19)

Analysis and Modeling of Capacitive Coupling Along Metal Interconnect Lines

by

Andrew K. Percey

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

Bachelor of Science in Electrical Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Author ....
Department of Electrical Engineering and Computer Science
May 28, 1996

Eng.

Certified by ....
Jacob White
Professor of Electrical Engineering
Thesis Supervisor

Accepted by ....
F. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Analysis and Modeling of Capacitive Coupling Along Metal Interconnect Lines

by

Andrew K. Percey

Submitted to the Department of Electrical Engineering and Computer Science
on May 28, 1996, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Electrical Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Electrical signals propagating along metal interconnect lines within contemporary microchips experience significant delay and noise due to capacitive coupling effects. Analysis and modeling of these effects was performed at the author's VI-A Internship company. Numerous CAD circuit simulations were performed to acquire a better understanding of these coupling effects. A program was created to assist circuit designers in analyzing these effects for their particular circuit topologies. Two signal buses that were potentially at high-risk for coupling problems were analyzed in detail, and recommendations were made for reducing such problems. Finally, CAD circuit simulation data was compared with experimental data to determine the modeling accuracy of the CAD tools being used with respect to capacitive coupling.

Thesis Supervisor: Jacob White
Title: Professor of Electrical Engineering

Acknowledgments

I would like to express my sincere gratitude to David Chin, my supervisor for my final six-month internship assignment at Intel, for providing me with the opportunity to engage in this thesis work within his group and for supplying the guidance and support necessary to make it a successful endeavor.

I owe much of my happiness and success at Intel to George Dallas, Technical Program Manager with Corporate College Recruiting, who spent much time and effort in helping me to find worthwhile and challenging internships within the company for the past three years and who was always been generous with his honest and friendly advice.

Many thanks to Professor Jacob White, my thesis advisor at MIT, who provided encouragement and direction for the thesis and was very open to the ever-changing plans for the focus and scope of my work.

I especially want to recognize my wonderful parents, Kenneth and Ellen Percey, who have made both my academic and non-academic successes possible by always giving me nothing less than wholly unconditional love, support, and understanding for all of my chosen endeavors. I hope I will always make them proud of me.

Finally, I owe deep thanks to my dear sister, Deborah O'Leary, who first got my foot in the door of the high-tech world, which paved the way for my acceptance into the VI-A program with Intel and all of my successes that have since followed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Master's thesis represents the bulk of the work that I performed during my final VI-A internship assignment at Intel Corporation in Santa Clara, CA. I completed this work on metal interconnect coupling effects in the group responsible for the design of an enhanced version of the Pentium Pro^{TM} Microprocessor. While my work was intentionally focused to be directly usable by this design group, the effects studied and their implications are generally useful and will be even more important for future microchip generations.

Important note about proprietary material:
Given the setting of this work, sensitive proprietary information must, by necessity, be excluded from this thesis document. This censoring manifests itself in several ways:

1) Microprocessor specifications are left out, including architectural descriptions, and functional unit names.

2) For references to values such as clock speeds and process sizes, which are essential to this work, actual numbers have been replaced with variables.

3) Detailed equations for proprietary simulation models are glossed over with only relatively simple, top-level relations being included.

4) Large tables of raw data are excluded in favor of edited summary tables that still fully detail the most important observations and trends.

5) References to material from internal Intel technical memos, reports, and other sources are not included.

I feel that these compromises in no way detract from the understandability or significance of this thesis work.

1.1 Motivation for this project

Capacitive and inductive coupling effects (also termed "cross-talk", since it is the signal dynamics of adjacent lines which cause most of the effects) are always present to some extent in electronic systems. Within high-density, high-performance microchips, these effects become very significant. As device and interconnect sizes decrease and circuit frequencies increase, signal degradation due to electrical coupling becomes more prominent. These trends are prevalent in microchip design, especially leading microprocessor design where circuit density and clock frequencies are continually pushed to their physical limits.

Within microchips it is the long metal interconnect lines which cause the most concern when considering coupling effects. Buses and other sets of parallel interconnect lines in particular must be closely scrutinized as they are composed of many thin, closely spaced wires, running in parallel for considerable distances on the chip. Coupling along these lines not only slows down the circuit, but also introduces noise (voltage fluctuations), which is especially dangerous for sensitive receiving circuits such as pass gates, domino logic, and clocks.

This work focuses on capacitive coupling effects, disregarding the effects of inductance. The decision to exclude inductance effects was made based on previous investigations performed within the microprocessor group which indicated that while capacitive effects were very significant, inductive coupling would have less than a 10% effect on noise and delay given the project's interconnect geometries and target clock frequencies. Since the simulation tools are only accurate to within about 10% overall, the variation due to inductance is safely disregarded, which greatly simplifies the cross-talk simulation models.

1.2 Goals for the thesis work

Specific goals for this thesis work were not set at the beginning of the project. Rather, the work developed as each stage led naturally to further applications and investigations. A general goal was to create a better understanding of not only the ways in which numerous circuit parameters contribute to capacitive coupling but also the significance of these effects and what can be done to minimize their negative impact.

Several smaller and very specific capacitive coupling investigations had been completed within the group before the commencement of this thesis work. Those reports looked mainly at variations in noise and delay when a particular parameter was varied through some specified range. They did not attempt to create a comprehensive understanding of the complex interactions of all of the circuit parameters involved. An important goal of this work is to provide a broader perspective than previously attained on the issue of capacitive coupling.

Another motivating factor for this work was derived from the circuit design practices of the microprocessor group engineers. Many of the integrated circuit engineers used simulation models to account for the performance effects of capacitive coupling but not for the signal integrity effects because including the latter effect greatly increases the complexity of the cross-talk model[1]. This thesis work uses comprehensive models to consider both performance and signal integrity simultaneously.

Ultimately, the results of this work were to be used to aid circuit designers by providing a more informed understanding of capacitive coupling effects and how to restructure designs to minimize them.

[1] In this thesis, good "performance" means short signal delays along the interconnect lines, and good "signal integrity" means minimal noise.

1.3 Organization of the document

Chapter 2 develops the mathematical background and circuit models used for capacitive coupling effects.

Chapter 3 describes the Unix script created to help engineers consider coupling effects in their circuit analyses.

Chapter 4 describes the results of numerous circuit simulations performed to investigate cross-talk effects.

Chapter 5 contains an analysis of the capacitive coupling experienced by the two longest buses on the microprocessor chip.

Chapter 6 provides a comparison of capacitive coupling effects as estimated with simulations against the same effects as measured with physical test chip data.

Chapter 7 contains conclusions based upon the results of the work described in the previous chapters.

Chapter 2

An Overview of Capacitance

2.1 Capacitance physics and modeling

Any two objects that are capable of being electrically charged and which are separated by some dielectric material [1] will form a capacitor. Applying a voltage difference between two such objects leads to a buildup of charge on those objects that is proportional to the capacitance between them. This relationship is described mathematically in Equation 2.1, where Q = charge (measured in Coulombs), C = capacitance (Farads), and V = voltage (Volts). Larger capacitances lead to greater accumulation of charge for a given voltage difference. [1, p. 650]

$$Q = CV \qquad [Coulombs] \tag{2.1}$$

Capacitance can be produced between objects of any shape. The capacitances of concern in this project occur between interconnect lines which, as we shall see, can be fairly accurately modeled as parallel-plate capacitors. For parallel-plate capacitors, the capacitance is given by the relationship in Equation 2.2, where W = the width of the parallel plates (meters), L = the length of the plates (meters), h = the distance

[1] A dielectric is any material, including free space, which does not permit current flow, thus allowing charge to accumulate. Current *will* flow through dielectrics if the voltage difference is greater than the breakdown voltage of the dielectric. However, for the voltage levels and dielectrics used in this project, dielectric breakdown is not a concern.

between the plates (meters), and $\epsilon = \epsilon_0 \cdot \epsilon_r$ (Farads/meter) where ϵ_0 is the permittivity of free space ($8.854 \cdot 10^{-6}$ pF/μm) and ϵ_r is the relative dielectric constant of the material between the plates. [1, p. 653]

$$C = \epsilon \cdot \frac{WL}{h} \qquad [Farads] \tag{2.2}$$

These geometries are shown in Figure 2-1(a). The plates are idealized as having zero thickness. In the remainder of this thesis, references to capacitors will imply parallel-plate capacitors.

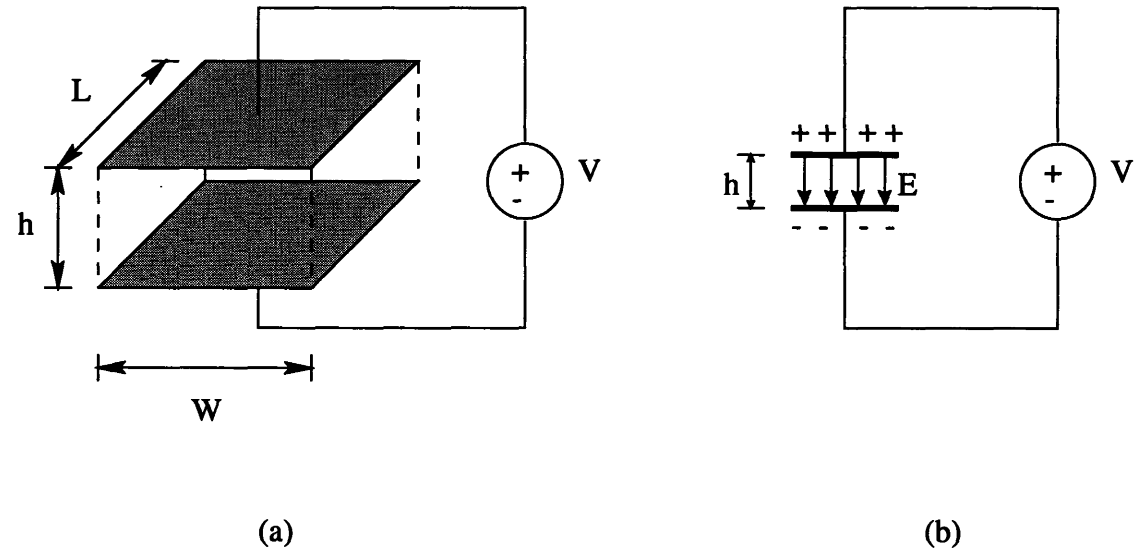


Figure 2-1: A parallel-plate capacitor: a) physical depiction, b) circuit representation

Placing a voltage across a capacitor produces an electric field, E, between the plates of the capacitor as shown in Figure 2-1(b). The electric field, by definition, is directed from the more positively charged plate towards the more negatively charged plate and is inversely proportional to the distance separating them, as given in Equation 2.3. [1, p. 653]

$$E = \frac{V}{h} \qquad [Volts/meter] \tag{2.3}$$

If the voltage difference between these parallel plates is time-varying, then current will "flow" through the capacitor corresponding to Equation 2.4, where i_C is the current flowing through the capacitor (Amperes), V_C is the voltage across the capacitor (Volts), and t is time (seconds). [2, p. 126]

$$i_C = \frac{dQ}{dt} = C \cdot \frac{dV_C}{dt} \qquad [Amperes] \tag{2.4}$$

As already mentioned, current will not actually flow through a dielectric unless it experiences break down, but the effect here is equivalent. Figure 2-2 demonstrates this graphically.

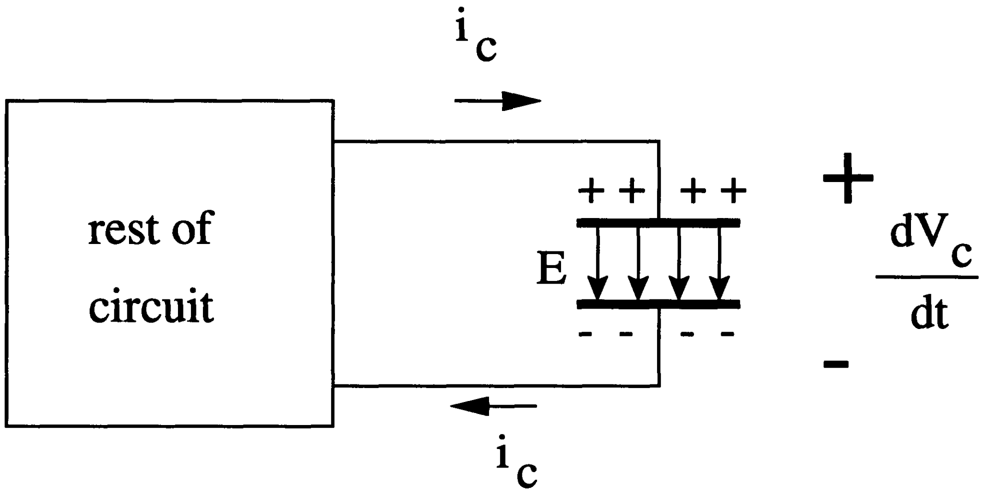


Figure 2-2: Current flow "through" a capacitor

As the voltage across the capacitor in Figure 2-2 increases (due to voltage changes in the "rest of circuit" black box), electrons (negative, mobile charge carriers) are *pushed* from the upper plate of the capacitor to elsewhere in the circuit, while electrons are *pulled* to the lower plate. Electrons exiting the upper plate leave behind a concentration of positive charge on that plate, while electrons accumulating on the bottom plate create a buildup of negative charge there. This effectively creates current flow[2] "through" the capacitor from the upper plate to the lower plate. When the voltage level finally stabilizes, the top plate is left with Q $(= CV)$ Coulombs of

[2] Current is defined to flow from the more positive voltage to the more negative voltage, which is opposite the flow of electrons.

positive charge, while the bottom plate has Q Coulombs of negative charge. [2, p. 126-7]

A parallel-plate capacitor can also be created between a single thin plate and a thick plane. Figure 2-3 shows two cases of a single metal plate forming a capacitor with a ground plane. Here, the metal plate is charged to some positive voltage, while the ground plane is at 0 Volts (also termed "V_{ss}"). Imagine that the plates are rectangular with length (the dimension going "into" the page) about equal to the width.

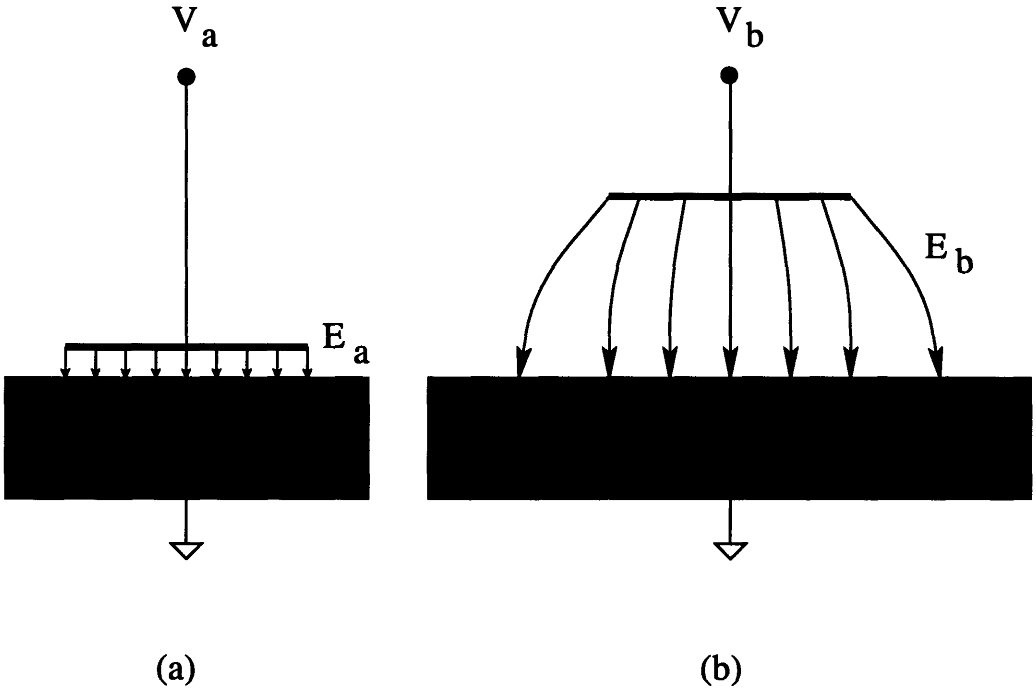


Figure 2-3: Electric fields of a single metal plate over a ground plane for two cases: a) h ≪ W,L, and b) h comparable to W,L

In Figure 2-3 (a), the height above the V_{ss} plane, h, is much less than both the width and the length of the metal plate. For this case, the total electric field is dominated by the vertical field and the small "fringing" fields (fields that extend beyond the area of the plate) can be ignored. In Figure 2-3 (b), however, the three dimensions are comparable and the fringing fields are no longer negligible. These fringing fields act to increase the effective area of the plate. Accounting for these fringing fields is complex and often ignored when modeling simplicity is important. It should be noted, however, that Equation 2.3 only explicitly holds within regions where E is orthogonal to the parallel plates (i.e., where there is no fringing effect). [3, p. 191]

2.2 Capacitive coupling

Analyzing real metal lines adds another dimension to this problem because metal lines also have a non-zero thickness, t. A representation of the fields produced by a metal interconnect line is shown in Figure 2-4. The metal line is charged to V_{cc} (logical "1", the highest voltage level intentionally present on the chip), while the substrate below is tied to V_{ss} (logical "0", the lowest voltage level intentionally present on the chip). Note that now the fringing fields are even more prominent than before. Also note that actual interconnect wires may have rectangular cross-sections of any dimensions and are certainly not limited to the square shape shown in the figure. [3, p. 191]

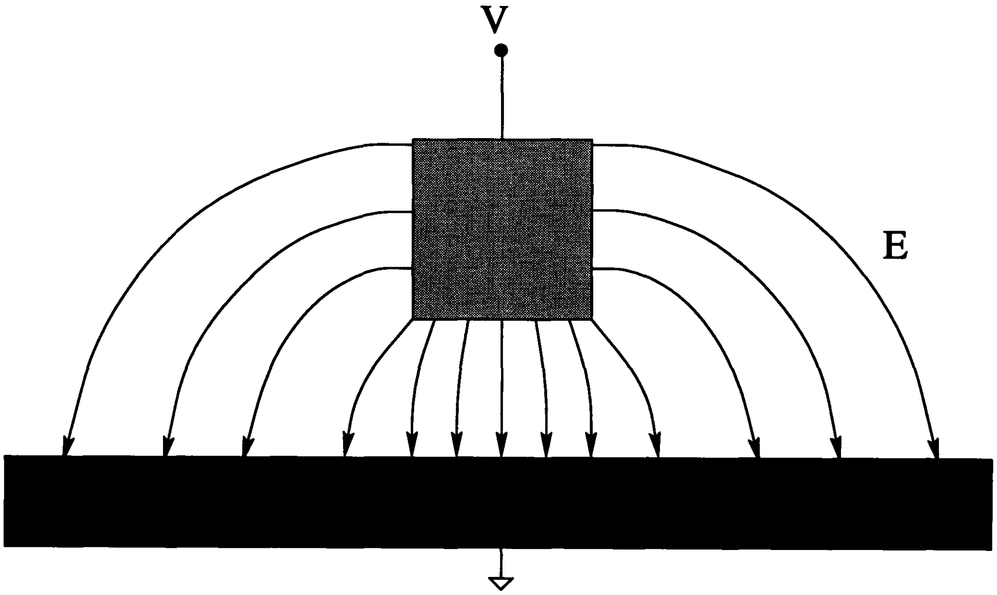


Figure 2-4: Electric fields of a single metal line over substrate

Capacitive coupling along interconnect lines refers to the effects experienced by an electrical signal wire due to voltage fluctuations on nearby signal wires. When dealing with interconnect, these local voltages are produced either by neighboring metal lines (to the side, above, or below) or by the substrate below them. When voltage levels on interconnect lines change, charge flows between the lines due to the capacitive coupling present, as was revealed by Equation 2.4. This flow of charge affects both signal propagation delay (circuit performance) and noise levels (signal integrity). The capacitances present are called "parasitic" capacitances because they are almost always undesired and act to harm performance and signal integrity.

Modern microprocessor designs use multiple layers of metal, each of which has parasitic capacitance to every other layer as well as to the same layer. A cross-section showing the different layers of a microchip is presented in Figure 2-5. Note that the fabrication process used by the design group allows for four distinct metal layers. The metal and dielectric thicknesses are on the order of tenths of microns (millionths of a meter).

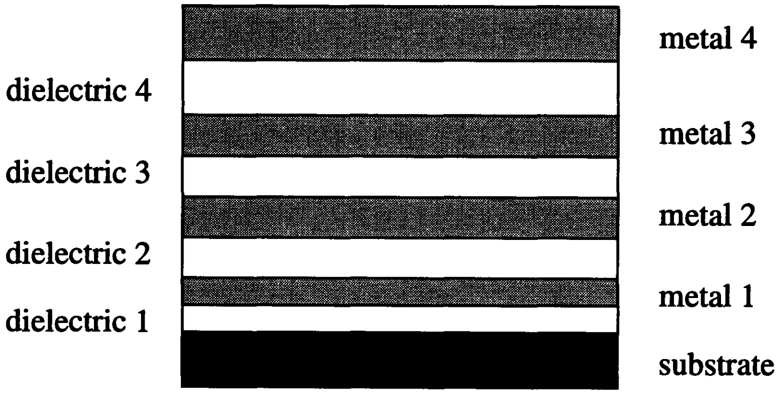


Figure 2-5: Abstract cross-section of a microchip

The problem of estimating capacitance for calculating its effects becomes even more complicated when multiple metal lines are considered. Figure 2-6 shows a complete picture of the electric fields present in a 3-metal parallel configuration.

When running waveform simulations on large circuits it becomes prohibitive to attempt to analyze every electric field present to get a completely accurate view of the capacitances and their effects. The "compromise" model used by the simulation tools in this project is shown in Figure 2-7. The upper and lower rectangles simulate

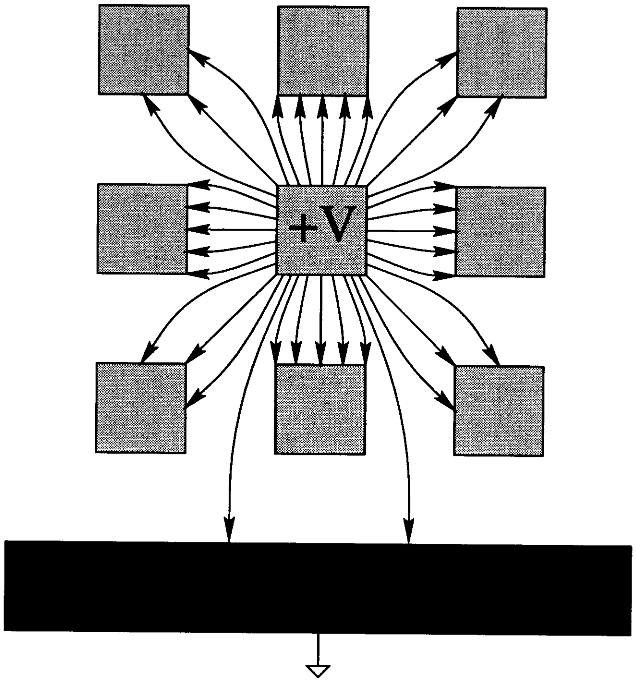


Figure 2-6: Electric fields of multiple metal lines over substrate

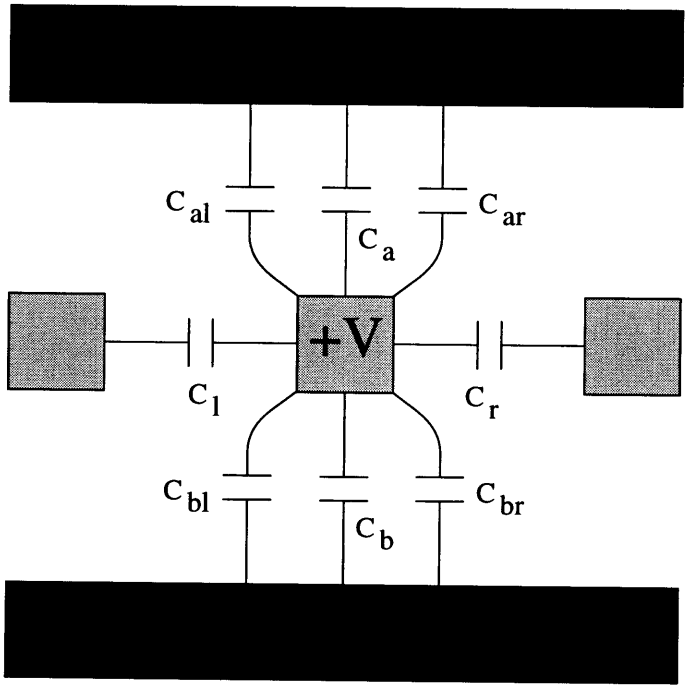


Figure 2-7: Capacitance modeling of multiple metal lines

full metal coverage. The top rectangle can be removed, indicating that there is no metal coverage above the signal line. There may also be no metal coverage below the signal line, in which case the lower rectangle represents the substrate.

The equation used to calculate the overall capacitance seen by the central metal line of Figure 2-7 is given in Equation 2.5 with the term definitions following. This model has been experimentally proven in previous investigations using an electromagnetic field solver program to yield values for total capacitance that are accurate to within 10% of the actual values.

$$C_{total} = C_a + C_b + C_{al} + C_{ar} + C_{bl} + C_{br} + K \cdot C_l + K \cdot C_r \qquad (2.5)$$

C_a = area capacitance to the layer above
C_b = area capacitance to the layer below
C_{al} = fringing capacitance to the above left
C_{ar} = fringing capacitance to the above right
C_{bl} = fringing capacitance to the below left
C_{br} = fringing capacitance to the below right
C_l = line-line capacitance to the left
C_r = line-line capacitance to the right
K = Miller capacitance factor (constant)

The constant, K, is used to try to compensate for adjacent lines in the same metal layer which may be experiencing voltage changes different from the line of interest in the center. This is very often the case in buses. K can be assigned by the circuit designer the values 0, 1, or 2, depending on the application, and it will be explained in detail later.

2.3 Schematic modeling

2.3.1 Simple RC models

Delay along long interconnect wires is "RC dominated", meaning it is the combination of the resistances and capacitances along the line which has the most significant influence on delay. For a simple RC circuit, as shown in Figure 2-8, the output is given by Equations 2.6 and 2.7 for an ideal voltage step input. It is apparent from these equations that a larger RC product corresponds to slower voltage swings across the capacitor. [2, p. 139-47]

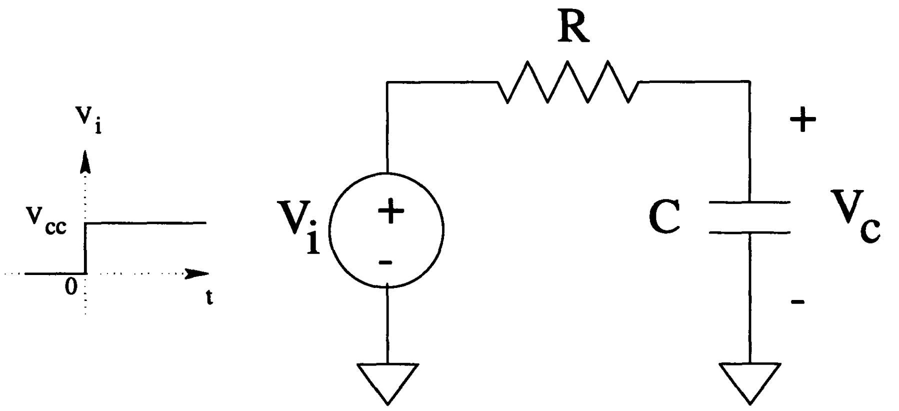


Figure 2-8: A simple RC circuit model

$$(charging) \qquad V_C = V_{cc} \cdot (1 - e^{-t/RC}) \qquad [Volts] \qquad (2.6)$$

$$(discharging) \qquad V_C = V_{cc} \cdot e^{-t/RC} \qquad [Volts] \qquad (2.7)$$

Fully modeled interconnect lines add much complexity to this model. A model for interconnect is shown in Figure 2-9, where C_s is the capacitance of each segment, C_l is the capacitance of the receiving element (also called the "load"), R_s is the resistance of each segment, and R_d is the resistance of the driving element.

One simple and widely used method of calculating the total RC delay for this

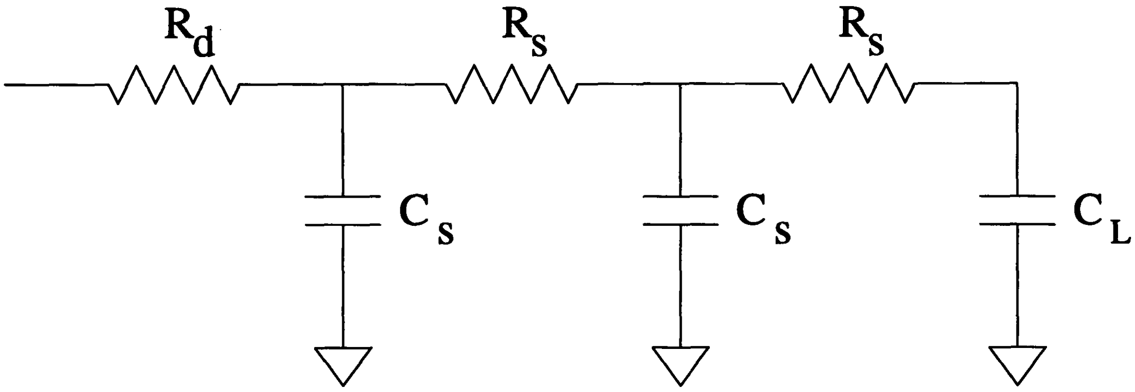


Figure 2-9: An RC model for interconnect

circuit is given in Equation 2.8. Any number of R_s/C_s segments could be used in this model. The more segments used, the more the model will approach a “true” description of the wire, which can best be thought of as being made up of an infinite number of infinitesimally small RC segments. [3, p. 198-200]

$$(RC)_{total} \simeq \sum_{i=1}^{m}[C_i \cdot \sum_{j=1}^{n}(R_j)] = (R_d) \cdot C_s + (R_d + R_s) \cdot C_s + (R_d + 2R_s) \cdot C_l \qquad (2.8)$$

For interconnect lines, R is calculated from Equation 2.9, where ρ is the resistivity of the metal (Ohm · meters) [1, p. 678-81], and C is calculated from Equation 2.2.

$$R = \rho \cdot \frac{L}{W \cdot t} \qquad [Ohms] \qquad (2.9)$$

2.3.2 Idealized capacitive coupling model

To understand capacitive coupling, it is instructional to consider the idealized case presented in Figure 2-10, which shows two coupled lines with all resistances removed from the model. C_S is the self-capacitance of each interconnect line, C_L is the capacitance of each receiving load device, and C_C is the cross-coupling capacitance present between the two parallel lines.

First consider how coupling affects delay. Assume both X and Y are driven high

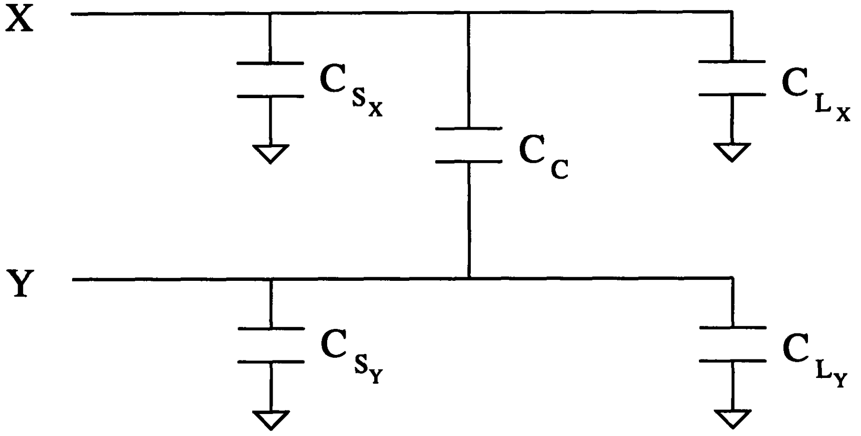


Figure 2-10: Idealized model for analyzing the physics of capacitive coupling

(to V_{cc}). If X now switches low (to V_{ss}), then it has to discharge the charge stored on capacitors, C_{S_X}, C_{L_X}, and C_C. Capacitance adds when capacitors are in parallel, so the C_{total} that needs to be considered in the instantaneous transient for signal line X is $C_{S_X} + C_{L_X} + C_C$. If the coupling to line Y was not present, then there would not be a C_C capacitor and hence no C_C term in C_{total}. Therefore, with capacitive coupling there is more stored charge to remove in order to switch the line low, which results in slower signal propagation and transition times.

To understand how voltage glitching occurs, consider what happens to Y in the previous example. When X and Y are both high, there is no voltage drop across C_C. Assuming that Y is not being actively driven (i.e., it is a “floating” node), then the voltage on Y is maintained by the charge on C_{S_Y} and C_{L_Y}. Thus, the initial voltage on Y is:

$$V_{Y_i} = \frac{Q_Y}{(C_{S_Y} + C_{L_Y})} \tag{2.10}$$

But when X switches low, C_C is suddenly thrown into the picture. When the transient on X has passed, Y now “sees” three capacitors to ground, but the total charge stored on Y has not changed because it is an isolated node. So now, the final voltage on Y is:

$$V_{Y_f} = \frac{Q_Y}{(C_{S_Y} + C_{L_Y} + C_C)} \tag{2.11}$$

Combining Equations 2.10 and 2.11 yields Equation 2.12, which clearly shows the final voltage on Y to be less than the initial voltage when there is capacitive coupling present.

$$V_{Y_f} = \frac{(C_{S_Y} + C_{L_Y})}{(C_{S_Y} + C_{L_Y} + C_C)} \cdot V_{Y_i} \tag{2.12}$$

This analysis is idealized but easily expanded to include realistic situations. If Y were a long, driven, resistive line, then the portion of the line at the load end can be considered to be a temporarily floating node. When X switches, V_Y will drop as noted above, but will recover through the influence of its driver. The voltage dip and recovery will both be exponential with RC rather than instantaneous as in the idealized case. This dip is dangerous even when temporary because it may cause the receiving element to change its logical output, a logical error which could in turn propagate through the rest of the circuit.

2.3.3 π-RC models

The values used to calculate the individual terms of C_{total} in Equation 2.5 are acquired from two places: from the process file for the fabrication process being used, and from the designer, who provides the length, width, and spacing of the lines, the metal layer and coverage by other layers, and a value for K. This information, as well as resistance information that is also read from the process file, is incorporated into circuit schematics by using the π-RC model shown in Figure 2-11. This π-RC model is instantiated in schematics to model delay due to resistance of the line and due to the self- and cross- capacitances present.

The π-RC model is often instantiated several times along a single schematic wire in order to yield a more accurate, distributed RC model rather than a simple, lumped (single R, single C) RC model. The π-RC model can only be specified within the simulation tool to have either zero or complete overlap with any metals above it

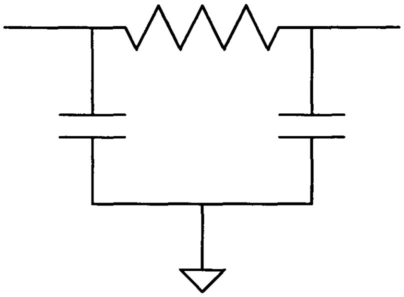


Figure 2-11: π-RC model used for estimating interconnect delay

or below it or with the substrate. This "all or nothing" modeling shows the value of using multiple π-RC models along the same line so that overlap variations can be approximated with different specifications for each π-RC model. All of this information is then used to select the appropriate parameter values from the process file for the resistance and capacitance calculations.

One significant limitation of this π-RC model is that it has no facility for incorporating noise effects due to capacitive coupling to adjacent lines. This shortcoming is present because each interconnect line has no information about the dynamic voltage levels on adjacent lines. To incorporate this important information, explicit cross-coupling capacitors must be included in the schematic. A complete section of 3 adjacent, same metal layer interconnect lines with distributed π-RC modeling and explicit cross-capacitors is shown in Figure 2-12.

Simulation results for circuits using π-RC modeling vary considerably depending on the value chosen for K, which is used in Equation 2.5. To understand the role of K, refer to Figure 2-12, but ignore the cross-capacitors for now and consider delay only. Setting $K = 0$ is the best case for delay. It assumes that both adjacent lines are switching along with the center line, either all from low to high or all from high to low, so that $dV/dt = 0$ (in Equation 2.4) and the capacitances between them, therefore, have no effect. Since dV/dt cannot be set to zero within the model, K is set to zero, which eliminates the capacitance contribution from the adjacent lines, thus achieving the same goal. Setting $K = 1$ means that the voltages on both adjacent lines are remaining constant while the center line switches, so that capacitance to these lines

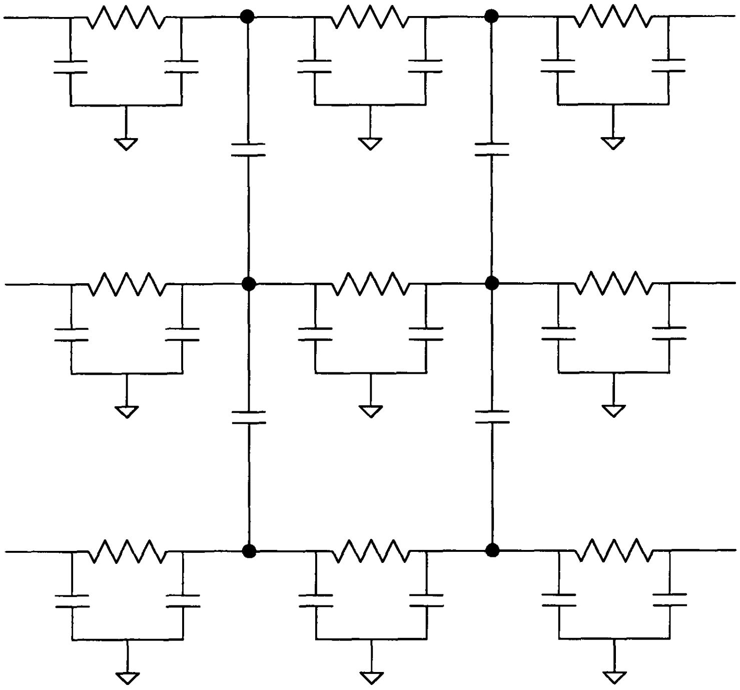


Figure 2-12: Segment of complete cross-talk model for both noise and delay

is felt. Finally, setting $K = 2$ is the worst case for delay. It means that both adjacent lines are switching in the opposite direction of the center line so that the dV/dt is essentially doubled. Since in this case there are no explicit cross-capacitors to model dV/dt in the simulation, the K value is needed to appropriately scale C_l and C_r in Equation 2.5. Note that only delay can be modeled by varying K, not noise.

Now consider the full model of Figure 2-12. Not only do the explicit cross-capacitors allow switching on the adjacent lines to cause voltage glitching (noise) on the center line, but dV/dt is explicit in the simulation so that K values are not needed. Indeed, for this complete simulation schematic, the K values of all the π-RC models are set to zero, else the capacitance effects would be counted twice! By setting $K = 0$, we are removing all line-line coupling estimation from the π-RC model, instead accounting for such coupling explicitly.

The values for the cross-coupling capacitors must also be retrieved from the process file, based upon metal layer, spacing, and segment length. It was the additional complexity introduced through using explicit cross-coupling capacitors that led many designers to forsake this full coupling model for the simplicity of the simple π-RC model. Note that this full model is not only more satisfying in its completeness in providing a way to measure noise, but it is also more accurate in its delay calculation in most cases, as will be shown in Chapter 6.

It is important to understand why all of this circuit estimation is performed when much more accurate circuit values can be extracted from layout and more accurate cross-coupling analysis performed at a later stage of the project. The benefits are gained because improving the circuit level timing and noise analysis has been shown to greatly reduce the amount of debugging required after the layout phase. Furthermore, as circuit designers become accustomed to accurately modeling parasitics, they will naturally learn to optimize their circuits for such effects from the outset, again minimizing debugging time.

Chapter 3

A Simulation Program for Modeling Cross-Talk Effects

My initial approach for improving engineers' understanding of cross-talk effects within the project group was to create a series of graphs that plotted variations in signal delay and noise due to each contributing parameter. It soon became apparent, however, that this approach would have limited usefulness because of the number of parameters involved and their intricate interactions. Faced with this problem, I thought of and developed the idea of creating a program, called "Xtalk_sim", which could account for all relevant parameters simultaneously.

The goal of this program was to provide a straightforward means for engineers to analyze capacitive coupling in their circuits without needing to create elaborate models themselves. The simulation program was written using the C-Shell programming language. This language was chosen for its straightforward means of interfacing with the circuit simulation tool. The C-shell script creates an interactive interface through which engineers may specify numerous parameters relevant to their interconnect lines of interest. After the relevant data has been entered, a simulation is spawned using a circuit model whose values are governed by those input parameters. When the simulation completes, the results are displayed in an Emacs window. The hope was that this program would become a useful addition to the current design flow in the microprocessor group.

3.1 Simulation circuit model

The full circuit model, complete with variable parameters, is shown in Figure 3-1. This cross-talk model is an embellished version of the circuit of Figure 2-12. The inputs, X_i, Y_i, and Z_i are fed into inverting drivers, which drive the signals down the long interconnect lines into inverting receivers, to produce the outputs, X_o, Y_o, and Z_o.

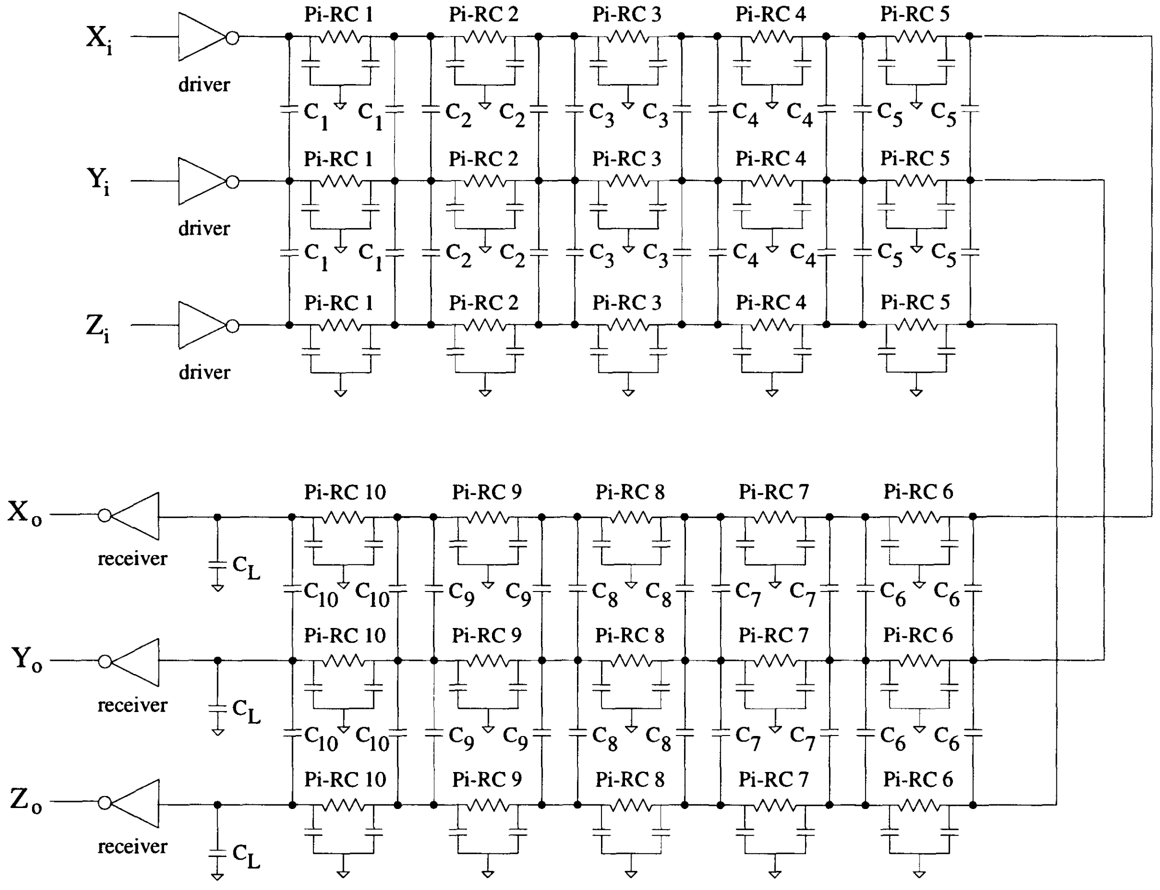


Figure 3-1: Circuit model for Xtalk_sim program

Note that every circuit element in the simulation model has a variable assignment which is determined from the user inputs. The drivers, receivers, metal layer, metal coverage, metal length, metal spacing, metal width, cross-coupling capacitances, optional additional load capacitances, and even input signal rise and fall times are all variable. Some of these parameters are assigned directly from the user inputs, while others are either calculated or looked up in the process file for the project. The Xtalk_sim program handles all of these variable assignments. Note that each of the three adjacent signal paths are identical with the exception that the outer paths are not coupled to further metal lines.

3.2 Program interface screen

A sample of the interface created when Xtalk_sim is run is shown in Figure 3-2. Every choice for which the user is prompted has a default value (listed in <>, and given here as a variable name) which can be kept by simply pressing return ($< CR >$). The program requests information about the drivers, the receivers, the input waveforms, and the interconnect, and it also includes a few extra options.

For both the driver and receivers, specific inverter-like library cells can be specified, as these are the most common driver and receiver elements for long interconnect lines. A non-inverter-like gate can be approximated by specifying the effective p and n transistor sizes of the gate, which are then used to calculate output drive and input capacitance. Adding the capability to directly specify any type of gate from the libraries available would have greatly complicated the program while adding little utility, so was not included. For the receiver specification there is a third option: specifying a capacitive load (in which case the receiving gates are removed from the circuit). This is useful for cases either where there are multiple receiving elements or else where the receiver is not an inverter and the input capacitance of the gate is more readily available or more accurate than a value calculated from the effective p and n sizes.

The sample shown assumes that the default choice of specifying explicit library cells was selected. In this case, the further options of choosing inverter libraries and exact drivers are presented. If the choice of indicating effective p and n transistor sizes had been made, then the program would have prompted for those values. Finally, if the capacitive load option had been selected, a capacitive value in units of picofarads would have been requested.

```
##########################
# XTALK_SIM                    version 2.0   #
#                                                              #
# Created on 10/13/95 by Andrew Percey      #
##########################

Enter the following parameters for the interconnect lines
(or hit < $CR$ > for default values):

Do you want to specify the driver with:
        (1) an explicit library cell
        (2) effective p and n sizes for an inverter model
        choice = < 1 >:
                driver library [basic, complex, etc] < $basic$ >:
                driver type < $inv_7$ >:

Do you want to specify the receiver with:
        (1) an explicit library cell
        (2) effective p and n sizes for an inverter model
        (3) a capacitive load
        choice = < 1 >:
                receiver library [basic, complex, etc] < $basic$ >:
                receiver type < $inv_7$ >:

20%-80% rise time for input to drivers < $t_{rise}$ $NS$ >:
80%-20% fall time for input to drivers < $t_{fall}$ $NS$ >:
process corner < $proc_{def}$ >:

$\pi$-RC modeling:

        total line length < $len$ >:

                $\pi$-RC model name < $default: metal$ 2 $over$ $substrate$ >:
                $\pi$-RC spacing < $s_{m_2}$ >:
                $\pi$-RC width < $w_{m_2}$ >:
                fraction of line using this $\pi$-RC model < 1.0 >:

Run extra simulations without explicit cross-capacitors
        to compare delays? (y/n) < $n$ >: y

What $K$ value should be used for modeling the delay?
        $K$ (0,1,2) < 0 >: 2

Run xtalk_sim again? (y/n) n
```

Figure 3-2: Sample Xtalk_sim interface screen

After the drivers and receivers have been characterized, the program requests information about the input waveforms. Specifically, it will prompt for the rise and fall times of the input signals, where rise and fall times are measured from the 20% to 80% of V_{cc} points. With this information Xtalk_sim generates closely-approximated waveforms to be used as the inputs. For example, if the input rise time is t_r, then then the generated rising transition will consist of three linear voltage ramps each of duration t_r: one from V_{ss} to 20% of V_{cc}, one from 20% of V_{cc} to 80% of V_{cc}, and one from 80% of V_{cc} up to V_{cc}.

Next is a request for the "process corner" to be modeled. This parameter defines assumptions about the quality of silicon produced - whether is has faster or slower devices than expected, etc. This specification was included for completeness, but at the time this work was done, the simulation tool did not differentiate between different process corners.

Several parameters are needed to fully characterize the interconnect: the total length of each of the interconnect lines, the metal/substrate coverage model to use, the spacing between each pair of lines, the width of each line, and the percentage of each interconnect line that should use the parameters just entered. The coverage model indicates the metal layer for the current line and also the interactions with other metals above and below or the substrate below. For example, the default model, *metal 2 over substrate*, indicates no significant overlap with other metals; its only source of coupling capacitance comes from the substrate below. If the "fraction of line" option is kept at 1.0, then all of the π-RC models in the simulation schematic will be assigned the parameters that were just selected. Optionally, a fraction in multiples of 0.1 (since there are 10 π-RC models along each interconnect line) can be specified. In that case, the π-RC modeling input section will repeat, allowing different characterization for further sections of the interconnect until 100% characterization is achieved.

Next, a further option is presented. If desired, a second set of simulations can be run. In this optional set of runs, the explicit cross-coupling capacitors are effectively removed from the circuit and instead K values are specified. This option was made available so that designers who were accustomed to the simple K value modeling could easily see how much of a difference the full characterization makes in delay analysis. After this option, the circuit database is built and the simulations are spawned.

Finally, the user may choose to re-run Xtalk_sim. This can be done immediately because the simulations running are spawned as background processes in Unix. If the program is re-run, all of the values that the user just specified become the default values.

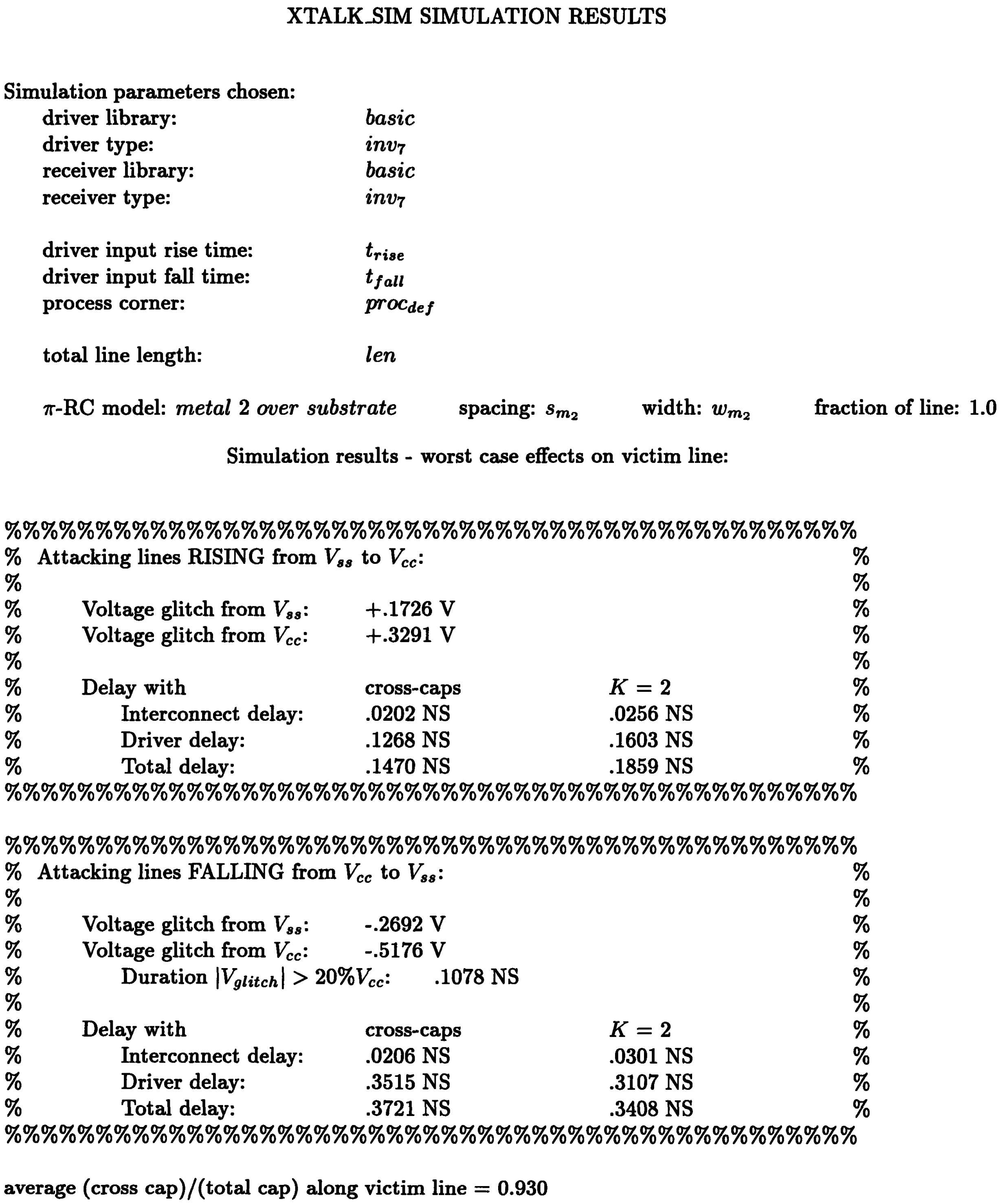
The simulations performed are for worst case scenarios. For noise simulations, the victim line is first held high while the attacking lines switch from high to low then from low to high in unison. Then the victim line is held low and the attacking line pattern is repeated. For delay simulations, the victim line first switches high to low while the attacking lines simultaneously both switch opposite the victim line. Then the victim line switches low to high while the attackers again switch oppositely.

3.3 Program results screen

Figure 3-3 shows a sample Xtalk_sim output screen that resulted from running simulations with the parameters that were chosen from within the interface screen of Figure 3-2. This output screen has two main sections. The first half restates the input values that were specified by the designer. The second half gives the noise and delay values calculated by the circuit simulator.

The simulation results section is further broken down into two sets of data. In the first, the attacking lines (the two lines surrounding the "victim" line of interest) are switching from low to high (rising), while in the second, the attacking lines are switching from high to low (falling). A glance at the numbers will reveal that both noise and delay are considerably worse in the second set of data. This trend is explained by the fact that in the default driver as well as in most of the drivers in the cell libraries used, the pull-down devices (NMOS transistors) are stronger than the pull-up devices (PMOS transistors).[1]

[1] This difference in strength is an intrinsic feature owing to the physical differences between N



```
                         XTALK_SIM SIMULATION RESULTS

Simulation parameters chosen:
    driver library:                  basic
    driver type:                     $inv_7$
    receiver library:                basic
    receiver type:                   $inv_7$

    driver input rise time:          $t_{rise}$
    driver input fall time:          $t_{fall}$
    process corner:                  $proc_{def}$

    total line length:               len

    $\pi$-RC model: metal 2 over substrate    spacing: $s_{m_2}$    width: $w_{m_2}$    fraction of line: 1.0

                 Simulation results - worst case effects on victim line:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Attacking lines RISING from $V_{ss}$ to $V_{cc}$:                                %
%                                                                          %
%       Voltage glitch from $V_{ss}$:     +.1726 V                                %
%       Voltage glitch from $V_{cc}$:     +.3291 V                                %
%                                                                          %
%       Delay with                 cross-caps            $K = 2$                %
%            Interconnect delay:   .0202 NS              .0256 NS               %
%            Driver delay:         .1268 NS              .1603 NS               %
%            Total delay:          .1470 NS              .1859 NS               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Attacking lines FALLING from $V_{cc}$ to $V_{ss}$:                               %
%                                                                          %
%       Voltage glitch from $V_{ss}$:     -.2692 V                                %
%       Voltage glitch from $V_{cc}$:     -.5176 V                                %
%            Duration $|V_{glitch}| > 20\% V_{cc}$:     .1078 NS                     %
%                                                                          %
%       Delay with                 cross-caps            $K = 2$                %
%            Interconnect delay:   .0206 NS              .0301 NS               %
%            Driver delay:         .3515 NS              .3107 NS               %
%            Total delay:          .3721 NS              .3408 NS               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

average (cross cap)/(total cap) along victim line = 0.930
```

Figure 3-3: Sample Xtalk_sim output screen

Therefore, the attacking lines switch more quickly during the falling transitions, leading to a greater "current" flow through the cross-coupling capacitors in the model, increasing both noise and delay along the victim line.

Both sets of data give the same information. Consider the set with the attacking lines rising. The first piece of data is the level of the voltage glitch when the victim line is being driven low. The voltage glitch is measured at the input to the receiver, where noise is of most concern. In this case, the victim line rises to about $0.17V$ temporarily, due to the low to high transitions of the attacking lines, before being driven back down to $0V$ by its driver.

The second data is the level of the voltage glitch when the victim line is being driven high. In this case, the victim line actually rises *above* V_{cc} by about $0.33V$ before settling back down to V_{cc}. The trend within the noise data is that the glitch is always worse when the victim line is being driven high than when it is being driven low. Again, this effect is a result of the driving pull-up device being weaker than the pull-down device and therefore less able to maintain a steady voltage level in the presence of capacitive coupling.

Note that the second set of results data has an additional line following the "Voltage glitch from V_{cc}" value. If any of the output voltage glitches exceeds 20% of V_{cc} in magnitude, then an additional line of data is reported which reveals the duration of time for which that threshold is exceeded. This information is provided to help the user visualize the noise to distinguish between a very brief noise spike and a longer-lasting glitch. The latter is more dangerous in terms of causing the receiving stage to switch, which could lead to a logic error.

Finally, delay information is reported, split into the sub-categories of interconnect delay (delay from the output of the driver to the input of the receiver), driver delay (delay from the input of the driver to the output of the driver), and total delay (the sum of interconnect and driver delay). As throughout this thesis, delay is measured

and P type MOSFET transistors, which can only be overcome by making the P devices larger than the N devices, which is often done. By default, however, drivers with same size N and P transistors can drive the output low more strongly (i.e., more quickly) than they can drive the output high.

at the 50% of V_{cc} points.

At least one, and optionally two, columns of data are given for delay information. The first column presents the results of the simulations run with the full cross-coupling capacitor circuit model and $K = 0$. The second, optional column gives delays from simulations in which the cross-coupling capacitors are removed and instead the K value is used to approximate coupling effects. As explained previously, this option was included for engineers who were accustomed to such modeling.

Note that for the parameters given for this simulation run, the driver delay is much more significant than the interconnect delay. For longer interconnect lines, the interconnect delay component becomes a much more substantial and, eventually, dominant part of the full signal delay.

At the bottom of the output screen is a report of the average cross-coupling along the victim line in terms of a $(cross - couplingcapacitance)/(totalcapacitance)$ ratio. This number was included also as a reference for engineers accustomed to using this ratio to determine potentially problematical interconnect lines. This ratio by itself, however, does not encompass all of the relevant parameters. For example, changing any of the length, driver, receiver, or input waveform parameters will alter both noise and delay along a capacitively coupled line, but will not change the % cross-cap ratio. Too much importance, therefore, must not be placed on this ratio without full knowledge of all of the circuit parameters.

3.4 Final evaluation

Xtalk_sim was designed to have many features desirable for circuit designers who are concerned with capacitive coupling in their circuits. The program offers a much quicker alternative for accurate coupling analysis than creating a full π-RC and cross-capacitance model for each metal line in a circuit schematic. The interface is straightforward and simple to use. Any number of simulations can be spawned simultaneously, with values from one simulation conveniently appearing as the default values for the next so that testing of limited parameter tweaking is very simple.

Finally, the set of simulations run by each execution of Xtalk_sim requires only about 1 1/2 minutes of CPU time to complete, so results are very quickly attainable.

Xtalk_sim, as written, performed all simulations at T_0 Celsius. As both noise and delay exhibit slight changes in performance due to temperature (as described in Chapter 4), including temperature as a specifiable variable in the interface screen would be a worthwhile future addition.

Chapter 4

Cross-talk Simulation Data

During the course of the work I performed at Intel, I ran a large number of circuit simulations to investigate cross-talk effects. The results of some of the more revealing ones, which provide insight into the effects of capacitive coupling, are included in this chapter.

4.1 Qualitative summary of simulation results

Performance problems (delay) and signal integrity problems (noise) are affected to different degrees and, in certain cases, in different ways by the same circuit parameters. These parallels and differences are discussed below and exemplified in the tables and graphs of this chapter.

Signal propagation delay is governed by the time required to charge or discharge capacitances, parasitic and otherwise, along the interconnect line. In general, the capacitance of the receiving element is the most critical, but as line lengths and, therefore, cross-coupling, increase, interconnect capacitance becomes more and more important.

Signal noise is introduced by dynamic voltage switching on adjacent interconnect lines. As line lengths increase, the metal lines behave more like floating nodes because the driving elements are relatively far away so that it takes significant time to recover from voltage fluctuations. This effect makes long metal lines more susceptible to cross-talk noise, especially at points close to the receiving elements (the farthest from the driver a line segment can be), which, unfortunately, is where voltage stability is the most important.

The circuit conditions leading to noise are a slightly different set than those leading to delay, but they both have many factors in common. Decreasing spacing between metal lines increases the cross-capacitance between the lines, adversely affecting both noise and delay. Decreasing the metal width of the wire increases noise susceptibility because the self-capacitance is reduced so that the cross-capacitances play a more significant role, while smaller width also means larger resistance through the metal so that delay is increased. Also, a smaller driver for the victim line (buffer, inverter or whatever it may be) means the signal is driven more weakly so is more susceptible to noise effects and also takes longer to (dis)charge line capacitances, thus increasing delay. Finally, as circuit temperatures increase, wire resistances increase, which directly increases delay (through the RC product) and indirectly increases noise by further isolating the receiver from its driver.

There are also several parameters that affect noise and delay oppositely. A large capacitive load at the end of the line due to the receivers helps maintain voltage levels and hence reduces noise effects, but it also increases (dis)charge time. Fast input transitions (short rise and fall times) cause more noise because of the increased dV/dt, but can actually help delay when measured as we are measuring it, using 50% of V_{cc} points, because the line and load capacitances are (dis)charging while the input has yet to swing to 50%, thus giving the output signal a "head start". Having non-switching metal or substrate above and/or below the line helps reduce noise by adding "quiet" capacitance, but it increases delay because of this same additional capacitance.

4.2 Graphs and tables of simulation results

A number of tables and graphs of data were produced that explicitly show these noise and delay effects due to capacitive cross-coupling. Most of the data was generated using Xtalk_sim and the full π-RC model shown in Figure 3-1, which includes explicit cross-coupling capacitors with the K coefficient set to zero. Any deviations are noted within the relevant sections. The simulations that were run to produce all of these tables and graphs use the parameter values below as defaults except for the particular parameter that is being varied.

default π-RC model	= metal x over substrate with no overlap of other metal layers
default spacing	$= s_{m_x} =$ minimum spacing of metal x
default width	$= w_{m_x} =$ minimum width of metal x
default length	$= len$
default driver size	$= buf_7$
default receiver size	$= buf_7$
default rise time	$= t_{rise}$
default fall time	$= t_{fall}$
default temp	$= T_0$

These default parameters were chosen based upon average or customary parameters for circuits in which cross-talk concerns arise. The default π-RC model of "metal x over substrate with no overlap of other metal layers" makes the cross-coupling to the adjacent metal lines dominant, leading to worst case noise glitching. Spacing and width are set to minimum for worst case noise and delay modeling. The length was made sufficient to produce substantial variations in noise and delay when other parameters were varied. The driver and receiver were set to the typical bus driver/receiver inverter. The rise and fall times were set to typical times for the target clock speed. The default temp was set to the minimum test value.

Initially, a model with 20 π-RC elements modeling each interconnect line was used for performing simulations. Changing the model to use only 10 π-RC elements, however, led to less than a 1% difference in noise and delay calculations while decreasing computation time by more than 20%. Thus, the 10 element model was chosen for performing simulations.

4.2.1 Broad-range sensitivity tests

The following "sensitivity" experiments were performed to get a "feel" for the effects of individually varying all the parameters that change the influence of capacitive coupling. A summary of the changes in noise and delay for metals 1 through 4 are presented in Tables 4.1 through 4.4, respectively.

The first column of each of these tables lists the parameter that is being varied from its default value. For example, the second data row reads: "spacing: $1.5 \cdot s_{m_x}$", which indicates that the data in that row was produced by simulations run using the default parameters except for spacing, which was increased by 50%. Also, "buf_8" is one inverter size larger than "buf_7", and so on. For simplicity and symmetry, the tests were run with $t_{rise} = t_{fall} = t_{tran}$. Reading the data, in Table 4.1 increasing the spacing by 50% decreases the worst case voltage glitch by 23% and worst case total delay by 10%. The percentage changes listed in the sensitivity tables are the worst case changes found from examining the results of all input transition combinations. Total delay is the sum of line (interconnect) delay and driver delay; For these examples driver delay makes up a more significant portion of the total delay. The final column gives the change in the cross-cap ratio, which was introduced in Chapter 3.

Studying these tables reveals many prominent trends which hold in general across all of the metal layers. In fact, taking metal 3 as a base line data set, all of the other metals exhibit the exact same behavior for all of the given parameter changes to within 5% (except for metal coverage values which vary up to 10%). Notice that changing the π-RC model from one with no metal coverage above or below the current metal layer to one with full coverage has the greatest impact on reducing the level of the voltage glitching experienced. Increasing the spacing also significantly reduces noise. Increasing the length of the interconnect greatly increases noise, of course. And making the rise/fall transitions sharper also worsens noise. In terms of delay, length is the most influential dimension, obviously. A stronger driver will significantly improve signal propagation time, as will faster input transitions and wider spacing. All of these trends are exactly as discussed in Section 4.1, which details the reasoning

Table 4.1: Sensitivity tests: metal 1 data

parameter changed	voltage glitch % change	total delay % change	line delay % change	driver delay % change	(cross cap)/ (total cap) % change
π-RC model: full coverage	-32	-1	+7	-2	-33
spacing: 1.5$\cdot s_{m_1}$	-23	-10	-8	-10	-13
width: 1.5$\cdot w_{m_1}$	-5	-1	-34	+3	-3
length: 1.5$\cdot len$	+31	+28	+89	+20	–
driver: buf_8	-6	-14	-3	-15	–
receiver: buf_8	-7	+4	+14	+2	–
rise/fall time: 0.5$\cdot t_{tran}$	+14	-16	+3	-18	–

Table 4.2: Sensitivity tests: metal 2 data

parameter changed	voltage glitch % change	total delay % change	line delay % change	driver delay % change	(cross cap)/ (total cap) % change
π-RC model: full coverage	-47	-4	+10	-5	-48
spacing: 1.5$\cdot s_{m_2}$	-21	-10	-9	-10	-9
width: 1.5$\cdot w_{m_2}$	-4	-1	-34	+2	-2
length: 1.5$\cdot len$	+32	+26	+85	+22	–
driver: buf_8	-8	-14	-3	-15	–
receiver: buf_8	-6	+4	+17	+3	–
rise/fall time: 0.5$\cdot t_{tran}$	+16	-16	+1	-17	–

Table 4.3: Sensitivity tests: metal 3 data

parameter changed	voltage glitch % change	total delay % change	line delay % change	driver delay % change	(cross cap)/ (total cap) % change
π-RC model: full coverage	-42	-6	+3	-7	-40
spacing: 1.5$\cdot s_{m_3}$	-20	-9	-8	-9	-9
width: 1.5$\cdot w_{m_3}$	-3	-1	-35	+1	-1
length: 1.5$\cdot len$	+32	+26	+87	+22	–
driver: buf_8	-7	-14	-3	-15	–
receiver: buf_8	-6	+4	+17	+3	–
rise/fall time: 0.5$\cdot t_{tran}$	+16	-16	+1	-17	–

Table 4.4: Sensitivity tests: metal 4 data

parameter changed	voltage glitch % change	total delay % change	line delay % change	driver delay % change	(cross cap)/ (total cap) % change
π-RC model: full coverage	-34	+2	+12	+2	-39
spacing: $1.5 \cdot s_{m_4}$	-28	-11	-6	-11	-15
width: $1.5 \cdot w_{m_4}$	-2	+1	-31	+1	-2
length: $1.5 \cdot len$	+31	+23	+84	+23	–
driver: buf_8	-11	-15	-6	-15	–
receiver: buf_8	-6	+4	+19	+4	–
rise/fall time: $0.5 \cdot t_{tran}$	+18	-17	+0	-17	–

behind these results.

4.2.2 Noise and delay vs. cross-talk models

Tables 4.5 and 4.6 show the results of simulations run with various circuit configurations. Table 4.7 shows the results of simulations run with various cross-talk models. All of these tables use metal 2 for the simulation models.

Table 4.5: Signal noise vs. simulation model

	glitch with			
	attacking lines rising		attacking lines falling	
model used	from V_{ss}	from V_{cc}	from V_{ss}	from V_{cc}
default	+0.173	+0.329	-0.269	-0.518
1st half: no coverage 2nd half: full coverage	+0.120	+0.237	-0.194	-0.377
full metal coverage	+0.087	+0.169	-0.146	-0.272
four attacking lines	+0.182	+0.338	-0.272	-0.516
only one attacking line	+0.088	+0.179	-0.140	-0.285
two attacking lines separated as if shielded	+0.081	+0.160	-0.127	-0.249
two attacking, shielded lines	+0.013	0.022	-0.019	-0.033

Tables 4.5 and 4.6 show two sets of results for the same set of π-RC models (in the first column). The default model is metal 2 over substrate with no overlap of other metals and with two adjacent, parallel lines set at minimum spacing. The second row

Table 4.6: Signal delay vs. simulation model

	delay with					
	attacking lines rising			attacking lines falling		
model used	line	driver	total	line	driver	total
default	0.020	0.127	0.147	0.021	0.351	0.372
1st half: no coverage 2nd half: full coverage	0.021	0.132	0.153	0.022	0.346	0.368
full metal coverage	0.021	0.136	0.157	0.023	0.337	0.360
four attacking lines	0.023	0.135	0.158	0.021	0.348	0.369
two attacking, shielded lines	0.016	0.117	0.133	0.019	0.231	0.250
two attacking lines separated as if shielded	0.014	0.104	0.118	0.016	0.263	0.279
only one attacking line	0.012	0.093	0.105	0.015	0.262	0.277

shows data for a model where the first half is the default model, and the second half has full metal coverage above and below. For the third row the model uses complete metal coverage. The final four rows cover cases of different layouts within the same metal layer. First, two additional attacking lines are added at minimum spacing from the initial attacking lines. Second, all but one attacking line are removed. The third and fourth cases investigate the effects of "interleaving", where V_{cc} or V_{ss} lines are placed between the victim line and the attacking lines (all with minimum spacing). This "shields" the victim line from the attackers, reducing cross-talk. For the third case, the attacking lines are spaced as if there were shielding lines in place, though there are not. For the fourth and final case, there are in fact shielding lines in place between the victim and the attackers.

There are very significant differences in noise for the various simulation models, as shown in Table 4.5. The worst case noise (the final column) improves a great deal through the introduction of overlapping metal layers into the model, as expected since the additional neutral capacitances help maintain voltages along the victim line. Using four attacking lines instead of two is inconsequential as the nearest attacking lines almost fully shield the effects of the outer two. Modeling only one attacking line reduces the noise by nearly a factor of two from the default case. Using two attacking lines both of which are separated from the victim line as if there were lines interleaved between them also reduces noise by almost 50% from the default case. Finally, actually interleaving V_{ss} lines between the victim and the attackers reduces noise to negligible levels, demonstrating the utility of this shielding technique.

The effects on delay of varying the modeling are less stark because interconnect delay makes up only a small portion of total delay for these simulations. But still the trends, as shown in Table 4.6, are instructional. Line (interconnect) delay increases with increasing metal coverage. Adding 2 extra attacking lines has no appreciable effect. Shielding decreases the delay, but not as much as spacing out the attacking lines to the same distance without the interleaved V_{ss} lines because any cross-capacitance, switching or static, contributes to delay. Finally, having only one attacking line is the best case for interconnect delay, as would be expected since this produces the minimum coupling capacitance of all the models.

Table 4.7: Signal delay vs. level of cross-talk present

	delay with					
	attacking lines rising			attacking lines falling		
model used	line	driver	total	line	driver	total
default	0.020	0.127	0.147	0.021	0.351	0.372
no explicit cross-caps $K = 2$	0.026	0.160	0.186	0.030	0.311	0.341
no explicit cross-caps $K = 1$	0.016	0.119	0.135	0.020	0.236	0.256
no explicit cross-caps $K = 0$	0.008	0.061	0.069	0.009	0.149	0.158

Table 4.7 gives a comparison of cross-talk models. Rows 2, 3, and 4 were created by running the extra simulations option in Xtalk_sim, removing the explicit cross-coupling capacitors and assigning values for K. The inability of K value modeling to realistically model dynamic switching effects leads to the discrepancies between the default model and row 2, even though setting $K = 2$ is done to approximate the case where the attacking lines are switching opposite the victim line. Row 3 data shows less delay, as now the model assumes the attacking lines are static. Finally, row 4

data has the least delay since the model now assumes that the attacking lines are switching along with the victim line so that no cross-coupling is "seen" by the victim line.

4.2.3 Noise and delay vs. interconnect length, width, and spacing

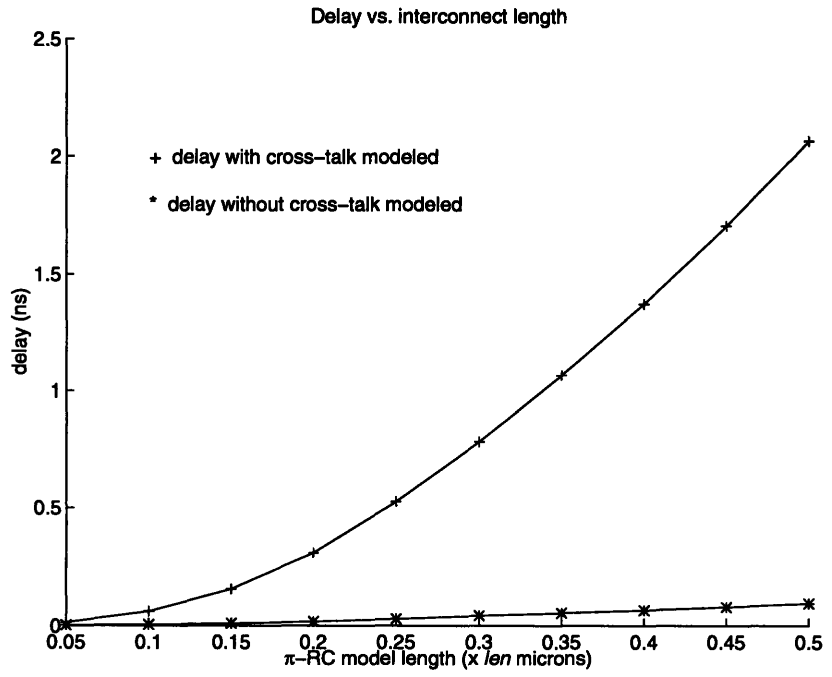


Figure 4-1: Signal delay vs. length

Figures 4-1, 4-2, and 4-3 show plots of signal propagation delay versus interconnect length, width, and spacing, respectively, with all other parameters retaining their default values. There are two lines plotted on each graph, one for delay results *with* cross-talk modeled and one for delay results *without* cross-talk modeled. Each y-axis shows delay, measured in nanoseconds. Each x-axis is measured in microns, and the numbers on the axis are scale factors for the dimension variable listed in the axis label. For example, in Figure 4-1, 0.2 on the x-axis means that the π-RC model is $0.2 \cdot len$ microns long. The entire interconnect line, therefore, is $4 \cdot len$ microns (since the interconnect models used to produce these simulation results used 20 π-RC models for each line). Note that for these 3 plots, "delay" refers to interconnect delay only.

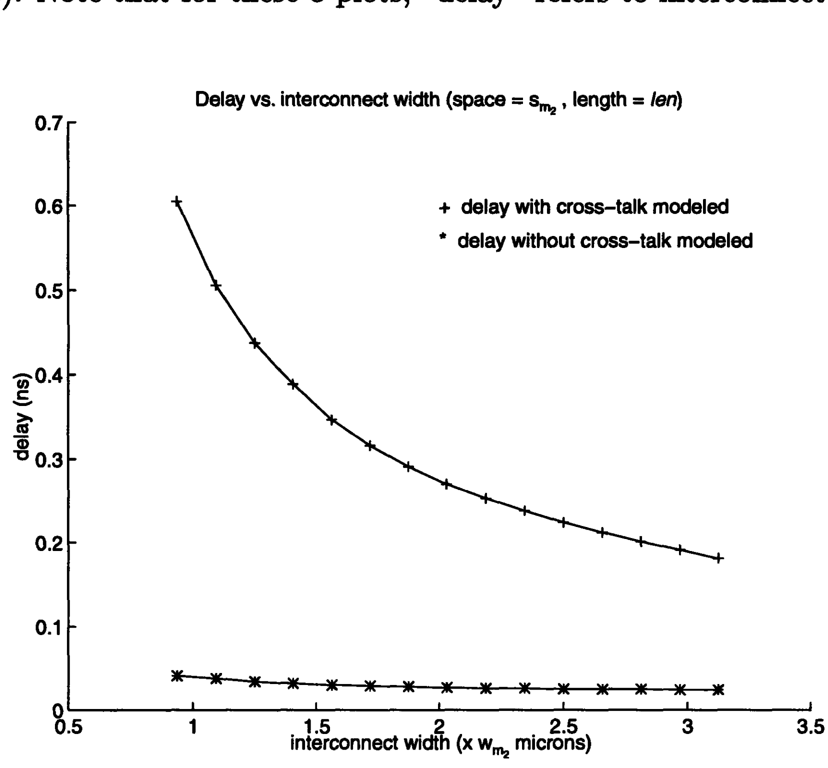


Figure 4-2: Signal delay vs. width

These three figures clearly show the danger of neglecting cross-talk modeling when running circuit simulations. Delay obviously increases with line length, but *much* more so when there is cross-talk present. Delay increases with decreasing width due to the increased resistance, but increases much more quickly when there is cross-talk because the decreased self-capacitance of the victim line makes it more susceptible to these effects. Also, under the influence of cross-talk delay increases as spacing decreases, though less sharply than with decreasing width.

Interestingly, when cross-talk effects are not considered, delay actually *decreases* slightly with decreasing metal spacing. This is because of the modeling situation created with this unrealistic "no cross-talk" scenario. To model zero cross-talk, the explicit cross-capacitors were removed from the circuit and the K values were set to zero. However, the π-RC models still see adjacent lines s_{m_2} distance away on either side, which are just switching along with the victim line so that they appear as if they were not even present. But those adjacent lines are present, and when they are moved

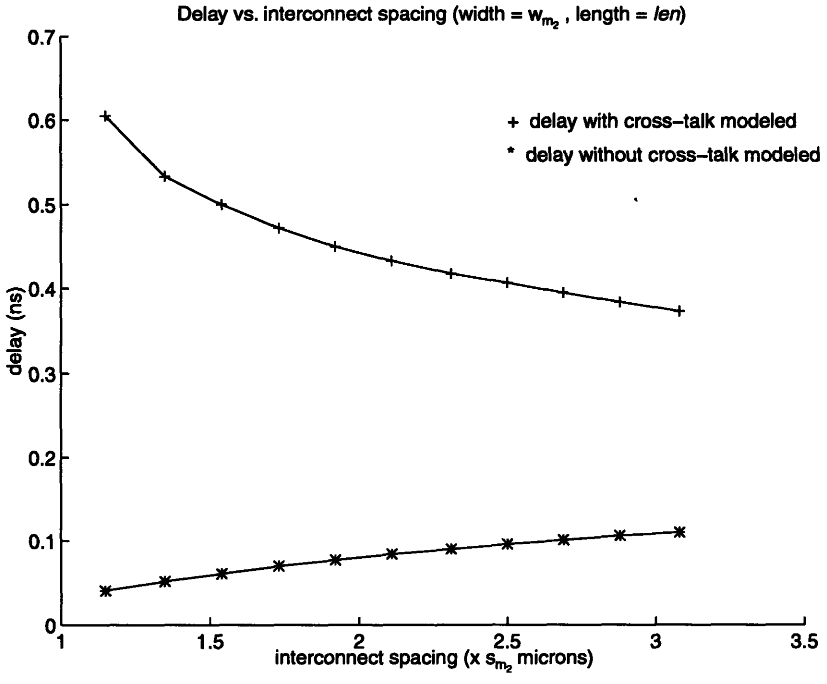


Figure 4-3: Signal delay vs. spacing

farther away, more of the fringing electric fields from the victim line now terminate on the substrate instead of those adjacent lines. This increases the capacitance seen by the victim line, hence increasing the interconnect delay.

Note that not modeling cross-talk leads to substantially optimistic results, and is never recommended. These graphs were included simply to demonstrate the importance of modeling cross-talk effects.

Figures 4-4 and 4-5 plot interconnect delay and noise each versus spacing and width together (with cross-talk modeled). As already demonstrated in the previous three graphs, width and spacing have substantial effects on delay. As Figure 4-5 reveals, however, width plays a much smaller role in noise effects than does spacing.

4.2.4 Noise and delay vs. temperature

Noise and delay (interconnect delay, in these figures) both depend to some small extent on the operating temperature of the interconnect lines. Figures 4-6 and 4-7 show these dependencies over the full range of standard operating temperatures.

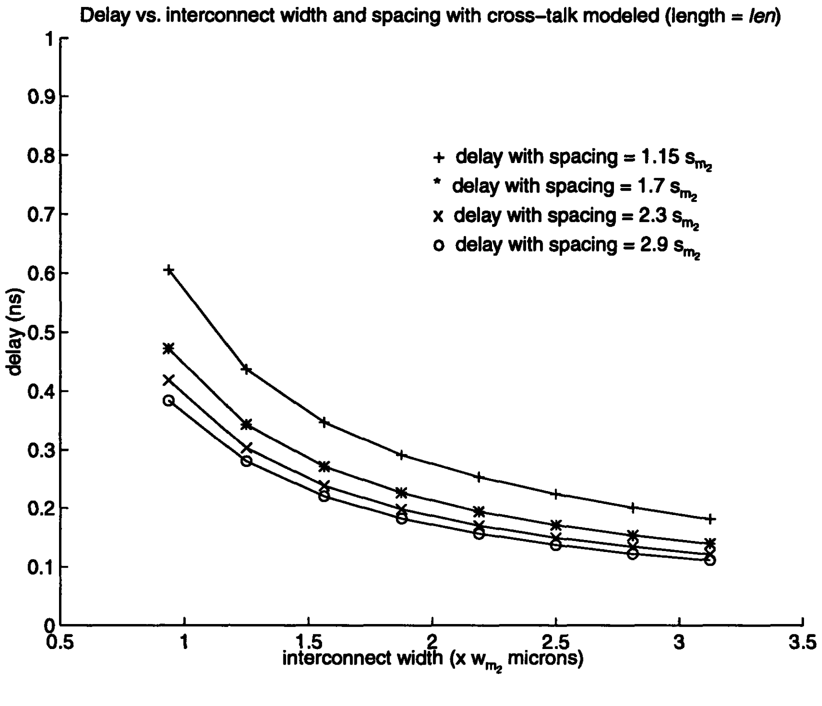


Figure 4-4: Signal delay vs. width and spacing

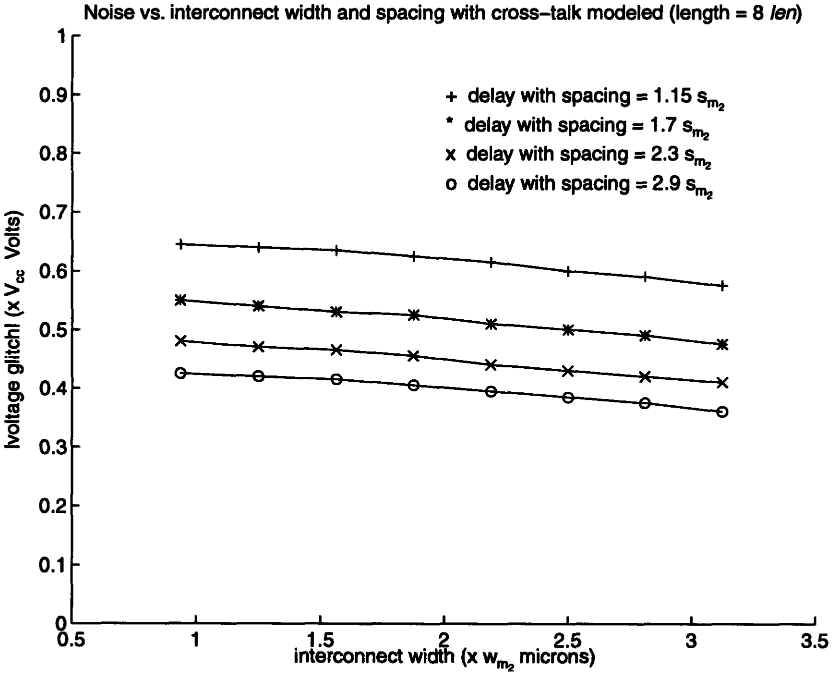


Figure 4-5: Signal noise vs. width and spacing

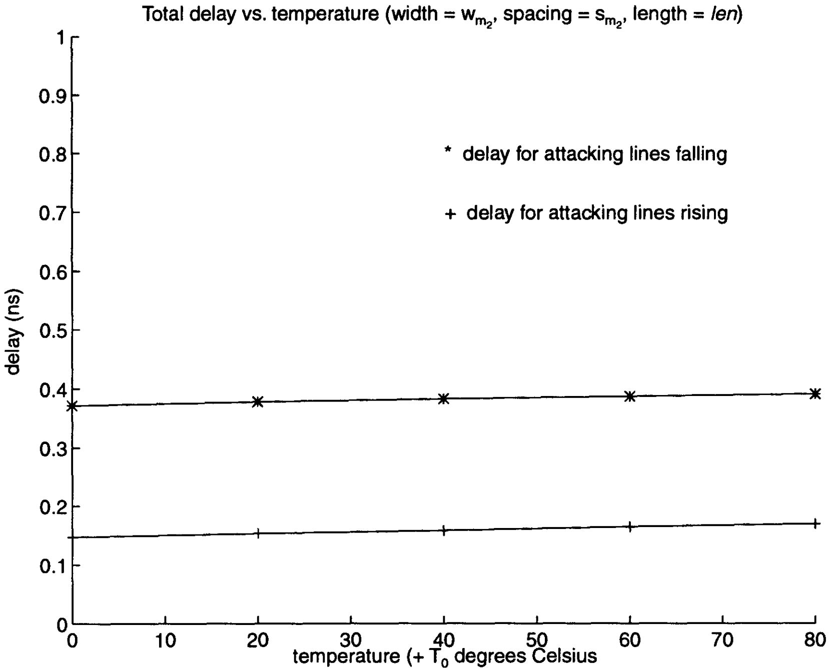


Figure 4-6: Signal delay vs. temperature

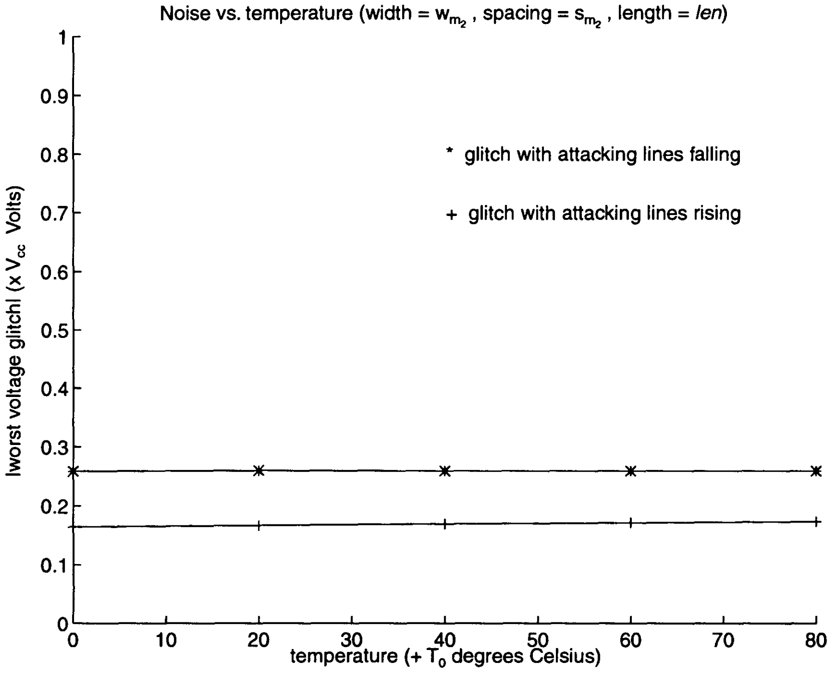


Figure 4-7: Signal noise vs. temperature

Chapter 5

Cross-talk Analysis of Two Microprocessor Buses

Bus lines are particularly susceptible to serious cross-coupling problems because they are often very long, which reduces the ability of the drivers to maintain signal values along the entire line, and because they are usually laid out closely together and in parallel, which leads to potentially substantial cross-capacitances.

Engineers in the project were very concerned about coupling effects along the two longest buses on the chip (the longest one was about $24 \cdot len$ microns in length!). These were large data buses routed along the pad ring outside of the main functional blocks. The die size of the microchip was very aggressively constrained for the microprocessor design, so there was little room for generosity in widening the bus lines or increasing the spacing between them, yet the bus lines themselves had aggressive timing requirements and needed to experience limited noise.

I was asked to build precise capacitive coupling models to very accurately estimate the delay and noise effects of coupling along those lines. I was also asked to recommend acceptable spacing and width dimensions for the lines as well as techniques that could be used to reduce the cross-talk effects to more acceptable levels.

5.1 Simulation modeling

The buses were laid out in metal 4, which is the best metal layer available in the fabrication process used for limiting coupling effects as it has the greatest minimum spacing and minimum width of all the metal layers. Unfortunately, besides having adjacent metal 4 lines with which to experience coupling, these bus signals also had metal 3 signal lines running in parallel one layer below! The effects of these lines were non-negligible, so the simulation model needed to be expanded to include these lines.

The tool generally used to generate the cross-coupling capacitance values was not capable of calculating capacitance values for multiple metal lines in different metal layers. For this reason and to improve the overall accuracy of the models for these buses, a more precise capacitance calculation tool was employed.

5.1.1 Precision calculation of cross-coupling capacitance

A tool called Maxwell Spicelink was used to generate very precise capacitance values for the interconnect geometries analyzed for these buses. Instead of making estimates based on general values from the process file as the tool used by Xtalk_sim did, Spicelink actually solves explicitly for all of the electric and magnetic fields around a given conductor arrangement. This approach yields very accurate capacitance values, but is also time-consuming to set up and run and extremely computationally intensive. For these reasons, using a tool like this for general simulations is unwieldy.

One of the models used in the Spicelink analysis is shown in Figure 5-1. This figure shows a cross-section of the bus routing on the chip. The ϵ symbols denote dielectric values. Note that no metal 2 or metal 1 layers were included in this model. This exclusion was intentional as metal 1 and metal 2 were compensated for in the π-RC model coverage specifications.

Other configurations were also run to determine the worst case model to use for the overall circuit simulation. For example, it was not known prior to analyzing the Spicelink results that configuration (a) in Figure 5-2 yielded slightly worse capacitive coupling than configuration (b) for the center metal 4 line.

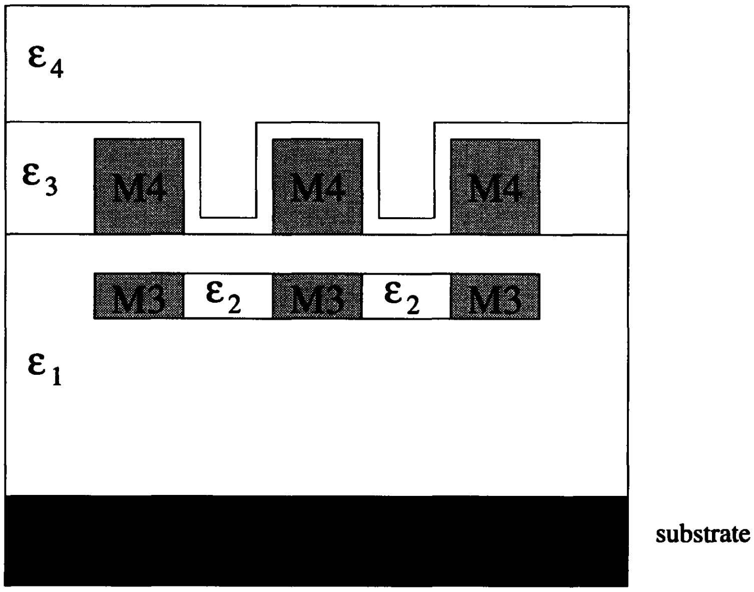


Figure 5-1: Cross-section used for precise capacitance analysis

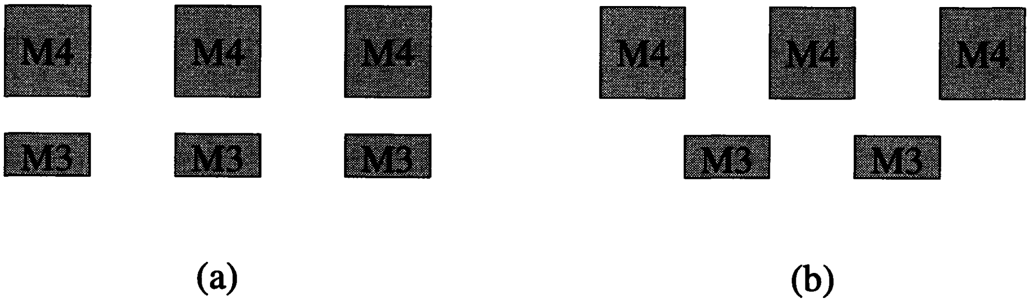


Figure 5-2: Two layout possibilities for worst-case capacitive coupling

5.1.2 Circuit model

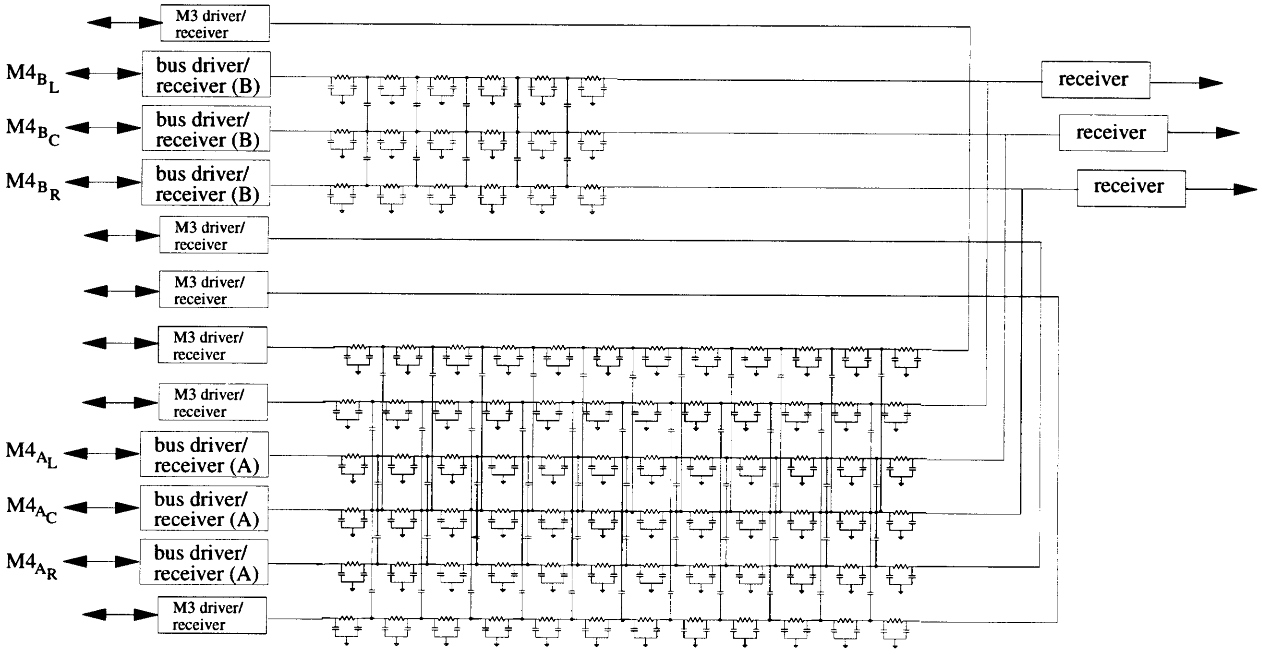


Figure 5-3: Simulation model for analyzing two specific buses

The circuit simulation model used in Xtalk_sim was woefully inadequate to handle these special bus cases. The bus drivers were tri-state buffers. The buses were both bi-directional with multiple loads at both ends of the line. One of the buses had additional receivers halfway along the length of the bus. Also, as mentioned above, the metal 3 lines beneath had appreciable effects on the metal 4 lines. None of these contingencies could be adequately modeled by the circuit model used by Xtalk_sim.

Therefore, a very specific circuit model was made for each of the two buses analyzed. In addition to having cross-coupling capacitances to five other lines (two metal 4 lines and three metal 3 lines, the drivers and receivers in the simulation schematic are the actual driver and receiver logic blocks for the buses. Figure 5-3 shows the circuit simulation model used for the two buses. The values used for all of the cross-coupling capacitors were calculated using the Spicelink model of Figure 5-1. Note that the metal 3 lines did not run the entire length of either metal 4 bus, hence in the top portion of the model, only metal 4 interactions are considered.

Simulations were performed using worst-case modeling. Also, the inputs were constructed such that the metal 3 lines and metal 4 lines switched simultaneously, which was a possible occurrence.

5.2 Analysis of bus #1

The first bus analyzed was the longer of the two and the longest on the entire chip. It was first analyzed at minimum spacing and width for metal 4 with no coupling reduction techniques employed. These simulations revealed a worst-case voltage glitch in one direction along the bus of greater than $0.5 \cdot V_{cc}$!

This large glitch caused the receiving element to temporarily flip its output, creating a logic error which also propagated slightly to the third logic stage. Fortunately, the signals going into this receiver did not encounter registering elements (flip-flops) until the fifth logic stage, so that logical failures were difficult to latch. However, given additional noise from other sources such as inductance and the nearby V_{cc} and ground pad connections, the voltage glitch could become even worse and potentially propagate all the way to the registers.

This presented a serious danger which needed to be addressed. As mentioned previously, increasing the pitch (width + spacing) of the metal 4 lines was problematic because of the tight layout area constraints in the pad ring. Therefore, other techniques to reduce cross-talk were investigated. Interleaving bus lines with V_{cc} or ground lines was infeasible, again because of the pitch constraints. Another idea was to capitalize on the fact that not all 64 bits of the bus line ran for the entire $24 \cdot len$ microns. Bits were added four at a time as the bus progressed. This layout could be taken advantage of by widening out and increasing the spacing between the lines to fill the entire 64-bit pitch when less than all 64 bits were present. However, upon further investigation it was discovered that the "extra" area available in those regions had already been routed with other unrelated signals.

Finally, the option of using repeaters along the lines was investigated. Repeaters are just buffers that are used along long interconnect lines to boost the signal strength. Though they add a logic gate delay, for long sections of interconnect they will actually decrease the total delay due to their strong drive. They also help to reduce noise by decreasing the distance from driver to receiver so that the voltage at the receiver can be better maintained. The idea of using repeaters was not popular either, because large metal 4 bi-directional repeaters, as would be needed for these buses, take up considerable area. At most, only one bank of repeaters per bus would have been permitted.

Table 5.1: Summary of simulation results: bus #1

		timing margins at		worst case	other
		f_1 MHz	f_2 MHz	voltage glitch	observations
A − > B path					
without repeaters	wid = w_{m_4} spa = s_{m_4}	+1.0 ns	+0.4 ns	1.02 V	glitch propagates to second receiver stage
	wid = $1.39 \cdot w_{m_4}$ spa = $1.25 \cdot s_{m_4}$	+1.2 ns	+0.6 ns	0.92 V	slight glitch propagation
with repeaters	wid = w_{m_4} spa = s_{m_4}	+1.55 ns	+0.95 ns	0.77 V	glitch safe for this path
	wid = $1.39 \cdot w_{m_4}$ spa = $1.25 \cdot s_{m_4}$	+1.6 ns	+1.0 ns	0.69 V	glitch safe for this path
B − > A path					
without repeaters	wid = w_{m_4} spa = s_{m_4}	-0.1 ns	-0.7 ns	0.81 V	glitch safe for this path
	wid = $1.39 \cdot w_{m_4}$ spa = $1.25 \cdot s_{m_4}$	+0.1 ns	-0.5 ns	0.62 V	glitch safe for this path
with repeaters	wid = w_{m_4} spa = s_{m_4}	+0.3 ns	-0.3 ns	0.56 V	glitch safe for this path
	wid = $1.39 \cdot w_{m_4}$ spa = $1.25 \cdot s_{m_4}$	+0.4 ns	-0.2 ns	0.52 V	glitch safe for this path

As further analysis showed, using the repeaters about midway along the bus line reduced delay and noise to much more acceptable levels. A summary of the results of the main simulations performed is presented in Table 5.1. The first column lists the simulation parameters. For example, the first line of data corresponds to analyzing signals propagating up the bus from point A to point B, without using repeater along the lines, and with the lines set to minimum spacing and width. The second and third columns show the timing margins calculated for the given simulation run. In the first data row, the positive margin of +1.0 ns means the timing requirement was met with a nanosecond to spare. A negative margin means the timing requirement has not been met. The fourth column gives the magnitude of the worst-case voltage glitch calculated by the simulation. The final column comments on the seriousness of the observed noise.

The recommendation made for this bus was to use the best overall case from the table: using repeaters with width = $1.39 \cdot w_{m_4}$ and spacing = $1.25 \cdot s_{m_4}$, which was the maximum pitch the designers would be willing to accept. These changes not only reduced voltage glitching by over 30% to safe levels for this bus, but also reduced delay - a double win. The repeaters alone reduced glitching by over 20%.

5.3 Analysis of bus #2

Table 5.2: Summary of simulation results: bus #2

		worst case voltage glitch	other observations
A − > B path			

without repeaters	wid = w_{m_4}
	spa = s_{m_4}
		0.88 V	glitch safe for this path

	wid = $1.39 \cdot w_{m_4}$
	spa = $1.25 \cdot s_{m_4}$
		0.79 V	glitch safe for this path

B − > A path			

without repeaters	wid = w_{m_4}
	spa = s_{m_4}
		0.60 V	glitch safe for this path

	wid = $1.39 \cdot w_{m_4}$
	spa = $1.25 \cdot s_{m_4}$
		0.57 V	glitch safe for this path

Learning from the analysis of bus #1, bus #2 was analyzed for the same parameter combinations of Table 5.1. The results summary for bus #2 is shown in Table 5.2. Note that since this bus was significantly shorter, even the worst-case voltage glitching did not cause any logic errors, and all of the timing constraints were met (so were left out of the table). This bus was deemed acceptable without modification.

A problem specific to this bus, however, did arise. Some of the bus lines fed directly into a transmission gate, which could potentially be turned on in reverse in the presence of sufficient voltage glitching above V_{cc} or below V_{ss}. Acting upon my request, the designer in charge of those receivers added buffering elements to eliminate this potential problem.

Chapter 6

Correlation of Experimental to Simulation Data

Of great concern within all VLSI design groups is the accuracy of the circuit simulation tools used. Designers rely very heavily upon these tools to report accurate waveform and timing information, which incorporate a multitude of physical effects such as explicit and parasitic RC delays, electrical coupling, leakage currents, body effect, voltage thresholds, device sizing, and carrier mobilities.

Many of these considerations are simple to model accurately, such as transistor sizes, while others involve complex modeling techniques and significant approximations, such as capacitive coupling. As part of my work within the processor group, I took on the task of analyzing the accuracy of our simulation models and tools with regard to characterizing RC delays and capacitive coupling effects.

6.1 Approach used for comparisons

Data generated from a fabricated "test" chip was used to analyze the accuracy of the group's CAD tools. The test chip contained dozens of experiments to test our CAD tool accuracy with a variety of circuit configurations. One set of experiments was designed exclusively to contain patterns of long RC-dominated interconnect lines, both with and without cross-coupling from adjacent lines and with and without V_{ss}

shielding, for all levels of metal. Each individual experiment was surrounded by V_{ss} lines to fully isolate the experiment from the others. Data from analysis of these specific experiments was used to evaluate CAD tool modeling accuracy.

The test chips were analyzed with specific test vectors run on an electron beam test machine. This meant that only the high-level circuit inputs and outputs of each set of interconnect lines could be observed and affected, which somewhat limited the usefulness of the tests. Because of this restriction, voltage glitching could not be measured. Delay through each path was measurable, but unfortunately included the driver and receiver delays. Worse still, the internal signals were latched before reaching the output pins, so that to measure interconnect delay required indirect methods. Therefore, the circuit delay values were calculated by running the test chip circuit at incrementally higher frequencies until eventually the output registers failed to latch the new values. CAD simulation models were created for all of these circuits, and identical tests performed on them.

6.2 Models and assumptions

Two different simulation models were used. The first model corresponded to the general modeling approach used by the engineers in the group. The second simulation model was the complete capacitive cross-coupling analysis circuit of Figure 2-12. The models are detailed below:

RC simulation model #1:
- π-RC model with
  - $K = 1$ for lines without cross-talk (adjacent lines are V_{ss})
  - $K = 2$ for lines with cross-talk
  - no explicit cross-coupling capacitors

RC simulation model #2:
- π-RC model with
  - $K = 0$
  - explicit cross-coupling capacitors from adjacent lines included in schematic

To match the test chip testing environment, simulations were performed with: $V_{cc_{test}} = V_{cc} + V_{variation}$, $Temp = Temp_{max}$, and a few other tool-specific settings.

The circuits for the cross-talk experiments were set up such that the data signal for the victim line passed through an additional inverter before reaching the attacking lines. Therefore the adjacent lines did not switch at exactly the same time as the victim line, leading to slightly less than worst-case delay.

Previous wafer analysis demonstrated that the metal sheet resistances on the silicon matched the simulation values to within 5%. Also, other experiments on the test chip proved that the simulation accuracy of the inverters and registers was very, very good. Therefore, any differences in simulation data and actual data for the interconnect experiments was presumed to lie mostly with the capacitance modeling.

6.3 Model comparisons

Though the RC delays could not be completely decoupled from the device delays in these experiments, the results were still useful for general circuit analysis. The circuits' RC delay ratio of about 50% of total delay is reasonable for many real circuit situations where interconnect performance is an issue.

Tables 6.1, 6.2, and 6.3 summarize the experimental and simulation data for metal layers 1, 2, and 3, respectively. Four experiments were performed for each metal layer: a control experiment (ctl) consisting of a signal line surrounded by V_{ss} lines, two cross-talk experiments (xcapa, xcapa2) consisting of a signal line surrounded by attacking lines, and a repeater experiment (repd) consisting of a signal line surrounded by V_{ss} lines but broken in the middle by a repeater. The first column of data gives the minimum clock period (maximum frequency) at which the experiment on the actual silicon successfully ran. The second and third columns show the same information for the CAD simulation models along with the percentage difference from the test chip data in parenthesis. The final column reports the percentage of total delay in the experiment that was due to interconnect delay.

Table 6.1: Interconnect experimental/simulation correlation - metal 1

experiment		experimental data: min clock period (average)	simulation data: min clock period (RC model #1)	simulation data: min clock period (RC model #2)	% of delay due to RCs (average)
ctl	0 − > 1	3.7 ns	4.2 ns (+13.5%)	4.2 ns (+13.5%)	42%
	1 − > 0	[no data]	3.6 ns	3.6 ns	
	average		—	—	
xcapa	0 − > 1	4.51 ns	5.3 ns (+17.5%)	5.4 ns (+20.0%)	53%
	1 − > 0	4.24 ns	4.9 ns (+15.5%)	5.0 ns (+18.0%)	
	average		(+16.5%)	(+19.0%)	
xcapa2	0 − > 1	4.45 ns	4.8 ns (+8.0%)	4.9 ns (+10.0%)	48%
	1 − > 0	3.9 ns	4.3 ns (+10.0%)	4.4 ns (+13.0%)	
	average		(+9.0%)	(+11.5%)	
repd	0 − > 1	4.49 ns	5.6 ns (+26.0%)	5.6 ns (+26.0%)	61%
	1 − > 0	4.72 ns	5.0 ns (+6.0%)	5.0 ns (+6.0%)	(including
	average		(+16.0%)	(+16.0%)	repeater)

Table 6.2: Interconnect experimental/simulation correlation - metal 2

experiment		experimental data: min clock period (average)	simulation data: min clock period (RC model #1)	simulation data: min clock period (RC model #2)	% of delay due to RCs (average)
ctl	0 − > 1	3.96 ns	4.4 ns (+11.0%)	4.4 ns (+11.0%)	45%
	1 − > 0	3.56 ns	3.9 ns (+9.5%)	3.9 ns (+9.5%)	
	average		(+10.3%)	(+10.3%)	
xcapa	0 − > 1	4.45 ns	6.2 ns (+39.5%)	5.2 ns (+17.0%)	54%
	1 − > 0	4.65 ns	5.6 ns (+20.5%)	4.7 ns (+1.0%)	
	average		(+30.0%)	(+9.0%)	
xcapa2	0 − > 1	4.44 ns	5.5 ns (+24.0%)	4.9 ns (+10.5%)	50%
	1 − > 0	4.62 ns	4.6 ns (-0.5%)	4.3 ns (-7.0%)	
	average		(+11.8%)	(+1.8%)	
repd	0 − > 1	4.43 ns	5.8 ns (+31.0%)	5.8 ns (+31.0%)	62%
	1 − > 0	4.65 ns	5.2 ns (+12.0%)	5.2 ns (+12.0%)	(including
	average		(+21.5%)	(+21.5%)	repeater)

Table 6.3: Interconnect experimental/simulation correlation - metal 3

	experiment	experimental data: min clock period (average)	simulation data: min clock period (RC model #1)	simulation data: min clock period (RC model #2)	% of delay due to RCs (average)
ctl	0 − > 1	4.0 ns	4.4 ns (+10.0%)	4.4 ns (+10.0%)	45%
	1 − > 0	3.56 ns	3.9 ns (+9.5%)	3.9 ns (+9.5%)	
	average		(+9.8%)	(+9.8%)	
xcapa	0 − > 1	4.48 ns	6.2 ns (+38.5%)	5.3 ns (+18.5%)	54%
	1 − > 0	4.68 ns	5.2 ns (+32.5%)	4.7 ns (+0.5%)	
	average		(+35.5%)	(+9.5%)	
xcapa2	0 − > 1	4.48 ns	5.5 ns (+23.0%)	4.9 ns (+9.5%)	50%
	1 − > 0	4.63 ns	5.1 ns (+10.0%)	4.3 ns (-7.0%)	
	average		(+16.5%)	(+1.3%)	
repd	0 − > 1	4.47 ns	4.9 ns (+9.5%)	4.9 ns (+9.5%)	62% (including repeater)
	1 − > 0	4.67 ns	5.3 ns (+13.5%)	5.3 ns (+13.5%)	
	average		(+11.5%)	(+11.5%)	

6.3.1 Accuracy with and without cross-talk effects

Using RC model #1 and RC model #2 yields identical results when the victim line is shielded on both sides by V_{ss} (in the "ctl" and "repd" experiments). For this case (no cross-talk) CSE simulated delay is pessimistic relative to wafer data by about 10-20%. For lines with cross-talk, RC model #2 is more accurate on average, yielding delays that are pessimistic by about 0-20%, while RC model #1 yields delays that are pessimistic by about 10-30%.

RC model #1 is less accurate than RC model #2 for the cross-talk experiments because the attacking lines are not switching at the exact same time (they feed through an extra inverter stage to create the opposite switching polarity), a worst-case assumption that is built into the first model. On average, these delay simulations err on the pessimistic side (indicating that a lower maximum clock frequency is permitted) compared to actual wafer data for both RC models. An inverter chain analysis performed separately within the project group showed simulated delay to be pessimistic by about 15% for these devices. The interconnect analysis above shows simulated delay to be pessimistic by close to the same amount on average, indicating that our RC models are also somewhat pessimistic on average since interconnect and driver delay are about equal.

6.3.2 Rising/falling transitions

In almost all cases, the simulated delay for the 0 − > 1 transition is more pessimistic than for the 1 − > 0 transition. In fact, in a few cases the simulated 1 − > 0 transition delays are optimistic compared to the silicon delays, though all of the transition averages still fall on the pessimistic side.

This behavior is expected since the p devices have been made stronger than anticipated relative to the n devices due to fabrication process enhancements.

6.3.3 Metal width

For all of the cross-talk experiments, increasing the metal width improves the simulation accuracy significantly.

Since the simulation sheet resistance values match almost exactly to the wafer values, it is reasonable to conclude that most inaccuracies in the RC models are a result of inexact capacitance parameters and modeling structures (the "sandwich model" of Figure 2-7 that is used for capacitance simulation does involve significant estimation - on the pessimistic side normally - without which simulations would become unacceptably time-consuming). Increasing the metal width decreases the relative impact of capacitive effects, thus leading to simulation delay values that better match the experimental values.

6.3.4 Overall correlation results

Overall, this analysis showed the simulation models and methods to be pessimistic compared to the experimental wafer data by about 20%, on average, for interconnect modeling. This was a welcome finding as it indicates good modeling and simulation accuracy for very complex circuit interactions such as cross-talk. Also, since circuit designers are required to meet the given project requirements using simulation data, the fact that the simulations yield worst than actual results leaves a comfortable error "buffer" that may lead to better than expected actual chip performance.

Chapter 7

Conclusions

The capacitive coupling modeling and analysis provided in this thesis work made significant strides towards improving understanding and management of cross-talk effects at the circuit simulation level within the microprocessor project group. Xtalk_sim proved useful for many circuit designers in the group and was even explored by some members of the next-generation microprocessor group. The project manager was put at ease by the recommendations made for the two, long buses, especially after a set of repeaters was added to the most problematic bus of the two. Finally, the accuracy of the CAD circuit simulation tools with respect to interconnect modeling was proven to be acceptable by the comparisons performed with actual test chip data. Overall, the thesis work helped to provide a broader perspective on and understanding of capacitive coupling issues.

The trends plotted in Chapter 4 clearly show that as technology continues to advance, cross-talk effects become ever more prominent. Technical improvements in fabrication and layout always lead to smaller, thinner, and longer wires on average as well as to sharper signal transitions due to higher clock speeds. Steps must be taken to not only better model cross-talk effects, but to reduce them.

Since technological advances work to increase cross-talk effects, the task of reducing them falls to good engineering strategies. Chapter 5 mentions several possible approaches including interleaving buses with V_{ss} or V_{cc} lines, widening and spacing sections of the interconnect where area permits, avoiding running different metal layers in parallel, and using repeaters along long interconnect lines to boost the signal strength. These approaches along with other innovative engineering ideas will, by necessity, be much more widely used in future designs.

With higher clock speeds and longer interconnect lines, inductive coupling effects will become non-negligible. Techniques to reduce inductive coupling are less well-known and in general more difficult to implement than those used to reduce capacitive coupling. Much work will need to be done in this area in the very near future.

Technological advances are rarely achieved without cost. The increasing prominence of electrical coupling effects on microchips is the cost incurred as higher performance devices are continuously sought. Improved methods of dealing with these effects will certainly be developed for some time to come, as parasitic capacitance and inductance are unavoidable consequences of semiconductor microchip physics.

Appendix A

Xtalk_sim files

```
### XTALK_SIM version 2.0

#! /bin/csh -f

### This program is designed to accurately simulate noise and delay
### effects on interconnect lines due to cross-talk.
### date created: 9/5/95


###### VARIABLE INITIALIZATION ##########################

### make sure process file is set
if (! -e $CSEPRCFILE) then
setenv CSEPRCFILE $proc_file
endif

### make sure technology file is set
if (! -e $CSETECHFILE) then
setenv CSETECHFILE $tech_file
endif

### set input default values
set dr_choice=1
set driver_lib=basic
set driver=zi0bin01g
set p_dr=21.88
set n_dr=16.48
set rec_choice=1
set receiver_lib=basic
set receiver=zi0bin01g
set p_rec=21.88
set n_rec=16.48
set cap_load = 0.1
set cap_load_save = 0.1
set t_rise=0.3
set t_fall=0.3
set skew="tttt"
set length_total=1000
set model="rm2fu"
set width=0.64
set spacing_l=0.52
set spacing_r=0.52
set fraction=1.0
set opt_delay = 0
set opt_choice = n
set mcf_choice = 0

### set process id number – will be incremented with each iteration
###   of script (everytime that ’y’ is selected and a new run begins)
### need unique identification for each run so that different users
###   or different runs from the same user don’t overwrite data from
###   a concurrent run
set proc_id_beg = $$
@ proc_id = $proc_id_beg – 1



##### MAIN PROGRAM LOOP ################################

LOOP:

### increment run id number
@ proc_id = $proc_id + 1

clear

echo "################################################"
echo "# XTALK_SIM      version 2.0                   #"
echo "#                                              #"
echo "# Created on 10/13/95 by Andrew Percey         #"
echo "#                                              #"
echo "# If you have any questions or comments,       #"
echo "# contact Andrew Percey at 5-6304              #"
echo "#                                              #"
echo "################################################"
echo " "
echo " "

### define input directory variables
set andy_dir = /p6c/pd/fcpvrv/work/apercey/cse/NC
set input_dir = $andy_dir/Inputfiles
# set input_dir = /p6c/pd/fchip/floorplan/work/scripts/p854/xtalk

### write to log file to monitor script usage
echo ‘whoami‘ ‘date‘ >> /tmp/xs_$$.log
set date = ‘awk '{print $3 $4}' /tmp/xs_$$.log‘
set last_date = ‘awk '{print $3 $4}' $andy_dir/xs.log | tail –1‘
if ($date != $last_date) then
```

```
  (echo "  "; \
  echo "*******************************************************"; \
  echo "  ") >> $andy_dir/xs.log
endif
cat /tmp/xs_$$.log >> $andy_dir/xs.log
\rm /tmp/xs_$$.log


###### GET USER INPUTS AND MODIFY FILE TEMPLATES ############

echo "Enter the following parameters for the interconnect lines "
echo "(or hit <CR> for default values):"
echo " "

###### Driver information -------------------------------------

DRIVER_LOOP:
echo "Do you want to specify the driver with:"
echo "  (1) an explicit zi0lib cell"
echo "  (2) effective p and n sizes for an inverter model"
echo -n "  choice = <$dr_choice>:  "
set answer = $<
if ("$answer" != "") then
  set dr_choice = "$answer"
  if ("$dr_choice" != 1 && "$dr_choice" != 2) then
    echo "    *** "$dr_choice" is not a valid choice ***"
    goto DRIVER_LOOP
  endif
endif
echo " "

if ($dr_choice == 1) then
  DRLIB_LOOP:
  echo -n "    zi0lib driver library [basic, complex, etc] <$driver_lib>:  "
  set answer = $<
  if ("$answer" != "") then
    set driver_lib="$answer"
    more /libdir/zi0lib/$driver_lib >& /tmp/dr_lib_"$USER""$proc_id"
    if (`cat /tmp/dr_lib_"$USER""$proc_id" | wc -l` != 3) then
      echo "    *** zi0lib/$driver_lib library not found ***"
      goto DRLIB_LOOP
    endif
    \rm /tmp/dr_lib_"$USER""$proc_id"
  endif
  DR_LOOP:
  echo -n "    driver type <$driver>:  "
  set answer = $<
  if ("$answer" != "") then
    set driver="$answer"
    ls /libdir/zi0lib/$driver_lib/aux/$driver.aux \
        >& /tmp/dr_"$USER""$proc_id"
    if (`cat /tmp/dr_"$USER""$proc_id" | wc -w` != 1) then
      echo "    *** $driver not found in zi0lib/$driver_lib library ***"
      goto DR_LOOP
    endif
    \rm /tmp/dr_"$USER""$proc_id"
  endif

else if ($dr_choice == 2) then
  P_DR_LOOP:
  echo -n "    Effective p transistor size of driver <z = $p_dr>:  "
  set answer = $<
  if ("$answer" != "") then
    set p_dr = "$answer"
    echo "scale = 3; $p_dr * 1" | bc >& /tmp/p_dr_"$USER""$proc_id"
    if (‘cat /tmp/p_dr_"$USER""$proc_id" | wc -w‘ != 1) then
      echo "    *** $p_dr is not a number ***"
      goto P_DR_LOOP
    endif
    set int_num = ‘echo $p_dr | awk ’{print int($1 * 1000)}’‘
    if ($int_num < 0 || $p_dr == 0) then
      echo "    ### input value must be greater than zero ***"
      goto P_DR_LOOP
    endif
    \rm /tmp/p_dr_"$USER""$proc_id"
  endif
  N_DR_LOOP:
  echo -n "    Effective n transistor size of driver <z = $n_dr>:  "
  set answer = $<
  if ("$answer" != "") then
    set n_dr = "$answer"
    echo "scale = 3; $n_dr * 1" | bc >& /tmp/n_dr_"$USER""$proc_id"
    if (‘cat /tmp/n_dr_"$USER""$proc_id" | wc -w‘ != 1) then
      echo "    *** $n_dr is not a number ***"
      goto N_DR_LOOP
    endif
    set int_num = ‘echo $n_dr | awk ’{print int($1 * 1000)}’‘
    if ($int_num < 0 || $n_dr == 0) then
      echo "    ### input value must be greater than zero ***"
      goto N_DR_LOOP
    endif
    \rm /tmp/n_dr_"$USER""$proc_id"
  endif

endif

###### Receiver information ----------------------------------------

RECEIVER_LOOP:
echo " "
echo "Do you want to specify the receiver with"
echo "  (1) an explicit zi0lib cell"
echo "  (2) effective p and n sizes for an inverter model"
echo "  (3) a capacitive load"
echo -n "  choice = <$rec_choice>:  "
set answer = $<
if ("$answer" != "") then
```

```
  set rec_choice = "$answer"
  if ("$rec_choice" != 1 && "$rec_choice" != 2 && "$rec_choice" != 3) then
    echo "     *** "$rec_choice" is not a valid choice ***"
    goto RECEIVER_LOOP
  endif
endif
echo " "

if ($rec_choice == 1) then
  RECLIB_LOOP:
  echo −n "     zi0lib receiver library [basic, complex, etc] <$receiver_lib>:  "
  set answer = $<
  if ("$answer" != "") then
    set receiver_lib="$answer"
    more /libdir/zi0lib/$receiver_lib >& /tmp/rec_lib_"$USER""$proc_id"
    if (‘cat /tmp/rec_lib_"$USER""$proc_id" | wc −l‘ != 3) then
      echo "     *** zi0lib/$receiver_lib library not found ***"
      goto RECLIB_LOOP
    endif
    \rm /tmp/rec_lib_"$USER""$proc_id"
  endif
  REC_LOOP:
  echo −n "     receiver type <$receiver>:  "
  set answer = $<
  if ("$answer" != "") then
    set receiver="$answer"
    ls /libdir/zi0lib/$receiver_lib/aux/$receiver.aux \
          >& /tmp/rec_"$USER""$proc_id"
    if (‘cat /tmp/rec_"$USER""$proc_id" | wc −w‘ != 1) then
      echo "     *** $receiver not found in zi0lib/$receiver_lib library ***"
      goto REC_LOOP
    endif
    \rm /tmp/rec_"$USER""$proc_id"
  endif

else if ($rec_choice == 2) then
  P_REC_LOOP:
  echo −n "     Effective p transistor size of receiver <z = $p_rec>:  "
  set answer = $<
  if ("$answer" != "") then
    set p_rec = "$answer"
    echo "scale = 3;  $p_rec * 1" | bc >& /tmp/p_rec_"$USER""$proc_id"
    if (‘cat /tmp/p_rec_"$USER""$proc_id" | wc −w‘ != 1) then
      echo "     *** $p_rec is not a number ***"
      goto P_REC_LOOP
    endif
    set int_num = ‘echo $p_rec | awk '{print int($1 * 1000)}'‘
    if ($int_num < 0 || $p_rec == 0) then
      echo "     ### input value must be greater than zero ***"
      goto P_REC_LOOP
    endif
    \rm /tmp/p_rec_"$USER""$proc_id"
  endif
  N_REC_LOOP:
```

```
  echo -n "    Effective n transistor size of receiver <z = $n_rec>:  "
  set answer = $<
  if ("$answer" != "") then
    set n_rec = "$answer"
    echo "scale = 3; $n_rec * 1" | bc >& /tmp/n_rec_"$USER""$proc_id"
    if (‘cat /tmp/n_rec_"$USER""$proc_id" | wc -w‘ != 1) then
      echo "    *** $n_rec is not a number ***"
      goto N_REC_LOOP
    endif
    set int_num = ‘echo $n_rec | awk ’{print int($1 * 1000)}’‘
    if ($int_num < 0 || $n_rec == 0) then
      echo "    ### input value must be greater than zero ***"
      goto N_REC_LOOP
    endif
    \rm /tmp/n_rec_"$USER""$proc_id"
  endif

else if ($rec_choice == 3) then
  CAP_LOOP:
  echo -n "    capacitive load  <$cap_load pf>:  "
  set answer = $<
  if ("$answer" != "") then
    set cap_load = $answer
    set cap_load_save = $cap_load
    echo "scale = 3; $cap_load * 1" | bc >& /tmp/cap_load_"$USER""$proc_id"
    if (‘cat /tmp/cap_load_"$USER""$proc_id" | wc -w‘ != 1) then
      echo "    *** $cap_load is not a number ***"
      goto CAP_LOOP
    endif
    set int_num = ‘echo $cap_load | awk ’{print int($1 * 1000)}’‘
    if ($int_num < 0 || $cap_load == 0) then
      echo "    ### input value must be greater than zero ***"
      goto CAP_LOOP
    endif
    \rm /tmp/cap_load_"$USER""$proc_id"
  endif

endif


###### Input rise/fall times ------------------------------------

echo " "
RISE_LOOP:
echo -n "  20%-80% rise time for input to drivers <$t_rise NS>:  "
set answer = $<
if ("$answer" != "") then
  set t_rise="$answer"
  echo "scale = 3; $t_rise * 1" | bc >& /tmp/rise_"$USER""$proc_id"
  if (‘cat /tmp/rise_"$USER""$proc_id" | wc -w‘ != 1) then
    echo "    *** $t_rise is not a number ***"
    goto RISE_LOOP
  endif
  set int_num = ‘echo $t_rise | awk ’{print int($1 * 1000)}’‘
  if ($int_num < 0 || $t_rise == 0) then
    echo "    ### input value must be greater than zero ***"
    goto RISE_LOOP
  endif
  \rm /tmp/rise_"$USER""$proc_id"
endif

FALL_LOOP:
echo -n "  80%-20% fall time for input to drivers <$t_fall NS>:  "
set answer = $<
if ("$answer" != "") then
 set t_fall="$answer"
  echo "scale = 3; $t_fall * 1" | bc >& /tmp/fall_"$USER""$proc_id"
  if (`cat /tmp/fall_"$USER""$proc_id" | wc -w` != 1) then
    echo "    *** $t_fall is not a number ***"
    goto FALL_LOOP
  endif
  set int_num = `echo $t_fall | awk '{print int($1 * 1000)}'`
  if ($int_num < 0 || $t_fall == 0) then
    echo "    ### input value must be greater than zero ***"
    goto FALL_LOOP
  endif
  \rm /tmp/fall_"$USER""$proc_id"
endif

###### Process corner ----------------------------------------
### this parameter does not currently matter since only 'tttt' is being
###   used, but that may change in the future

SKEW_LOOP:
echo -n "  process corner <$skew>:  "
set answer = $<
if ("$answer" != "") then
  set skew="$answer"
  if (`echo $skew | wc -c` != 5) then
    echo "    *** $skew is not a valid process corner ***"
    goto SKEW_LOOP
  endif
endif

###### Set driver ----------------------------------------
### currently, for choice = 1, p and n sizes are retrieved from the .aux files

if ($dr_choice == 1) then
  set p_driver = `awk '($2 ~ /zpullup/) {print $4}' /libdir/zi0lib/$driver_lib/aux/{$driver}.aux`
  set n_driver = `awk '($2 ~ /zpulldn/) {print $4}' /libdir/zi0lib/$driver_lib/aux/{$driver}.aux`
else if ($dr_choice == 2) then
  set p_driver = $p_dr
  set n_driver = $n_dr
endif

###### Set receiver ----------------------------------------

if ($rec_choice == 1) then
```

```
  set p_receiver = `awk '($2 ~ /zpullup/) {print $4}' /libdir/zi0lib/$receiver_lib/aux/{$receiver}.aux`
  set n_receiver = `awk '($2 ~ /zpulldn/) {print $4}' /libdir/zi0lib/$receiver_lib/aux/{$receiver}.aux`
  set cap_load = 0
else if ($rec_choice == 2) then
  set p_receiver = $p_rec
  set n_receiver = $n_rec
  set cap_load = 0
else if ($rec_choice == 3) then
  set p_receiver = 0
  set n_receiver = 0
endif

###### Setup netlist and CSE command file templates ----------------

if ($rec_choice == 1 || $rec_choice == 2) then
  sed "s/xcap_model/xcap_model"$proc_id"/g" $input_dir/xcap_model_3.1.isp
        > $CSEDIR/xcap_model"$proc_id".isp
  sed "s/xcap_model/xcap_model"$proc_id"/g" $input_dir/xcap_model_3.1.iif
        > $CSEDIR/xcap_model"$proc_id".iif
  sed "s/xcap_model/xcap_model"$proc_id"/g" $input_dir/xcap_model_3.1.gux
        > $CSEDIR/xcap_model"$proc_id".gux
  sed  "s/xcap_model/xcap_model"$proc_id"/g; \
        s/XCAP_MODEL/XCAP_MODEL"$proc_id"/g; \
        s/gbdfa/"$proc_id"/g; \
        s/load_cap/$cap_load/; \
        s/var_rise_time/$t_rise/; \
        s/var_fall_time/$t_fall/; \
        s/var_p_dr/$p_driver/; \
        s/var_n_dr/$n_driver/; \
        s/var_p_rec/$p_receiver/; \
        s/var_n_rec/$n_receiver/; " \
     $input_dir/sim_xtalk.cmd > /tmp/sim_xtalk_"$USER""$proc_id".cmd

else if ($rec_choice == 3) then
  sed "s/xcap_model/xcap_model"$proc_id"/g" $input_dir/xcap_model_3.1.isp | \
   awk '($0 !~ /psize_rec/) && ($0 !~ /OUTPUT/) {print $0}' > $CSEDIR/xcap_model"$proc_id".isp
  sed "s/xcap_model/xcap_model"$proc_id"/g; s/a a1 b b1 c c1/a b c/" $input_dir/xcap_model_3.1.iif
        > $CSEDIR/xcap_model"$proc_id".iif
  sed "s/xcap_model/xcap_model"$proc_id"/g" $input_dir/xcap_model_3.1.gux
        > $CSEDIR/xcap_model"$proc_id".gux
  sed  "s/xcap_model/xcap_model"$proc_id"/g; \
        s/XCAP_MODEL/XCAP_MODEL"$proc_id"/g; \
        s/gbdfa/"$proc_id"/g; \
        s/load_cap/$cap_load/; \
        s/var_rise_time/$t_rise/; \
        s/var_fall_time/$t_fall/; \
        s/var_p_dr/$p_driver/; \
        s/var_n_dr/$n_driver/; "\
     $input_dir/sim_xtalk.cmd | awk '($4 !~ /var_p_rec/) && \
        ($4 !~ /var_n_rec/) {print $0}' > /tmp/sim_xtalk_"$USER""$proc_id".cmd
else
  echo "ERROR: Invalid receiver choice"
  exit 0
endif
```

```
set cap_load = $cap_load_save


###### FTRC Modeling -----------------------------------

echo " "
echo "FTRC modeling:"
echo " "

###### Interconnect length ---------------------------------

LENGTH_LOOP:
echo -n "  total line length <$length_total>:  "
set answer = $<
if ("$answer" != "") then
 set length_total = "$answer"
 echo "scale = 3; $length_total * 1" | bc >& /tmp/len_"$USER""$proc_id"
 if (‘cat /tmp/len_"$USER""$proc_id" | wc -w‘ != 1) then
   echo "    *** $length_total is not a number ***"
   goto LENGTH_LOOP
 endif
 set int_num = ‘echo $length_total | awk ’{print int($1 * 1000)}’‘
 if ($int_num < 0 || $length_total == 0) then
   echo "    ### input value must be greater than zero ***"
   goto LENGTH_LOOP
 endif
 \rm /tmp/len_"$USER""$proc_id"
endif
set length = ‘echo "scale = 2; $length_total / 10" | bc‘
echo " "

### FTRC specification loop

### initialize variables for calculations
set total_fraction=0.0
set total_xcap=0
set total_tot_cap=0

FTRC_LOOP:

###### FTRC type ----------

MODEL_LOOP:
echo -n "  FTRC model name [rm2fu, rm2m1m3, etc] <$model>:  "
set answer = $<
if ("$answer" != "") then
 set model="$answer"
 awk -f $input_dir/get_parameter.awk skew_type=$skew model="$model" \
        $CSEPRCFILE > /tmp/param_"$USER""$proc_id"
 if (‘cat /tmp/param_"$USER""$proc_id" |wc -l‘ < 4) then
   echo "    *** $model ftrc model not found ***"
   goto MODEL_LOOP
 endif
```

```
  \rm /tmp/param_"$USER""$proc_id"
endif

###### FTRC spacing -----------

SPACING_LOOP:
echo -n "     FTRC spacing <$spacing_l>:  "
set answer = $<
if ("$answer" != "") then
  set spacing_l="$answer"
  set spacing_r="$answer"
  echo "scale = 3; $spacing_l * 1" | bc >& /tmp/spc_"$USER""$proc_id"
  if (`cat /tmp/spc_"$USER""$proc_id" | wc -w` != 1) then
    echo "     *** $spacing_l is not a number ***"
    goto SPACING_LOOP
  endif
  set int_num = `echo $spacing_l | awk '{print int($1 * 1000)}'`
  if ($int_num < 0 || $spacing_l == 0) then
    echo "     ### input value must be greater than zero ***"
    goto SPACING_LOOP
  endif
  \rm /tmp/spc_"$USER""$proc_id"
endif

###### FTRC width -----------

WIDTH_LOOP:
echo -n "     FTRC width <$width>:  "
set answer = $<
if ("$answer" != "") then
 set width="$answer"
  echo "scale = 3; $width * 1" | bc >& /tmp/wid_"$USER""$proc_id"
  if (`cat /tmp/wid_"$USER""$proc_id" | wc -w` != 1) then
    echo "     *** $width is not a number ***"
    goto WIDTH_LOOP
  endif
  set int_num = `echo $width | awk '{print int($1 * 1000)}'`
  if ($int_num < 0 || $width == 0) then
    echo "     ### input value must be greater than zero ***"
    goto WIDTH_LOOP
  endif
  \rm /tmp/wid_"$USER""$proc_id"
endif

###### Fraction of line using this FTRC model -----------

FRACTION_LOOP:
echo -n "     fraction of line using this FTRC model <$fraction>:  "
set answer = $<
if ("$answer" != "") then
  set fraction = "$answer"
  echo "scale = 3; $fraction * 1" | bc >& /tmp/fraction_"$USER""$proc_id"
  if (`cat /tmp/fraction_"$USER""$proc_id" | wc -w` != 1) then
    echo "     *** $fraction is not a number ***"
    goto FRACTION_LOOP
  endif
  \rm /tmp/fraction_"$USER""$proc_id"
endif

### Check to make sure fraction is 0.1 or greater
set fraction_comp = ‘echo "scale = 3; $fraction * 100" | bc‘
set fraction_int = ‘echo $fraction_comp | awk '{print int($1)}'‘
if ($fraction_int < 10) then
  echo "    *** fraction must be 0.1 or greater ***"
  goto FRACTION_LOOP
endif

### Check to make sure fraction is a multiple of 0.1
set factor_float = ‘echo "scale = 3; $fraction / .1" | bc‘
set factor_comp = ‘echo "scale = 3; $factor_float * 1000" | bc‘
set mod = ‘echo $factor_comp | awk '{print ($1 % 1000)}'‘
if ($mod != 0) then
  echo "    *** fraction must be a multiple of 0.1 ***"
  goto FRACTION_LOOP
endif

### Check to make sure total fraction does not exceed 1.0
set factor = ‘echo $factor_float | awk '{print int($1)}'‘
set total_fraction = ‘echo "scale = 3; $total_fraction + $fraction" | bc‘
set comp_float = ‘echo "scale = 3; $total_fraction * 10" | bc‘
set comp = ‘echo $comp_float | awk '{print int($1)}'‘
if ($comp > 10) then
  echo "   *** total fraction cannot exceed 1.0 ***"
  set total_fraction = ‘echo "scale = 3; $total_fraction - $fraction" | bc‘
  goto FRACTION_LOOP
endif

### Once inputs are accepted, write FTRC info to file for later
###   retrieval for output screen
echo "  FTRC model:  $model   spacing:  $spacing_l   width:  $width
        fraction of line:  $fraction" >> /tmp/ftrc_modeling_"$USER""$proc_id".out


set total_factor_float = ‘echo "scale = 3; $total_fraction / .1" | bc‘
set total_factor = ‘echo $total_factor_float | awk '{print int($1)}'‘
set model_num_float = ‘echo "scale = 3; $total_factor - $factor + 1" | bc‘
set model_num_start = ‘echo $model_num_float | awk '{print int($1)}'‘

### Calculate line–line cap values for each FTRC for schematic coupling caps
echo " "
echo "Calculating capacitances and setting FTRC model(s)...    "
echo "(FTRC parameters read from $CSEPRCFILE)"
@ i = 0

while ($i < $factor)

  set model_num = ‘echo "scale = 3; $model_num_start + $i" | bc‘
  awk -f $input_dir/get_parameter.awk skew_type=$skew model=$model \
         $CSEPRCFILE > /tmp/param_"$USER""$proc_id"

  if (‘cat /tmp/param_"$USER""$proc_id" |wc -l‘ < 4) then
    echo "$model not found"
  else
    sed ’s/\=/\ = /g’ /tmp/param_"$USER""$proc_id" \
         | awk -f $input_dir/rc_prc_parameter_orig.awk space_l=$spacing_l \
         space_r=$spacing_r wid=$width len=$length > /tmp/rcc_"$USER""$proc_id"
         \rm /tmp/param_"$USER""$proc_id"
  endif

  set lcap = ‘awk ’($1 ~ /left/) {print $5}’ /tmp/rcc_"$USER""$proc_id"‘
  set total_xcap = ‘echo "scale = 3; $total_xcap + $lcap * 2" | bc‘
  set tot_cap = ‘awk ’($2 ~ /interconnect/) && ($3 ~ /capacitance/) {print $5}’
         /tmp/rcc_"$USER""$proc_id"‘
  set total_tot_cap = ‘echo "scale = 3; $total_tot_cap + $tot_cap" | bc‘

  \rm /tmp/rcc_"$USER""$proc_id"

### Replace default values in split with user inputs

sed "s/ftrc_"$model_num"_model/$model/" $CSEDIR/xcap_model"$proc_id".isp
        > /tmp/xcap_model"$proc_id".isp
\mv /tmp/xcap_model"$proc_id".isp $CSEDIR/xcap_model"$proc_id".isp

sed "    s/var_"$model_num"_cap/$lcap/; \
         s/var_"$model_num"_length/$length/; \
         s/var_"$model_num"_spacing/$spacing_l/; \
         s/var_"$model_num"_width/$width/; " \
         /tmp/sim_xtalk_"$USER""$proc_id".cmd > /tmp/tmp_"$USER""$proc_id".cmd

\mv /tmp/tmp_"$USER""$proc_id".cmd /tmp/sim_xtalk_"$USER""$proc_id".cmd

@ i = $i + 1

end #while

echo "...done"
echo " "

### $total_factor = int ($total_fraction / 0.1)
if ($total_factor < 10) then
  goto FTRC_LOOP
else if ($total_factor > 10) then
  echo "ERROR: total fraction exceeds 1.0 *****"
  exit 0
endif
set ave_xcap = ‘echo "scale = 3; $total_xcap / $total_tot_cap" | bc‘


### Optional additional delay splits ----------------------------------
```

```
OPT_DELAY_LOOP:
echo "Run extra simulations without explicit cross-capacitors"
echo -n "  to compare delays?  (y/n) <$opt_choice>:  "
set answer = $<
if ($answer != "") then
  set opt_choice = $answer
endif
if ("$opt_choice" == "y" || "$opt_choice" == "yes") then
  set opt_delay = 1
  goto MORE_SPLITS
else if ("$opt_choice" == "n" || "$opt_choice" == "no") then
  goto SET_OPT_VARS
else
  goto OPT_DELAY_LOOP
endif

MORE_SPLITS:
echo " "
echo "What MCF value should be used for modeling the delay?"
MCF_LOOP:
echo -n "  MCF (0,1,2) <$mcf_choice>:  "
set answer = $<
if ("$answer" != "") then
  set mcf_choice = "$answer"
  if ("$mcf_choice" != 0 && "$mcf_choice" != 1 && "$mcf_choice" != 2) then
    echo "     *** "$mcf_choice" is not a valid value ***"
    goto MCF_LOOP
  endif
endif

SET_OPT_VARS:

if $opt_delay then
  sed  "s/opt_mcf/$mcf_choice/; \
         s/opt_spl/ACTIVE/;" \
         /tmp/sim_xtalk_"$USER""$proc_id".cmd > /tmp/tmp_"$USER""$proc_id".cmd
else
   sed "s/opt_mcf/$mcf_choice/; \
         s/opt_spl/NOT ACTIVE/;" \
         /tmp/sim_xtalk_"$USER""$proc_id".cmd > /tmp/tmp_"$USER""$proc_id".cmd
endif

\mv /tmp/tmp_"$USER""$proc_id".cmd /tmp/sim_xtalk_"$USER""$proc_id".cmd


###### OUTPUT SCREEN ####################################

OUTPUT_SCREEN:

echo " "

(echo " "; \
echo "XTALK_SIM SIMULATION RESULTS"; \
echo " "; \
echo " "; \
echo "Simulation parameters chosen:"; \
echo " ") > $CSEDIR/xtalk_sim_"$proc_id".output

if ($dr_choice == 1) then
 (echo "  zi0lib driver library:           $driver_lib"; \
 echo "  driver type:                                $driver") >> \
        $CSEDIR/xtalk_sim_"$proc_id".output
else if ($dr_choice == 2) then
 (echo "  driver effective p size:      $p_driver"; \
 echo "  driver effective n size:      $n_driver") >> \
        $CSEDIR/xtalk_sim_"$proc_id".output
endif

if ($rec_choice == 1) then
 (echo "  ziolib receiver library:      $receiver_lib"; \
 echo "  receiver type:                            $receiver") >> \
        $CSEDIR/xtalk_sim_"$proc_id".output
else if ($rec_choice == 2) then
 (echo "  receiver effective p size:   $p_receiver"; \
 echo "  receiver effective n size:   $n_receiver") >> \
        $CSEDIR/xtalk_sim_"$proc_id".output
else if ($rec_choice == 3) then
 echo "  receiver capacitive load:     $cap_load pf" >> \
        $CSEDIR/xtalk_sim_"$proc_id".output
endif

(echo " "; \
echo "  driver input rise time:         $t_rise NS"; \
echo "  driver input fall time:         $t_fall NS"; \
echo "  process corner:                          $skew"; \
echo " "; \
echo "  total line length:                   $length_total microns"; \
echo " "; \
awk '{print $0}' /tmp/ftrc_modeling_"$USER""$proc_id".out; \
\rm /tmp/ftrc_modeling_"$USER""$proc_id".out; \
echo " ") >> $CSEDIR/xtalk_sim_"$proc_id".output

echo " " >> $CSEDIR/xtalk_sim_"$proc_id".output
echo "Simulation results - worst case effects on victim line:"
        >> $CSEDIR/xtalk_sim_"$proc_id".output
echo " " >> $CSEDIR/xtalk_sim_"$proc_id".output


###### INVOKE CSE AND EXECUTE COMMAND FILES #############

echo "Building CSE circuit database and performing TIM simulation..."
echo "  Please be patient - this will take a few minutes."
echo "  Results will be displayed in an emacs window."
echo " "

### Need to use modified job file (modified to not echo commands)
setenv CSEJOBFILE $input_dir/jobman.csh
```

```
### Need to set up these dummy files to avoid error messages being
###   printed to the screen if they do not exist when being checked
###   by xc_finish
\cp $input_dir/testfile.txt $CSEDIR/xmodel"$proc_id"_dur_S0.results
\cp $input_dir/testfile.txt $CSEDIR/xmodel"$proc_id"_dur_S1.results
\cp $input_dir/testfile.txt $CSEDIR/xmodel"$proc_id"_dur_S2.results
\cp $input_dir/testfile.txt $CSEDIR/xmodel"$proc_id"_dur_S3.results

### need to pass run id number to xc_finish
set var = "$proc_id"

### calculates x and y coordinates of emacs results windows for "tiling" effect
@ diff = $proc_id − $proc_id_beg
@ num = $diff − (($diff / 10) * 10)
@ xcoord = (40 * $num)
@ ycoord = (15 * $num)

### the following set of commands in () are spawned to the background
### 1: the CSE .cmd file is run and the simulation is sent
### 2: xc_finish is called to format and output the results
### 3: the output file is set to read−only
### 4: the results file is brought up in an emacs window
(echo "@/tmp/sim_xtalk_"$USER""$proc_id"" | $CSE/cse −t
          > $CSEDIR/xtalk_sim_"$proc_id".log; \
$input_dir/xc_finish $var $ave_xcap $opt_choice $mcf_choice
          >> $CSEDIR/xtalk_sim_"$proc_id".output; \
chmod 444 $CSEDIR/xtalk_sim_"$proc_id".output; \
/usr/local/bin/emacs −geometry =80x65+$xcoord+$ycoord
          $CSEDIR/xtalk_sim_"$proc_id".output &) &


###### DONE WITH CALCULATIONS ######################

EXIT_LOOP:
echo " "
echo " "
echo −n "Run xtalk_sim again?  (y/n) "
set answer=$<
echo " "
if ("$answer" == "y" || "$answer" == "yes") then
          goto LOOP
      exit 0
else if ("$answer" == "n" || "$answer" == "no") then
      exit 0
else goto EXIT_LOOP
endif
```

```
# xc_finish: called by xtalk_sim to output simulation results

#! /bin/csh -f

set proc_id = $argv[1]
set extra_delay_opt = $argv[3]
set mcf = $argv[4]

set vss = 0.0
set vcc = vcc_project


###### OUTPUT RESULTS ###################################

### Split0 output – glitches and duration

echo  " %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"
echo  " % Attacking lines RISING from Vss to Vcc:            %"
echo  " %                                                    %"

set test0 = ""
set test0 = `awk '(/mV/) {print $3}' $CSEDIR/xmodel"$proc_id"_glitch_S0.results | sed 's/mV//'`
if ($test0 != "") then
  set max_glitch0 = `echo "scale = 4; $test0 / 1000" | bc`
else
  set max_glitch0 = `awk '(/V/) {print $3}' $CSEDIR/xmodel"$proc_id"_glitch_S0.results | sed 's/V//'`
endif
set glitch_mag0 = `echo "scale = 3; $max_glitch0 - $vss" | bc`
set glitch_val0 = "+$glitch_mag0 V"
echo  " %     Voltage glitch from Vss:      $glitch_val0      %"

set test = 0
set test = `grep CROSS $CSEDIR/xmodel"$proc_id"_dur_S0.results | sed 's/TABLE//'`
if ($test == CROSS) then
  set dur_beg = `awk '($3 ~ /RISE/) {print $4}'
        $CSEDIR/xmodel"$proc_id"_dur_S0.results | sed 's/nS//' | head -1`
  set dur_end = `awk '($3 ~ /FALL/) {print $4}'
        $CSEDIR/xmodel"$proc_id"_dur_S0.results | sed 's/nS//' | tail -1`
  set dur_val = `echo "scale = 3; $dur_end - $dur_beg" | bc`
  echo " %       Duration glitch > 0.4V:   $dur_val NS        %"
endif
echo  " %                                                    %"


### Split1 output – glitches and duration

set test0 = ""
set test0 = `awk '(/mV/) {print $3}' $CSEDIR/xmodel"$proc_id"_glitch_S1.results | sed 's/mV//'`
if ($test0 != "") then
  set max_glitch1 = `echo "scale = 4; $test0 / 1000" | bc`
else
  set max_glitch1 = `awk '(/V/) {print $3}' $CSEDIR/xmodel"$proc_id"_glitch_S1.results | sed 's/V//'`
```

```
endif
set glitch_mag1 = ‘echo "scale = 3; $max_glitch1 - $vcc" | bc‘
set glitch_val1 = "+$glitch_mag1 V"
endif
echo  " %     Voltage glitch from Vcc:      $glitch_val1     %"

set test = 0
set test = ‘grep CROSS $CSEDIR/xmodel"$proc_id"_dur_S1.results | sed 's/TABLE//'‘
if ($test == CROSS) then
  set dur_beg = ‘awk '($3 ~ /RISE/) {print $4}'
          $CSEDIR/xmodel"$proc_id"_dur_S1.results | sed 's/nS//' | head -1‘
  set dur_end = ‘awk '($3 ~ /FALL/) {print $4}'
          $CSEDIR/xmodel"$proc_id"_dur_S1.results | sed 's/nS//' | tail -1‘
  set dur_val = ‘echo "scale = 3; $dur_end - $dur_beg" | bc‘
  echo " %        Duration glitch > 0.4V:   $dur_val NS        %"
endif
echo  " %                                                    %"


### Split4 output - delay

set driver_in = ‘awk '($2 ~ /B@xm/) {print $4}'
          $CSEDIR/xmodel"$proc_id"_delay_S4.results | sed 's/nS//'‘
set driver_out = ‘awk '($2 ~ /B_I/) {print $4}'
          $CSEDIR/xmodel"$proc_id"_delay_S4.results | sed 's/nS//'‘
set receiver_in = ‘awk '($2 ~ /B1_I/) {print $4}'
          $CSEDIR/xmodel"$proc_id"_delay_S4.results | sed 's/nS//'‘
set driver_del = ‘echo "scale = 3; $driver_out -  $driver_in"  | bc‘
set line_del   = ‘echo "scale = 3; $receiver_in - $driver_out" | bc‘
set total_del  = ‘echo "scale = 3; $driver_del + $line_del"| bc‘

if ($extra_delay_opt == "n" || $extra_delay_opt == "no") then
  echo   " %     Delay with cross-coupling capacitor model:    %"
  echo   " %        Interconnect delay:    $line_del NS              %"
  echo   " %        Driver delay:                $driver_del NS        %"
  echo   " %        Total delay:                  $total_del NS        %"
  echo   " %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"
  echo   " "

else if ($extra_delay_opt == "y" || $extra_delay_opt == "yes") then
  set driver_in = ‘awk '($2 ~ /B@xm/) {print $4}'
          $CSEDIR/xmodel"$proc_id"_delay_S6.results | sed 's/nS//'‘
  set driver_out = ‘awk '($2 ~ /B_I/) {print $4}'
          $CSEDIR/xmodel"$proc_id"_delay_S6.results | sed 's/nS//'‘
  set receiver_in = ‘awk '($2 ~ /B1_I/) {print $4}'
          $CSEDIR/xmodel"$proc_id"_delay_S6.results | sed 's/nS//'‘
  set driver_del2 = ‘echo "scale = 3; $driver_out -  $driver_in"  | bc‘
  set line_del2   = ‘echo "scale = 3; $receiver_in - $driver_out" | bc‘
  set total_del2  = ‘echo "scale = 3; $driver_del2 + $line_del2"| bc‘
  echo   " %     Delay with           cross-caps  MCF = $mcf %"
  echo   " %        Interconnect delay:    $line_del NS    $line_del2 NS %"
  echo   " %        Driver delay:                $driver_del NS    $driver_del2 NS %"
  echo   " %        Total delay:                  $total_del NS    $total_del2 NS %"
  echo   " %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"
  echo " "
endif

### Split2 output - glitches and duration

echo " %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"
echo " % Attacking lines FALLING from Vcc to Vss:            %"
echo " %                                                     %"

set test0 = ""
set test0 = `awk '(/mV/) {print $3}' $CSEDIR/xmodel"$proc_id"_glitch_S2.results | sed 's/mV//'`
if ($test0 != "") then
  set min_glitch2 = `echo "scale = 4; $test0 / 1000" | bc`
else
  set min_glitch2 = `awk '(/V/) {print $3}' $CSEDIR/xmodel"$proc_id"_glitch_S2.results | sed 's/V//'`
endif
set glitch_mag2 = `echo "scale = 3; $min_glitch2 - $vss" | bc`
set glitch_val2 = "$glitch_mag2 V"
echo " %      Voltage glitch from Vss:       $glitch_val2      %"

set test = 0
set test = `grep CROSS $CSEDIR/xmodel"$proc_id"_dur_S2.results | sed 's/TABLE//'`
if ($test == CROSS) then
  set dur_beg = `awk '($3 ~ /FALL/) {print $4}'
        $CSEDIR/xmodel"$proc_id"_dur_S2.results | sed 's/nS//' | head -1`
  set dur_end = `awk '($3 ~ /RISE/) {print $4}'
        $CSEDIR/xmodel"$proc_id"_dur_S2.results | sed 's/nS//' | tail -1`
  set dur_val = `echo "scale = 3; $dur_end - $dur_beg" | bc`
  echo " %        Duration glitch < -0.4V:  $dur_val NS         %"
endif
echo " %                                                     %"

### Split3 output - glitches and duration

set test0 = ""
set test0 = `awk '(/mV/) {print $3}' $CSEDIR/xmodel"$proc_id"_glitch_S3.results | sed 's/mV//'`
if ($test0 != "") then
  set min_glitch3 = `echo "scale = 4; $test0 / 1000" | bc`
else
  set min_glitch3 = `awk '(/V/) {print $3}' $CSEDIR/xmodel"$proc_id"_glitch_S3.results | sed 's/V//'`
endif
set glitch_mag3 = `echo "scale = 3; $min_glitch3 - $vcc" | bc`
set glitch_val3 = "$glitch_mag3 V"
endif
echo " %      Voltage glitch from Vcc:       $glitch_val3      %"

set test = 0
set test = `grep CROSS $CSEDIR/xmodel"$proc_id"_dur_S3.results | sed 's/TABLE//'`
if ($test == CROSS) then
  set dur_beg = `awk '($3 ~ /FALL/) {print $4}'
        $CSEDIR/xmodel"$proc_id"_dur_S3.results | sed 's/nS//' | head -1`
  set dur_end = ‘awk ’($3 ~ /RISE/) {print $4}’
        $CSEDIR/xmodel"$proc_id"_dur_S3.results | sed ’s/nS//’ | tail –1‘
  set dur_val = ‘echo "scale = 3; $dur_end - $dur_beg" | bc‘
  echo " %        Duration glitch < -0.4V:  $dur_val NS        %"
endif
echo  " %                                                      %"


### Split5 output – delay

set driver_in = ‘awk ’($2 ~ /B@xm/) {print $4}’
        $CSEDIR/xmodel"$proc_id"_delay_S5.results | sed ’s/nS//’‘
set driver_out = ‘awk ’($2 ~ /B_I/) {print $4}’
        $CSEDIR/xmodel"$proc_id"_delay_S5.results | sed ’s/nS//’‘
set receiver_in = ‘awk ’($2 ~ /B1_I/) {print $4}’
        $CSEDIR/xmodel"$proc_id"_delay_S5.results | sed ’s/nS//’‘
set driver_del = ‘echo "scale = 3; $driver_out -  $driver_in"  | bc‘
set line_del  = ‘echo "scale = 3; $receiver_in - $driver_out" | bc‘
set total_del  = ‘echo "scale = 3; $driver_del + $line_del"| bc‘

if ($extra_delay_opt == "n" || $extra_delay_opt == "no") then
  echo  " %     Delay with cross-coupling capacitor model:   %"
  echo  " %       Interconnect delay:    $line_del NS              %"
  echo  " %       Driver delay:                $driver_del NS        %"
  echo  " %       Total delay:                  $total_del NS        %"
  echo  " %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"
  echo  " "
  echo  " average (cross cap)/(total cap) along victim line = $argv[2]"
  echo  " "
  echo  " "

else if ($extra_delay_opt == "y" || $extra_delay_opt == "yes") then
  set driver_in = ‘awk ’($2 ~ /B@xm/) {print $4}’
        $CSEDIR/xmodel"$proc_id"_delay_S7.results | sed ’s/nS//’‘
  set driver_out = ‘awk ’($2 ~ /B_I/) {print $4}’
        $CSEDIR/xmodel"$proc_id"_delay_S7.results | sed ’s/nS//’‘
  set receiver_in = ‘awk ’($2 ~ /B1_I/) {print $4}’
        $CSEDIR/xmodel"$proc_id"_delay_S7.results | sed ’s/nS//’‘
  set driver_del2 = ‘echo "scale = 3; $driver_out -  $driver_in"  | bc‘
  set line_del2  = ‘echo "scale = 3; $receiver_in - $driver_out" | bc‘
  set total_del2  = ‘echo "scale = 3; $driver_del2 + $line_del2"| bc‘
  echo  " %     Delay with              cross-caps  MCF = $mcf %"
  echo  " %       Interconnect delay:    $line_del NS   $line_del2 NS %"
  echo  " %       Driver delay:                $driver_del NS   $driver_del2 NS %"
  echo  " %       Total delay:                  $total_del NS   $total_del2 NS %"
  echo  " %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%"
  echo  " "
  echo  " average (cross cap)/(total cap) along victim line = $argv[2]"
  echo  " "
  echo  " "

endif
```

```
###### DONE WITH NOISE CALCULATIONS #####################

echo "This file is in $CSEDIR"
echo "Note the full file name (in the Emacs status bar below) for reference"
echo " "
echo "To close this window, type CTRL-x-c"
echo " "

\rm /tmp/sim_xtalk_"$USER""$proc_id".cmd

\rm $CSEDIR/xmodel"$proc_id"_glitch_S0.results
\rm $CSEDIR/xmodel"$proc_id"_glitch_S1.results
\rm $CSEDIR/xmodel"$proc_id"_glitch_S2.results
\rm $CSEDIR/xmodel"$proc_id"_glitch_S3.results

\rm $CSEDIR/xmodel"$proc_id"_dur_S0.results
\rm $CSEDIR/xmodel"$proc_id"_dur_S1.results
\rm $CSEDIR/xmodel"$proc_id"_dur_S2.results
\rm $CSEDIR/xmodel"$proc_id"_dur_S3.results

\rm $CSEDIR/xmodel"$proc_id"_delay_S4.results
\rm $CSEDIR/xmodel"$proc_id"_delay_S5.results

if ($extra_delay_opt == "y" || $extra_delay_opt == "yes") then
  \rm $CSEDIR/xmodel"$proc_id"_delay_S6.results
  \rm $CSEDIR/xmodel"$proc_id"_delay_S7.results
endif

\rm $CSEDIR/xcap_model"$proc_id".isp
\rm $CSEDIR/xcap_model"$proc_id".iif
\rm $CSEDIR/xcap_model"$proc_id".gux
\rm $CSEDIR/xcap_model"$proc_id".sdp

\rm $CSEDIR/xmodel"$proc_id".def
\rm $CSEDIR/xmodel"$proc_id".runinput
\rm $CSEDIR/xmodel"$proc_id".spo
\rm $CSEDIR/xmodel"$proc_id".timout
\rm $CSEDIR/xmodel"$proc_id"_1.info
\rm $CSEDIR/xmodel"$proc_id"_1.op
\rm $CSEDIR/xmodel"$proc_id"_1.split
\rm $CSEDIR/xmodel"$proc_id"_2.info
\rm $CSEDIR/xmodel"$proc_id"_2.op
\rm $CSEDIR/xmodel"$proc_id"_2.split
\rm $CSEDIR/xmodel"$proc_id"_3.info
\rm $CSEDIR/xmodel"$proc_id"_3.op
\rm $CSEDIR/xmodel"$proc_id"_3.split
\rm $CSEDIR/xmodel"$proc_id"_4.info
\rm $CSEDIR/xmodel"$proc_id"_4.op
\rm $CSEDIR/xmodel"$proc_id"_4.split
\rm $CSEDIR/xmodel"$proc_id"_5.info
\rm $CSEDIR/xmodel"$proc_id"_5.op
```

```
\rm $CSEDIR/xmodel"$proc_id"_5.split
\rm $CSEDIR/xmodel"$proc_id"_6.info
\rm $CSEDIR/xmodel"$proc_id"_6.op
\rm $CSEDIR/xmodel"$proc_id"_6.split

if ($extra_delay_opt == "y" || $extra_delay_opt == "yes") then
  \rm $CSEDIR/xmodel"$proc_id"_7.info
  \rm $CSEDIR/xmodel"$proc_id"_7.op
  \rm $CSEDIR/xmodel"$proc_id"_7.split
  \rm $CSEDIR/xmodel"$proc_id"_8.info
  \rm $CSEDIR/xmodel"$proc_id"_8.op
  \rm $CSEDIR/xmodel"$proc_id"_8.split
endif

exit 0
```

References

[1] David Halliday and Robert Resnick. *Physics, Part II.* John Wiley & Sons, Inc., New York, NY, third edition, 1978.

[2] Stephen D. Senturia and Bruce D. Wedlock. *Electronic Circuits and Applications.* Krieger, Malabar, FL, 1993.

[3] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design.* Addison-Wesley, Reading, MA, second edition, 1993.