occurs when the current's locations fall in different branches of the network; switching between locations causes one branch's node voltages to increase while those of the other branch decrease.

The highest marginal cost function seems to be modestly better than just picking current locations at random, but not spectacularly so. In the three examples, 65% of the nodes had equal or higher voltages for the highest marginal cost pattern than for the lowest marginal cost. Despite the relative insensitivity of ECL current patterns to the state of the circuit, I decided to continue using the highest marginal cost current pattern. Since the network solution techniques described in the next chapter require formation of

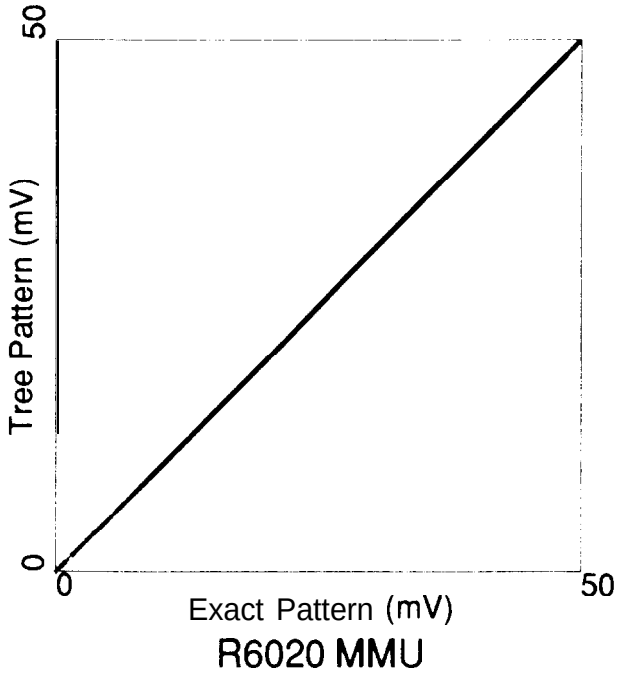


Figure 65: Effects of Path Resistance Estimate on Voltages

a spanning tree for the network anyway, the overhead involved in implementing the cost function is minimal. Further, more specialized circuits that make greater use of wired-or circuits, such as Ling Adders[33], may have a greater pattern dependence than do the R60X0 chips, which are composed primarily of ordinary gates.

Given the relative insensitivity of the ECL current distribution to the circuit state, and the ease with which a single, static current pattern can be calculated, the static current estimation algorithm described in this chapter is sufficient for current estimation in ECL designs.

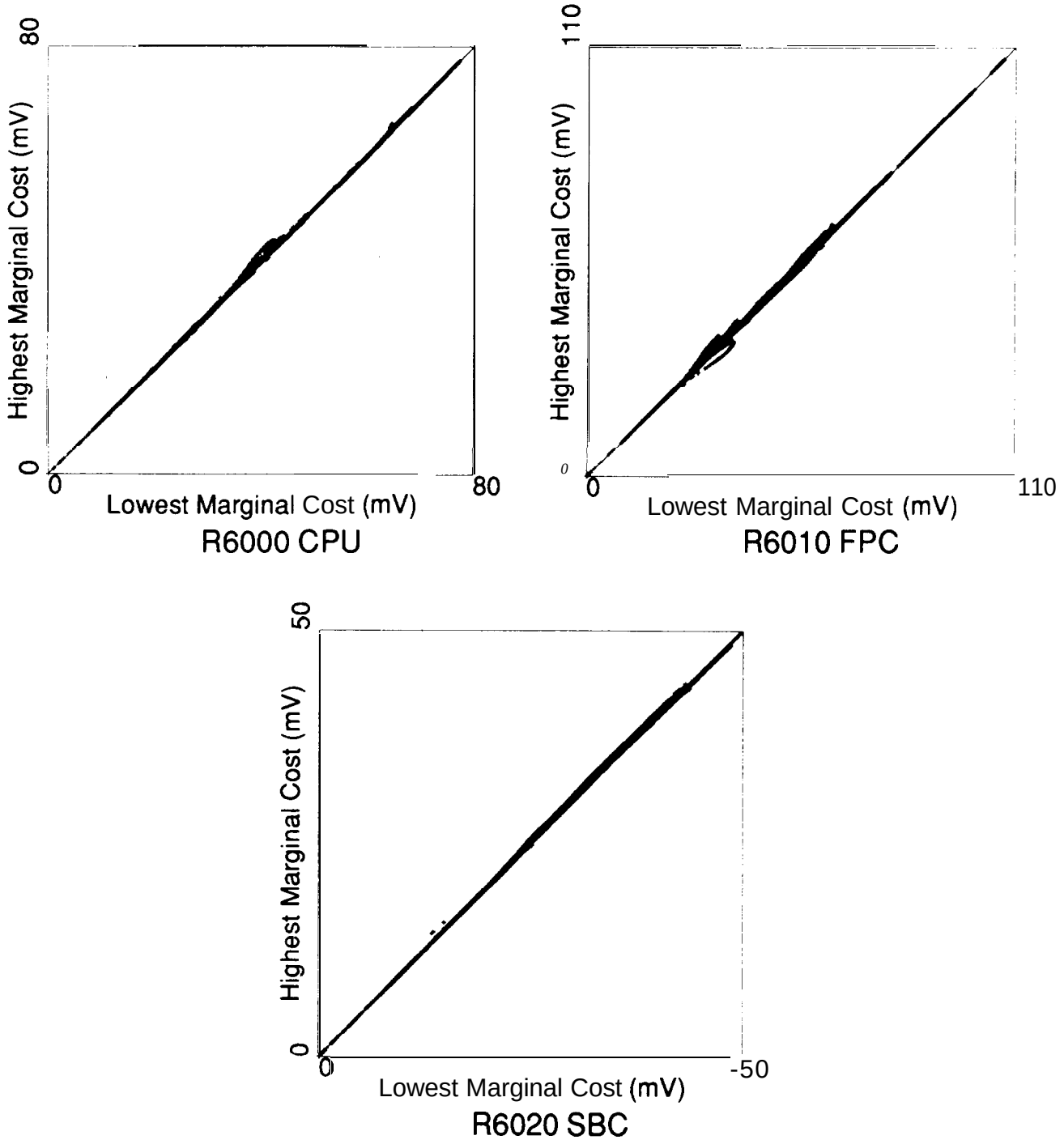


Figure 66: Current Pattern Dependence of ECL circuits

Chapter 5

Network Solution

> Our life is frittered away by detail. An honest man has hardly need to count more than his ten fingers, or in extreme cases he may add his ten toes, and lump the rest. Simplicity, simplicity, simplicity!
>
> Henry David Thoreau
> *Walden (1854)*

The last three chapters described how to derive a resistance network from the design's layout and a current profile from its circuit; the remaining step is solving for the network's node voltages and branch currents. When run on an entire design, the number of node equations can be fairly large; networks of order 20,000 are not uncommon[54]. Since the current distribution can vary with time, these systems may have to be solved more than once. Fortunately, power supply networks have some special properties that allow for faster calculation of voltages.

The first section of this chapter outlines previous approaches that have been taken to solve these networks. Following this is an examination of some of the special properties of resistor networks and how they can be exploited, an outline of applicable sparse matrix techniques, and a discussion of the system's performance on the test designs.

5.1 Previous Work

Branin[4] solves power supply networks using tree-link analysis. He envisions power supply networks as being basically trees, with an occasional loop created when the designer adds cross links. Figure 67 shows what happens when a single link resistor R_{link} is added to a tree, forming a loop. The link resistor takes some amount of current from one branch of the tree and redistributes it to the other. Current is only affected in the loop formed by the link and the two sides of the tree connecting to it; the currents above the top node N_0, where the two branches merge, is unchanged.

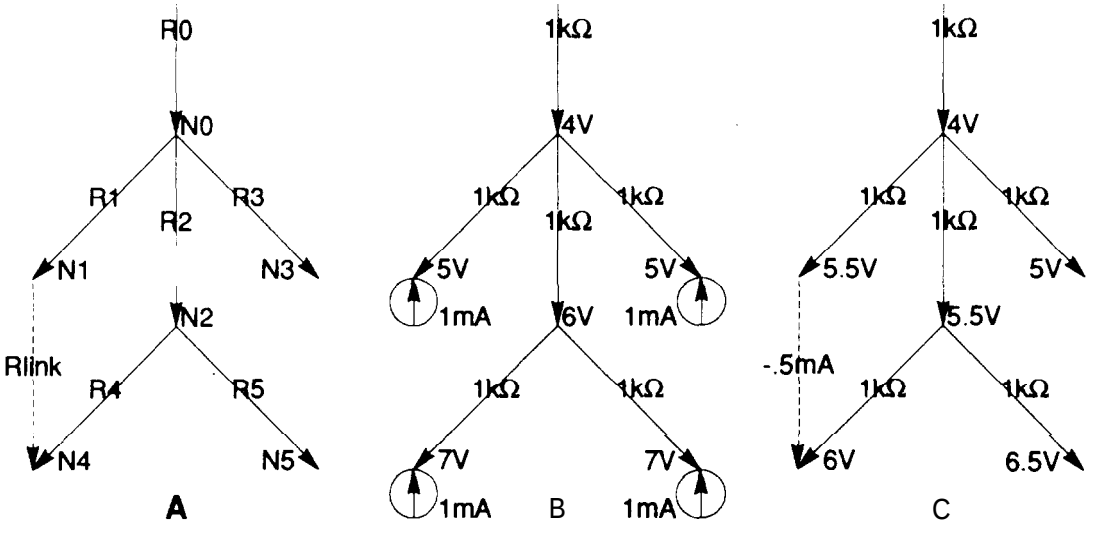


Figure 67: A Single Link Resistor in a Tree

Isolated link resistors in a tree-like structure thus add local perturbations to the underlying tree voltage-current distribution. This suggests a method for solving this class of network:

1. Form a spanning tree for the network.

2. Solve the tree network, ignoring all non-tree resistors.

3. Calculate the perturbations in the current distribution caused by the link (non-tree) resistors.

4. Solve the network again using the revised current distribution.

Branin's algorithm is efficient for systems where the number of loops is quite small. The most expensive step in the algorithm is usually calculating the loop currents. As the number of loops go up, the complexity of the system of loop equations also goes up. While it is reasonable to assume that the number of loops is small in circuits fabricated with a single layer of metal, it is not true for newer devices that have meshes embedded in their power distribution networks. However, this algorithm can still be used to advantage in some limited cases, as discussed in Section 5.3.

Tyagi[58] uses a method similar to Branin's, except that the range of topologies that it can handle is more limited. He also assumes that the network is basically a tree, except for some comb sections at the tree's leaves. Combs are sections of power bus that have parallel wires tied together at both ends. Each comb section is treated as a supemode; all the current that enters the network through the comb section is injected at node where the comb is attached to the tree. The tree and comb are then solved separately; the tree using the linear two-pass algorithm discussed in the next section and the comb by solving the node-current equations. For a comb, Tyagi shows that the node equations can be solved in linear time.

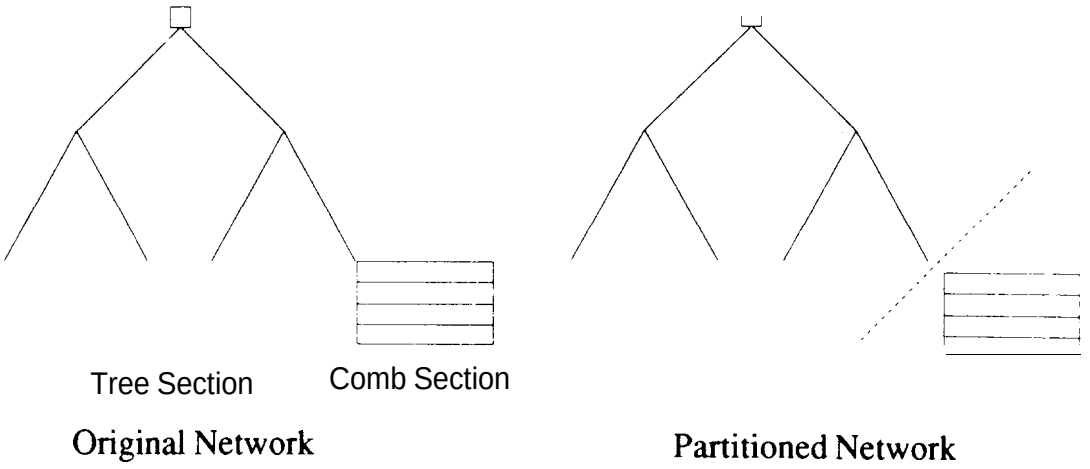


Figure 68: Tyagi's Algorithm for Treelike Systems

Most of the power networks in custom designs could not be solved using this algorithm because they do not have a tree-comb topology. Even if more general networks than

combs are allowed for the non-tree sections, Tyagi's approach will be fairly slow; a single loop near the root of a power bus will cause nearly all of the bus to be put in a single supemode. Solution time for this supemode will be essentially the same as the time required for directly solving the original network. Since any distribution topology that has more than one pad connection will have *all* nodes (including the root node) as part of a loop, Tyagi's method is not particularly useful for contemporary multilayer power networks.

Chowdhury[10] takes a different approach to analyzing the system. Rather than calculating the actual node voltages and branch currents, he instead calculates an upper bound on the current that can flow through each branch assuming that some subset of the network branches fail via electromigration. Figure 69 shows an example of how his algorithm works. For each node, the estimator enumerates all possible paths to ground. Picking such a path divides the nodes in the design into two classes: those whose current may flow to the node and through the path to ground, and those whose current will not. In the example, starting at node n_2 and using the path $e_1 - e_0$, current from nodes n_1, n_2, and n_3 fall into the first class and nodes n_0 and n_4 fall into the second one. Once the nodes have been classified, the maximum current for the first element in the path can be determined; it is the sum of the currents from all the nodes in the first set. In the example, the maximum current in element e_1 for the given starting node and path is $I_{n1} + I_{n2} + I_{n3}$. By repeating this operation for all nodes and paths in the system, the maximum current in each branch can be calculated.'

Unfortunately, the upper bound that this algorithm gives for each element is much too conservative to be useful. Figure 70 represents the power distribution for the datapath section of the design. There is one horizontal power line for each bit in the datapath, with vertical connections to pads at either end. Using Chowdhury's algorithm, the upper bound on the current in each element of each row is very nearly the entire current for the datapath. The solid line represents one node-path pair; current may flow from nearly all the nodes in the circuit to the node-path pair via the dotted lines. Each bit's power connection would have to be wide enough to support the current for all the bits. No

[1] Chowdhury does not actually evaluate every path; by using branch and bound techniques, he can show that some paths will not give a larger current for the element than a path already enumerated, and can therefore be ignored.

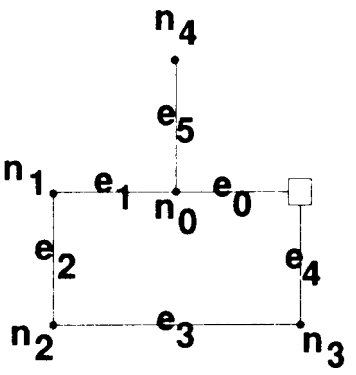


Figure 69: Chowdhury's Max Current Estimation Algorithm

aggressive design can afford to be this conservative.

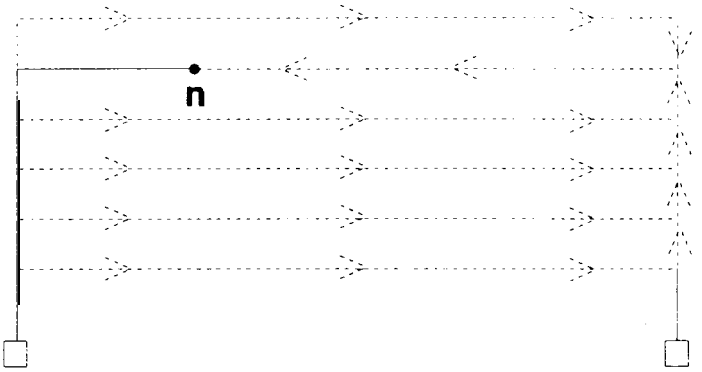


Figure 70: Overestimation in Chowdhury's Algorithm

Since none of these algorithms were entirely satisfactory, I decided to take a closer look at the designs to see what special properties they exhibited and how these properties might be exploited. The techniques that the analyzer currently uses take some of the more useful components of Brat-tin's algorithm for trees and simple links, and combine them with a new algorithm for handling sections of resistors in series with interspersed current sources.

5.2 Trees of Resistors

A typical power supply network in a VLSI design (Figure 71) consists of two sections: a "backbone" that provides global distribution of power from the pads to the various modules on chip, and a "feeder" network that connects individual cells and transistors to the backbone. The most common style for these feeder networks is a tree, with the root being the connection to the backbone and the leaves being each transistor connected to the supply. Solving these tree networks is simple: starting at the leaves (Figure 72), currents are summed up toward the root. Once the current is known in each branch, the node voltages can be found by back-substitution; each node's voltage is that of its parent node plus the voltage drop across the connecting resistor.

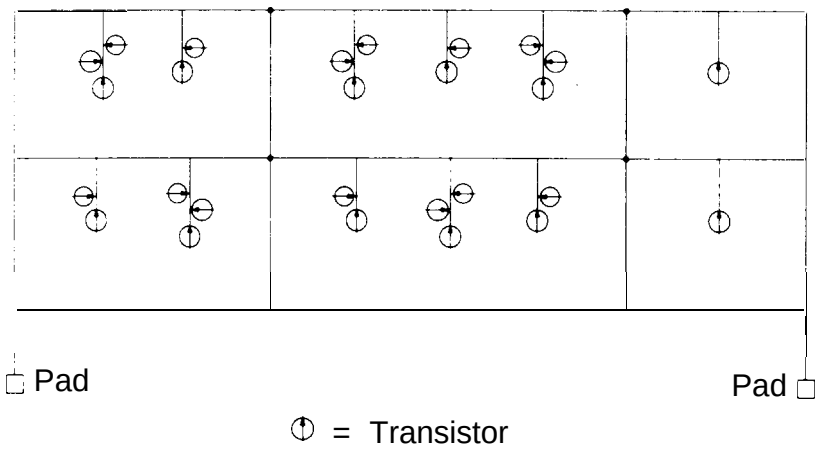


Figure 71: A Typical Power Network

In the linear solver, these trees are replaced on the network backbone by a current source representing all the current injected into the tree. After the backbone network is solved and the voltage at the root node of each tree is known, the leaf voltages can be found by the method described above.

5.3 Simple Loops and Kirchoff's Voltage Law

This section explores Branin's algorithm in more depth. The key to his method is quickly calculating the current disturbance caused by the link resistors. This adjustment can be

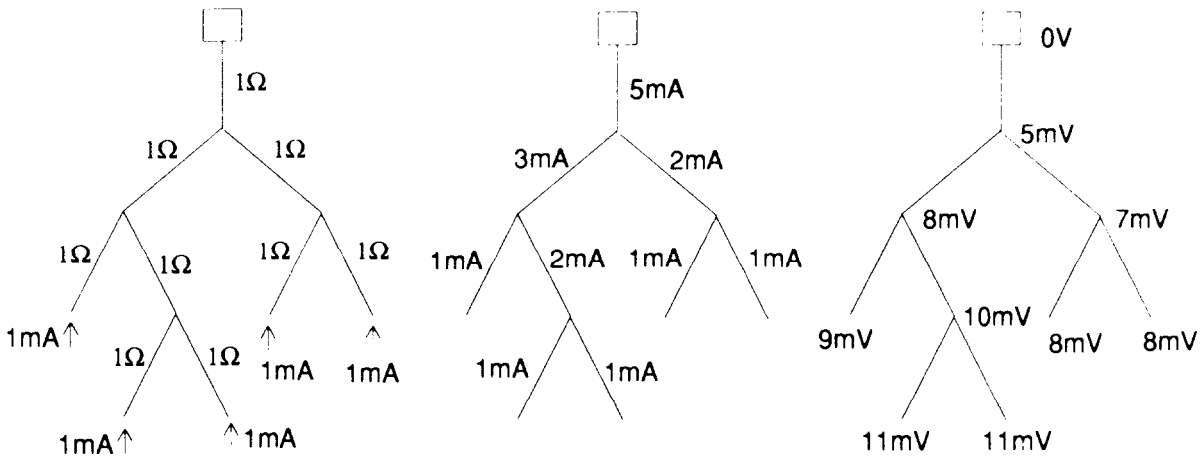


Figure 72: Solving for the Tree Voltages

found by applying Kirchoff's voltage law to the loop formed by the link resistor and the spanning tree. Enough current must flow through the link so that the sum of the voltage drops across all the resistors in the loop is zero. Suppose that the voltages at each node are initially calculated without the link resistors present using the tree algorithm described in the previous section and shown in Figure 67. Each unit of current flowing through the link resistor reduces the voltage drop between the nodes N_1 and N_0 by $I_{link}R_1$, and increases the voltage drop between nodes N_4 and N_0 by $I_{link}(R_2 + R_4)$. Given the voltage distribution calculated for the tree part of the network, the link resistor current must be:

$$I_{link} = \frac{V_{link}}{\Sigma R_{loop}} = \frac{-2V}{4K\Omega} = -0.5mA$$

ΣR_{loop} is the sum of resistors forming the loop, and V_{link} is the initial voltage drop between the link resistor's connecting nodes. In Example 67, $I_{link} = -0.5mA$. Once I_{link} is known, the tree portion of the network can be solved with the modified current distribution (Figure 67c), giving the correct node voltages. For isolated loops, this solve-perturb-solve method is considerably faster than doing a more general sparse solution of the network.

The problem becomes more complicated when there is more than one loop, as shown in Figure 73. Because the two loops share a resistor, we cannot apply the previous

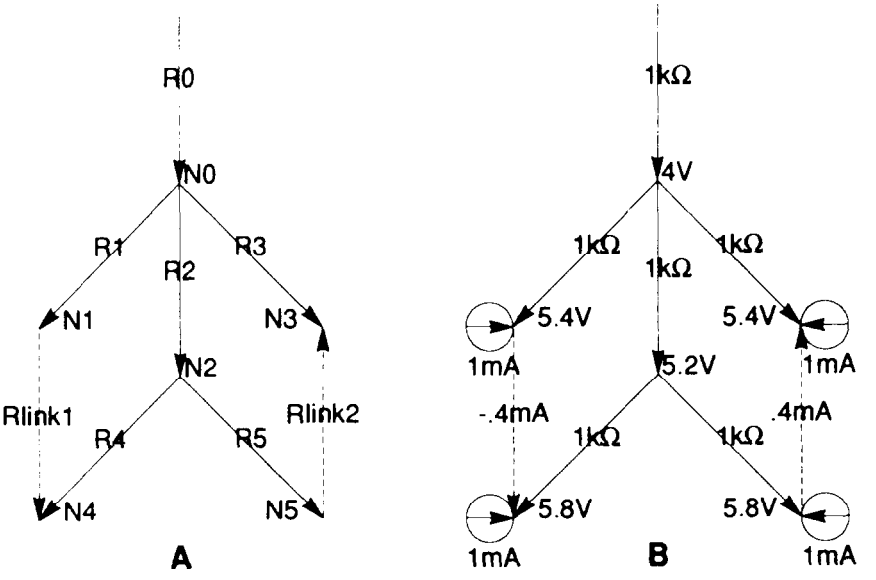


Figure 73: Multiple Link Resistors in a Tree

formula to the loops individually; the change in voltage drop across R_2 that each causes will affect the other. Kirchoff's voltage law must be satisfied in both loops simultaneously. Summing up the drops across each resistor in both loops gives two linear equations:

$$\begin{bmatrix} \Sigma R_{loop1} & R_{mutual_{1,2}} \\ R_{mutual_{1,2}} & \Sigma R_{loop2} \end{bmatrix} \begin{bmatrix} I_{link1} \\ I_{link2} \end{bmatrix} = \begin{bmatrix} V_{link1} \\ V_{link2} \end{bmatrix}$$

$$R_{mutual_{1,2}} = \Sigma(R_{loop1} \cap R_{loop2}) * (Incidence_1 * Incidence_2)$$

$R_{mutual_{1,2}}$ is the sum of the resistance that the two loops have in common times an *incidence* value of -1 or 1; the incidence value tells the direction through the common resistor which the extra link current is defined to flow. (The direction chosen is arbitrary, so long as it is consistent throughout the calculation.) Loop resistors in the tree branch connected to the ***head*** of the link resistor will have an incidence of 1, while those in the *tail* branch will have an incidence of -1. In Example 73a, there is a single common resistor, R_2, whose incidence is 1 for loop 1 and -1 for loop 2, giving $R_{mutual_{1,2}} = -R_2$. *The 2x2* matrix for the example is:

$$\begin{bmatrix} R_1 + R_2 + R_4 + R_{link1} & -R_2 \\ -R_2 & R_2 + R_3 + R_5 + R_{link2} \end{bmatrix} \begin{bmatrix} I_{link1} \\ I_{link2} \end{bmatrix} = \begin{bmatrix} V_{link1} \\ V_{link2} \end{bmatrix}$$

This 2x2 system is trivially invertible, giving $I_{link2} = -I_{link1} = 0.4mA$. With the link currents known, the tree can be solved again, giving the voltage distribution shown in Figure 73b.

This method can be extended for an arbitrary number of link resistors. The resulting system will have one equation per link resistor:

$$\sum_{j=1}^{N} R_{mutual_{i,j}} * I_{link_j} = V_{link_i}$$

Once this system is solved, currents in the tree portion of the network can be modified and the corresponding node voltages calculated.

Unfortunately, this algorithm does not work well for most power networks. A network's backbone section generally contains many link resistors, each of which share part of their loop with another link. This gives a large number of nonzero R_{mutual} terms, making the resulting system, while lower order, considerably less sparse than the original. Table 13 shows the sizes and number of nonzero terms for the power networks of the three CMOS designs described in Section 1.2. The first two columns show the order and sparsity for the conductance matrix after the simplification techniques have been performed, while the latter two show the corresponding values for the mutual resistance matrix. Using the sparse techniques discussed in Section 5.5, the higher order, sparser systems can be solved more quickly than the more dense, lower order ones.

Unlike the original matrices, however, the mutual resistance matrix is usually disconnected. Each subgraph of the matrix corresponds to a set of link resistors with a nonzero mutual resistance to at least one other member of the set, but no nonzero mutual resistance values to anything else. The order of a subgraph is equal to the number of link resistors it contains. Figures 67 and 73 show subgraphs of order 1 and 2, respectively. Table 14 shows the distribution of subgraph orders for the designs mentioned above. All the networks contain a random scattering of low order subgraphs and one or two high order ones that contain the network "backbone".

Several of the networks have a large number of subgraphs containing one or two members. These sections usually arise from strapping a region with high sheet resistance, such as diffusion, with a much less resistive one, such as metal. They are often part of a

Power Network	Simplified Original Matrix		Mutual Resistance Matrix	
	Matrix Order	Nonzero Terms	Matrix Order	Nonzero Terms
MIPS-X gnd	4095	18019	2893	12940 1
MIPS-X vdd	2881	11651	2469	35241
SPIM gnd	2139	8737	1165	31085
SPIM vdd	601	2443	324	26616
μTitan gnd	3066	12366	1669	13021
μTitan vdd	2123	3244	1157	32925

Table 13: Original and Mutual Resistance Matrices

Power Network	Subgraph Order					
	1	2	3-10	11-100	101-1000	1000
MIPS-X gnd	15	3	5	4	0	1
MIPS-X vdd	75	444	3	0	0	1
SPIM gnd	0	0	1	3	0	1
SPIM vdd	0	0	1	1	1	0
μTitan gnd	11	53	64	2	2	0
μTitan vdd	39	9	36	3	1	0

Table 14: Subgraph Sizes for Various Networks

feeder network that could have otherwise been removed by the tree algorithm described above. Since these subgraphs are trivial to solve and removing them often allows an entire subtree to be separated from the main graph, Ariel processes them specially. It searches the resistor network for one and two element graphs that are part of trees which can be removed from the network. When it finds one, Ariel replaces the subnetwork with a current source as described in Section 5.2. Once the backbone network has been solved, the voltages in the loop are determined by calculating the initial tree voltages, inverting and solving the 1x1 or 2x2 loop-link matrices, and recalculating the tree voltages using the modified current distribution.

Moderate sized subgraphs, containing less than 100 elements, can also be solved

relatively cheaply. Unfortunately, they are rarely parts of a feeder tree and therefore cannot be separated from the main graph. Because of this, Ariel limits this type of processing to 1 and 2 element subgraphs.

5.4 Series Connections of Resistors.

Once the trees and simple loops are removed from a network, the remaining backbone consists primarily of long strings of resistors, one per bit pitch, with occasional cross links, as shown in Figure 74. These long sections of series resistors, with a current source at each intervening node, can be replaced by a single resistor with a current source at each end. An intuitive derivation of the equivalent circuit is given in the following section. For the skeptical, a more formal derivation is given in Section 5.4.2.

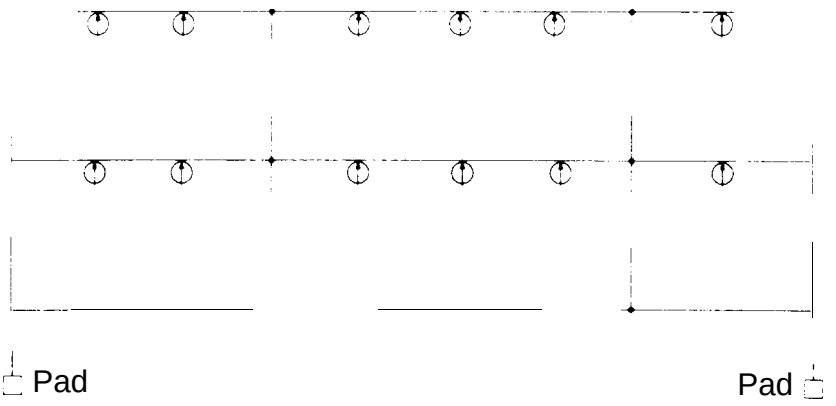


Figure 74: Power Network with Trees and Simple Loops Removed

54.1 Equivalent Circuit for Series Resistors

Consider the series chain in Figure 75a. There will be some voltage, Vs. between the two series ends, N_1 and N,. Conceptually, a voltage source with value Vs can be added between nodes N_1 and N_n without disturbing the network. Superposition can be applied to this network to produce the equivalent circuit shown in figure 75b. The equivalent resistance R_S is just the sum of all the resistors in series. Superposition can be used to

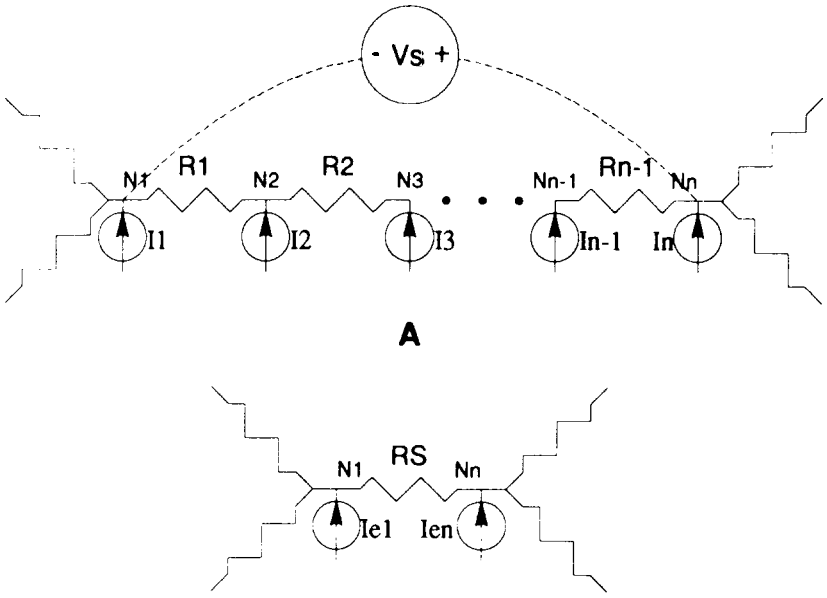


Figure 75: Series Equivalent Circuit

determine how the current from each current source divides between the two ends. All current sources except the one in question are replaced by open circuits, while the voltage source between nodes N_1 and N_n is replaced by a short circuit. The resulting system is a simple current divider, and the additional current at N_1 and N_n is just the sum of all the divided currents:

$$R_S = \sum_{i=1}^{n-1} R_i \tag{52}$$

$$I_{e_1} = \sum_{i=1}^{n} \frac{\sum_{j=i}^{n-1} R_j}{R_S} I_i \tag{53}$$

$$I_{e_n} = \sum_{i=1}^{n} \frac{\sum_{j=1}^{i-1} R_j}{R_S} I_i \tag{54}$$

An arbitrarily long series of resistors can thus be replaced by a single resistor and two current sources.

Once the network has been solved with the equivalent series circuit and the voltages

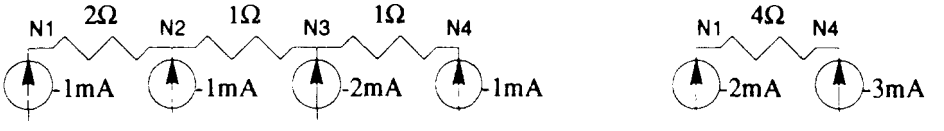
at the end nodes are known, the solver must determine the voltages at the series nodes. Superposition can be used to calculate the intermediate values for this system:

$$V_i = V_{i-1} + \underbrace{\frac{R_{i-1}}{R_S} V_S}_{\text{Voltage Divider Drop}} + \underbrace{I_{e_i} R_{i-1}}_{\text{Current Source Drop}}$$

$$I_{e_{i+1}} = I_{e_i} - I_i$$

Starting at one end, the voltage at each successive node is equal to the value at the previous node plus the drop in the intervening resistor. This drop has two parts. The first, labeled *Voltage Divider* Drop, is caused by V_S, the voltage imposed by the rest of the network across this series section. To calculate it, V_S is split across the resistors in proportion to their values. The second component, labeled *Current Source Drop,* is caused by current injected at nodes within the series. Between each pair of nodes, this drop is equal to the connecting resistor's value times the current that would flow through this resistor if V_S was replaced by a short circuit.

Figures 76 and 77 show an example of series replacement and intermediate node voltage calculation. The three resistors in the circuit on the left are replaced by a single one, and the current at N_2 and N_3 is redistributed to the end nodes, N_1 and N_4. This equivalent circuit is connected to the rest of the network and solved.



Original Series Equivalent Circuit

Figure 76: Series Circuit Example

In the example, suppose that the drop V_S between nodes N_1 and N_4 is -4mV. The next step is setting the voltages at N_2 and N_3. Figure 77 shows the different voltage

components for each node. The voltage divider drops are calculated by partitioning V_S between the three resistors. The current source drop is found by calculating the current that would flow through each resistor with N_1 and N_4 grounded. Beginning at the left, the current between nodes N_1 and N_2 is $I_{e_2} = I_{e_1} - I_1 = -1mA$. Similarly, the current between N_2 and N_3 is $I_{e_3} = I_{e_2} - I_1 = 0$. The current source drop in each section is the resistance times these currents. The total drop for each node is calculated by adding the two components.

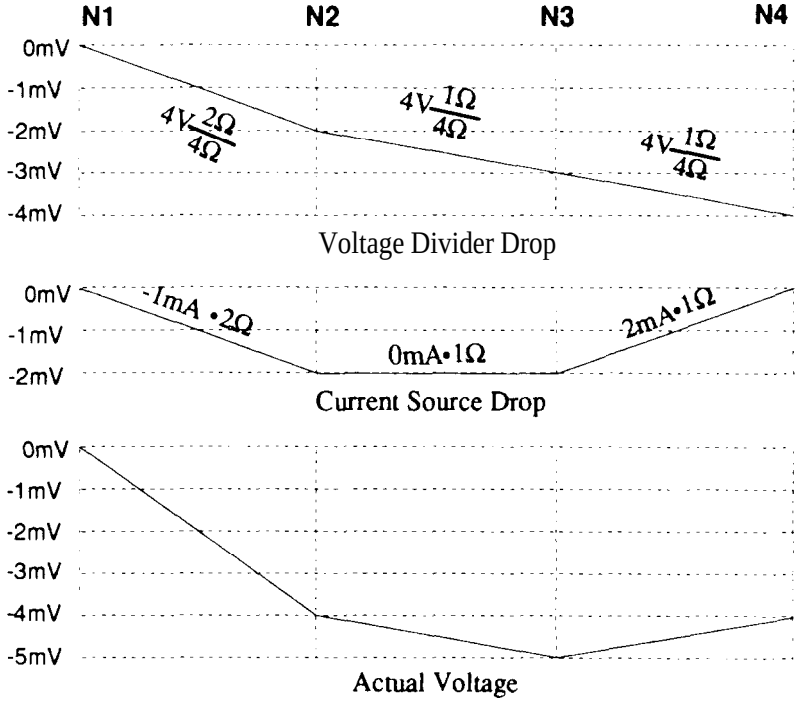


Figure 77: Voltages for Series Circuit Example

5.4.2 Norton Equivalent Circuits for the Series Systems

The previous derivation and example were intended to be intuitive; this section gives a more formal derivation of the series equivalent circuit. Figure 78a shows a series of *n* resistors interspersed with current sources. R_L and I_L form the Norton equivalent circuit representing the rest of the network as seen from the left end of the series. Figure 78b shows the series equivalent circuit for this configuration, as derived in the previous

section. The following section shows that the two circuits are equivalent when viewed from the right side terminals; by symmetry, the same can be shown from the left side.

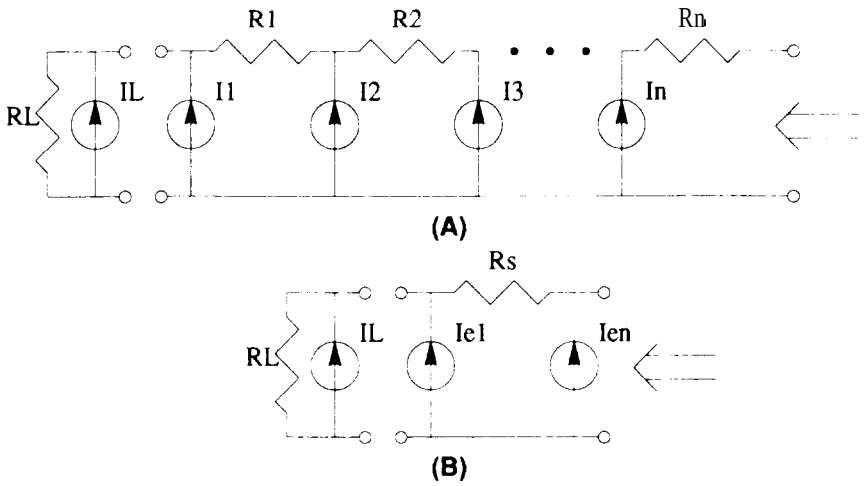


Figure 78: Series and Equivalent Circuit

All the resistors in Figure 78a are in series, so the equivalent resistance is just their sum, labeled R-r in Equation 55. When the right terminals are shorted, $n+1$ current dividers are formed; by superposition, the short-circuit current is just the sum of these dividers, as indicated in Equation 56.

$$R_T = R_L + \sum_{i=1}^{n} R_i \tag{55}$$

$$I_{SC_A} = \sum_{i=1}^{n} \frac{R_L + \sum_{j=i-1}^{i} R_j}{R_T} I_i + \frac{R_L}{R_T} I_L \tag{56}$$

By inspection, the equivalent resistance of Circuit 78b is identical to that of 78a. The short circuit current (Equation 57) is the sum of the series equivalent current I_{e_n} and the fraction of I_L and I,, supplied by the resistive divider. Combining terms under the same summation gives Equation 58. Substituting Equation 55 into Equation 58, and combining terms (Equation 59) allows the $\sum_{j=1}^{i-1} R_j$ and $\sum_{j=i}^{n} R_j$ summations to be combined. The combined terms cancel, giving Equation 60, identical to Equation 56. The two circuits thus have identical Norton Equivalent circuits.

$$I_{SC_B} = \sum_{i=1}^{n} \frac{\sum_{j=1}^{i-1} R_j}{\sum_{j=1}^{n} R_j} I_i + \frac{R_L}{R_T}(I_L + \sum_{i=1}^{n} \frac{\sum_{j=i}^{n} R_j}{\sum_{j=1}^{n} R_j} I_i) \quad (57)$$

$$I_{SC_B} = \sum_{i=1}^{n} \frac{R_T(\sum_{j=1}^{i-1} R_j) + R_L(\sum_{j=i}^{n} R_j)}{R_T(\sum_{j=1}^{n} R_j)} I_i + \frac{R_L}{R_T} I_L \quad (58)$$

$$I_{SC_B} = \sum_{i=1}^{n} \frac{(\sum_{j=1}^{i-1} R_j)(\sum_{j=1}^{n} R_j) + R_L(\sum_{j=1}^{i-1} R_j + \sum_{j=i}^{n} R_j)}{R_T(\sum_{j=1}^{n} R_j)} I_i + \frac{R_L}{R_T} I_L \quad (59)$$

$$I_{SC_B} = \sum_{i=1}^{n} \frac{\sum_{j=1}^{i-1} R_j + R_L}{R_T} I_i + \frac{R_L}{R_T} I_L \quad (60)$$

5.5 Network Solution Techniques

Once all the above simplifications have been performed on a network, the remainder is basically a mesh representing the backbone of the power distribution system. The voltage distribution of this network must be calculated. The following two sections examine methods for solving this network.

5.5.1 Direct Methods

The power network can be be written as an *n-1* by *n-1* system of equations Gv = i, where G is the conductance matrix, v are the voltages at the *n-1* nodes (referenced to ground), and i are the currents injected at these nodes. Since G is positive definite,[2] there exists a Cholesky decomposition G = LL', where L is lower triangular. Once this decomposition is known, the node voltages v can be calculated by forward solving the lower triangular system Ly = i, then back substituting into L'v = y to derive v.

The key to this algorithm is being able to calculate the LL' factorization quickly. This is done using an envelope implementation of the bordering method as described by

[2] A positive definite matrix is one for which $x^t Ax > 0$ for all x with at least one nonzero element. For the conductance matrix G, the term $v^t Gv$ has a physical interpretation; since Gv is the current injected at each node, $v^t Gv = v^t i$. The inner product v'i is the power dissipated by the network, which is always greater than zero for nonzero excitation. Conductance matrices are thus positive definite due to the law of conservation of energy.

George[16]. The matrix G can be written as a matrix M of order $n-1$, a column vector u representing the nth row, and a diagonal term s. Equation 63 gives the factorization of G assuming that M's factorization, $L_M L_M^t$, is known.

$$G = \begin{vmatrix} M & u \\ u^t & s \end{vmatrix} \tag{61}$$

$$M = L_M L_M^t \tag{62}$$

$$G = \begin{vmatrix} L_M & 0 \\ w^t & \sqrt{s - w^t w} \end{vmatrix} \begin{vmatrix} L_M^t & w \\ 0 & \sqrt{s - w^t w} \end{vmatrix} \tag{63}$$

$$w = L_M^{-1} u \tag{64}$$

The factorization can thus be computed starting at the upper left comer by setting $L_0 = \sqrt{g_{0,0}}$, then solving the lower triangular system $Law = u$ to calculate the second row. Adding this row to L_0 gives the decomposition $L_1 L_1^t$ of the leading submatrix M_1. Each subsequent row can be solved in the same manner, using the factorization L, calculated in the previous step.

The only remaining problem is deciding the node ordering for the matrix. Ariel uses the Reverse Cuthill-McKee method[18]. Given a starting node, its neighboring nodes are added to the matrix in increasing order of degree. Neighbors of the nodes just added are in turn sorted and included; this process continues until all nodes have been processed. This ordering is then reversed because the opposite ordering has been proven to always have an equal or smaller envelope than the forward one.

In processing CMOS power supply networks, the system of equations is usually solved more than once. Once the conductance matrix G is factored, the cost of calculating each time step is relatively small; it is just the time required to solve the two triangular systems. Direct techniques thus look attractive for technologies where the power supply current is dynamic.

5.5.2 Iterative Methods

Of the iterative solution methods, Successive Overrelaxation (SOR) seems to have the most promise. For resistor networks, SOR is simply successive application of Kirchoff's Current Law:

$$V_{i_t} = (1 - \omega)V_{i_{t-1}} + \omega\frac{I_i + \sum_{k=1}^{n} G_k V_k}{\sum_{k=1}^{n} G_k}$$

Each new guess at a node's voltage is derived by looking at the voltages of the neighboring nodes, then setting the value so that KCL is obeyed for this node. The overrelaxation factor ω is required for fast convergence; without it, voltages approach their final values very slowly.

The main disadvantage to overrelaxation is its sensitivity to the "diameter" of the network, i.e. the number of edges separating the two nodes which are furthest apart. Since each node directly affects only the voltages of its immediate neighbors, the number of iterations it takes for changes in one node to propagate throughout the network is proportional to the diameter. Fortunately, the simplification techniques previously described either remove many of the "tall", stringy sections of the network or replace them with a single resistor, so the simplified network's height is considerably less than that of the original system.

5.6 Results

The effectiveness of the above simplification procedures on several designs is shown in Figure 79. The total reduction in network size varied strongly with the network's topology; it ranged from a factor of 5 for one of the microprocessors to a factor of 50 for the multiplier. The tree and series simplifications were effective on all three designs, but the success of removing simple loops from the network varied widely. The pad drivers for MIPS-X have one configuration that yields many such loops; removing them reduced the total network size by 19%. However, in the multiplier, no simple loops were found. Whether this simplification is worth implementing is dependent on design style.

Solution times using direct factorization are shown in Table 15. The program was run on a Titan, an ECL RISC machine developed at the Digital Equipment Corporation

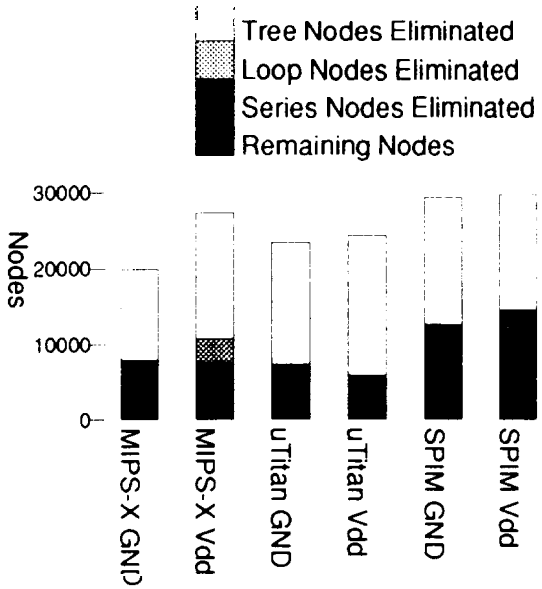


Figure 79: Results of Network Reduction

Western Research Laboratory that is about 15 times faster than a VAX-11/780. "Setup" includes the time required to calculate the LL' factorization, and "Solve" is the time required to solve the two triangular systems Ly = i and L'x = y and to do the back annotation for the trees and series. (For comparison, solving the first network, 'MIPS-X gnd', using spice2g6[39] on the same machine took about 14 hours.) The speedups here for the smaller networks are less impressive because the solution time for a single voltage vector is dominated by simply reading in the network. However, the speedup for each additional solution (the ratio of the 'Solve' columns) is considerably higher; it ranges between 4.5 and 254. This lower marginal cost for each additional solution permits a larger number of current distributions to be applied to the network.

Solution times for both the original and simplified networks using the iterative method are shown in Table 16. In this case, "Read" is the time required to parse the input file, "Setup" is the time to identify the tree, loop, and series configurations, and "Solve" is the time required to solve the core network and back-annotate the tree and series voltages. The speedup obtained is roughly proportional to the amount by which the original network could be reduced. The unsimplified networks for the second two designs require significantly longer to solve than the first because their network diameter is higher.

Power Network	Time in Seconds						
	Unsimplified			Simplified			
	Read	Setup	Solve	Read	Setup	Solve	Speedup
MIPS-X gnd	26.8	293.6	9.64	26.8	56.8	2.0	3.8
MIPS-X vdd	34.4	949.5	38 1.6	34.5	30.2	1.5	20.6
μTitan gnd	28.4	130.1	5.2	27.7	8.7	1.0	4.4
μTitan vdd	29.4	126.5	5.375	29.8	6.3	1.1	4.3
SPIM gnd	36.6	161.1	6.4	36.5	6.67	1.3	4.6
SPIM vdd	37.4	93.7	5.0	37.3	3.9	1.1	3.2

Table 15: Direct Method Solution Times

Of the two methods, direct solution looks more useful. It was faster for all but one of the test cases listed, and for cases where the network is to be solved more than once using different current distributions, each additional solution is relatively cheap.

Power Network	Time in Seconds						
	Unsimplified			Simplified			
	Read	Setup	Solve	Read	Setup	Solve	Speedup
MIPS-X gnd	26.8	3.2	235.1	27.5	2.9	42.8	3.6
MIPS-X vdd	34.4	4.6	399.0	36.5	19.7	28.8	5.2
μTitan gnd	27.7	3.7	1640.0	28.3	3.8	220.8	6.6
μTitan vdd	29.3	3.8	1097.4	29.3	2.7	60.7	12.2
SPIM gnd	36.6	5.0	1044.0	38.4	3.8	22.5	16.8
SPIM vdd	37.3	5.0	2281.4	38.0	3.6	6.3	48.5

Table 16: Iterative Method Solution Times

5.7 Conclusions

Three algorithms for efficient partitioning of resistor networks into small sections that can be solved quickly have been presented. The first algorithm analyzes trees by noting that current always flows toward the tree's root, the second processes simple loops using

Kirchoff's Voltage Law, and the third solves series configurations using superposition. The system remaining after such partitioning was readily solved using standard sparse positive definite matrix techniques. Two such techniques were explored: direct solution using the bordering method and iterative solution using successive overrelaxation. Solving using the partitioning algorithms provided an overall speedup between 3 and 50 for the designs tested.

Chapter 6

Results

Glendower: I can call spirits from the vasty deep.
Hotspur: Why, so can I, or so can any man,
But will they come for you when you call for them?

William Shakespeare
1 Henry IV

The previous four chapters described how the system produces resistance, current, and voltage profiles for the network. Now that the system is assembled, the next step is to see what it can discover about some real designs. This chapter contains plots showing the voltages and currents calculated by the system for the designs tested, and discusses some of their salient features.

The voltage distribution of the metal ground bus from the MIPS-X microprocessor datapath is shown in Figure 80. The values shown are the maximum over ten clock cycles. Dots that are completely black represent nodes where the voltage drop is half a volt or more. The datapath is connected to the padframe at the upper left and lower right comers. There are several notable features in this plot. The black column of nodes in the left side are part of the register file. In the MIPS-X architecture, Register 0 always returns a value of 0; half of each register file cell in this column is hardwired to ground. This connection is made in a peculiar way; the source of the access transistor is grounded via a polysilicon wire. Most of the drop seen in the drawing is due to the polysilicon, but it appears in the plot because of the small strip of metal required to connect between

polysilicon and diffusion.

The dark section in the upper left comer of the design is the floating point coprocessor interface. As evidenced by the sharp gradient, power for this interface is not connected to the rest of the datapath; instead of connecting to the clean power below it, the designer inexplicably extended down lines from the clock driver circuitry that sits just above [1]. The coprocessor interface bounces with the large transient voltages that occur when the clocks switch.

There is also a horizontal dark patch in the upper right comer. At the right of the datapath are the match tags for the instruction cache; sitting at the top of these tags are some fairly large buffers that drive the block select signals into the instruction cache, which sits above the datapath. These buffers are all powered via a minimum width metal line, which bounces substantially when they change state.

Despite these shortcomings, however, the power distribution for MIPS-X is roughly consistent with the designer's expectations. The designers were assuming a 0.5V drop on the supply lines; aside from the dummy register file column where the drop is not a problem, this estimate seems reasonable.

For μTitan, however, the voltage distribution plot highlights a substantial flaw in the power bus wiring. Figure 81 shows the ground voltage drops for the metal section of the microprocessor. At the top of the plot is the datapath, with the instruction cache sitting below it. The internal power wiring of the instruction cache was not modeled, so this section of the plot is blank. To the right of the instruction cache are a set of large drivers for the cache's column select and sense amplifier circuitry. The power wiring for these drivers is 66μm wide in the driver cells themselves, but is connected to the main power bus by a 4μm wide, 4mm long wire. The drop calculated along this wire is nearly two volts. Since the amount of current drawn by the drivers will be noticably reduced by the large ground bounce, the actual drop is somewhat lower.

Root-mean-square current densities for μTitan's ground bus are shown in Figure 82. Predictably, the same icache wire that gave high voltage drops also carries a large current density. (The actual density calculated for the wire is over 5 $mA/\mu m^2$; the scale has been compressed to show detail in some of the lower current density areas.) There are some

[1] Sadly, I designed this section of the chip.

other areas where the current density is rather high. In the datapath, ground is wired horizontally, with each two bits sharing a power line. There are few vertical connections between these horizontal buses, and they seem susceptible to electromigration. Depending on the data, the current drawn by each bit in the path will be different, as will the current that each horizontal power line must carry. This sometimes gives substantial voltage gradients between adjacent lines. At the occasional connections between these buses, large currents flow to try and equalize the potential.

Figure 83 shows the ground voltage drops for the SPIM multiplier. Here the scale runs from 0 to 1 volts; since the design is fully static, it does not have any functionality problems from the high drops, but it is losing some speed. Power runs horizontally in the design, with no vertical cross links, and adjacent power lines often have quite different drops. One possible solution would be to make the array a little wider and add vertical cross ties to spread the current move evenly among all the buses.

Figure 84 shows the root-mean-square current densities for SPIM. This plot is basically an inverse of the last one, with low current density grey in the center of the array and high density black at the edges. The scale here runs to 5 $mA/\mu m^2$, indicating a fairly aggressive design. It is probably worth considering widening these lines or adding vertical cross links.

Figure 85 shows the voltage distribution for the ground bus of the R6000 microprocessor, including package and bond wire drops. The on-chip voltage drops range from 20mV to 130mV. The design uses a standard-cell methodology, with power running horizontally in rows. The highest drops occur in the center section of standard cells, which are the farthest from the pads. The two open squares in the bottom left comer contain the register file cells; “ground” for the cells is a special reference voltage several hundred millivolts below real ground. This extra space is necessary to allow the file’s sense amplifier to operate properly. Since this virtual ground is not part of the actual ground network, the arrays appear blank.

Total solution times for the test circuits are given in Table 17. The CMOS designs were run on a 22 MHz Titan, an ECL RISC machine, while the ECL designs were run on a MipsCo RC3260, which runs at 25 Mhz.

Unsurprisingly, solution times for the CMOS designs are dominated by generating current patterns and solving the system. The SPIM times are the shortest, primarily because the SPIM simulation uses the smallest number of current patterns (about 200) compared to over 800 for MIPS-X and 600 for μTitan.[2] The solution times for Vdd and Ground are roughly consistent for each design except μTitan. This primarily arises from two factors: substrate contacts and icache power routing. Since this design contains no frontside substrate contacts, the resistance network for Ground is simpler, and the current profile contains no additional image current pulses. In the instruction cache, Vdd is gridded and Ground is not, giving Vdd a more complex network. These factors lead to the wide disparity in solution times.

For the ECL designs, the situation is reversed; resistance extraction dominates. Since the technology for these chips includes three layers of metal and and some 45-degree angles, there are substantially more rectangles to process. Producing currents and solving the system is fast because only one current distribution is involved, and the chips' power buses do not contain many meshes.

	Time (seconds)			
Circuit	Resistance Extraction	Current Estimation	System Solution	Total
MIPS-X Ground	933	1113	4829	6875
MIPS-X Vdd	586	1488	2996	5070
SPIM Ground	431	940	650	202 1
SPIM Vdd	567	1320	833	2720
μTitan Ground	408	1232	2612	4252
μTitan Vdd	843	3768	22130	2674 1
R6000 vcc	9168	537	106	9811
R6010 vcc	9392	360	106	9858
R6020 Vcc	3788	802	88	4679

Table 17: Total Analysis Times

[2] These solution times differ from those in the last chapter for two reasons. First, the networks used here are somewhat larger; the extractor was not allowed to merge diffusion and metal resistors, so that metal-only drops could be calculated. Second, extra consistency, voltage and current density checks were performed on the networks after each solution step.

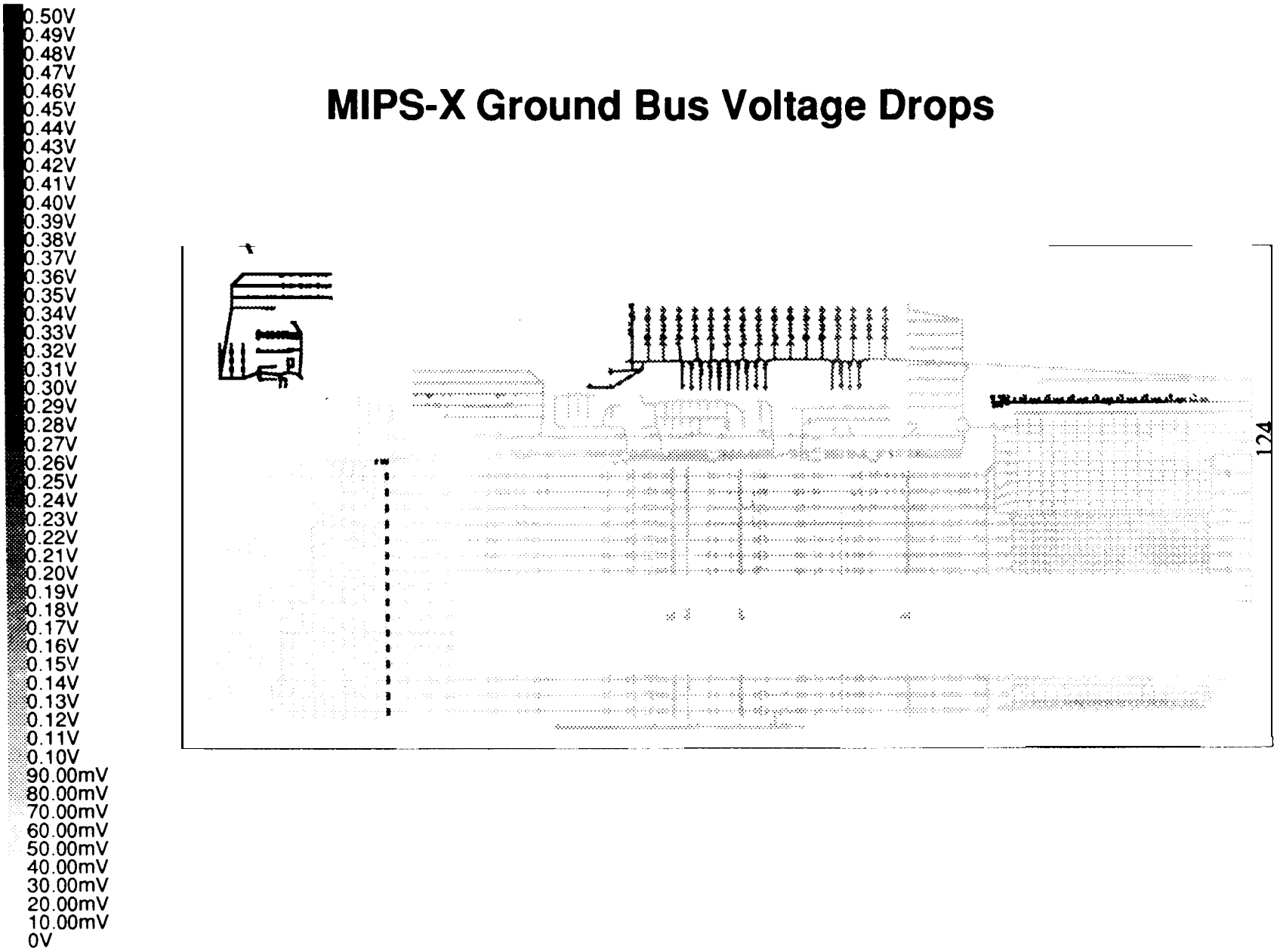


Figure 80: MIPS-X Ground Bus Voltage Plot

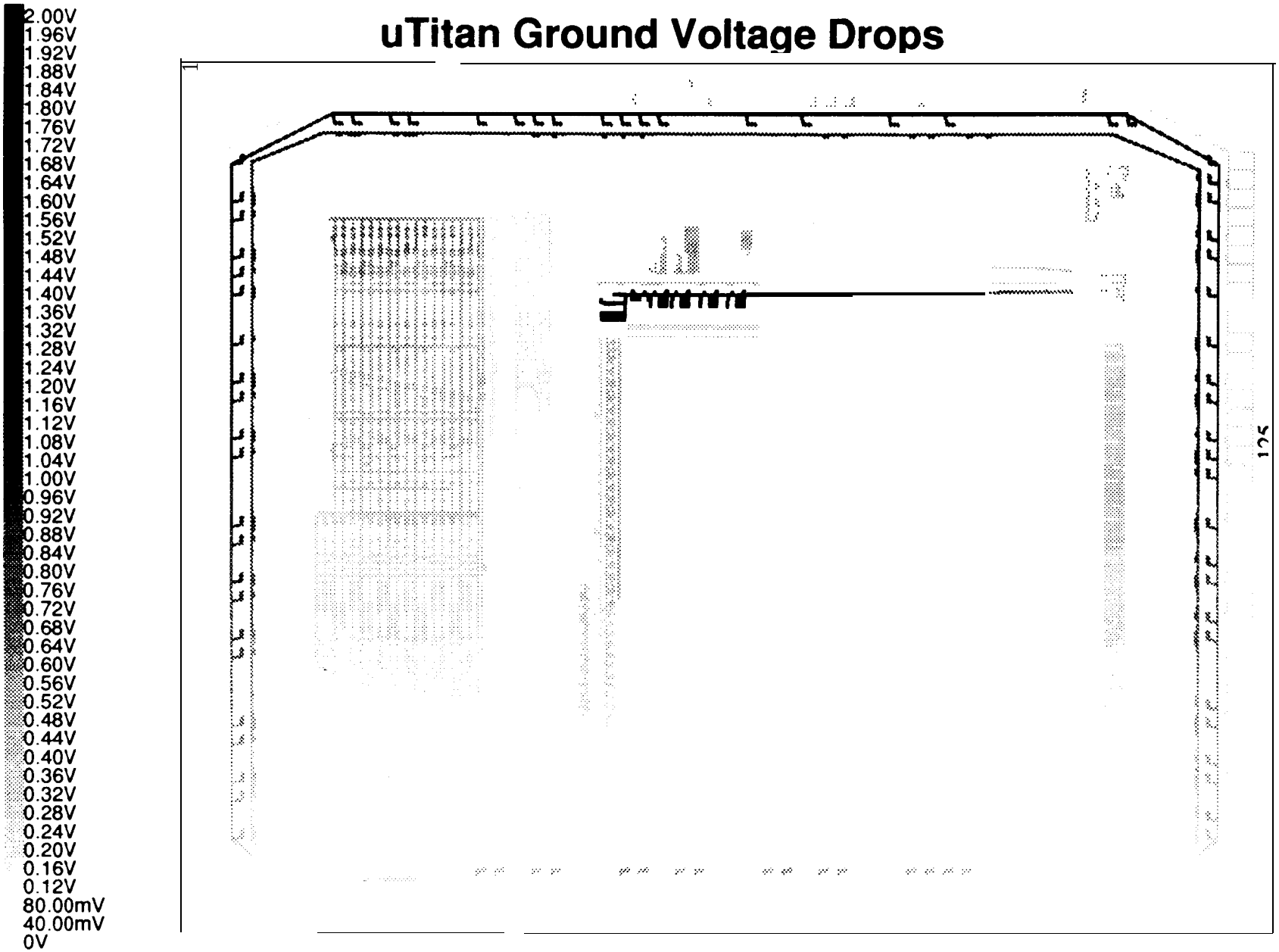


Figure 8 1: **uTitan** Ground Bus Voltage Plot

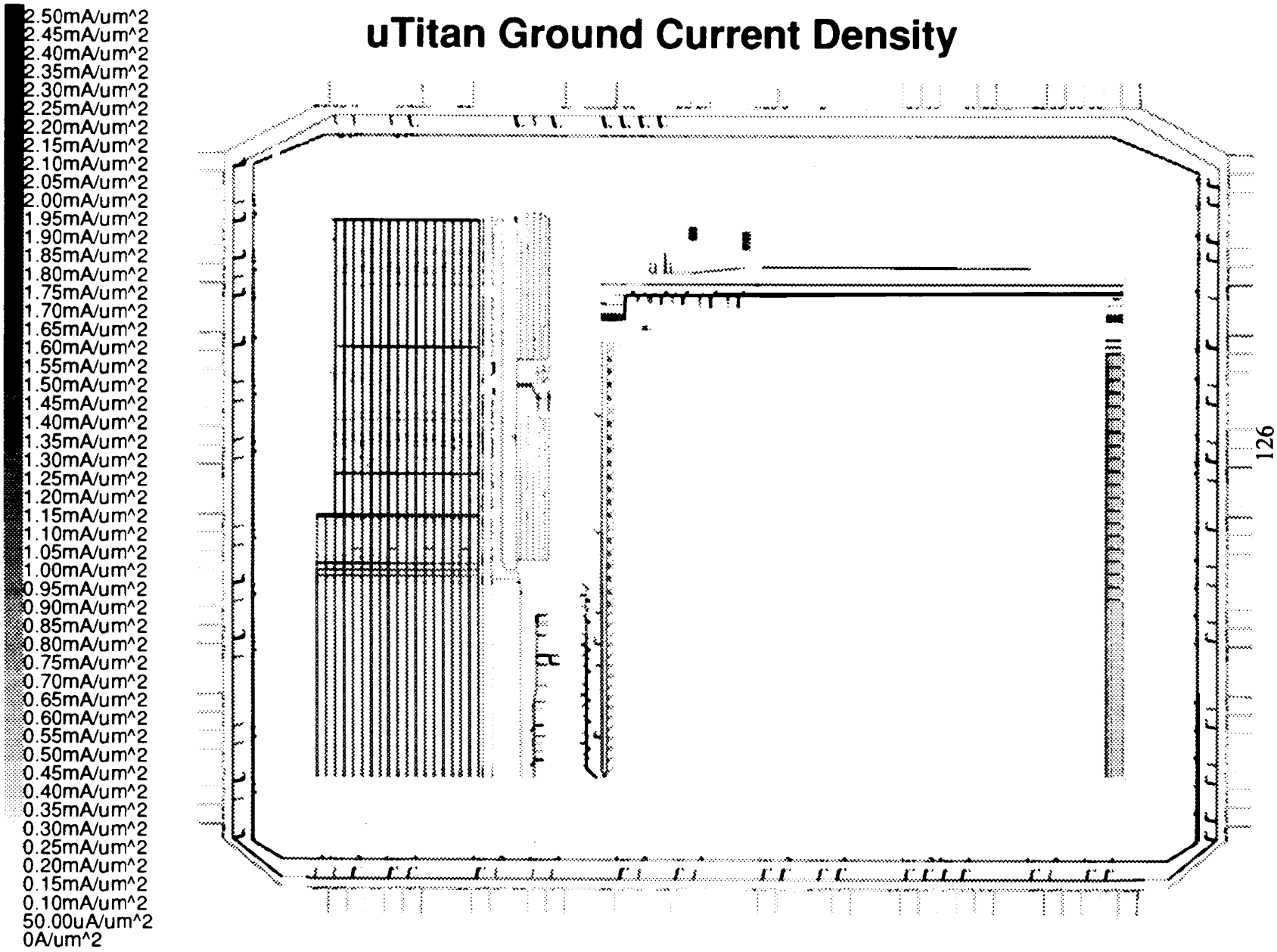


Figure 82: **uTitan** Current Densities (rms)

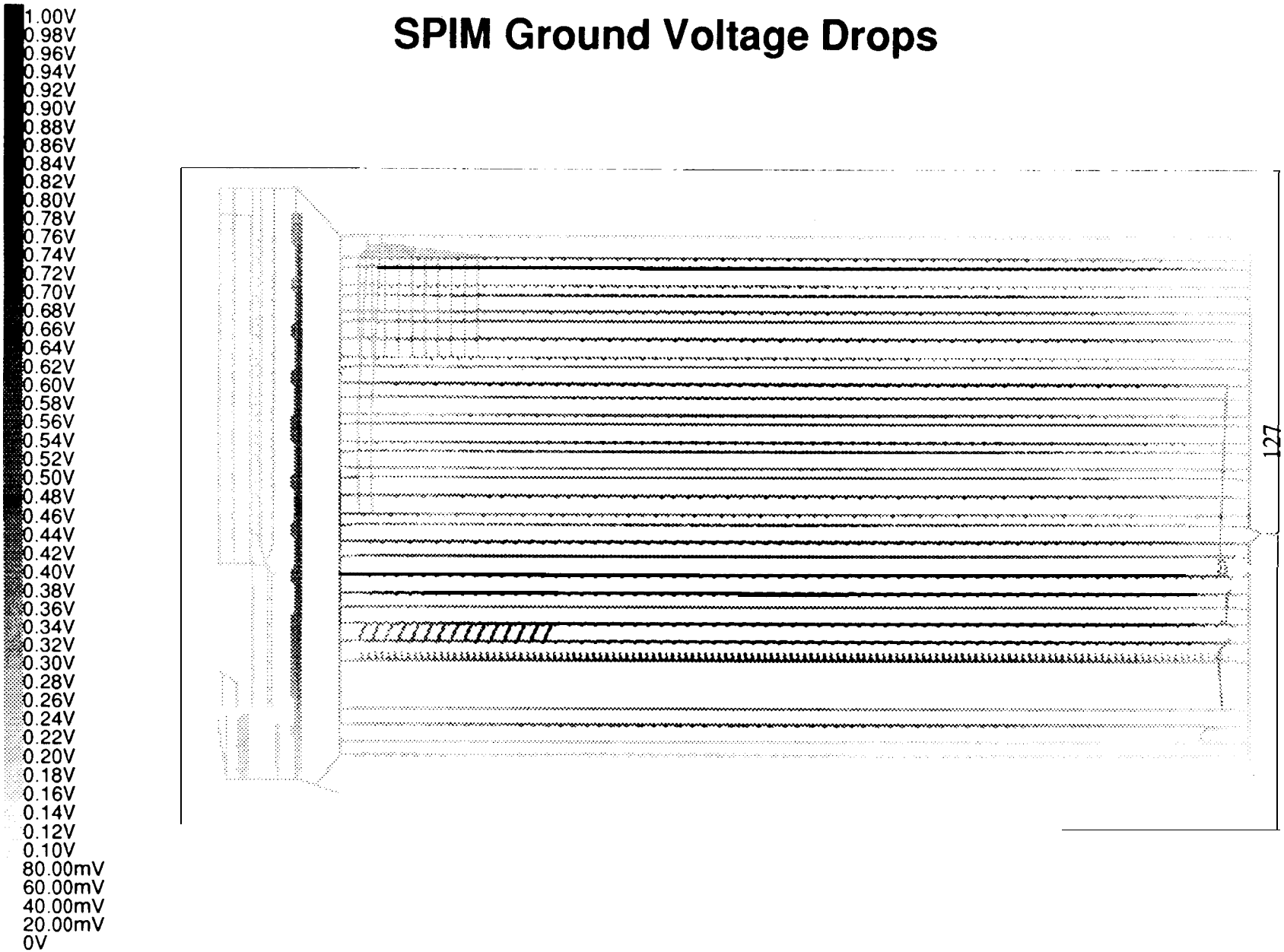


Figure 83: SPIM Ground Bus Voltage Plot

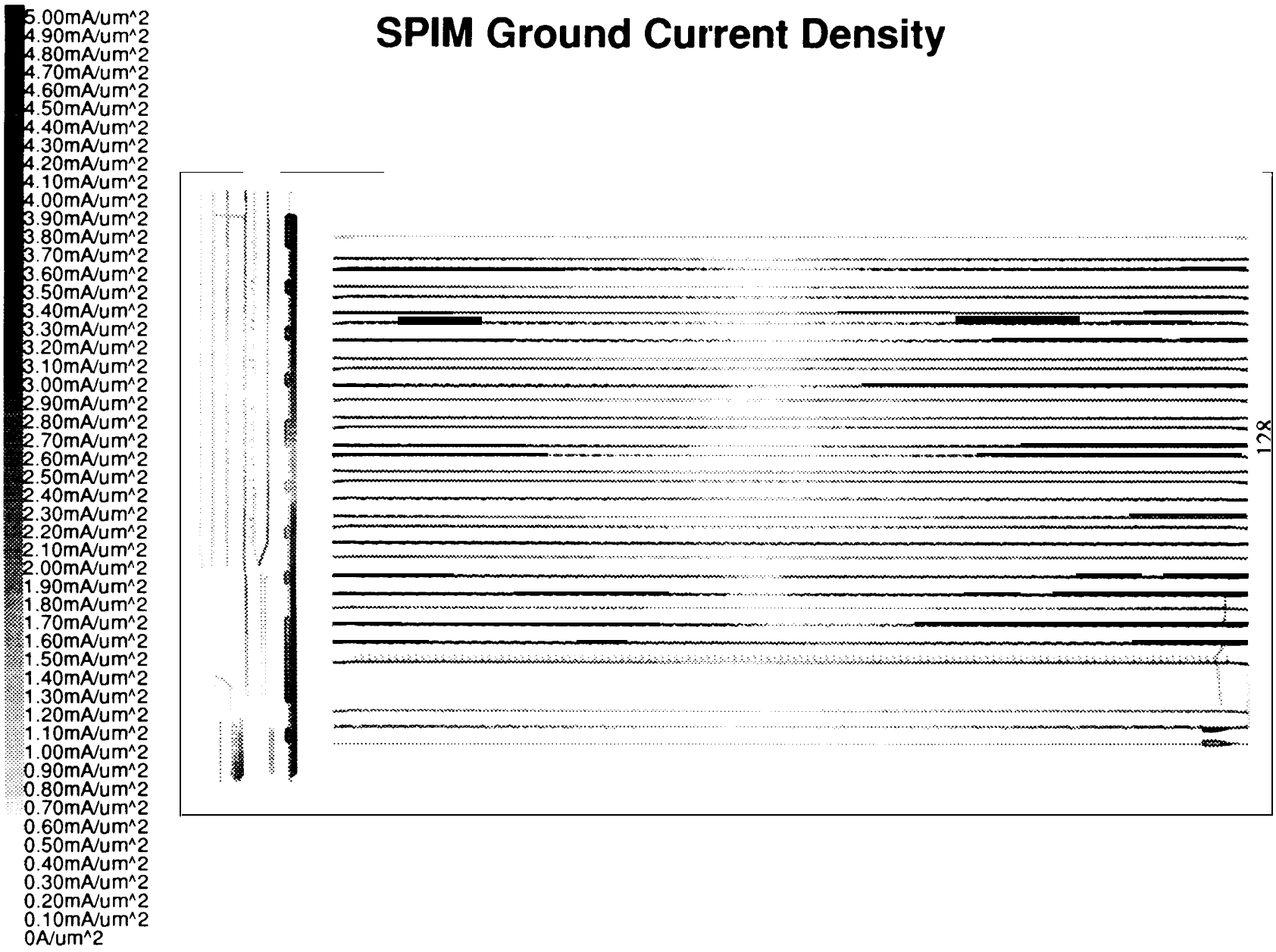


Figure 84: SPIM Current Densities (rms)

R6000 Vcc Bus Voltage Drops

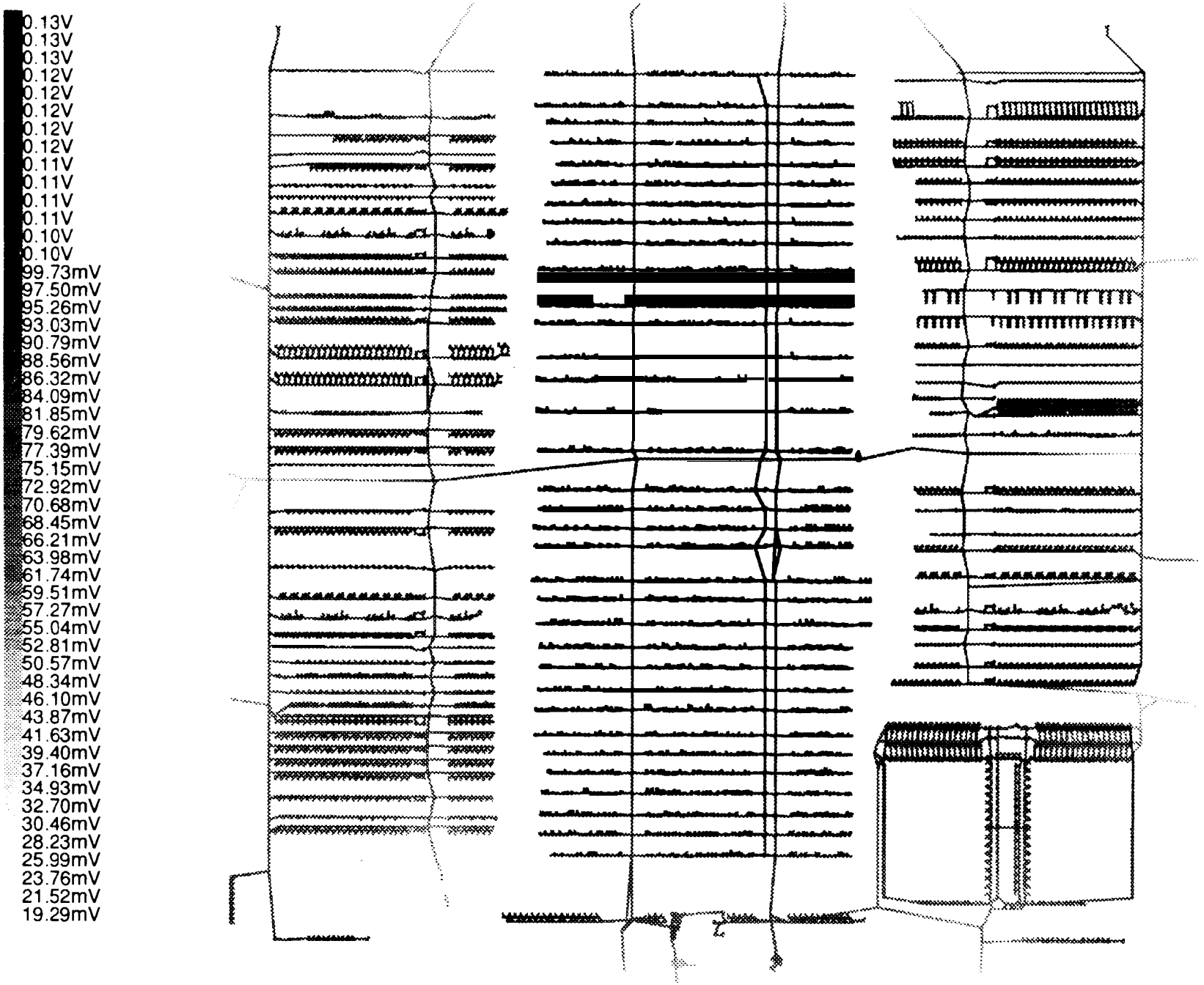


Figure 85: R6000 Vcc Voltage Drops

Chapter 7

Conclusions

I prithee,
Remember I have done thee worthy service,
Told thee no lies, made thee no mistakings, serv'd
Without grudge or grumblings.

William Shakespeare
The Tempest

An integrated circuit whose design is correct logically still may not function as intended if the system violates some of the underlying electrical constraints. Analyzing an entire chip can be difficult, however, due to its size. Design of a CAD tool to automate power and ground checks requires careful attention to the tradeoffs between speed and accuracy for each component.

For resistance extraction, a simple extractor that uses the standard square-counting approximation is best suited for power supply analysis. This method is surprisingly accurate, giving nearly the same results as finite element analysis in a fraction of the time. Even when compared to a fairly elaborate finite element extractor that used a library of previous solutions, the simple method is still two orders of magnitude faster.

In CMOS current estimation, the speed-accuracy tradeoffs surround time and pattern dependence. The design's maximum current consumption cannot be calculated without trying most or all of the input patterns; such current estimation is much too expensive to perform on large circuits. Instead, my estimator is based on the switch level simulator

Rsim. Estimating currents with Rsim allows the designer to see how her circuit performs under real operating conditions and to test particular patterns that seem to be causing problems. Each event inside Rsim produces a triangular current pulse whose rising and falling edges depend on the slope of the gate's inputs and the intrinsic delay of the gate's output. These triangular pulses provide a reasonable approximation to the actual current waveform and can be produced and manipulated quickly. Pulses are also produced for the image currents that flow into the substrate. Experiments indicated that ignoring these image currents entirely led to an unacceptably inaccurate current distribution, but that they could be estimated fairly easily using a simple algorithm based on each node's bounding box.

For ECL, there is also potentially a tradeoff to be made in selecting a current pattern or patterns; although the current magnitude is fairly constant, its distribution across the design varies with the input state. After describing a static estimator that finds and traces currents through the system, I showed that the current pattern is relatively insensitive to the circuit state; perturbations in the voltage and current distributions are minor, localized effects. A single current pattern is sufficient for ECL designs, making the resistance-current network fast to solve.

Given a tractable resistance network and current pattern, the linear solver must efficiently solve the system. I investigated three techniques for partitioning the network into smaller, more easily solved sections. First, subnetworks that form trees can be pruned from the graph and replaced by current sources with values equal to all the current injected in the tree. Second, simple configurations of loops can be solved using Kirchoff's Voltage Law. Finally, sections of the network that form long series chains can be replaced by a single resistor with current sources at either end. Once all these simple subnetworks have been removed, the remaining core system can be solved using standard sparse matrix techniques, and the subnetwork voltages can then be back-annotated.

The key to this system is making the individual components operate correctly in tandem. The resistance extractor had to produce a network that contained the essence of the power distribution system, yet was still amenable to solution. The two current estimators needed to produce current profiles that accurately modelled the system's activity, but were not too complicated to produce or to use. The linear solver had to be tailored to

capitalize on the special topologies of power networks and efficiently handle the most common configurations. With each part tailored to mesh with the others, Ariel provides insight for entire designs efficiently and accurately.

Although Ariel is a usable system now, there are both changes and extensions to the existing system that would enhance its utility. The biggest obstacle to analyzing chips of arbitrary size is the memory usage of the resistance extractor; because it flattens the power bus before extracting it, it is quite profligate in memory use. Extraction without flattening is not an easy problem, however. Hierarchical extraction is quite difficult unless cell overlap is extremely restricted; such restrictions are a nuisance to designers and inconsistent with Magic's philosophy. Incrementally flattening a section, then extracting it is also not trivial; the extractor must determine which rectangles in a region correspond to the power bus (or else extract everything), it must produce the correct resistance even for rectangles that are intersected by the region boundary, and it must correctly assemble the parts.

There are several areas in current estimation that deserve additional attention. To date, no entirely satisfactory method of estimating currents for CMOS circuits has been found. The probabilistic methods of Burch produce expected instead of peak voltages and currents for the system, while my simulation based approach is pattern dependent. A ideal approach would produce currents that near worst case and avoid gross overestimation for circuits with limited activity.

The next major extension will be providing current estimation for BiCMOS. The fastest BiCMOS circuits may use ECL configurations for critical paths and MOS circuits for less critical ones in order to save power. Operating ECL circuits in a noisy MOS environment will accentuate the need for power supply noise analysis. The biggest challenge in current estimation for these designs is correct analysis of hybrid circuits, where neither a distributed RC tree model nor a current steering model is entirely accurate.

Finally, I would like to develop a useful set of postprocessing tools. The voltage and current profiles that Ariel produces are basically raw data; what the designer would really like to know is how the system's power noise is going to affect his particular design. Does a given circuit have sufficient noise margin to tolerate the supply noise? How can the placement of cells be modified to equalize the current patterns? Can the circuit timing be

modified to redistribute power consumption more evenly across the clock cycle? These types of questions combine the raw data calculated by Ariel with additional concerns about the circuit, about the layout, and even about the underlying logic. This analysis is critical for fullest utilization of power supply analysis.

Appendix A

Triangular Finite Element Derivation

Finite element analysis is based on the calculus of variations. If the integral I(v) of a functional F (Equation 65) has a stationary point where $\partial F/\partial v = 0$ for some value of v, then it can be shown that Equation 66 must be valid[13].

$$I(v) = \int \int_R F(x, y, v, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}) dx dy \tag{65}$$

$$\frac{\partial}{\partial x}\frac{\partial F}{(\partial \frac{\partial v}{\partial x})} + \frac{\partial}{\partial y}\frac{\partial F}{(\partial \frac{\partial v}{\partial y})} - \frac{\partial F}{\partial v} = 0 \tag{66}$$

When Formula 66 is applied to

$$F(v) = \frac{\sigma}{2}((\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2) \tag{67}$$

the result is Laplace's equation.

$$\sigma(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) = 0 \tag{68}$$

Finding the solution to Laplace's equation for a given region is thus equivalent to finding a function $v(x, y)$ which yields a stationary point when the functional of Equation 67 is applied, integrated with respect to x and y, then differentiated with respect to v. Finding the potential in this manner does not appear any easier than solving for v directly. However, the region R can be broken up into a set of smaller elements, E, and the total integral approximated as the sum of the integrals of the parts.

$$I(v) \approx \sum_{E} \int \int_{E} F(x, y, v, \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}) dx dy \tag{69}$$

If the Function v used for each small element is simple enough so that Equation 67 is independent of x and y, then calculating the previous integral is simple; it is **just the area** of the element times the function $F(v)$. Stationary points of this integral can **be** calculated by setting $\partial F(v)/\partial v_e = 0$, where v_e is the voltage function for the element. A solution for the entire system requires that these partial derivatives for each element be satisfied simultaneously. The entire integral is thus replaced by a set of simultaneous equations.

To illustrate how this method works, this section derives the element equations for a triangular element, then describes how the equations for each individual element can be combined. Triangles are the most commonly used finite element because any two-dimensional polygon may broken down into them. A typical triangular element is shown in Figure 17. It has vertices $\{i, j, k\}$ at the points (s,. y_i),(x_j. y_j), and (x_k. y_k); the potentials at these points are V_i, V_j, and V_k, respectively. The potential for any point in the element will be defined as some linear combination of vertex potentials (Equation 70).[1] The interpolating functions ϕ will be linear functions of x and y.

$$V(x, y) = \phi_i(x, y)V_i + \phi_j(x, y)V_j + \phi_k(x, y)V_k \tag{70}$$

$$\phi_i = a_i + b_i x + c_i y$$

$$\phi_j = a_j + b_j x + c_j y$$

$$\phi_k = a_k + b_k x + c_k y$$

The first step is to calculate the coefficients for these functions ϕ. The equation for the voltage forms a plane,

$$V(x, y) = A + Bx + cy \tag{71}$$

which has to have values V_i,V_j,and V_k at the three vertices. The constants A, B, and C can be determined by solving for these three values simultaneously.

'Allowing the voltage to vary linearly is equivalent to requiring the current density in the **element to be constant,** since $J = -\sigma \nabla V$.

$$\begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} V_i \\ V_j \\ V_k \end{bmatrix} \tag{72}$$

$$\begin{aligned} A &= ((x_j y_k - x_k y_j)V_i + (x_k y_i - x_i y_k)V_j + (x_i y_j - x_j y_i)V_k)/Area \\ B &= ((y_j - y_k)V_i + (y_k - y_i)V_j + (y_i - y_j)V_k)/Area \\ C &= ((x_k - x_j)V_i + (x_i - x_k)V_j + (x_j - x_i)V_k)/Area \\ Area &= x_j y_k + x_i y_j + x_k y_i - x_j y_i - x_i y_k - y_j x_k \end{aligned} \tag{73}$$

$Area$, the determinant of the 3x3 matrix, is equal to twice the area of the element. This can be seen by shifting and rotating the element. If the element is moved so that point i is at the origin, the equation for Area can be rewritten with the transformed coordinates.

$$\begin{aligned} x'_i &= 0, \quad y'_i = 0 \\ x'_j &= x_j - x_i, \quad y'_j = y_j - y_i \\ x'_k &= x_k - x_i, \quad y'_k = y_k - y_i \\ Area &= x'_j y'_k - y'_j x'_k \end{aligned} \tag{74}$$

The element can then be rotated by an angle θ so that the edge ij is coincident with the x-axis.

$$\begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} = \begin{bmatrix} \frac{x'_j}{\sqrt{x'^2_j + y'^2_j}} & \frac{-y'_j}{\sqrt{x'^2_j + y'^2_j}} \\ \frac{y'_j}{\sqrt{x'^2_j + y'^2_j}} & \frac{x'_j}{\sqrt{x'^2_j + y'^2_j}} \end{bmatrix} \tag{75}$$

The element's area is half the length of edge ij times the height, which is y'_k after transformation by Equation 75. This area is half the value for $Area$ given in Equation 74.

$$\triangle_e = \left(\frac{1}{2}\sqrt{x'^2_j + y'^2_j}\right)\frac{-y'_j x'_k + x'_j y'_k}{\sqrt{x'^2_j + y'^2_j}} = x'_j y'_k - y'_j x'_k \tag{76}$$

The coefficients C are linear functions of the vertex voltages, and may be broken into their constituent components by inspection.

$$a_i = (x_j y_k - x_k y_j)/Area,\ a_j = (x_k y_i - x_i y_k)/Area,\ a_k = (x_i y_j - x_j y_i)/Area$$
$$b_i = (y_j - y_k)/Area, \qquad b_j = (y_k - y_i)/Area, \qquad b_k = (y_i - y_j)/Area$$
$$c_i = (x_k - x_j)/Area, \qquad c_j = (x_i - x_k)/Area, \qquad c_k = (x_j - x_i)/Area \tag{77}$$

The next step is calculating the integral l(v). When Formula 67 is applied to the element's voltage relation (Equation 70), the resulting integrand is independent of x and y, and the integral is just the area of the element.

$$I(v) = \int\int_{\triangle} \frac{\sigma}{2}\left(\left(\frac{b_i V_i + b_j V_j + b_k V_k}{Area}\right)^2 + \left(\frac{c_i V_i + c_j V_j + c_k V_k}{Area}\right)^2\right) dx\,dy \tag{78}$$

$$I(v) = \frac{\sigma}{4\,Area}\left((b_i V_i + b_j V_j + b_k V_k)^2 + (c_i V_i + c_j V_j + c_k V_k)^2\right) \tag{179}$$

This expression for I(v) can be differentiated to give three equations for the node potentials in terms of one another.

$$\frac{\partial I}{\partial \phi_i} = 0, \frac{\partial I}{\partial \phi_j} = 0, \frac{\partial I}{\partial \phi_k} = 0 \tag{80}$$

$$\frac{\sigma}{4A}\begin{bmatrix} b_i^2 + c_i^2 & b_i b_j + c_i c_j & b_i b_k + c_i c_k \\ b_i b_j + c_i c_j & b_j^2 + c_j^2 & b_j b_k + c_j c_k \\ b_i b_k + c_i c_k & b_j b_k + c_j c_k & b_k^2 + c_k^2 \end{bmatrix}\begin{bmatrix} V_i \\ V_j \\ V_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{81}$$

When the values of a. *b.* c from Equation 77 are substituted in this array, it can be rewritten in terms of three new constants, $G_1, G_2,$ **and** G_3.

$$\begin{bmatrix} -(G_1 + G_2) & G_1 & G_2 \\ G_1 & -(G_1 + G_3) & G_3 \\ G_2 & G_3 & -(G_2 + G_3) \end{bmatrix}\begin{bmatrix} V_i \\ V_j \\ V_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{82}$$

$$G_1 = \frac{\sigma}{4A}(x_j x_k + x_i x_k - x_k^2 - x_i x_j + y_j y_k + y_i y_k - y_k^2 - y_i y_j) \tag{83}$$

$$G_2 = \frac{\sigma}{4A}(x_i x_j + x_j x_k - x_j^2 - x_i x_k + y_i y_j + y_j y_k - y_j^2 - y_i y_k)$$
$$G_3 = \frac{\sigma}{4A}(x_i x_j + x_i x_k - x_i^2 - x_j x_k + y_i y_j + y_i y_k - y_i^2 - y_j y_k) \quad (84)$$

These constants have the units of conductance; the finite element defined by nodes $\{i, j, k\}$ can be replaced by three discrete resistors with values $1/G_1, 1/G_2, 1/G_3$. The matrix for the entire system is thus equivalent to the discrete resistor network defined by the finite elements.

Bibliography

[I] Alfred Aho, John Hopcraft, and Jeffrey Ullman, ***The Design and Analysis of Computer*** *Algo-* ***rithms,*** Addison-Wesley, Reading, Massachusetts, 1974,172-176.

[2] Erich Barke, "Resistance Calculation From Mask Artwork Data by Finite Element Method", ***Proceedings of the 22nd Design Automation Conference,*** 1985, 305-3 11.

[3] J.D. Bastian, M. Ellement, P.J. Fowler, C.E. Huang, and L. P. McNamee, "Symbolic Parasitic Extractor for Circuit Simulation (SPECS)", ***Proceedings of the 20th*** *Design Automation* ***Conference, 1983, 346-352.***

[4] F.H. Branin Jr., 'The Analysis and Design of Power Distribution Nets on LSI chips", ***Proceedings of the International Conference on Circuits and Computers, 1980,*** 785-790.

[5] F.H. Branin Jr, and K. Huseyin, ***Problem Analysis in Science and Engineering,*** Academic Press, New York, 1977, 56-66.

[6] R.E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems", *IEEE* ***Transactions on Computers,*** Vol. C-33, No. 2, February 1984, pp. 160-177.

[7] R.E. Bryant, D. Beatty, K. Brace, K. Cho, and T. Sheffler, "COSMOS: A Compiled Simulator For MOS Circuits", ***Proceedings of the 24th Design Automarion Conference, 1987,*** pp 9-16.

[8] Basant R. Chawla and Hermann K. Gummel, "A Boundary Technique for Calculation of Distributed Resistance", ***IEEE Transactions on Electron Devices,*** Vol. ED-17, No. 10, October 1970, pp. 915-925.

[9] S. Chowdhury, "An Automated Design of Minimum-Area IC Power/Ground Nets", ***Proceedings of the 24th Design Automation Conference,*** 1987, 223-229.

[IO] S. Chowdhury and J.S. Barkatullah, "Estimation of Maximum Currents in MOS IC Logic Circuits", ***IEEE Transactions on Computer*** *Aided* ***Design,*** Vol. Cad-9, No. 6, June 1990. 642-*654.*

[1 1] Paul Chow, editor. ***The MIPS-X RISC Microprocessor,*** Kluwer Academic Publishers, Nor-well, Massachusetts, 1989, p. 122.

[12] Chomg-Yeong Chu, ***Improved Models for Switch Level Simulation***, Technical Report CSL-TR-88-368, Stanford University, November 1988.

[13] R. Courant and D. Hilbert, ***Methods of Mathematical Physics,*** Interscience Publishers, New York, 1953, 191-193.

[14] An-Chang Deng, Yan-Chyuan Shiau, and Kou-Hung Loh, "Time Domain Current Waveform Simulation of CMOS Circuits", ***1988 IEEE International Conference on Computer Aided Design Digest of Technical Papers,*** 1988, 208-211.

[15] D. T. Fitzpatrick, "MEXTRA: A Manhattan Circuit Extractor", Electronic Research Lab. Memo M82/42, Electronics Research Laboratory, University of California, Berkeley, January, 1982.

[16] Alan George and Joseph W-H Liu, ***Computer Solution of Large Sparse Positive Definite Systems,*** Englewood Cliffs, Prentice Hall, 1981.

[17] Alan George, "Computer Implementation of the Finite Element Method", Stanford Technical Report STAN-CS-208, Stanford University, 1971.

[18] Alan George and Joseph W-H Liu, "An Implementation of a Peripheral Node Finder", ***ACM Transactions on Math Software 5,*** 1979, 286-295.

[19] Lance A. Glasser and Daniel W. Dobberpuhl, ***The Analysis and Design of VLSI Circuits,*** 133-134. Addison Wesley, 1985.

[20] P.M. Hall, "Resistance Calculations for Thin Film Patterns", ***Thin Solid Films,*** Vol. 1, 1967/68, pp. 277-295.

[21] Joseph E. Hall, Dale E. Hocevar, Ping Yang, and Michael J. McGraw, "SPIDER-A CAD System for Modeling VLSI Metallization Patterns", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-6, No. 6, November 1987, 1023-1031.

[22] M. Glez Harbour and J. M. Drake, "Calculation of Multiterminal Resistances in Integrated Circuits", ***IEEE Transactions on Circuits and Systems***, Vol CAS-33, No. 4, April 1986, 462-***465.***

[23] Mark Alan Horowitz, ***Timing Models for MOS Circuits,*** Ph.D. Thesis, Stanford University, 1983.

[24] Mark Horowitz, Paul Chow, Don Stark, Richard T. Simoni, Arturo Salz, Steven Przybylski, John Hennessy, Glenn Gulak, Anant Agarwal and John Acken, "MIPS-X: A 20-MIPS Peak, 32-bit Microprocessor with On-Chip Cache", ***IEEE Journal of Solid State Circuits,*** Vol SC-22, ***No. 5,*** October 1987, 790-799.

[25] Mark Horowitz and Robert W. Dutton, "Resistance Extraction from Mask Layout Data", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-2, No. 3, July 1983, 145-150.

[26] Norman P. Jouppi, "Derivation of Signal ***Flow*** Direction in MOS VLSI", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-6, No. 3, May 1987, 480-490.

[27] Norman P. Jouppi, "Timing Analysis and Performance Improvement of MOS VLSI Designs", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-6, No. 4, July 1987, 650665.

[28] N. Jouppi, J. Tang, and J. Dion, "A 20 MIPS (Sustained) 32b CMOS Microprocessor with 64b Data Bus", ***1989 IEEE International Solid State Circuits Conference Digest of Technical Papers,*** 1989, 84-85.

[29] Russell Kao, Bob Alverson, Mark Horowitz and Don Stark, "Bisim: A Simulator for Custom ECL Circuits",*1* 988 ***IEEE International Conference on Computer Aided Design Digest of Technical Papers, 1988, 62-65.***

[30] Kuniyasu Kawarada, Masao Suzuki, Hisakazu Toyoda, and and Yoshisuke Kondo, "A Fast 7.5ns Access 1K-Bit RAM for Cache Memory Systems", ***IEEE Journal of Solid State Circuits,*** Vol SC-13, No. 5, October 1978, 656-663.

[31] Albertus J. Kemp, Jacobus A Pretorius, and Willem Smit, "The Generation of a Mesh for Resistance Calculation in Integrated Circuits", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-7, No. 10, October 1988, 1029-1037.

[32] M.R. Lightner et al, "CSIM: The Evolution of a Behavioral Level Simulator from a Functional Simulator: Implementation Issues and Performance Measurements", ***Digest of Technical Papers, 1985 IEEE International Conference on Computer-Aided Design, 350-352.***

[33] Hu Ei Ling, "High-Speed Binary Adder", ***IBM Journal of Research and Development,*** Vol. 25, Nos. ***2*** and 3, May 1981, 156-166.

[34] Jorge Machek and Siegfried Selberherr, "A Novel Finite Element Approach to Device Modelling", ***IEEE Transactions on Electron Devices,*** Vol ED-30, No. 9, September 1983, 1083-1091.

[35] Steven P. McCormick, "EXCL: A Circuit Extractor for IC Designs", ***Proceedings** of **the 21st Design Automation Conference,*** 1984, 616-623.

[36] Takashi Mitsuhasht and Kenji Yoshida, "A Resistance Calculation Algorithm and Its Application to Circuit Extraction", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-6, No. 3, May 1987, 337-345.

[37] Shojiro Mori and James A. Wilmore, "Resistance Extraction in a Hierarchical IC Artwork Verification System", ***Digest** of **Technical Papers, IEEE International Conference on Computer Aided Design, 1985,*** 196-198.

[38] Andrew S. Moulton, "Laying the Power and Ground Wires on a VLSI Chip", ***Proceedings** of **the 20th Design Automation Conference,*** 1983, 754-755.

[39] L.W. Nagel, "SPlCE2: A Computer Program to Simulate Semiconductor Circuits", Memorandum No. ERL-M520, Electronics Research Laboratory, University of California, Berkeley, 1975.

[40] Farid Najm, Richard Burch, Ping Yang, and lbrahim Hajj, "Crest - A Current Estimator for CMOS Circuits", ***1988 IEEE International Conference on Computer Aided Design Digest of Technical Papers,*** 1988, 204-207.

[41] F.N. Najm, R. Burch, P. Yang, and I.N. Hajj, "Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-9, No. 4, April 1990, 439-450.

[42] Zeev ***Nehari, Conformal Mapping,*** Dover Publications, New York, 1975, pp. 61-65.

[43] Yoshio Okamura, Yoshito Muraishi, Takashi Sato, and Yasuhiro lkemoto, "LAS: Layout Pattern Analysis System With New Approach", ***Proceedings** of **the IEEE International Conference on Circuits and Computers,*** 1982, 308-3 11.

[44] John K. Ousterhout, Gordon T. Hamachi, Robert N. Mayo, Walter S. Scott, and George Taylor, "Magic: A VLSI Layout System", ***Proceedings** of **the 21st Design Automation Conference,*** 1984. 152-159.

[45] J.K. Ousterhout, "Comer Stitching: A Data-Structuring Technique for VLSI Layout Tools", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-3, No. 1, January 1984, 87-100.

[46] J.K. Ousterhout, "A switch-level timing verifier for digital MOS VLSI", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-4, No. 3, July 1985, 336-348.

[47] Tomoyuki Ozaki, Jun Yoshida, Masahuro Kosaka, "PANAMAP-1: A Mask Pattern Analysis Program for IC/LSI – Centerline Extraction and Resistance Calculation –", ***Proceedings of the 1980 International Symposium on Circuits and Systems,*** 1979, 1020- 1024.

[48] David Roberts, Tim Layman, and George Taylor, "An ECL RISC Microprocessor Designed for Two Level Cache", ***Compcon 1990 Proceedings, 1990, 228-23*** 1.

[49] H-J. Rothermel and D. A. Mlynski, "Computation of Power Supply Nets in VLSI Layout", ***Proceedings of the 18th Design Automation Conference,*** 1981, 37-47.

[50] Jorge Rubenstein, Paul Penfield, Jr. and Mark A. Horowitz, "Signal Delay in RC Tree Networks", ***IEEE Transactions on Computer Aided Design,*** Vol. Cad-2, No. 3, July 1983, 202-2 10.

[51] Mark Santoro and Mark Horowitz, "A Pipelined 64 x 64b Iterative Array Multiplier", 1988 ***IEEE International Solid State Circuits Conference Digest of Technical Papers, 1988, 36-37.***

[52] Walter S. Scott and John K. Ousterhout, "Magic's Circuit Extractor", ***Proceedings of the 22nd Design Automation Conference,*** 1985, 286-292.

[53] Don Stark and Mark Horowitz, "REDS: Resistance Extraction for Use in Digital Simulation", ***Proceedings of the 24th Design Automation Conference,*** 1987, 570-573.

[54] Don Stark and Mark Horowitz, "Analyzing Power Supply Networks Using Ariel", ***Proceedings of the 25th Design Automation Conference***, 1988, 460-464.

[55] Shun Lin Su, Vasant B. Rao, and Timothy N. Trick, "HPEX: A Hierarchical Parasitic Circuit Extractor", ***Proceedings of the 24th Design Automation Conference,*** 1987, 566-569.

[56] Zahir A. Syed and Abbas El Gamal, 'Single Layer Routing for Power and Ground Networks in ***Integrated*** Circuits", ***Journal of Digital Systems,*** Vol. 6, No. 1, Spring 1982, 53-63.

[57] ***C.*** J. Terman, ***Simulation Tools for Digital LSI Design,*** Ph.D. Thesis, MIT Department of Electrical Engineering and Computer Science, 1983.

[58] Akhilesh Tyagi, ***The Role of Energy in VLSI Compurarions,*** Technical Report 88-06-05, University of Washington, Seattle, June 1988.

[59] Harry J.M. Veendrick, "Short Circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer ***Circuits", IEEE Journal of Solid State Circuits,*** Vol SC- 19, No. 4, August 1984, 468-473.

[60] T.E. Williams, M. Horowitz, R. L. Alverson, and T.S. Yang, "A Self-Timed Chip for Division", ***Advanced Research in VLSI: Proceedings*** *of the* ***1987 Stanford Conference,*** MIT Press, Cambridge, 1987, 75-96.

[61] Hiroshi Yoshimura, Kazuo Tansho, Norihiko Ohwada, and Tadashi Nishide, "An Algorithm for Resistance Calculation From IC Mask Pattern Information", ***Proceedings*** *of* ***the 1979 International Symposium on Circuits and Systems*** 1979, 478-48 1.