AN ABSTRACT OF THE THESIS OF

Patrick Birrer for the degree of Master of Science in Electrical and Computer Engineering presented on January 9, 2004.

Title: Silencer! A Tool for Substrate Noise Coupling Analysis.

Abstract approved: _____

Kartikeya Mayaram Terri S. Fiez

This thesis presents Silencer!, a fully automated, schematic-driven tool for substrate noise coupling simulation and analysis. It has been integrated in the CADENCE DFII environment and seamlessly enables substrate coupling analysis in a standard mixed-signal design flow. Silencer! aids IC designers in the analysis of substrate noise coupling at different levels of hierarchy - from a level where only an approximate layout of the transistors is known to a level that incorporates various parasitic elements. It can be used for layout optimization to reduce substrate cross-talk noise between circuitry that injects noise into the substrate and other circuitry that is sensitive to it. Examples in a TSMC 0.35µm heavily doped process have been simulated and the results are in good agreement with measured fabricated chips.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
09 JAN 2004		**00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Silencer! A Tool for Substrate Noise Coupling Analysis	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Oregon State University,School of Electrical Engineering and Computer Science,1148 Kelley Engineering Center,Corvallis,OR,97331-5501**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis presents Silencer!, a fully automated, schematic-driven tool for substrate noise coupling simulation and analysis. It has been integrated in the CADENCE DFII environment and seamlessly enables substrate coupling analysis in a standard mixed-signal design flow. Silencer! aids IC designers in the analysis of substrate noise coupling at different levels of hierarchy - from a level where only an approximate layout of the transistors is known to a level that incorporates various parasitic elements. It can be used for layout optimization to reduce substrate cross-talk noise between circuitry that injects noise into the substrate and other circuitry that is sensitive to it. Examples in a TSMC 0.35μm heavily doped process have been simulated and the results are in good agreement with measured fabricated chips.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**148**	

Silencer! A Tool for Substrate Noise Coupling Analysis

by

Patrick Birrer

A THESIS

Submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented January 9, 2004
Commencement June 2004

Master of Science thesis of Patrick Birrer presented on January 9, 2004

APPROVED:

Co-Major Professor, representing Electrical and Computer Engineering

---

Co-Major Professor, representing Electrical and Computer Engineering

---

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Patrick Birrer, Author

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude towards all the people who assisted me during my schooling here at Oregon State University in general and particularly while working on my research.

My sincere thanks go to Dr. Kartikeya Mayaram and Terri Fiez, my research advisors at Oregon State University for having given me an opportunity to join their research group in Substrate Noise Coupling and giving me tremendous support to accomplish my goals. I consider myself as very fortunate to have worked with them. Moreover, I remain thankful to SRC, NSF and the DARPA TEAM project for the financial support.

I would like to acknowledge Dr. Dean Jensen and Dr. Chenggang Xu for showing interest in my work and serving on my defense committee.

Furthermore, I thank Hans and Adele Neukomm, for their personal support and encouragement during the entire course of studies. The “Hans and Adele Neukomm Fellowship Fund” not only financially aids students from Burgdorf coming to Corvallis, but also establishes a long term relationship between Burgdorf School of Engineering and Oregon State University.

I would like to take this opportunity to thank Dr. Chris Bell, Dr. Franz Bachmann, Jean-Pierre Steger, and Marianne Schmalz, for encouraging me to apply for academic studies at OSU and for their recommendations. I am especially grateful to Dr. Chris Bell for his continuous help and advice during the time I was studying at Oregon State University.

I also want to thank Stefan, Christoph, Caroline, José, and Marcela for their friendship and support. Moreover, many thanks to all of my friends here in Corvallis for making the time I spent here a personal enrichment. Especially, I

would like to thank Donna and Larry Blus for being my home-stay family not only when I arrived in Corvallis but also at the end of my staying at OSU.

Thanks also to Ajit Sharma, Husni Habal, Shu Ching Hsu, Sirisha Adluri, Brian Owens, and the rest of the Substrate Noise Coupling group for the cooperation in the research project and all the thoughtful discussions.

Many thanks also to Taras Dudar, Volodymyr Kratyuk, Madhu Chennam, and Kannan Soundarapandian for their guidance in the process of getting familiar with the working environment, in the initial phase of my studies. Many thanks to Sasi Kumar Anurachalam and Prasad Talasila for helping me with measurements and simulations.

I remain thankful to Ferne Simendinger, April Melton, Sarah O'Leary, and all the office staff for all the help provided during my academic stay at OSU. I am also grateful to the school of Electrical Engineering and Computer Science at Oregon State University for providing me an excellent working environment.

Above all, I want to thank my parents for their support and encouragement during my schooling at Oregon State University. They always stood by me, also in less successful times of my academic studies, or when difficult decisions had to be made.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

LIST OF APPENDICES

LIST OF APPENDICES (Continued)

Appendix Page

To my family

SILENCER! A TOOL FOR SUBSTRATE NOISE COUPLING ANALYSIS

1. INTRODUCTION

1.1 Motivation

Over the past several years, technological advances have enabled engineers to produce smaller, yet more complex electronic equipment. Engineers are continually trying to fit more functionality on a chip, increase the speed, and accommodate more features in a smaller space. Consequently, there has been a considerable emphasis on deep submicron (DSM) System-on-a-Chip (SoC) solutions. An example of such a System-on-a-Chip is a wallet-sized cell phone that now offers capabilities which were unimaginable even a few years ago. Due to decreasing feature sizes, increasing clock frequencies, lower supply voltages, and larger interconnect parasitics, the digital portions of a mixed-signal chip become much noisier and often interfere with sensitive analog and RF circuitry. Therefore, it is a challenging task to design high-speed mixed-signal and RF ICs and substrate noise coupling becomes a key issue that must be considered for SoC applications [1], [2], [3], [17].

Due to the complexity of the problem, it is very difficult to accurately estimate substrate noise coupling in a mixed-signal design. Therefore, until recently, IC designers were forced to use rule-of-thumb guidelines when trying to minimize substrate noise coupling. However, these rule-of-thumb guidelines often led to either under- or over-engineered designs.

Being able to use a tool to solve substrate noise coupling issues leads not only to better designs, but it also reduces the time-to-market. Therefore, over the

past few years, several substrate noise coupling analysis tools have been developed [8], [9], [10], [11]. However, some disadvantages of these tools (a summary of these tools is given in Section 2.3) are the time and computational recourses required to extract a substrate network. Another common problem is the large netlists that lead to long simulation times, before a result can be obtained. In addition, a tool may be difficult to set up and to use, or a tool may not look at substrate noise coupling as a holistic problem that is it may not be seamlessly integrated into a mixed-signal design flow. Furthermore, most tools focus on the final post-layout verification. For example, they back annotate the substrate network, which was extracted from the process information, to the completed and extracted layout. However, once a layout is completed, it is usually too late and too expensive to make any major changes. In other words, even though the post-layout verification is an important part of a substrate noise coupling analysis tool and needs to be addressed, a holistic tool should also be usable in early stages of design, e.g., for floor-planning, or when only parts of the layout are developed.

It is very likely that the noise predicted in the pre-layout stage is less accurate. However, if the tool at least predicts some trends about substrate noise cross-talk between noisy digital and sensitive analog/RF circuitry, it can be very useful to the designer.

Silencer! tackles the disadvantages of existing substrate noise coupling analysis tools and builds a holistic approach and a platform for pre-layout substrate noise coupling analysis, addressed from a circuit design perspective. It uses the information that is present in the layout at a certain stage of design, and extracts (or estimates) a substrate network from that information. Then, it back annotates the network into the schematic view. Therefore, the concept of Silencer! is a schematic-driven tool that allows analysis and optimization of substrate noise coupling at different stages of design. Thus, noise suppression strategies can be used, if necessary, before the entire layout is finished. For instance, by using

Silencer!, designers can determine which one of the design changes actually achieved the successful result before they go to the next design step.

1.2 Substrate Noise

Substrate noise in mixed-signal ICs is due to the coupling of digital switching noise into the substrate from the switching transistors and interconnects. Other causes for noise injection are contacts that connect the substrate and wells to the power supplies. Current variations in the power supply and ground lines combined with the line inductances and routing resistances result in noise injection into the substrate [1]. Once injected into the substrate, noise can propagate via the shared silicon substrate and affect the performance of sensitive analog, RF, and other digital circuits on the same chip. Figure 1.1 illustrates how generated substrate noise propagates through the silicon substrate, and is picked up by a sensitive analog or RF circuit.

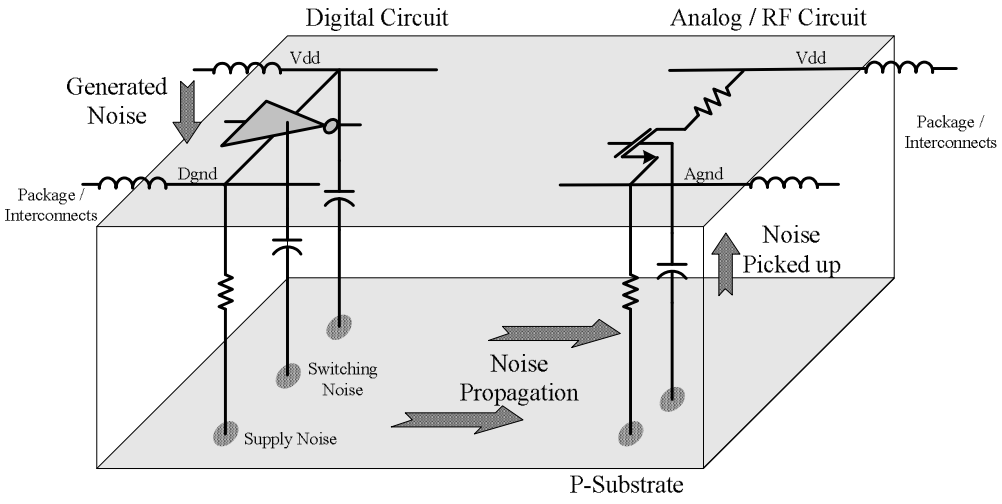

Figure 1.1 Substrate noise generated by a digital inverter propagates through the shared substrate to a sensitive analog NMOS transistor.

For example, a mixed-signal designer may be concerned about noise that couples from a digital signal processor (DSP) to a sensitive high gain amplifier. A RF designer, on the other hand, may worry about noise from the mixer coupling into the low noise amplifier (LNA). Moreover, the design of high-speed digital circuits with embedded sensitive circuitry, such as phase-locked loops (PLLs), presents another situation, where substrate noise can be a severe problem. PLLs are fundamental building blocks commonly used for data recovery in disk drives, in wireless communications to generate the local oscillator (LO), in high-speed micro-processors (for clock generation), and in memories. Substrate noise coupling into a PLL increases the jitter of the PLL, and in many cases, it not only degrades the performance, but even causes malfunctioning of an entire chip.

1.3 Summary of Previous Work

Previous research on substrate noise coupling simulation and analysis at Oregon State University (OSU) focused on substrate modeling, such as scalable macro-models [5], [6], boundary element models (EPIC) [7] and validation of these models with measurements from test structures, fabricated in various processes (TSMC 0.35µm heavily doped process, TSMC 0.25µm heavily and lightly doped process, SiGe 7HP lightly doped process, and SOI process through MIT). Moreover, research has been done in efficient and accurate probing of substrate noise using sensing options [16]. It has been determined, which sources of substrate noise are the dominant ones, and how different packages, interconnects, and other parasitic elements influence substrate noise coupling.

In the past two years, more complex circuits were fabricated at OSU to validate the substrate models and to correctly connect substrate networks to more complex layout structures. Due to the complexity of these circuits, it is inconvenient and time consuming to compute and connect a substrate network manually, and therefore, a prototype substrate coupling tool in Perl script was developed to extract a substrate network from the geometric information of a Caltech Intermediate Format (CIF) file. However, this tool was not seamlessly integrated into a mixed-signal design flow. In addition, the CIF file format, which is not a standardized file format, is very large for more complicated circuits and, therefore, is not practical to use.

1.4 Contributions of This Work

Silencer! combines the various aspects of a substrate noise coupling analysis methodology - based on research at Oregon State University over the past several years. It includes a self-guiding user interface that is fully integrated into the CADENCE DFII environment.

As already mentioned in Section 1.1, many of today's available substrate coupling tools [8], [9], [11] are not seamlessly integrated into a standard mixed-signal design flow, and consequently, do not look at the substrate noise coupling problem as a whole methodology. A substrate noise coupling tool may use the process parameters and a substrate model to extract and connect a substrate impedance network between some circuit nodes. However, from a circuit design point of view, there are more factors that have to be taken into account to be able to obtain accurate results. For example, if designers forgets to include the package models or the parasitic elements of badly designed ground traces to switching circuits, just because the tool does not provide such information, the simulation may be very inaccurate. This can result in an underestimation of substrate noise in a design and make a tool unreliable.

Other common problems with some substrate noise coupling tools are setting them up correctly for different processes or using them to optimize a layout during early stages of design. A setup often requires complicated modifications of the extraction rules files, before a substrate network can be extracted and correctly back annotated to the netlist. Furthermore, most tools are designed for a final verification of substrate noise coupling, using the fully extracted layout. However, at that stage, it is often too late, too expensive, and too time consuming to make major changes in the layout or the design.

Silencer! addresses the above mentioned problems and allows substrate noise coupling analysis, simulation, and optimization during the entire design and layout process. Depending on the stage of design and sizes of the circuit, fast scalable macro-models or a slower, but more accurate boundary element model (EPIC) can be used to calculate a substrate network. Scalable macro-models can tremendously increase the speed of substrate network extraction without much loss of accuracy. The integrated extractor EPIC, on the other hand, is meant to be used for smaller circuits or for validation. Switching between any of the different models is very simple (one mouse click) within the Silencer! framework.

1.5 Thesis Organization

The thesis is organized as follows: Chapter 2 discusses the sources of substrate noise and provides an overview of available substrate noise coupling tools. Chapter 3 describes the operation and tool capabilities and presents several substrate coupling models that have been incorporated into the tool. Chapter 4 presents a detailed example to illustrate the accuracy and efficiency of the tool. Using the example, it is shown how Silencer! can be used to perform substrate noise coupling analysis and optimization at different stages of the layout development. Chapter 5 compares substrate noise coupling measurements and Silencer! simulations between a noise injecting block and a noise sensitive block. Finally, Chapter 6 concludes the thesis and gives a perspective of future work.

2 SUBSTRATE NOISE COUPLING SIMULATION

For substrate coupling analysis, process and layout geometry information are needed. Process specific information, such as available process-layers (wells, buried layers, and other structures), are commonly encapsulated in a process technology file. Once the substrate port locations are known for noise coupling analysis, a substrate impedance network can be extracted and back annotated into the transistor level netlist. A SPICE like transistor level simulator can thereafter be used to perform a transient analysis and simulate the effect of substrate noise cross-talk between circuits. The output of the analysis may be the substrate noise at all the sensitive analog circuit locations in a chip. Designers can now determine the impact of substrate noise on circuit behavior and apply appropriate noise mitigation techniques.

The above approach to perform substrate noise coupling simulation and analysis has to be incorporated in a standard mixed-signal design flow. One general high-level design flow is illustrated in Figure 2.1.

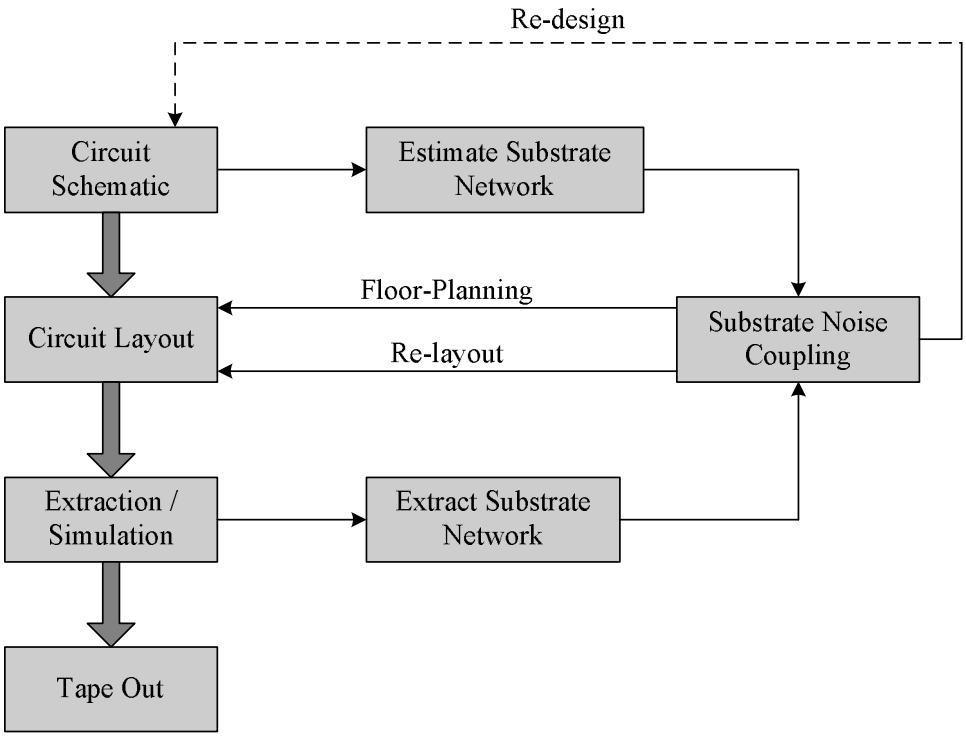

Figure 2.1 High-level design flow that incorporates substrate noise coupling simulation and analysis from circuit schematic level to tape out.

Once a circuit schematic of the designed circuit blocks is created, an approximate substrate network between the circuits can be estimated. The resistor values for the estimated network will not only depend on the size and separation of each individual block, but also on their locations on the chip. Large digital blocks may be represented as equivalent current sources with a certain noise signature and RMS value. Sensitive components of an analog circuit, on the contrary (e.g., transistors that have terminals connected to a high impedance node), may be parts of an analog block. Substrate noise coupling simulation and analysis at this stage of design - called the pre-layout design stage - helps during floor-planning. If too much substrate noise coupling from the digital blocks to the sensitive analog blocks occurs, a re-design may have to be considered, or noise isolation techniques may

have to be applied. One possible way of re-designing may be to relocate the I/O buffers in a digital circuit. If floor-planning is successful, routing interconnects from the blocks to the bond-pads can be estimated or extracted and connected. Furthermore, different package models can be used to experiment with how the package parasitic elements influence substrate noise coupling.

Once a more detailed layout of the circuits is available, the effect of substrate noise coupling can be simulated and analyzed more accurately. To include the parasitics, the layout and substrate network can be extracted and connected. If too much substrate noise coupling is observed at this stage, the layout of some blocks may have to be changed. Moreover, it may be necessary to even relocate an entire block, and redo the substrate coupling simulation and analysis.

When the entire chip layout is completed and the effects of substrate noise coupling are tolerable, a final verification using the entire extracted layout information can be performed. If the results are satisfactory, the chip is ready for tape out.

2.1 Sources of Substrate Noise

There are three main sources of substrate noise: impact ionization, switching noise, and supply noise [18]. Impact ionization is a phenomenon caused by high electric fields between the depleted part of the drain-channel and the substrate. Due to the high electric fields, electrons achieve high velocities and collide with silicon atoms in the substrate. Such collisions can create electron-hole pairs in the substrate and cause a substrate current. If the BSIM3 model is used at the transistor level, then impact ionization is automatically modeled.

Switching noise is caused by a large number of fluctuating digital nodes that are capacitively coupled to the substrate. When digital circuits switch, they inject current into the substrate via the source/drain junction capacitances. Owing to

the fact that the amount of injected current is directly proportional to the slew rate, dv/dt, of the switching voltage, substrate coupling increases with the speed of the circuit, i.e., i = C*dv/dt. Furthermore, the larger the transistors, the larger the junction capacitances, and thus, more noise can be coupled into the substrate. Figure 2.2 is a cross-Section of an NMOS transistor with its junction capacitances (left) and its symbol (right). It illustrates how switching noise is capacitively injected into the substrate. The junction capacitances are already included in the BSIM3 transistor models and not shown in the symbol.

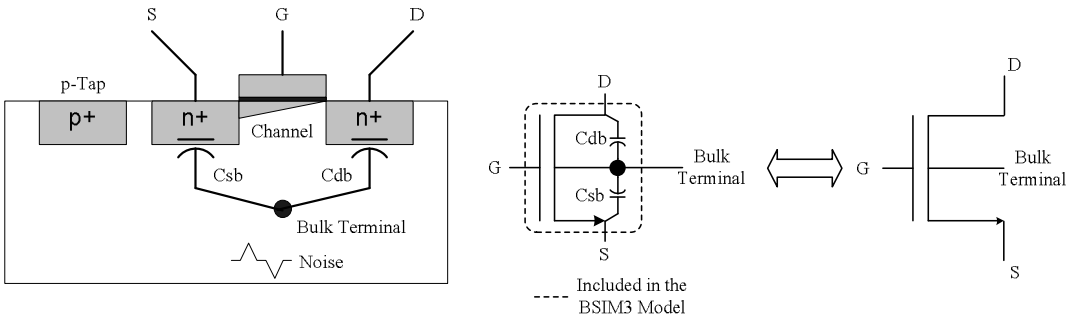

Figure 2.2 A cross-Section of an NMOS transistor (left) shows substrate noise coupling from the source and drain junction capacitances. The symbol (right) illustrates how the terminals of the NMOS transistor are represented.

The current injected will propagate through the substrate and affect other transistors that lie in its path through the body effect. The threshold voltage (V_t) of an NMOS transistor is given by,

$$V_t = V_{t0} + \gamma\left(\sqrt{2\Phi_F + V_{SB}} - \sqrt{2\Phi_F}\right) \tag{2.1}$$

where V_{SB} is the voltage between the source and the bulk terminals, V_{t0} is the zero bias threshold voltage, Φ_F is the Fermi level, and γ is the body effect parameter. As can be seen from this equation, a change of the voltage V_{SB} will also cause a change of the threshold voltage. Consequently, a fluctuation of the body potential due to a substrate current pulse will result in a change of V_{SB} and, therefore, also a change in V_t. The drain current of an NMOS transistor in both the linear and saturation regions is given by the following equations:

$$I_D(lin) = K_N' \frac{W}{L}\left\{(V_{GS} - V_t)V_{DS} - \frac{1}{2}V_{DS}^2\right\}\{1 + \lambda V_{DS}\} \tag{2.2}$$

$$I_D(sat) = \frac{K_N'}{2}\frac{W}{L}(V_{GS} - V_t)^2(1 + \lambda V_{DS}) \tag{2.3}$$

Because the drain current depends on the threshold voltage V_t, fluctuations in the transistor body potential directly lead to noisy drain currents.

The effect of non-ideal power supplies contributes significantly to the amount of substrate noise in an IC design. Package traces, bond-wires, and long interconnect routing associated with the substrate supplies have finite, and often large, inductances and resistances. Substrate current picked up by these supplies can cause glitches in the supply voltage. This phenomenon is also called inductive L*di/dt noise. Figure 2.3 shows a cross-Section of an NMOS transistor and its p+ substrate contact, also called p-tap (left) and its symbol (right). It illustrates how supply noise is injected directly into the substrate through the p+ contact. A transistor symbol does not include a terminal for the p-tap. In this thesis, if a p-tap terminal is necessary for illustration purposes, it will be represented as a circle between the bulk and source terminals.

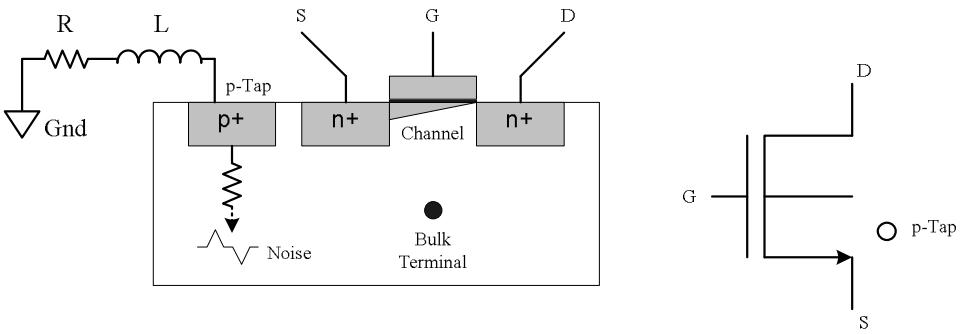

Figure 2.3 A cross-Section of an NMOS transistor (left) shows how supply noise (or inductive L*di/dt noise) is coupled into the substrate through a p-tap. The p-tap routing to the ground is non-ideal due to metal traces and package parasitics. The symbol (right) illustrates how the terminals of the NMOS transistor including the p-tap can be represented.

2.2 Lightly and Heavily Doped Substrates

Substrates are classified in two different types: lightly doped and heavily doped. Lightly doped substrates can be represented as two discrete layers with uniform doping concentrations and certain thicknesses. The bulk-layer of a lightly doped substrate has a high resistivity (e.g., 20-50 Ω-cm), consequently noise between substrate contacts propagates mostly on the surface. Separating circuits reduces crosstalk between noisy and sensitive circuitry. Therefore, the use of noise mitigation techniques, such as guard rings and trenches are very effective in reducing substrate noise coupling. However, the disadvantage of lightly doped substrates in mixed-signal ICs is latch-up problems.

Heavily doped substrates have been developed to prevent latch-up problems. They can be represented as three discrete layers with uniform doping

concentrations and certain thicknesses. The bulk-layer of a heavily doped substrate has a low resistivity (e.g., 1 mΩ-cm). Between the surface- and the bulk-layer, there is a thin, high resistivity epi-layer (e.g., resistivity = 10-15 Ω-cm). Due to the low resistance bulk- and a thin epi-layer, there is a low impedance path from the surface down to the backplane. For larger separations between substrate contacts, the cross-coupling impedance between the contacts is much higher than the impedance to the backplane. Substrate noise can therefore travel vertically to the backplane, propagate laterally through the low resistance bulk, and back up to the surface. Consequently, most of the substrate coupling between the circuits takes place through the heavily doped bulk-layer for larger separations (e.g., 30µm - 100µm). Beyond a certain separation, the amount of noise coupling becomes nearly independent of distance and consequently, placing the sensitive analog and noisy digital circuits farther apart from each other will not reduce the coupling further. Moreover, guard rings are not very effective in reducing the coupling, unless they are placed very close to the digital circuits. Noise can travel long distances and even affect circuits placed far away on the chip. However, grounding the backplane reduces noise significantly. Figure 2.4 shows the cross-Section of lightly and heavily doped substrates.

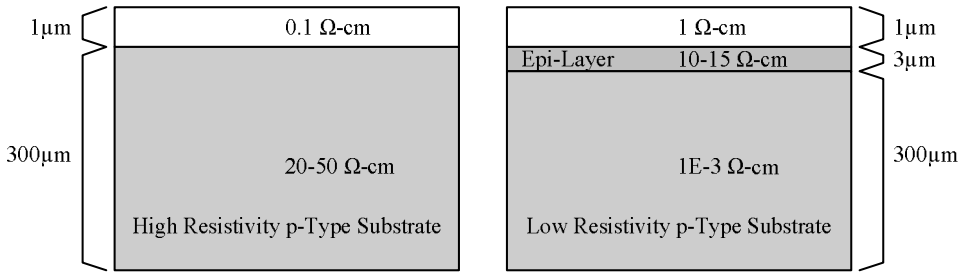

Figure 2.4 Cross-Section of a lightly doped two-layer (left) and a heavily doped three-layer substrate (right). In the lightly doped substrate, noise between substrate contacts will propagate on the surface and decreases if circuits are separated farther apart from each other. In the heavily doped substrate, beyond a certain separation between the circuits, most of the noise will propagate through the low resistivity substrate and cannot be further decreased by separating the circuits further apart.

2.3 Substate Noise Coupling Analysis Tools

This Section gives a brief overview of some of the commercially available substrate noise coupling simulation and analysis tools. The only tool we have evaluated is SCA. It is integrated in the CADENCE DFII framework version 4.4.6.

SCA [8] works with the Virtuoso layout editing tool and uses Assura™ interactive verification products (DIVA) to extract shapes. The substrate coupling model it generates is a resistor and capacitor network specifically designed for use by SPECTRE and SPECTRERF simulators. SCA also includes a command line interface and uses a core Poisson's equation based field solver together with a multilayer Green's function solver and matrix compression to extract a substrate impedance network. Insignificant matrix elements are discarded to reduce the netlist. SCA can be customized by adding different substrate port structures into the DIVA rules files. However, that requires detailed knowledge about DIVA programming and makes it difficult to set up for different processes. Further, SCA is not well integrated into a mixed-signal design flow, compared to other tools.

SubstrateStorm™ [9] is a suite of layout investigation tools developed by Simplex. It enables designers to take into account the substrate coupling effects of deep-submicron mixed-signal, analog, and RF designs. Furthermore, it helps engineers pinpoint crosstalk and parasitic coupling problems. SubstrateStorm supports complex structures, such as deep-well, triple-well, buried layers, and deep trenches for CMOS, BiCMOS, SiGe, and bipolar processes with lightly doped, epitaxial, or SOI bulks. It automatically generates a 3D mesh with vertical grid lines that fit the process doping profiles and a surface grid computed from the layout. An algorithm is used to reduce the model complexity from the original 3D mesh to a 2D information required to visualize the substrate noise distribution and to generate a SPICE-like netlist. If sufficient computation power is provided, the tool is capable of analyzing layouts with a resolution of up to one million surface nodes at the die level, which corresponds to approximately 50,000 devices. In

addition, it offers a 2D visualization of the noise distribution. SubstrateStorm™ is probably the most accurate tool on the market, but it is slow for a large number of substrate ports (a simulation can run for several days). As a result, it cannot be used to simulate very large circuits or entire chips.

SeismIC™ [10] is a substrate noise analyzer for mixed-signal designs. It extracts switching currents injected into the substrate from transistor bulks, interconnects, and power or ground connections. A boundary element method is used to calculate an impedance model Z(s) from the various switching sources to the analog components. The substrate is modeled as a 3D medium that consists of stacked layers of uniform doping, each characterized by a resistivity, dielectric constant, and thickness. Wells, buried layers, trenches, junction capacitances between well and substrate, and junction capacitances between buried layers and substrate are also modeled. SeismIC is well integrated into a mixed-signal design flow and lets IC designers perform substrate noise analysis on a full chip. For example, if no package parasitics and pin inductances are provided, it automatically assumes default values. It also gives the designer a list of recommended design changes to reduce noise at sensitive components. Furthermore, it supports substrate modeling for a variety of processes, including BiCMOS, twin well, triple well, SiGe, and standard CMOS. Noise voltage waveforms at the substrate of the noise sensitive components will be automatically displayed and noise sensitivity analysis capabilities are provided. For floor planning, the noise distribution is highlighted in the layout in the form of equipotential noise contours across the chip surface. SeismIC uses faster, but less accurate solvers and can be used to analyze large digital circuits and entire chips.

SPACE [12] is a practical layout-to-circuit extraction program developed at Delft University of Technology. It can be used to calculate a substrate resistance network between noise injecting and noise sensitive circuits. SPACE contains a 3D boundary element (BE) solver that can be used to calculate a substrate network for up to one hundred substrate ports. The solver uses the Laplace equation and then

transforms it with Green's theorem into a boundary integral equation. The BE solver is accurate, but becomes slow for large circuits and consumes large amounts of memory. Consequently, for large VLSI circuits, a less accurate, but faster interpolation technique has been implemented. In this method, direct coupling resistances are only computed between neighboring substrate ports. To determine whether or not two ports are neighbors, a Delaunay triangulation is used. The nodes of the Delaunay triangulation are the corners of the substrate ports. The values of the resistances are computed using interpolation formulas based on fitting parameters, area, perimeter of the substrate ports, and distance between the triangulation nodes. The output is a SPICE netlist that contains the substrate resistor network and other circuit components, such as interconnects. This netlist can then be used as an input file for a SPICE simulator. In addition to a batch-mode extraction, Xspace - an interactive graphical user interface to SPACE - is also available.

3 SILENCER! OPERATION AND CAPABILITIES

Silencer! is a simulation tool developed for substrate noise coupling analysis. It can be used to predict substrate noise coupling at three different stages of a mixed-signal chip design: pre-layout block level, layout level, and post-layout verification level, as illustrated in Figure 3.1. This thesis mainly focuses on the layout analysis level. This chapter describes how to use Silencer! to determine the effectiveness of layout changes, interconnect optimizations, package changes, and shielding techniques to reduce substrate noise coupling in a layout. A detailed example is given in Chapter 4 for demonstration. The pre-layout analysis and post-layout verification stages have not been fully integrated, and consequently are not described in great detail. However, with the current version of Silencer! it is still possible to analyze substrate noise coupling at these stages.

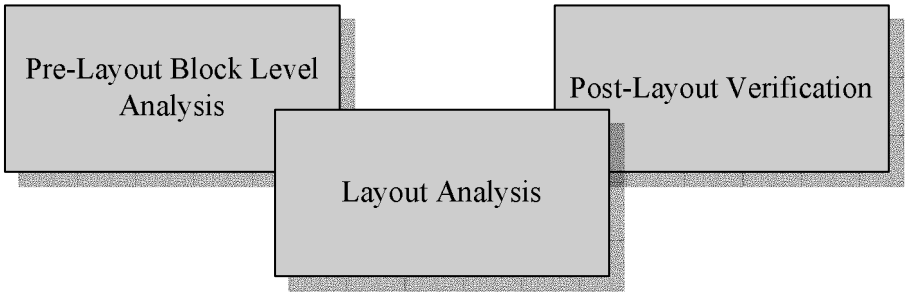

Figure 3.1 Substrate coupling simulation and analysis during the different stages of a mixed-signal IC design: pre-layout block level analysis, layout analysis, and post-layout verification.

3.1 Pre-Layout Block Level Analysis

The pre-layout block level analysis is useful for floor-planning, or to extract a substrate network between contacts for validating different substrate models. During floor-planning, Silencer! can be used to determine whether a circuit block will be adversely affected by substrate noise coupling problems. A block does not have to be fully designed yet. A layout of the most critical transistors and substrate contacts is usually accurate enough to get an approximate estimation of substrate noise coupling.

Large digital circuit blocks may be represented as equivalent current sources that inject noise into the substrate. This approach is essential for circuits that contain thousands of devices. A transistor level circuit simulator such as SPICE cannot handle a very large number of substrate ports connected to the circuit terminals. In such cases, the circuit netlist would become too large and simulation time would increase significantly.

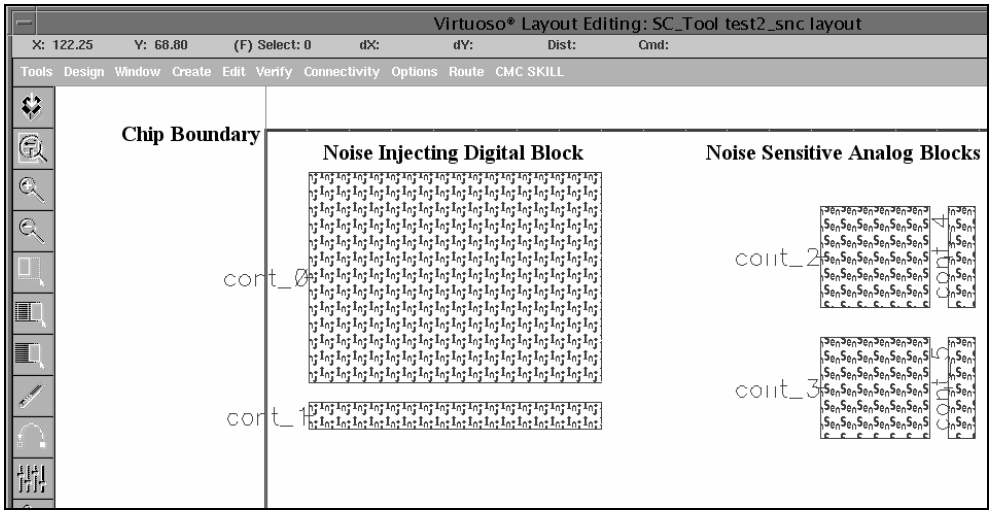

Figure 3.2 Example for pre-layout block-level analysis in Silencer! Critical Noise injecting and noise sensitive blocks can be drawn using 'SC_Inj' and 'SC_Sen' layers.

Figure 3.2 shows an example of a pre-layout block-level analysis. Circuits or critical parts inside a circuit can be represented as entire blocks. These blocks can be drawn in the same way any other layers are drawn in CADENCE. It is important to use the 'SC_Inj' and/or 'SC_Sen' layers, otherwise the blocks will not be recognized[1]. Silencer! has been designed to automatically distinguish between pre-layout block-level and layout-level analysis. Blocks can be labeled with 'SC_Inj' and 'SC_Sen' labels placed inside the blocks. The extractor in that case will use the label names as terminal connections of the substrate network. Otherwise, the blocks will be automatically labeled as 'cont_#'. After calculating the substrate network, a SPICE or SPECTRE netlist will be produced that can be back annotated and connected to power supplies, packages, and other circuit elements.

3.2 Post-Layout Verification

The post-layout verification level is the final and most accurate stage to analyze substrate noise coupling. In this stage, the entire chip layout is present, the layout has been extracted, and LVS has been passed. Figure 3.3 shows a possible flow for post-layout substrate noise coupling verification using the extracted layout that includes all the parasitic capacitances and interconnect inductances.

[1] Injector / sensor circuit blocks - in contrast to injector / sensor regions - do not have round corners. Injector / sensor regions are used to select noise injecting and noise sensitive circuitry in a layout. They are described in Section 3.4 and Chapter 4.

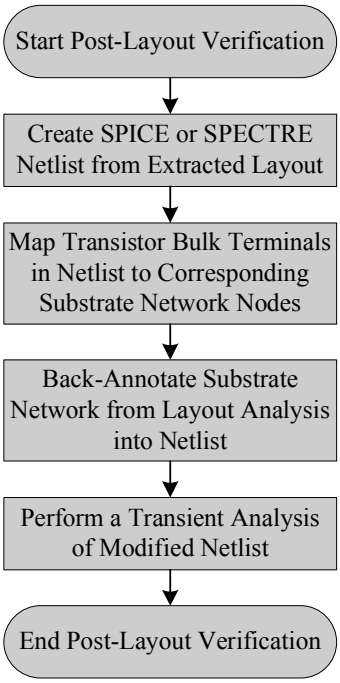

Figure 3.3 Flow for substrate noise coupling verification in Silencer! using the extracted layout including all the parasitic capacitances and interconnect inductances.

First, a circuit netlist is created from the extracted layout. For instance, Figure 3.4 shows how the SPECTRE netlist is created using the 'Affirma Analog Circuit Design Environment'. In the netlist, the bulk terminal connections of the NMOS transistors will most likely be connected to ground and those of the PMOS transistors to vdd, respectively. The bulk connection of the terminals will depend on the definitions in the extraction rules file. In order to connect the substrate network, these bulk terminal connections have to be mapped to the corresponding substrate network nodes. Labeling the bulk terminals of the transistors in the netlist is the second step in the post-layout verification process.

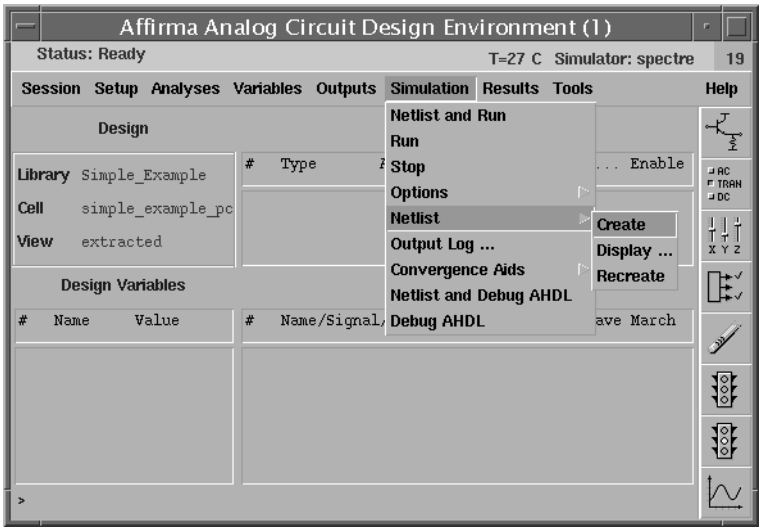

Figure 3.4 Using the 'Affirma Analog Circuit Design Environment' to create a SPECTRE netlist from the extracted layout.

Second, the substrate network created during the layout analysis can be back annotated into the netlist. The netlist can be used to perform a transient analysis and display the voltage waveforms at critical nodes of the sensitive circuits.

3.3 Silencer! Inputs and Outputs for Layout Level Analysis

To run Silencer! for layout level analysis, one needs to extract as much geometric information as there is available at the current stage of layout. For example, the only geometric information obtainable may be the physical locations of the NMOS transistors. However, some information about the routing from the p-taps to the power supply ground may also be available. In that case, this information would be another input to Silencer!. In other words, the more inputs

that are available - or the designer is able to estimate - the more precise the outputs created by Silencer! will become. More accurate outputs also mean more accurate prediction of switching and supply noise coupling into the substrate. Figure 3.5 illustrates the inputs and outputs that are incorporated in the Silencer! tool.

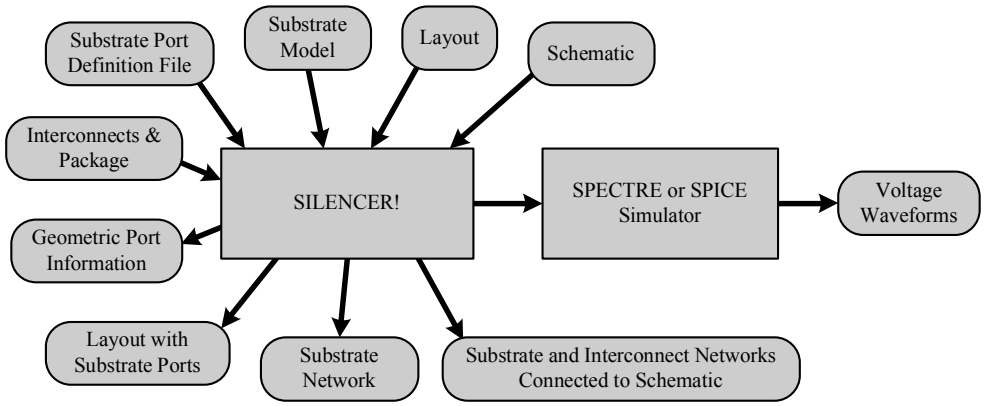

Figure 3.5 Inputs and outputs that are available in Silencer!

3.3.1 Schematic and Layout

The first two inputs to Silencer! are a completed and verified schematic of the circuits and a layout - or at least some key parts of it. The bulk terminals of the NMOS transistors in the schematic have to be labeled and left floating because Silencer! uses them as connection terminals to connect the substrate network. There are no conventions about how to label the bulk terminals, but it is suggested to use a label that contains the transistor name and an extension for bulk, e.g., 'bk'. In this way, the terminal can be easily identified in the layout or netlist, if necessary. The same names will be used to label the corresponding substrate ports in the layout, underneath the NMOS active regions.

Silencer! operates within the CADENCE DFII database and is integrated into the Virtuoso layout editor. Therefore, the circuit layout has to be designed in Virtuoso. As mentioned before, the layout does not necessarily have to be completed to perform substrate noise coupling analysis. However, the minimum requirements are an approximate estimation of the transistor sizes and locations, e.g., an approximate layout of the p-cells. To be able to model supply noise and apply noise shielding techniques, substrate contacts (p-taps), and eventually some critical interconnects that connect the circuits to the power supply should be present. It should be noted that if the interconnects are neglected, the p-taps must not be left floating, but have to be connected to ground. This may cause an underestimation of the noise present in the substrate due to the fact that supply noise is neglected.

3.3.2 Substrate Models

The next Silencer! input is a precise substrate model. Modeling of the physical properties of the substrate is required in order to get an accurate prediction

of how noise propagates through the medium. Over the past years, many methods have been developed to model the physical properties of the substrate. These methods can be divided into three groups: finite difference methods [13], [14], boundary element (BE) methods [2], [3], and scalable macro-models [4], [5], [15]. A detailed description of the various methods can be obtained from the appropriate references. Silencer! operates with two different types of substrate model extraction methods to obtain a substrate model: a boundary element (BE) method (EPIC) [7], and a fast, scalable macro-modeling approach. For the macro-modeling approach, two different algorithms have been implemented.

EPIC is a Green's function based boundary element substrate parasitic extractor. In contrast to the other Green's function based BE extractors, such as SCA, EPIC assumes a finite die size specified by the user. Silencer! provides a special layer called 'SC_Chip' to define the die size. EPIC treats the substrate area formed by the devices and contacts as two or three layers of uniformly doped semiconductor material with a certain thickness. A correct doping concentration and thickness of each of these layers is essential to calculate the substrate network. These parameters can be obtained from the spreading resistance profile (SRP) of the process. The output of EPIC is a pi-shaped substrate impedance network, which is resistive for low frequencies. Like the network obtained from the macro-modeling approach, the EPIC network includes cross-coupling resistances between the contacts and resistances from the contacts to the backplane.

The macro-model described in [6] is based on z-parameters and is scalable with contact geometry (area, perimeter) and separations. A z-parameter matrix of the substrate network can be calculated using the following equations:

$$Z_{11} = \frac{1}{K_1 Area + K_2 Perimeter + K_3} \tag{3.1}$$

$$Z_{12} = \alpha e^{-\beta x} \tag{3.2}$$

where K_1, K_2, K_3, α and β are process dependent parameters and x is the distance between the contacts. From the z-parameter matrix, a pi-network can be obtained.

3.3.3 Substrate Port Definition File

A 'Substrate Port Definition File' is another important input of the Silencer! tool. It contains information necessary to locate the substrate ports in a specific technology. Substrate ports may be bulk terminals of NMOS transistors, p-taps routed to ground, capacitances of wells connected to n-taps and routed to Vdd, parasitic interconnect capacitances, and other technology specific structures. The Silencer! substrate port definitions are specified in the 'Silencer.ini' file between the key-words 'ports=' and 'end', as illustrated in Figure 3.6.

```
;--------------------------------------------------------------------
; Setup file for substrate coupling analysis tool version 1.4
;--------------------------------------------------------------------

;--------------------------------------------------------------------
; Substrate port definitions (all different port types are listed
; here. Remark: The FIRST line defines the p-substrate-taps )
; portName (max. 8 characters) layer1 OVERLAPS layer2 OUTSIDE layer3
;--------------------------------------------------------------------
ports=
        ptap_           active          pplus           nwell
        nblk_           active          nplus           nwell
end
.
.
.
```

Figure 3.6 Substrate port definitions specified in 'Silencer.ini' file. The illustrated example is set up for the two dominating substrate ports present in the TSMC 0.35µm process.

Due to the layout topology and the low frequency of operation of the fabricated test circuits, Silencer! has been set up to distinguish between two types of substrate ports in a TSMC 0.35µm heavily doped CMOS process. However, more substrate ports could easily be added by extending the port definitions.

The first line after the key-word 'ports=' defines the substrate p-taps, which are responsible for substrate noise injection from the power supply. The second line classifies the connections between the substrate network and the NMOS transistor bulk terminals. From these ports, substrate noise can be picked up by the sensitive circuits. These ports also account for switching noise that is injected from digital circuits. Specific information about the 'Silencer.ini' file is given in Appendix B.

3.3.4 Interconnects and Package

The last two inputs available in Silencer! are the interconnect and package information. Interconnects are thin, long metal traces that connect the substrate and well connections to the power supply. Moreover, they provide electrical connections between circuit blocks to distribute analog and digital signals. Like the package parasitics, interconnects help improve the accuracy of the substrate coupling analysis, if they are included. The current version of Silencer! does not model substrate current that gets capacitively injected into the substrate from switching interconnects or wells since these capacitances are small and do not play an important role for frequencies below a few gigahertz. However, metal traces between the circuits and the substrate supplies (or power grids for larger circuits) have finite and often large resistances and are non-ideal connections to the power supplies. Therefore, noise in the ground traces will directly get coupled into the substrate through the substrate taps. Silencer! allows defining of such interconnects as L-R-series networks, which are also automatically connected to the circuit schematic.

Similar to the interconnects, it is essential to include the parasitic inductances and capacitances of a package, pin, bond-pad, bond-wire, and metal traces, in order to obtain an accurate prediction of the substrate noise coupling. Once a working schematic has been developed, package information can be included in the schematic as the last input to Silencer! Packages are usually characterized by the manufacturer and can be defined in a CADENCE library.

3.3.5 Geometric Port Information

Once the circuits of interest have been selected in the chip layout, the substrate ports for the process can be located. The geometric port information is saved in two different files; one contains the coordinates, area, and perimeter of the ports, the other one is an input file for the EPIC substrate network extractor. Both files provide the geometric information necessary to extract a substrate network. This information may be used in the future for an iterative optimization. The geometric port information is the first output of Silencer!

3.3.6 Layout with Substrate Ports

The second output is a copy of the Virtuoso layout that contains all located substrate ports. During the process of locating the substrate ports in the layout, all ports that were found will automatically be marked with an injector or sensor layer, and labeled - unless they were manually labeled beforehand. An automatic recognition of different port types is possible with the help of the ‘Substrate Port Definition File’ described in Section 3.3.3 and the CADENCE DFII Database. The layout and the detected substrate ports will be presented in a new layout cell view, whereas the original cell view remains untouched.

3.3.7 Substrate Network

The third Silencer! output is the substrate network that includes cross-coupling resistances between the contacts and resistances from the contact to the backplane. The substrate network can be extracted after the substrate ports have been located. It is be stored as a SPECTRE or SPICE netlist. The node names in the network match the port names in the layout.

3.3.8 Networks Back Annotated to Schematic

Once a substrate network has been calculated and an interconnect network has been defined, both networks can automatically be connected to the corresponding terminals in a newly created schematic view. A schematic that includes the connected substrate and interconnect networks is the fourth Silencer! output. The original schematic view remains untouched.

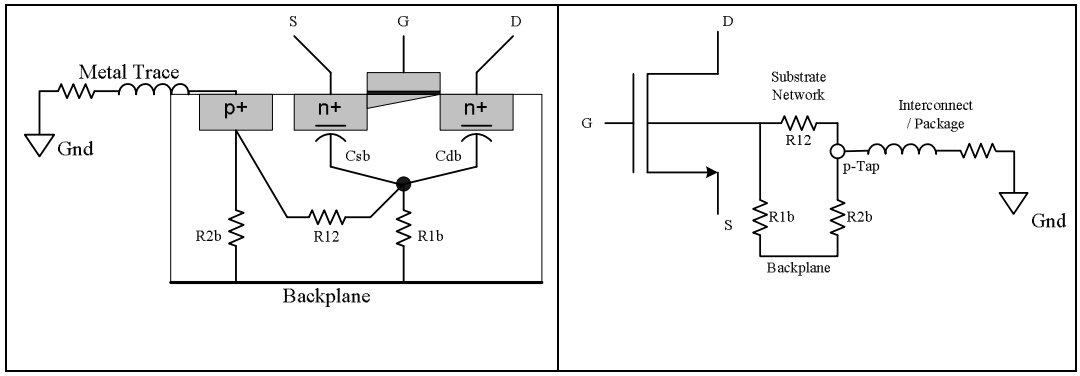

Figure 3.7 Connecting a resistive substrate network to a NMOS transistor. The junction capacitances Csb and Cdb are already included in the BSIM3 models.

Figure 3.7 illustrates how Silencer! connects the resistive substrate network between an NMOS transistor drain/source region and the p+ substrate contact (p-

tap) next to the transistor. In addition, the non-ideal p-tap to ground connection due to routing resistances and package inductances is shown.

It should be noted that the substrate models used in Silencer! were developed to calculate a resistive pi-network between multiple p+ contacts. However, the substrate port that connects the resistors R_{12} and R_{1b} to the transistor bulk terminal is actually a pn-junction. To be correct, it would therefore be necessary to extract a resistance between p+ and n+ contacts rather than between a p+ / p+ contact. Nevertheless, for the TSMC 0.35µm heavily doped process, Silencer! has been set up to use the entire region underneath an NMOS transistor as a substrate port to extract the substrate network. This simplification may cause some overestimation of substrate noise. However, it will be used until a p+ / n+ substrate model has been developed and implemented.

3.3.9 Voltage Waveforms

After computing and connecting the substrate and interconnect networks between the substrate ports, the impact of substrate noise, combined with the switching behavior of the digital circuits, can be analyzed. SPECTRE or any other transistor level simulator may be used to perform a transient analysis of the new schematic and display the voltage waveforms at the sensitive analog circuit nodes of interest.

3.4 Silencer! User Interface

The user interface of Silencer! is illustrated in Figure 3.8. A special layer is provided to define the chip boundary for Green's function based extractors. The menu item 'Define Chip Boundary' selects that layer so that the chip boundary can be drawn. If no boundary is defined, Silencer! will automatically suggest one.

Selection layers are available to select regions that inject noise in the chip and regions that are sensitive to it. This is useful in large designs if the designer wants to analyze only some specific circuits (or the most critical circuits) on the chip rather than the entire chip. Silencer! will only look for substrate ports within the boundary of the defined injector and sensor regions.

The menu item 'Add Networks into Schematic' can be used to connect an existing substrate and interconnect network to a schematic view. This is useful if changes in the schematic, but not in the layout have to be made. No changes in the layout means that the substrate and interconnect networks did not change and, therefore, do not have to be re-calculated.

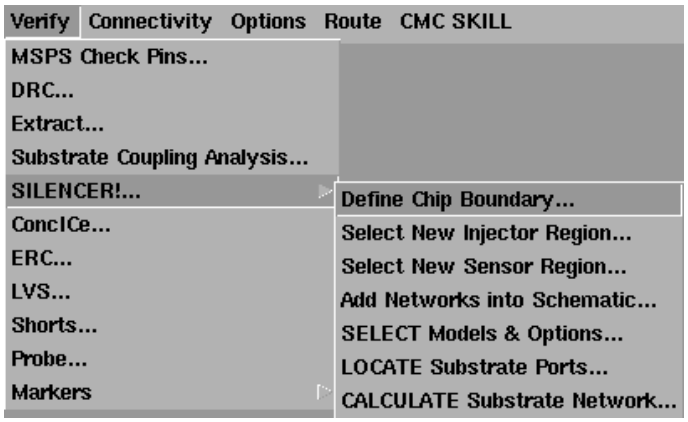

Figure 3.8 Silencer! user interface integrated into the CADENCE DFII Virtuoso layout editing tool.

A 'Models & Options' window, shown in Figure 3.9, will appear on the screen after the menu item 'SELECT Models & Options' was selected. It allows selecting between several options for substrate coupling simulation and analysis.

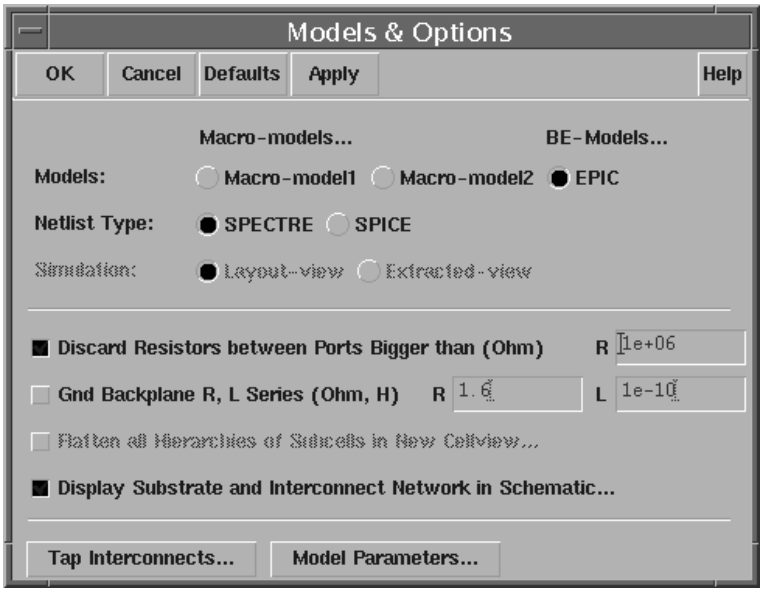

Figure 3.9 The Silencer! 'Models & Options' window.

The first option is the substrate model type. There are three different models available: Macro-model1, Macro-model2, and EPIC. The first two are z-parameter based scalable macro models [5], [6], [21] whereas EPIC is a Green's function based substrate parasitic extractor [7]. Figure 3.10 shows the model parameter window for EPIC parameters.

Once a model has been chosen, it is possible to select whether the resistor network will be generated in a SPICE or SPECTRE format. If there are very large resistor values present in the substrate network, they can be discarded to simplify the network. Further options are whether or not the backplane of the chip will be grounded, or the substrate resistor network should be displayed in the schematic. Finally, critical interconnect parasitics can be defined in the 'Models & Options' window by pressing the 'Tap Interconnects' button. Another window that contains a list of all the p-taps in the layout will pop up on the screen. The input mask allows entering all critical interconnects. If no interconnects are defined, it is necessary to

ground the p-taps. No p-tap should be left floating. However, before the list of p-taps is available, the substrate ports in the layout have to be detected.

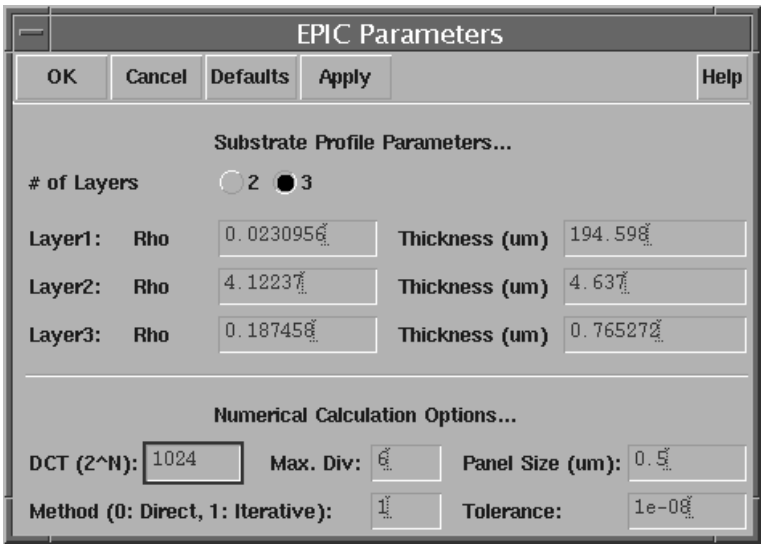

Figure 3.10 Model parameter window for EPIC parameters. This window contains the doping concentration and thickness of a two or three layer substrate and some options for the calculation.

The menu item 'LOCATE Substrate Ports' searches for all places on the chip within the defined injector/sensor regions, where substrate noise gets injected or picked up.

Finally, the menu item 'CALCULATE Substrate Network' reads the previously detected substrate port locations and extracts a substrate network connecting to these substrate ports. More information about locating the substrate ports, calculating the substrate network, and the user interface in general can be found in Chapter 4.

4 SUBSTRATE NOISE COUPLING ANALYSIS AND OPTIMIZATION AT DIFFERENT STAGES OF THE LAYOUT DEVELOPMENT

To demonstrate substrate noise coupling analysis and optimization, a simple example consisting of a substrate noise injecting inverter circuit and two substrate noise sensitive common source amplifiers with passive loads is described in this Section. First, the example will be designed and then, the layout will be improved step by step to reduce substrate noise coupling. During the different stages of design more detailed information, such as interconnects and package will be added.

4.1 Description and Simulation of a Simple Example

The example has been designed in the TSMC 0.35µm heavily doped process. All NMOS transistors are of size W/L =200µm/0.8µm with four fingers. The PMOS transistor is twice the size of the NMOS transistors, with 8 fingers.

The schematic for the example is shown in Figure 4.1. The bulk terminals of the transistors in the schematic are floating, but labeled with ‘M1bk’, ‘M2bk’, and ‘M3bk’, where ‘M#’ stands for the transistor name, and ‘bk’ for the bulk terminal. This labeling is used for the following reason: The bulk terminals as well as the substrate contacts are physical locations in the layout - called substrate ports. These are locations where noise can get coupled into the substrate or picked up by a sensitive circuit. In Silencer!, substrate ports can be manually labeled. However, usually the designer would only label substrate ports that are also visible in the schematic, such as the bulk terminals. Taps are not shown in the layout and, therefore, it makes sense to let Silencer! name them automatically. For a correct connection of the substrate network in the schematic or circuit netlist, the bulk terminal labels have to match with the labels given in the layout.

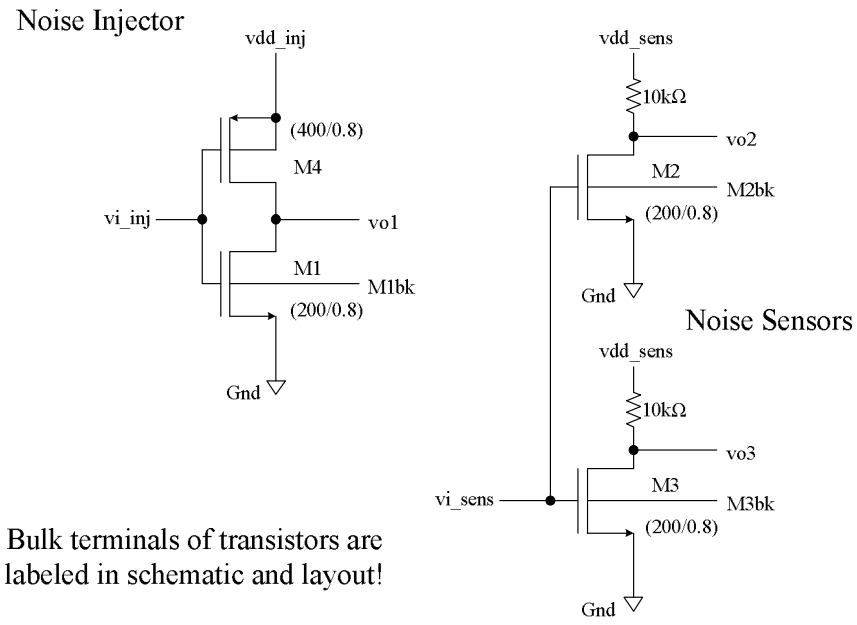

Figure 4.1 The schematic of the simple example consists of a digital inverter circuit and two common source amplifiers with a passive load. The bulk terminals are labeled and left floating.

Figure 4.2 shows the layout of the four transistor p-cells for the example. It should be noted that each of the NMOS transistors has a p-plus substrate contact (also called p-tap) placed at one of its sides. The PMOS transistor, on the contrary, is tied to the n-well with two n-plus contacts at each side. Furthermore, the common source amplifier transistors on the top are separated by $s = 10\mu m$ from the bottom inverter transistors.

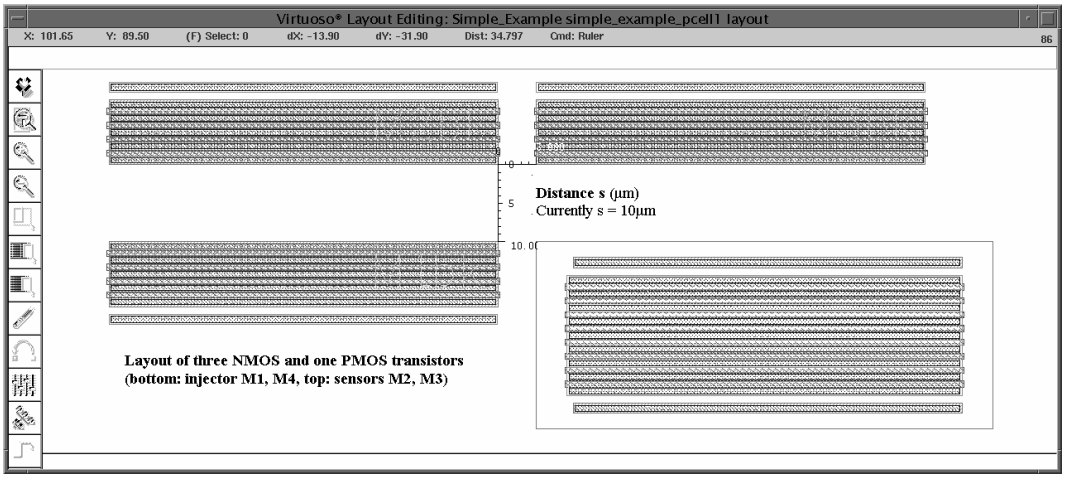

Figure 4.2 Layout of transistor p-cells and substrate taps for the simple example. The two top transistors belong to the common source amplifiers and the other two to the inverter.

As is typical for mixed-signal designs, the power supplies for the noise injecting circuits (vdd_inj) are kept separate from the supplies of the sensitive circuits (vdd_sens). The inverter input signal (vi_inj) is a 500kHz clock with rise/fall times of 1ns. Both the common source amplifiers are biased with a 620mV DC and have an input sine wave of 10mV/100kHz applied at their common input (vi_sen).

Once the transistors in the layout have been labeled with the same names as the bulk terminals in the schematic, a chip boundary can be defined (first Silencer! menu item). Then, circuits (or parts of a circuit) that are injecting substrate noise or are sensitive to it have to be selected by drawing an injector/sensor region layer across it (second and third Silencer! menu item).

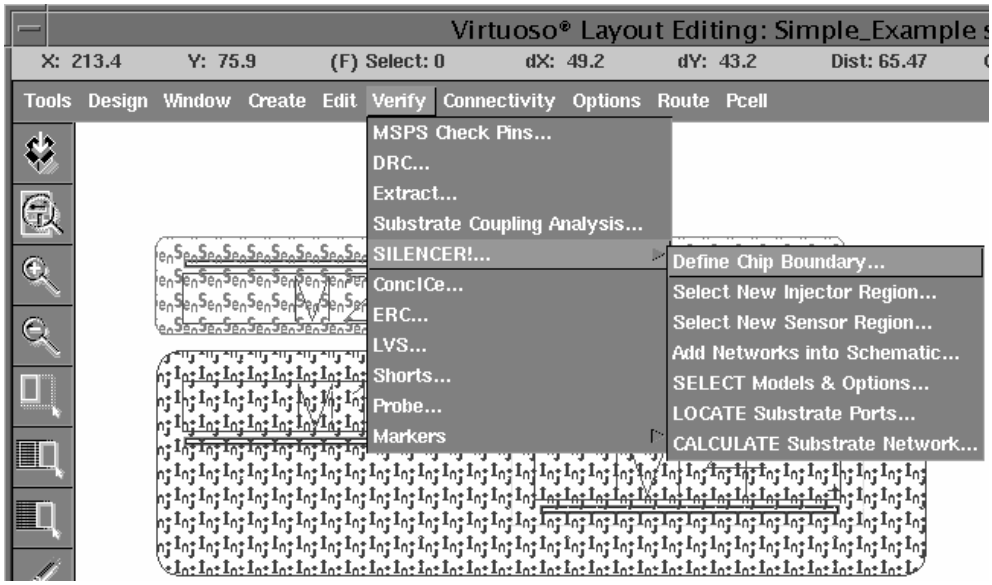

Figure 4.3 Define chip boundary and select injector/sensor regions for circuits that could be injecting substrate noise or be sensitive to it.

Figure 4.3 shows the injector and sensor region layers that were drawn on top of the inverter and amplifier transistors, respectively. After selecting the menu item 'LOCATE Substrate Ports…', the tool automatically detects p-tap and transistor bulk ports that are underneath a injector/sensor region layer. P-taps are defined in the first line of the 'Spectre.ini' file between the key-words 'ports=' and 'end'. More information about this file can be found in Appendix B. Furthermore, two files 'subsPort.txt' and 'EPICinput.dat' will be generated. The file 'subsPorts.txt' for the simple example is shown in Figure 4.4. It contains the coordinates, perimeters, and areas of the substrate ports and is used as the input file for the macro-models. The file 'EPICinput.dat' contains additional parameters, besides the geometric information of the substrate ports and is shown in Figure 4.5.

```
;----------------------Geometric-Port-Data-----------------------
;Name / lower left(x/y) / upper right(x/y) / Perimeter / Area
;-----------------------------------------------------------------
ptap_0     60.25      100.95      110.25      101.75      101.6      40
M1bk       60.25      103.25      110.25      111.15      115.8      395
ptap_2     60.25      141.1       110.25      141.9       101.6      40
ptap_3     115.8      141.1       165.8       141.9       101.6      40
M2bk       60.25      131.7       110.25      139.6       115.8      395
M3bk       115.8      131.7       165.8       139.6       115.8      395
```

Figure 4.4 The file 'subsPorts.txt' for the simple example contains the names, coordinates, perimeters, and areas of the substrate ports.

```
3      1000          1000
1      0.0230956     194.598
2      4.12237       4.637
3      0.187458      0.765272

4096   0.5           6
1      1e-08

6
0      0
0      60.25         100.95       110.25       101.75
1      60.25         103.25       110.25       111.15
2      60.25         141.1        110.25       141.9
3      115.8         141.1        165.8        141.9
4      60.25         131.7        110.25       139.6
5      115.8         131.7        165.8        139.6
6
1      1             1
2      1             2
3      1             3
4      1             4
5      1             5
6      1             6
0
```

Figure 4.5 The file EPICinput.dat for the simple example will be used as an input file for the substrate network extractor EPIC.

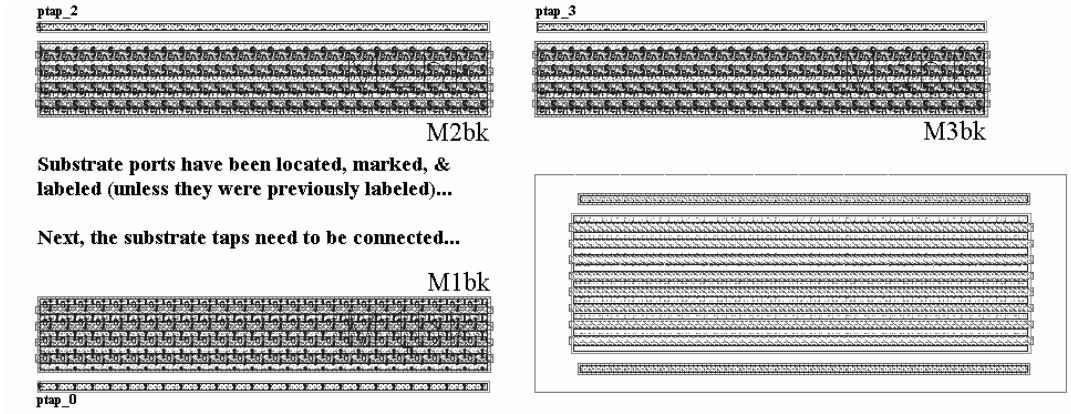

Figure 4.6 After detecting all the substrate ports underneath the injector/sensor region layers, the recognized ports will be marked, and labeled.

Figure 4.6 shows the layout after Silencer! has located the substrate ports. Since the noise injector switches below 1MHz, the network from the PMOS transistor and the n-well can be neglected. Consequently, no substrate ports for the PMOS transistors were defined in the file 'Silencer.ini' for this example.

In the next step, a pi-structure resistive substrate network to model the substrate properties has to be extracted. But before that, it is necessary to connect the p-taps to a specific potential, otherwise they would be floating. At the same time, several other options for the substrate network calculation can be selected.

After clicking on the menu item 'SELECT Models & Options…', the window shown in Figure 4.7 pops up. It allows entering the p-tap connections (and other interconnect of interest) and further options for the substrate network calculation. In this example, EPIC has been chosen to extract the substrate network. Moreover, options have been enabled to discard resistors greater than 1MΩ, leave the backplane floating, and finally, back annotate and connect the substrate and interconnect networks automatically to the schematic.

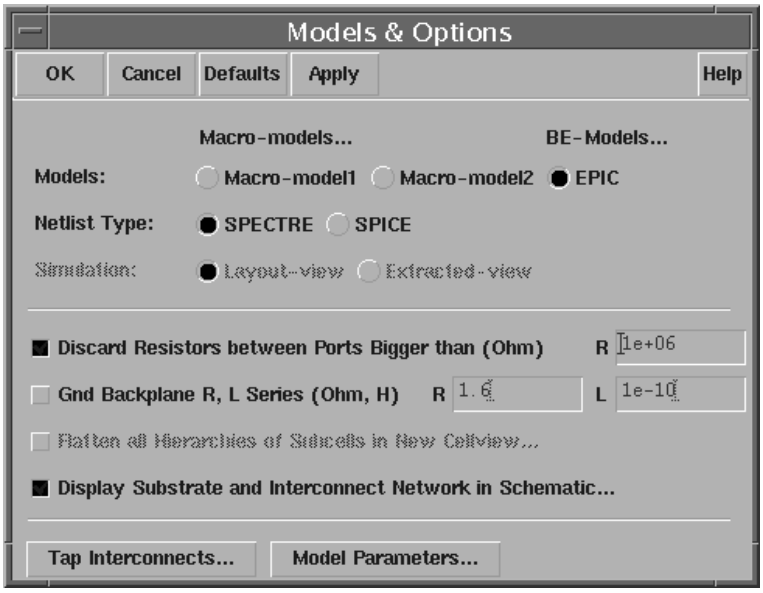

Figure 4.7 The 'Models & Options' window allows choosing between three substrate network extraction models. Further options as well as the p-tap connections can be defined at the same time.

To connect the p-taps, which Silencer! has labeled with 'ptap_#' in the layout (see Figure 4.6), the button 'Tap Interconnects…' has to be pressed. If done so, a new input mask to enter the p-taps will appear on the screen, as shown in Figure 4.8. This mask provides a list with all p-taps that Silencer! detected in the layout.

It should be evident that p-taps will usually be routed to a ground potential in a later stage of design. This can be done by connecting the p-taps to the source terminals of the NMOS transistors, or route them separately to a ground pin. Since there is no information about the routing available at this stage, p-taps can simply be connected to ground. However, it is important to keep in mind that noise injected through the supply is neglected in this case.

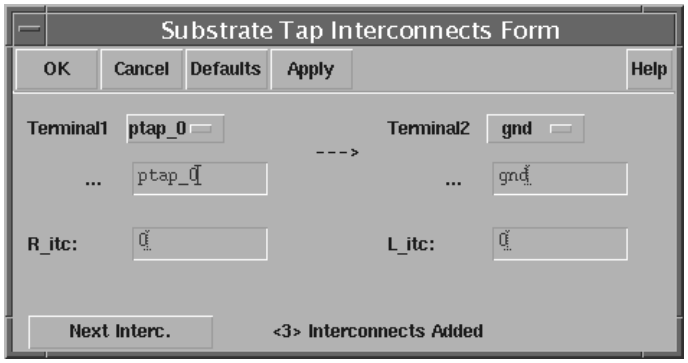

Figure 4.8 Input mask provided by Silencer! to define the connections of the substrate taps and the other interconnects.

Interconnects, such as the p-tap connections, are stored in the file 'ptapsItco.txt'. This file can be edited anytime by changing, deleting, or adding interconnects. For the example, all three p_taps have been connected to ground (R_itc = 0 and L_itc = 0 for ptap_0, ptap_1, and ptap_2). The command in the input masks will tell '<3> Interconnects Added'.

After all windows have been closed (after pressing the 'OK' button), the menu item 'CALCULATE Substrate Network…' can be selected. After a few seconds of calculation time (the calculation time depends on the substrate model and the number of substrate ports in the layout) the entire substrate and interconnect network will automatically be back annotated to the schematic view. The new schematic (name of original schematic and suffix '_snc') for the example is shown in Figure 4.9.

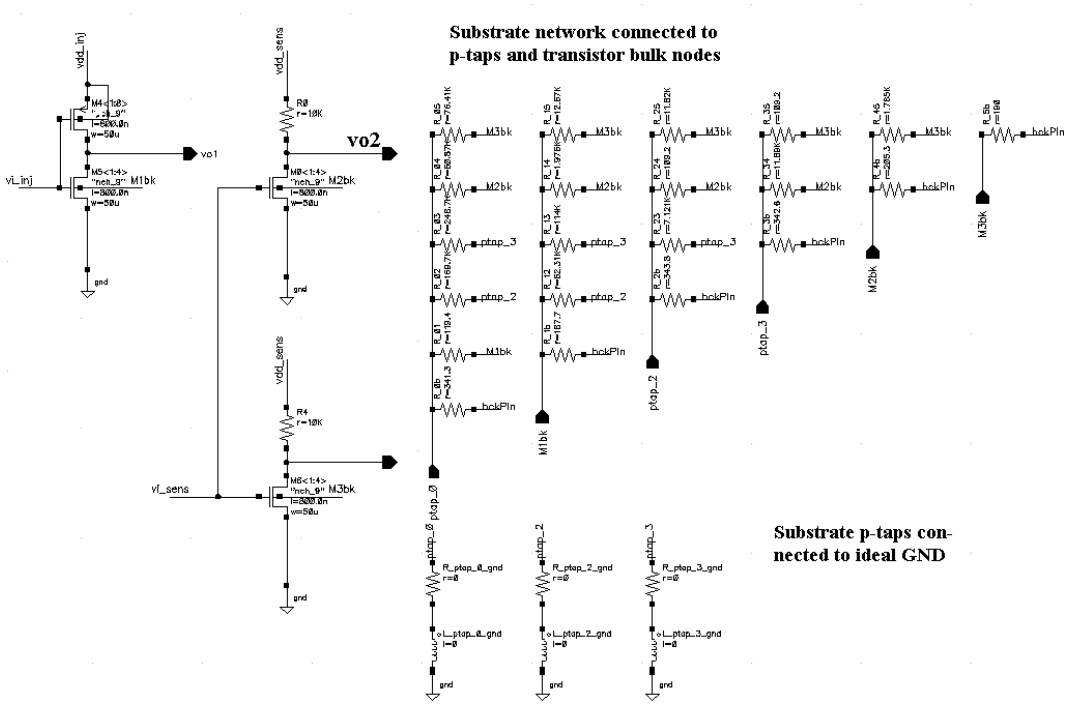

Figure 4.9 The substrate network and connection from the p-taps to ground have been back annotated to the new schematic view.

Furthermore, Silencer! has created the file ‘SCnetlist.txt’ that contains the substrate network in SPECTRE format in the ‘Silencer/Files’ folder. Due to the fact that EPIC was chosen to extract the substrate network, the logfile ‘epic_output’ has been created. In the case, where a network extraction would fail or encounter problems, an answer to the problem and possible solutions can be found in that file.

The newly created schematic is now ready for simulation and substrate coupling analysis. A transient analysis using SPECTRE can give the desired information about the substrate noise that gets coupled from the inverter circuit to the sensitive amplifiers. A simulation of the example using the ‘Affirma Analog Circuit Design Environment’ with appropriate models for the TSMC 0.35µm process is shown in Figure 4.10.

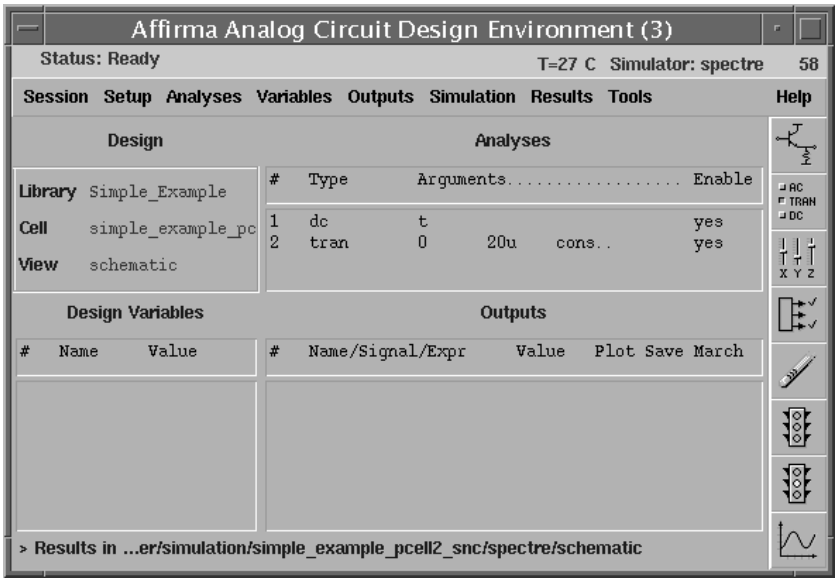

Figure 4.10 Substrate noise coupling analysis using SPECTRE from the 'Affirma Analog Circuit Design Environment'.

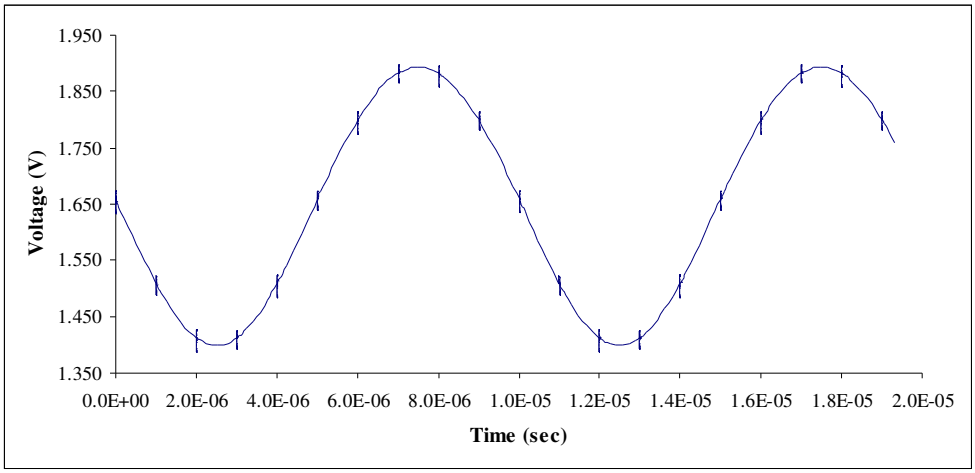

Figure 4.11 Common source amplifier output signal, vo2, degraded due to substrate noise.

Figure 4.11 shows the degraded, amplified output voltage observed at the amplifier output vo2. To minimize the substrate noise coupling crosstalk between these circuits, the layout has to be improved. The following Section explains how this can be achieved using the flow shown in Figure 4.12.

4.2 Optimization of Layout to Reduce Substrate Noise Coupling

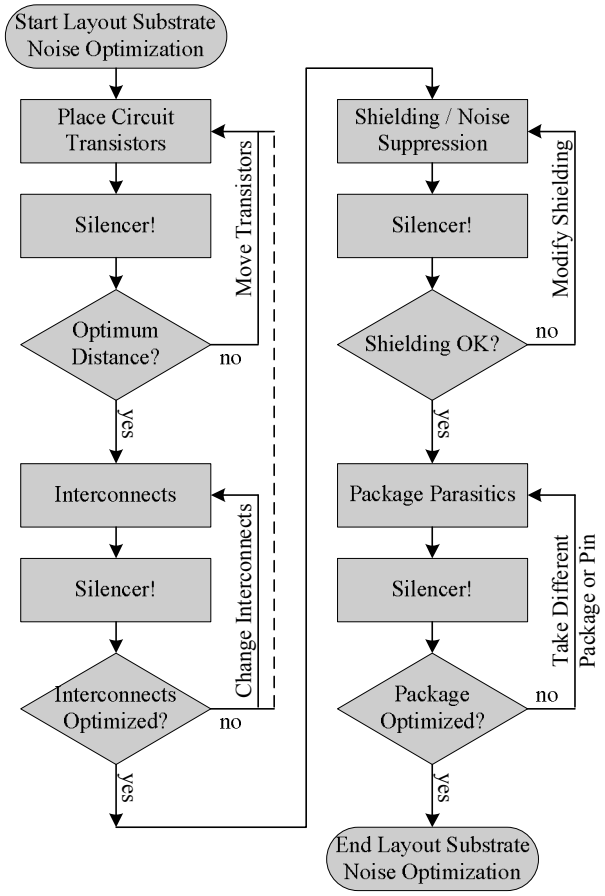

Figure 4.12 Flow for optimization of the layout to reduce the effects of substrate noise coupling at different stages of design.

Using the simple example described previously, the flow shown in Figure 4.12 is used to perform substrate noise coupling analysis and optimization at different stages of the layout.

First, an optimum spacing between the inverter and the amplifiers will be determined. Second, all interconnects between the ground and power supply lines will be added in the layout. This will cause ground bounce and increases the total amount of substrate noise injected. Third, the layout will be changed to minimize the resistor value in the metal traces to reduce ground bounce. Fourth, once the interconnect network has been added, additional grounded p-taps will be placed in the layout to suppress substrate noise cross-talk. Finally, a PGA132 package will be included which adds L*di/dt noise, due to the package line inductances.

4.2.1 Separating Noise Injecting and Noise Sensitive Circuits

This Section describes how the effect of substrate noise can be optimized by finding an optimum separation between the p-cells of the noise injector and noise sensor circuits. As previously shown in Figure 4.2, the switching inverter transistors are separated from the sensitive amplifiers by a separation *s*. To find the optimum separation *s*, we can vary *s* from 10µm to 100µm and recalculate the substrate network for each case[2]. It can be observed that with a larger separation, there is less noise coupling [19] due to the characteristics of a heavily doped substrate with floating backplane. Figure 4.13 illustrates that the maximum peak-to-peak noise at s = 10µm is 72mV. If the distance is *s* = 20µm, this value decreases to 48mV (curves are time-shifted for illustration purposes). Since the substrate is heavily doped and the backplane is floating, the noise will not decrease more beyond a certain separation. This is shown in Figure 4.14. To avoid wasting

[2] Finding the optimum spacing *s* between noise injecting and noise sensitive circuitry could be done in an iterative optimization loop.

valuable die area, an ideal separation between the circuits for our example is s = 30µm.

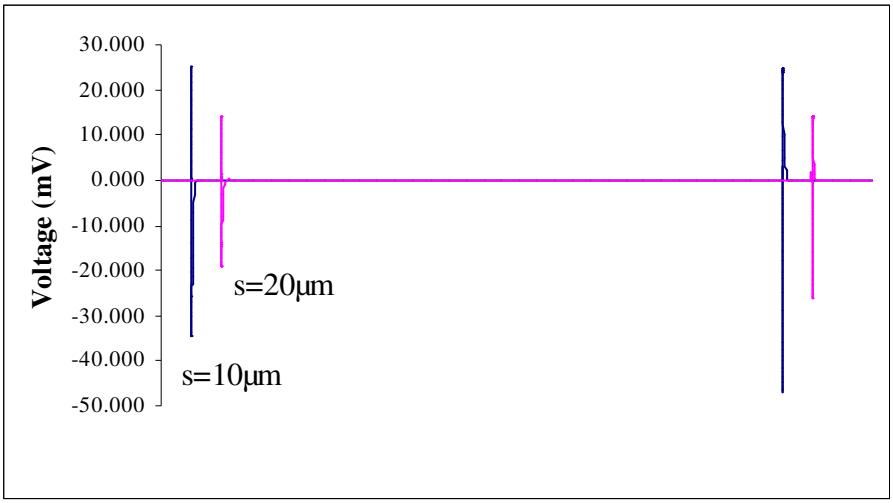

Figure 4.13 Substrate noise decreases by increasing the separation between the switching inverter and the amplifiers.

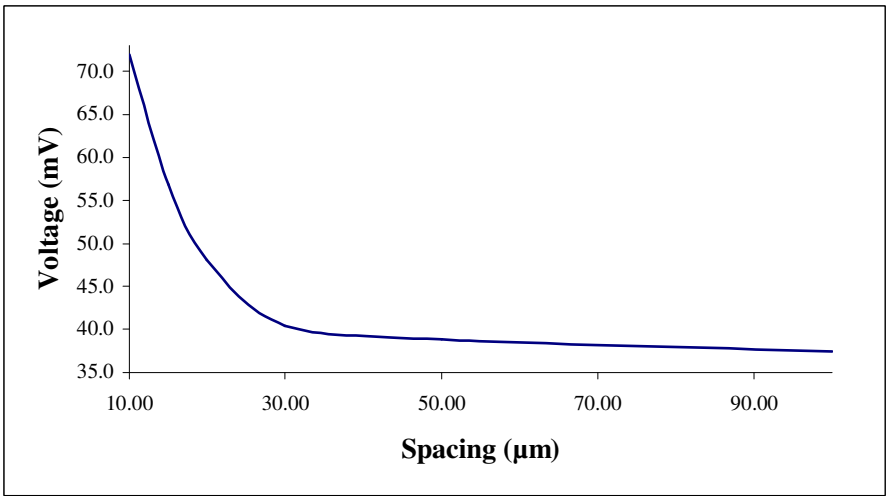

Figure 4.14 Substrate noise peak-to-peak voltage as a function of spacing s. After s = 30µm the noise does not decrease as the graph flattens out.

4.2.2 Reduction of Ground Bounce and Supply Noise

Substrate noise injected through the supply lines is significant and can not be neglected. Consequently, an estimated value for the routing impedance should be included in the simulation as the next step. To visualize this step, the routing of the p-taps to the source terminals and to the bond-pads has been included in the layout, as shown in Figure 4.15. The corresponding schematic is shown in Figure 4.16. The source terminals in the schematic are not connected to ground anymore, but left floating instead. They have been labeled with s1, s2, and s3 to connect the resistive interconnect network illustrated in Figure 4.17. The interconnect network connects a resistor between the source terminals and the ground/vdd pads. It has been defined in Silencer! using the input mask shown in Figure 4.8.

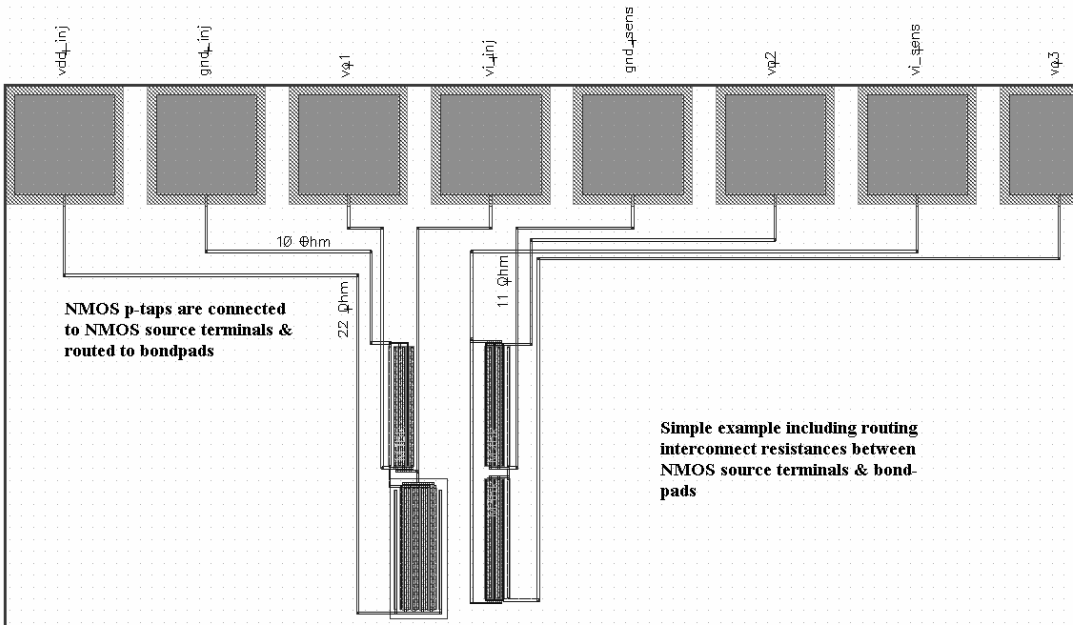

Figure 4.15 All transistor terminals are routed to the bond-pads. In particular, the p-taps are directly connected to the source terminals.

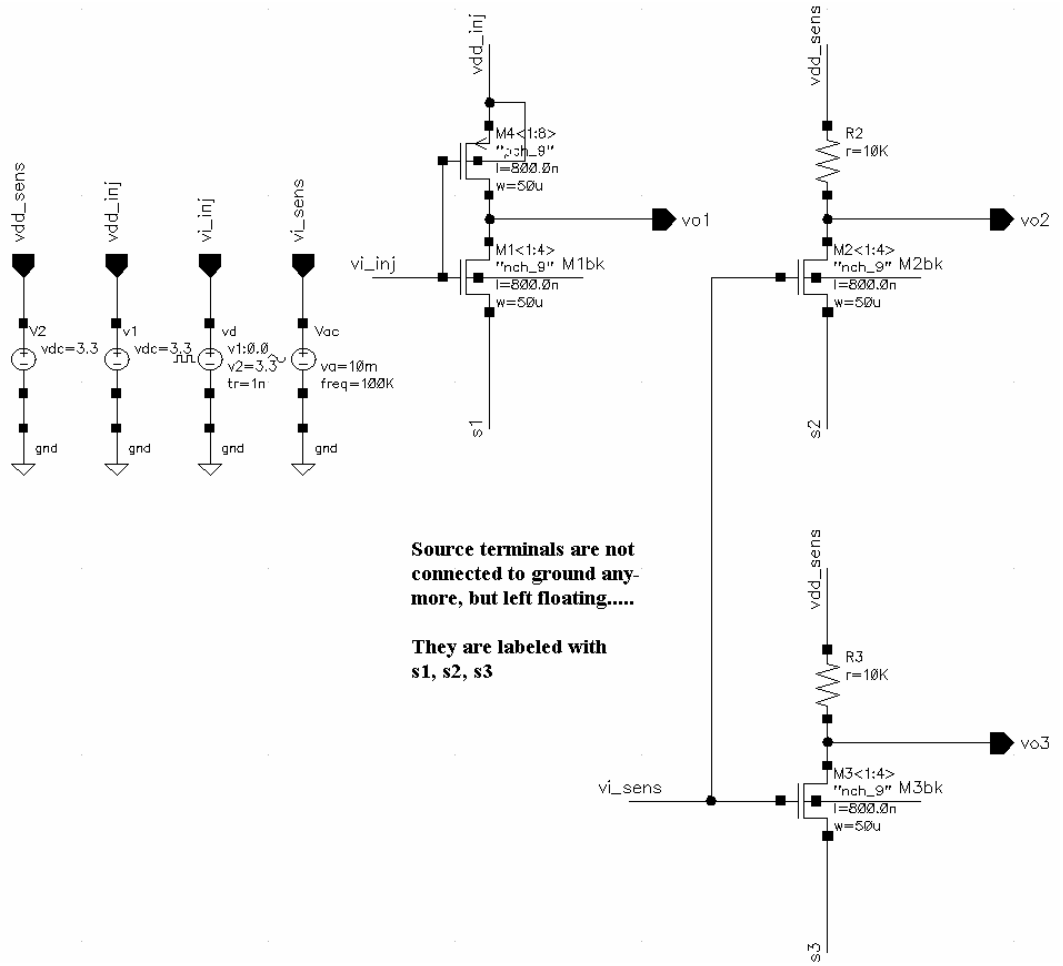

Figure 4.16 Schematic of the inverter and the amplifiers to which the substrate and interconnect networks will be back annotated.

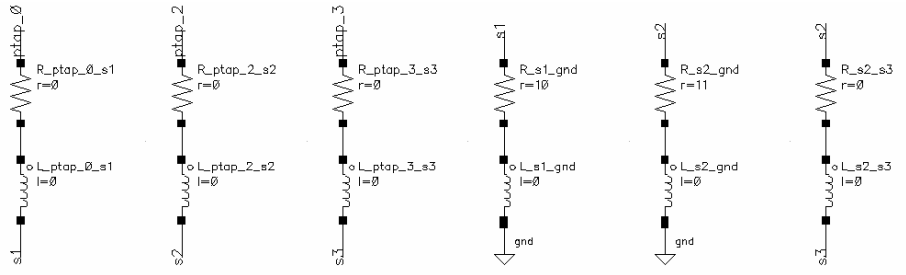

Figure 4.17 Interconnect network with the most critical routing resistor values in the ground and power supply lines. All inductances are zero (low frequencies) and the substrate p-taps (ptap_0, ptap_2, p_tap3) are routed ideally to the source terminals.

Due to the non-ideal ground connections, not only switching but also supply noise (ground bounce) can now be observed. In our example, the peak-to-peak value of the noise increases from 40.4 mV to 47.7mV.

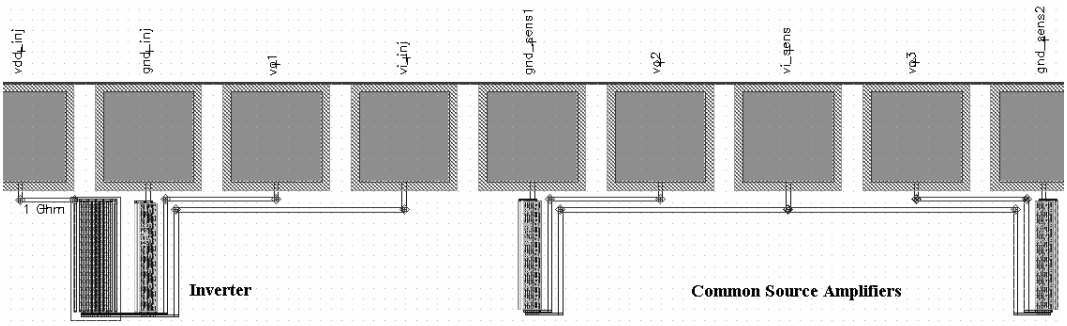

Figure 4.18 Improved layout with less interconnect routing to ground helps reduce substrate noise injected from the power supply.

To improve the routing from the source terminals and p-taps, the layout has been improved, as illustrated in Figure 4.18. As in the previous layout, the p-taps

are directly connected to the source terminals. However, the routing from the source terminals to the injector and the sensor ground pad is much shorter. Further, the separation between the inverter and the amplifiers is now greater than 170µm.

Analyzing the improved layout and comparing it with the previous one, the maximum peak-to-peak noise at the output vo2 has now decreased from 47.7mV to 42.5mV. Note that the noise in the ideal case (no supply noise) was 40.5mV.

4.2.3 Shielding Inverter from Amplifiers

The question now is how can the coupling from the inverter to the common source amplifiers be further decreased? Let us experiment by adding an additional substrate p-tap at the other side of the NMOS inverter transistor to shield the inverter better from the amplifiers, as illustrated in Figure 4.19. The p-tap absorbs noise from the substrate and deflects it away from the sensitive amplifiers. However, it should be noted that if the p-taps and the other noise mitigation techniques are used at the wrong place, they can aggravate the problem by bringing noisy supplies closer to sensitive devices!

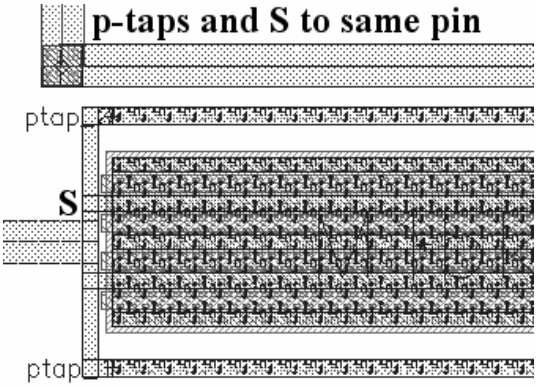

Figure 4.19 Additional p-tap substrate contact included next to the NMOS transistor of inverter. Both p-taps are routed to the source terminals and a ground pad.

If both p-taps are connected together and routed with a short interconnect to the injector ground bond-pad, there is a very low impedance path to ground which will absorb most of the noise. Simulations of the noise at the output vo2 of the amplifier show that the noise voltage decreases from the previous 42.5mV to 18.6mV, which is significant.

4.2.4 Separating Routing of p-Taps and Source Terminals

Another improvement may be made by isolating the p-taps of the noise injector from the source terminals, shown in Figure 4.20. However, this solution requires an additional bond-pad.

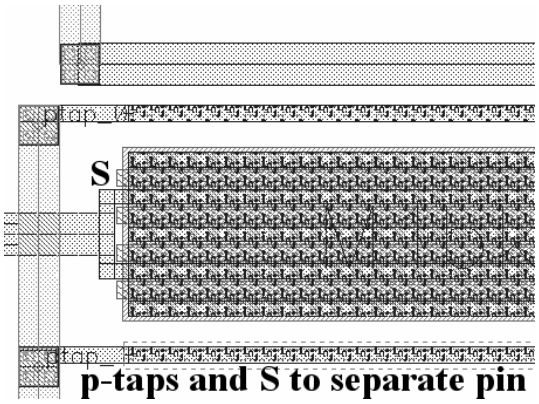

Figure 4.20 The two p-taps and the source terminal are routed to separate ground pads, as another noise mitigation technique.

It can be observed that the noise decreases only very little for this case; from 18.6mV to 17.9mV. Consequently, this solution is not recommended for the current layout since the improvement is only small, but requires an additional bond-pad. However, if this noise mitigation technique is used in a layout with long

ground and supply interconnects, like the layout in Figure 4.15, it may reduce ground bounce significantly.

Finally, including the package in the schematic, as shown in Figure 4.21, changes the noise signature again. The package acts as a low-pass filter and increases the substrate noise from 18.6mV to 34.1mV. The parasitic elements of the PGA132 package are shown in Figure 4.22. These elements vary from pin to pin. Therefore, choosing a pin with low metal trace and line inductance may help reducing supply noise injected to the substrate due to the package.

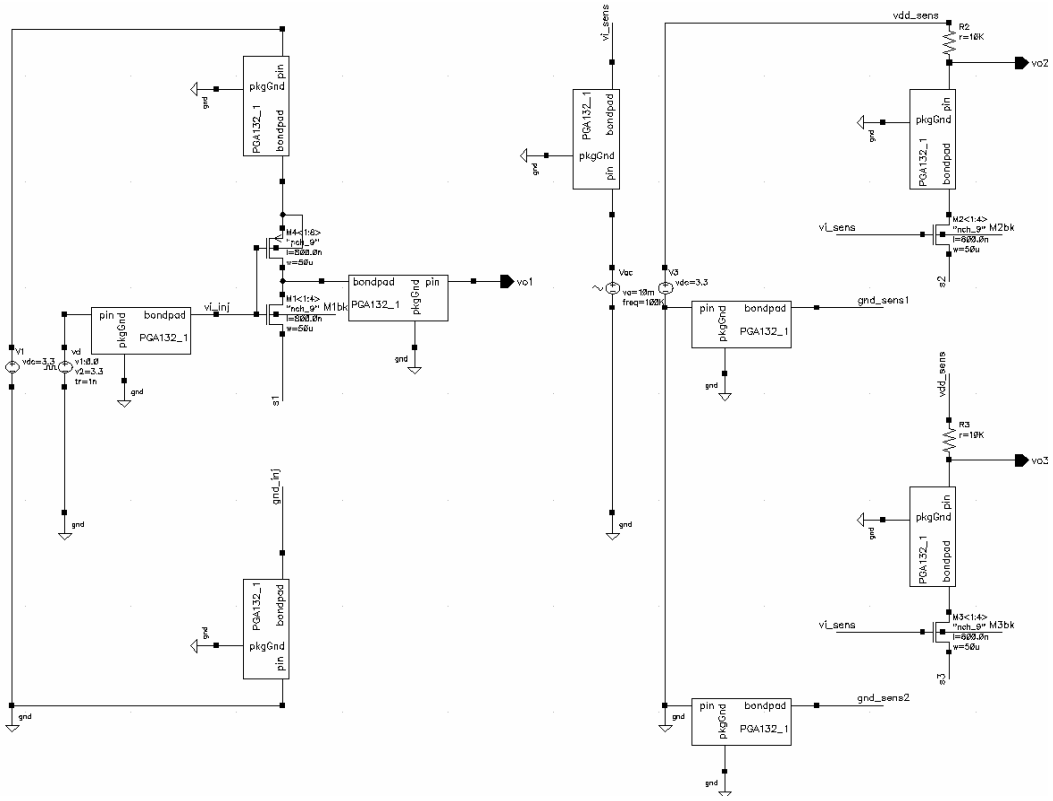

Figure 4.21 PGA132 package included in the schematic.

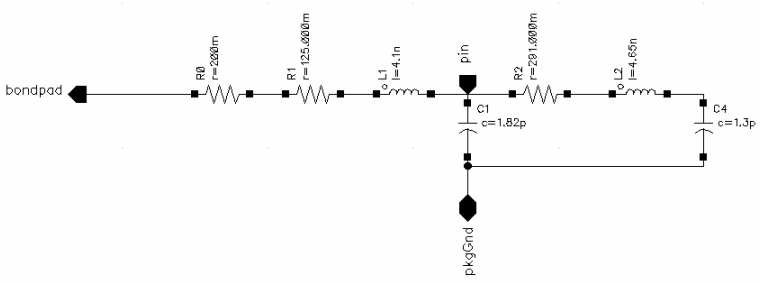

Figure 4.22 Schematic of PGA132 package parasitic elements between the package bond-pad and the package pin.

4.2.5 Summary of Layout Optimization

Table 4.1 and Figure 4.23 summarize the described methods to suppress substrate noise between the inverter and the amplifiers in the simple example. For better illustration, the switching events are time-shifted.

Table 4.1 Summary of optimization results for the simple example shown in Figure 4.23.

A	P-cell transistor layout with p-taps ideally grounded
B	Optimized distance between p-cells
C	Interconnects in ground and supply traces added
D	Optimized layout to reduce interconnects in ground and supply traces

E	Additional p-tap for injector added
F	P-taps and sources are separately routed to a pin
G	Including the package parasitics of PGA132 package

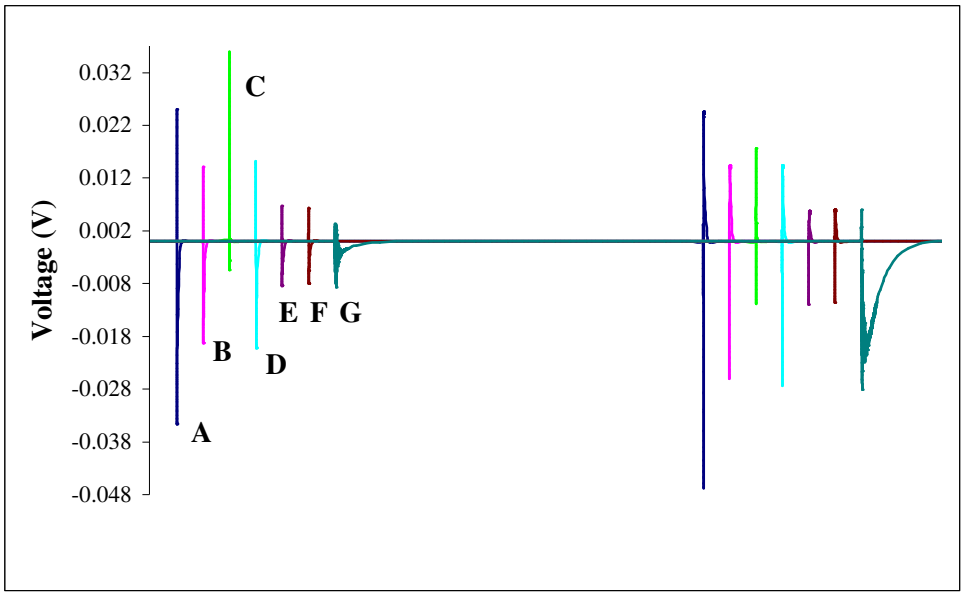

Figure 4.23 Summary of optimization results for the simple example. The substrate noise for the various conditions (A - G) is shown in Table 4.1.

Once the layout has been optimized, a final accurate substrate coupling simulation can be performed from the completed and extracted layout, as described in Section 3.2. If the results are satisfactory, the chip is ready for tape out. Otherwise, where necessary, more guard rings, trenches, separate power supplies can be added. In addition, the power and ground interconnects may have to be redesigned, or transistors may be moved to a different location. For heavily doped substrates in particular, substrate noise can be reduced significantly if the backplane of the chip is grounded.

5 SIMULATION AND MEASUREMENT RESULTS

A test chip fabricated in a TSMC 0.35µm heavily doped CMOS process was designed to test the capability of Silencer!. This chip, shown in Figure 5.2, contains several circuits that are injecting noise into the substrate (ring oscillators, stepped buffers) and others that are sensitive to it (operational amplifiers, sensing options, charge pump). This chapter describes simulation and measurements of substrate noise coupled from a stepped buffer to a folded cascode amplifier in unity-gain configuration, illustrated in Figure 5.1.

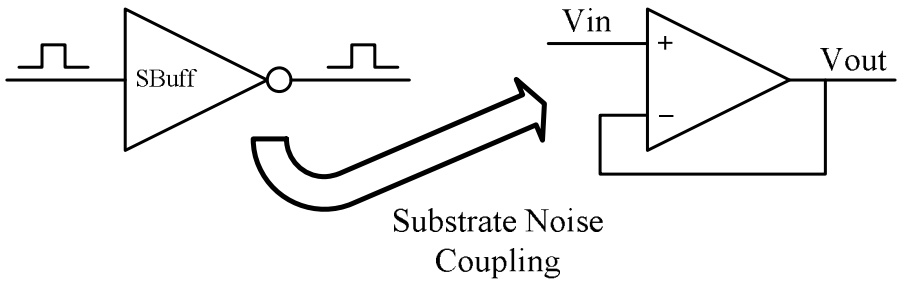

Figure 5.1 Noise coupling from a stepped buffer to a folded cascode amplifier in unity-gain configuration. The noise coupled from the digital block is measured at the output of the amplifier.

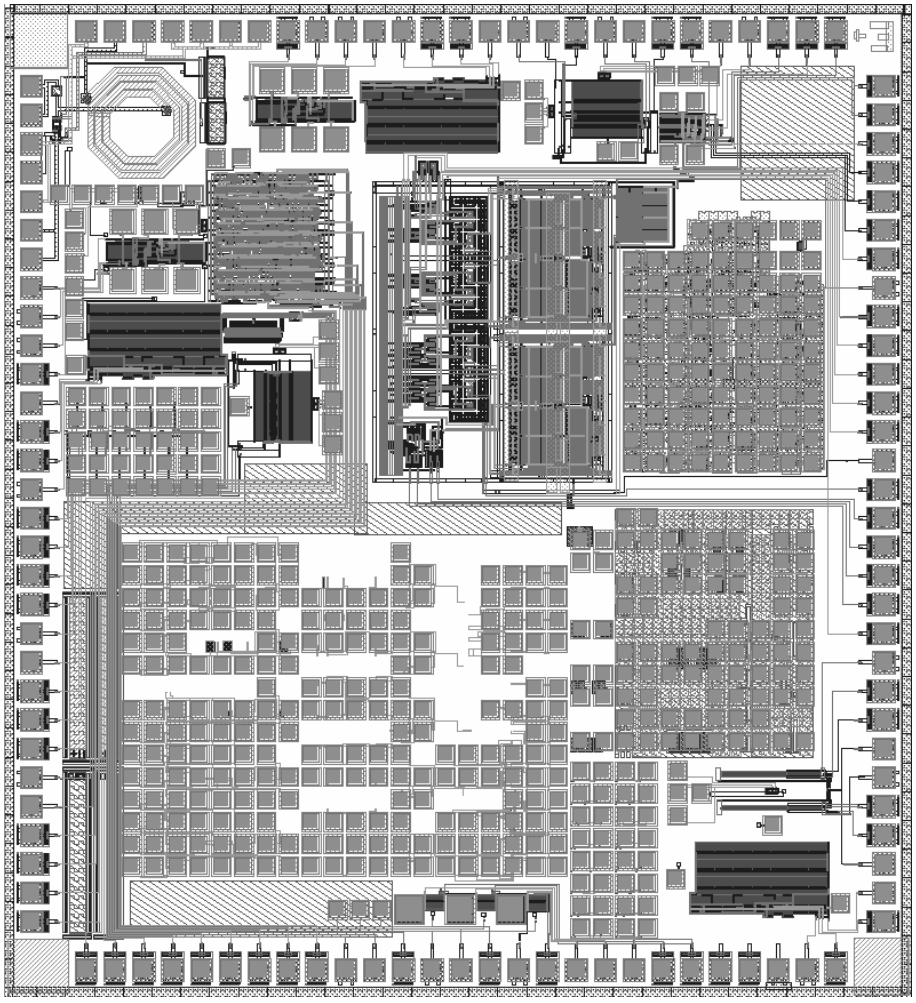

Figure 5.2 Layout of fabricated test chip in a TSMC 0.35µm process with epitaxial substrate.

A micrograph of two circuits on the test chip, a 7-stage stepped buffer with load and a folded cascode amplifier, is shown in Figure 5.3. Stepped buffers are used in many digital designs to provide buffering for clock and off-chip signals. For testing purposes, they serve as realistic digital noise generators. The amplifier represents a typical analog circuit that is sensitive to noise.

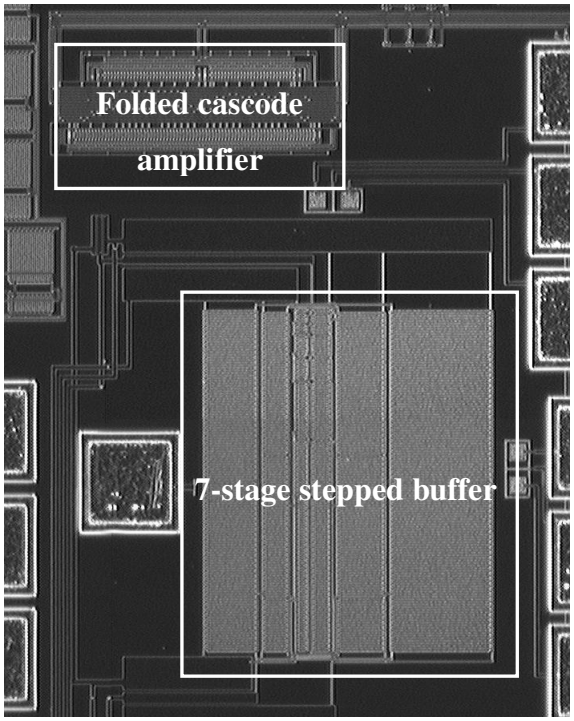

Figure 5.3 Micrograph of the portion of the test chip that was used for measuring substrate noise coupling from the stepped buffer to the opamp.

5.1 Measurement Setup

Figure 5.4 shows the setup for measuring substrate noise at the output of the folded cascode opamp in unity-gain configuration.

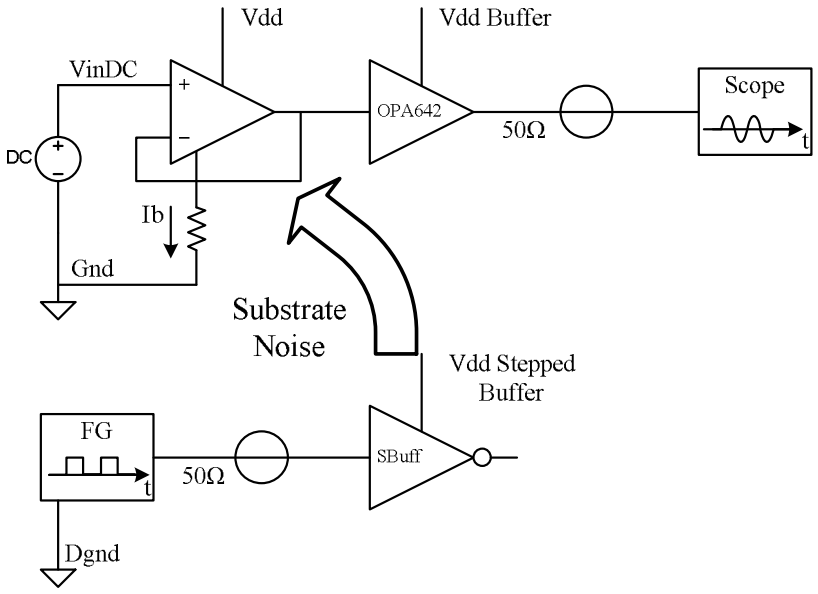

Figure 5.4 Setup for measuring substrate noise injected by the stepped buffer and picked up by the opamp (at the output of the opamp).

The stepped buffer is injecting substrate noise due to the switching behavior and the parasitics in the power supply lines (routing and package). The input signal for the stepped buffer is a 3Vpp/1MHz clock signal generated by a function generator. The stepped buffer has two ground pins, one of them is routed to the NMOS transistor source terminals, the other one to the p-taps. On the board, these two ground are shorted and connected to the same power supply. The setup for the stepped buffer is summarized in Table 5.1.

Table 5.1 Stepped buffer setup - supplies and equipment that was used for the measurements.

Input (50Ω termination)	Function Generator 0-3V / IMHz clock	HP 3325 B
Vdd = Svdd /gnd = Sgnd	3V / 0V	Agilent E361 A

The positive input terminal of the opamp is connected to a 1.5V DC voltage source (no sine wave is applied at the input). In addition, the output of the opamp is connected on the circuit board to a buffer with a high input impedance and 50Ω output impedance. The characteristics of the buffer are a very high bandwidth and a low distortion. Its output is directly connected to the oscilloscope with a 50Ω BNC cable and allows a low impedance and reflection-free connection between the output of the opamp and the oscilloscope. The setups for the opamp and the buffer are shown in Table 5.2 and Table 5.3, respectively. The configuration of the buffer is shown in Figure 5.5 (without 50Ω resistor between V1 and ground).

Table 5.2 Folded cascode opamp setup - supplies and equipment that was used for the measurements.

VinDC	1.5V dc	Agilent E361 A
output	The stepped buffer output is connected to input of buffer	
vdd/vss	3 V / 0V	Agilent E361 A
Rbias	100kΩ	

Table 5.3 Setup for OPA 642P that is used as 50Ω line driver.

output	The buffer output is connected to the oscilloscope (Tectronix TDS784D). Furthermore, a 3.3µF capacitor between the 50Ω output resistor and the output BNC connector is used to decouple DC (allows connecting the spectrum analyzer). The dc-decoupling does not affect frequencies greater than 100kHz.	
vdd/vss	6V / 0V	Agilent E361 A

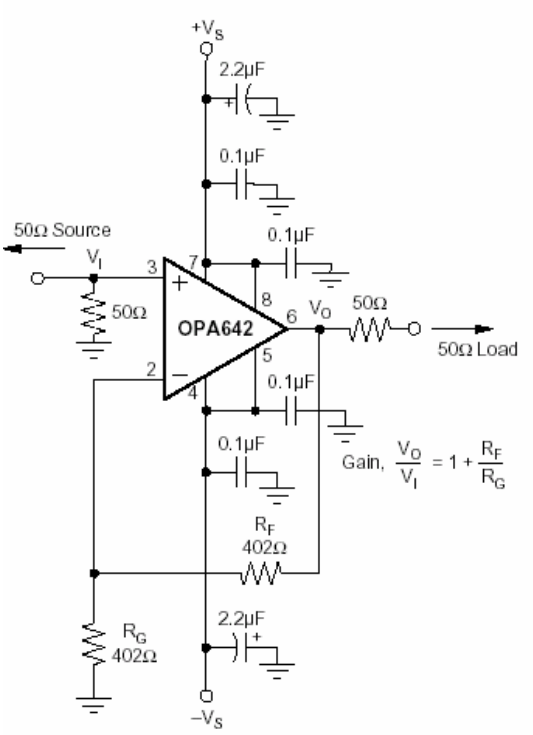

Figure 5.5 Wideband, low distortion, low gain opamp OPA642 used to connect a 50Ω BNC connector between the opamp output and the oscilloscope.

There are three separate power supplies for the stepped buffer, the opamp, and the buffer. All supplies are well decoupled with two parallel capacitors (electrolyte capacitor in parallel with a fast ceramic capacitor) placed right next to the supply/ground pins. To get exact results, it is essential to have good, quiet power supplies and prevent noise coupling into the chip from the circuit board.

5.2 Substrate Noise Coupling Simulation and Analysis

A complete schematic of the stepped buffer and common source amplifier in unity gain configuration is shown in Figure 5.6. The schematic contains all the necessary elements for substrate coupling analysis. Silencer! has been used to automatically extract a substrate resistor network from the circuit layout (only part of the network is visible). This network is back annotated in the schematic, together with the interconnect network and the package information. In addition, all the supplies and input signals are connected to the schematic.

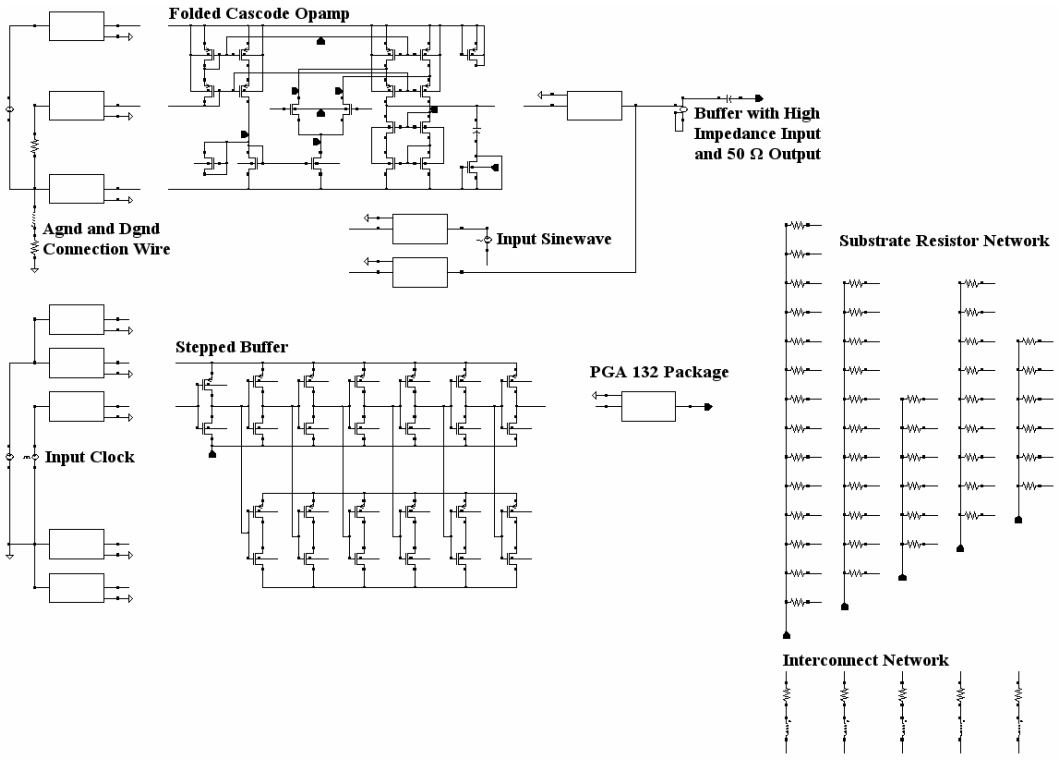

Figure 5.6 Circuit schematic for the stepped buffer and the folded-cascode amplifier including package parasitics, routing interconnects, and the substrate network.

After a SPECTRE simulation using the CADENCE 'Analog Environment', the transient voltage waveforms can be displayed. The simulated and measured transient output of the amplifier with the stepped buffer running at a clock frequency of 1MHz is shown in Figure 5.7. From these results it is clear that the simulation of the substrate noise coupling is in good agreement with measurements.

5.3 Simulations Compared with Measurements

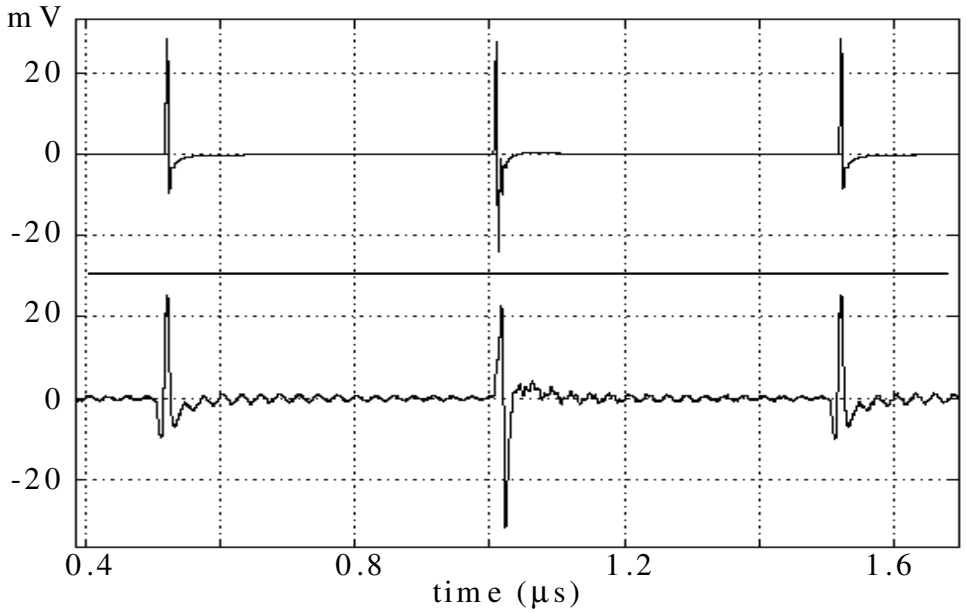

Figure 5.7 Simulation (top) and measurement (bottom) of the folded cascode amplifier output in unity-gain configuration when the stepped buffer is operating at 1MHz.

In heavily doped substrates, substrate noise coupling can be reduced if the backplane is grounded. The p+ diffusion ring (die-perimeter ring) placed around the perimeter of the chip, shown in Figure 5.2, allows grounding the backplane through a pin. It causes a low impedance connection to the backplane represented with the resistor Rdpr_ring in Figure 5.8. One terminal of this resistor is the backplane. The other terminal is connected to a package pin via bond-pad, bond-wire, bond-finger, and package metal trace.

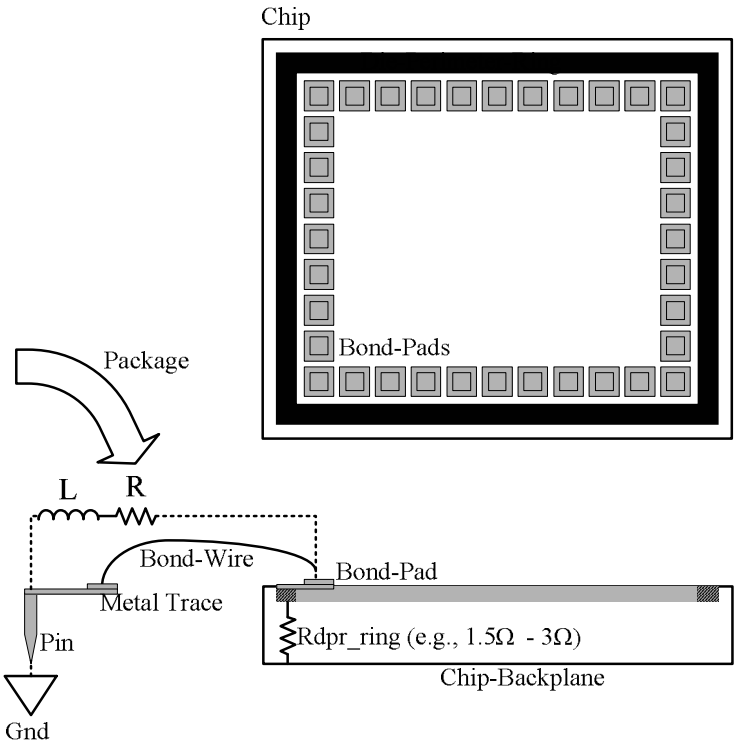

Figure 5.8 Top-view (top) and cross-Section (bottom) of a chip with die-perimeter ring. Due to the low impedance path from the die-perimeter ring to the backplane, it can be grounded through a pin.

Figure 5.9 compares the simulation and measurement results of the substrate noise coupling to the output of the opamp if the backplane is grounded through a pin. There is much less substrate noise present at the output of the amplifier.

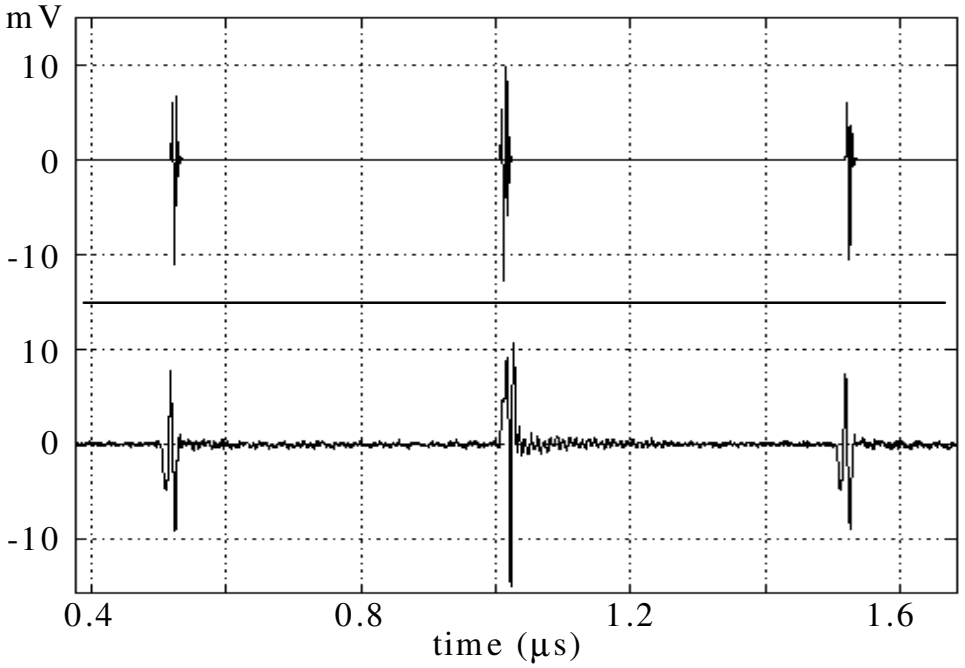

Figure 5.9 Simulation (top) and measurement (bottom) of the folded cascode amplifier output in unity-gain configuration when the stepped buffer is operating at 1MHz and the die perimeter ring is grounded through a pin.

5.4 Accuracy Versus Speed of Models

This Section compares the two substrate modeling approaches (macro-model and BE substrate network extractor) in accuracy and speed. In the fabricated example described in the Sections 5.2 and 5.3, supply noise was the dominant source of substrate noise. To predict the correct amount of substrate noise at the output of the opamp, it was essential to include not only the interconnect resistances of the ground metal traces but also the package parasitics.

To compare the substrate models, the same example with the stepped buffer and the opamp is used. However, only the substrate resistor networks are connected. The switching noise (1MHz frequency) injected by the stepped buffer NMOS transistors will propagate through the substrate and appear at the output of the opamp. Supply noise will be neglected by grounding all p-taps ideally (no interconnects) and removing the package information. In addition, the backplane is left floating.

Figure 5.10 shows the schematic for the simulation of the stepped buffer and the folded cascode amplifier in unity-gain configuration. Only the substrate resistor network is connected to the bulk terminals of the NMOS transistors.

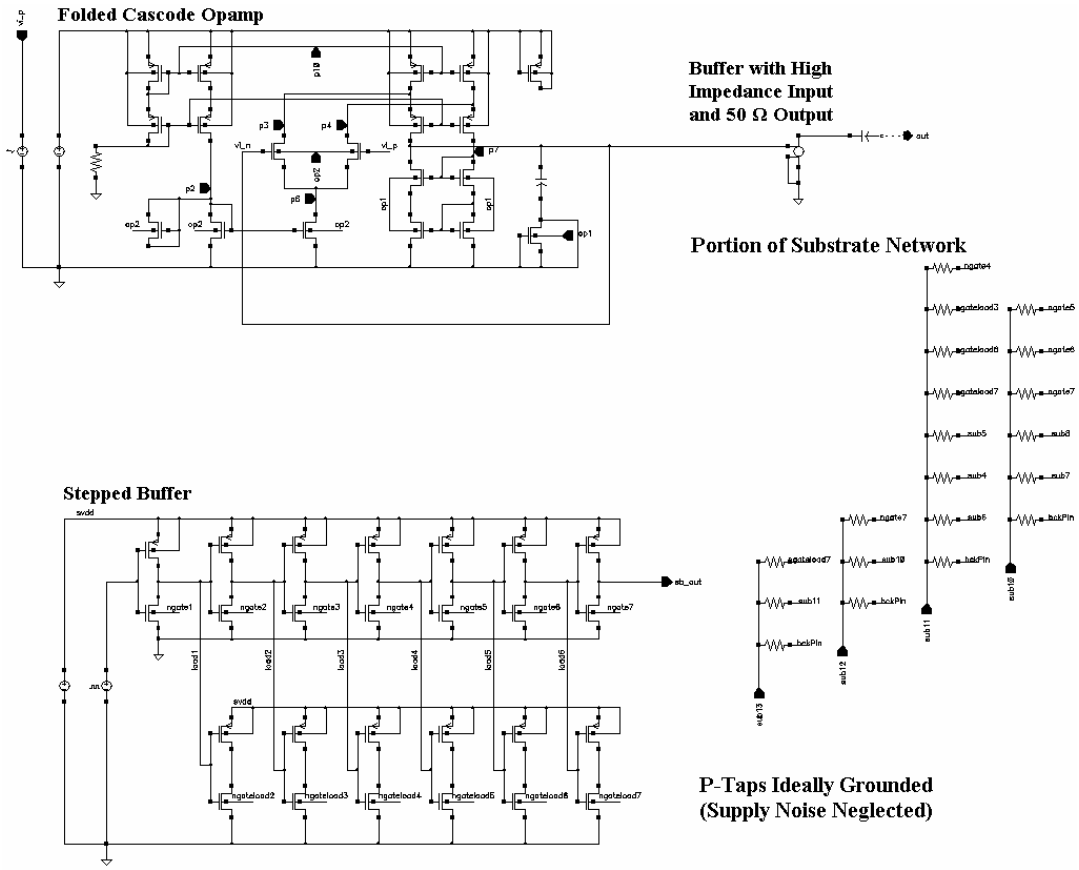

Figure 5.10 Circuit schematic for the stepped buffer and the folded cascode amplifier with only switching noise (no package, p-taps ideally grounded).

Figure 5.11 Shows the simulation results for both resistor networks, macro-model and EPIC. For this example that consists of 44 substrate ports, the difference between the noise predicted (peak-to-peak) is 29%. Table 5.4 summarizes the results (accuracy versus speed) of the two models.

If EPIC is the reference substrate network extractor, it is evident, that the error of the macro model is 29%. However, EPIC takes significantly more time to extract a substrate resistor network. As a result, the macro-model may be used for a fast, approximate prediction of substrate noise coupling during the initial stages of design. In these stages, the macro-model accurately predicts the trend of substrate

noise coupling and is useful for optimization. The BE-extractor EPIC, on the contrary may be used for a final, accurate verification.

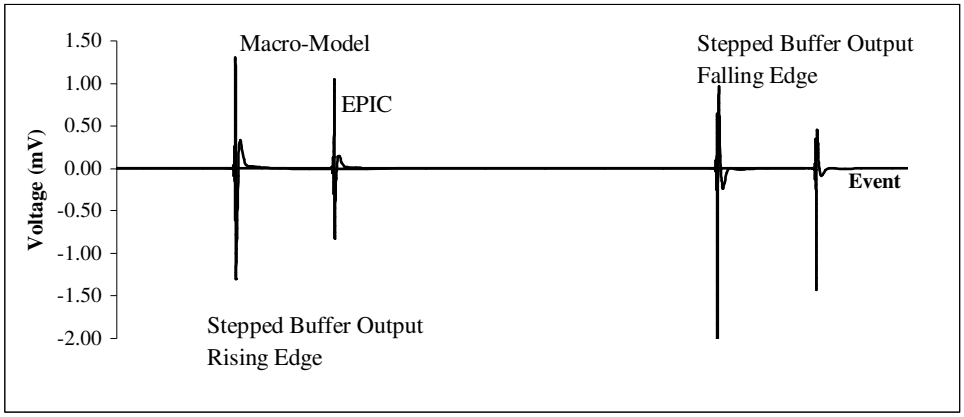

Figure 5.11 Simulation of the folded cascode amplifier output in unity-gain configuration when the stepped buffer is operating at 1MHz. Comparison between macro-model and EPIC. All substrate p-taps were grounded (no supply noise).

Table 5.4 Accuracy versus speed of macro-model and EPIC for the stepped buffer and folded cascode amplifier example. There are 44 substrate ports present in this example.

	Macro-Model	EPIC
ERROR (Vpp)	29%	-
Speed	2 seconds	174 seconds

6 CONCLUSION AND FUTURE WORK

Silencer! a new, schematic-driven tool for substrate coupling analysis has been described. It has been shown that Silencer! seamlessly fits into a standard mixed-signal design flow and is integrated into the CADENCE Design Framework II environment. The user interface naturally guides the designer by providing substrate noise coupling analysis and optimization during the different stages of design. Previous research relating to scalable macro-models, 3D-Green's function based substrate parasitic extractors, and package models are included in Silencer!

It has been shown for two examples, where Silencer! is used to change the placement of noisy or sensitive circuits, change the power distribution, and add additional noise isolation to have better shielding between noise injecting and sensitive circuits. The effect of substrate noise can be simulated accurately, so that it is possible to analyze how close a design is to failure and what eventually needs to be changed in an effort to solve a particular problem. As a result, it is no longer necessary to overdo noise mitigation techniques or waste precious chip area by separating circuits far apart from each other.

6.1 Extending Substrate and Interconnect Models

The Silencer! substrate models need to be extended to support more complex structures, such as triple-wells, buried layers, multiple well depths, and trenches. The methodology to extract a macro model from measured and simulated sets of test structures should be improved. For example, an optimal substrate characterization from process information should be fast and easy in order to develop further macro-models for CMOS, BiCMOS, SiGe, and bipolar processes

with lightly doped, epitaxial, or SOI bulk. Furthermore, for higher frequencies, lateral substrate coupling effects such as sidewalls of wells and resistive effects inside wells should be taken into account. In the current version of Silencer! only resistive substrate models have been implemented with the assumption that it is accurate for up to a few gigahertz. However, for higher frequency applications, such as those used in RF designs, this may not be the case anymore.

Silencer! models the interconnect resistances and inductances, but not the capacitances from the interconnects to the substrate. With decreasing feature sizes and increased frequencies of operation, the interconnect capacitances to the substrate can play a significant role. If fluctuating interconnects are present in the layout, designers may have to account for capacitively coupled substrate currents due to interconnects.

6.2 Switching Currents in Large Digital Circuit Blocks

A methodology of how to simplify large digital blocks and integrating them into the Silencer! framework has to be determined.

Several techniques for modeling currents injected by large digital circuits into the substrate have been developed [13]. One method is to simulate the entire digital block using a transistor level simulator such as SPICE. Switching currents on logic circuit nodes can then be probed and modeled as equivalent current sources. By monitoring the power supply current, a time window for the worst-case substrate current injection can be determined. Alternatively, an event-driven logic simulator can be used to simulate a digital block at the gate level. For a given input pattern to the logic circuit, such a simulator can record every transition in the circuit.

6.3 Pre-Layout Analysis and Post-Layout Verification

The Silencer! pre-layout analysis approach, such as floor-planning, has to be further developed. A pre-layout and floor-planning approach would be a tradeoff between accuracy and being able to simulate many more circuits on a large chip. Substrate noise coupling optimization and floor-planning may be started before the circuits are even laid out. For floor-planning, it is possible to experiment with various block placements. The placement of large noisy digital blocks and their effect on sensitive analog blocks and the optimum placement and isolation techniques can be identified.

The post-layout verification stage has not been fully automated yet. It is still necessary to either modify the extraction rules or re-label the bulk nodes in the netlist to connect the bulk terminals of the transistors to the corresponding terminal of the substrate network. Once layout versus schematic (LVS) completely matches, the LVS information about net and terminal connections may be used to automatically connect the substrate network to the extracted layout.

BIBLIOGRAPHY

[1] N. K. Verghese, T. J. Schmerbeck, and D. J. Allstot, *Simulation Techniques and Solutions for Mixed-Signal Coupling in Integrated Circuits*, Kluwer Academic Publishers, 1995.

[2] N. K. Verghese and D. J. Allstot, "Computer-aided design considerations for mixed-signal coupling in RF integrated circuits," *IEEE J. Solid-State Circuits*, vol. 33, pp. 314-323, March 1998.

[3] R. Gharpurey and R. G. Meyer, "Modeling and analysis of substrate coupling in integrated circuits," *IEEE J. Solid-State Circuits*, vol. 31, pp. 344-352, March 1996.

[4] A. J. van Genderen, N. P. van der Meijs and T. Smedes, "Fast computation of substrate resistances in large circuits," *Proc. European Design and Test Conf.*, pp. 560-565, Paris, France, March 1996.

[5] A. Samavedam, A. Sadate, K. Mayaram, and T. S. Fiez, "A scalable substrate noise coupling model for design of mixed-signal IC's," *IEEE J. Solid-State Circuits*, vol. 35, pp. 895-904, June 2000.

[6] D. Ozis, T. Fiez, and K. Mayaram, "A comprehensive geometry-dependent macromodel for substrate noise coupling in heavily doped CMOS processes," *CICC 2002*, May 2002, pp. 497-500.

[7] C. G. Xu, "EPIC: A program for extraction of resistance and capacitance of substrate with Green's function method," Dept. of ECE, Oregon State University, 2002.

[8] CADENCE, "Substrate Coupling Analysis User Guide," Product Version 4.4.6, April 2001.

[9] Simplex, "Substrate Storm," http://www.simplex.com, accessed January 2003.

[10] CADENCE, "SeismIC User Guide," Product Version 4.0, May 2002.

[11] A. J. van Genderen, N. P. van der Meijs, "Modeling Substrate Coupling Effects using a Layout-to-Circuit Extraction Program," In *ProRISC IEEE 8th Annual Workshop on Circuits, Systems and Signal Processing*, pp. 193-200, November 1997.

[12] CADENCE, "Substrate Noise Analysis of Mixed-Signal ICs," white paper, April 2003.

[13] B. R. Stanisic, N. K. Verghese, R. A. Rutenbar, L. R. Carley and D. J. Allstot, "Addressing substrate coupling in mixed-mode IC's: simulations and power distribution synthesis," *IEEE J. of Solid-State Circuits*, vol. 29, pp. 226-238, March 1994.

[14] I. L. Wemple and A. T. Yang, "Integrated circuit substrate coupling models based on Voroni-Tessellation substrate macromodels," *IEEE Transactions on Computer-Aided Design*, vol. 14, pp. 1459-1469, December 1995.

[15] D. Ozis, T. Fiez, and K. Mayaram, "A comprehensive geometry-dependent macromodel for substrate noise coupling in heavily doped CMOS processes," *CICC 2002*, May 2002, pp. 497-500.

[16] B. Owens, P. Birrer, S. Adluri, R. Shreeve, S. A. Arunachalam, H. Habal, S. Hsu, A. Sharma, K. Mayaram, and T. Fiez, "Strategies for simulation, measurement and suppression of digital noise in mixed-signal circuits," *CICC 2003*, September 2003, pp. 361-364.

[17] M. van Heijningen, J. Compiet, P. Wambacq, S. Donnay, M. G. E. Engels, and I. Bolsens, "Analysis and experimental verification of digital substrate noise generation for epi-type substrates," *IEEE J. Solid-State Circuits*, vol. 35, pp. 1002-1008, July 2000.

[18] J. Briaire and K.S. Krisch, "Principles of substrate crosstalk generation in CMOS circuits," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 645-653, June 2000.

[19] D. K. Su, M. J. Loinaz, S. Masui and B. A. Wooley, "Experimental results and modeling techniques for substrate noise in mixed signal integrated circuits," *IEEE J. Solid-State Circuits*, vol. 28, pp. 420 - 430, April 1993.

[20] Y. Zinzius, E. Lauwers, G. Gielen, and W. Sansen, "Evaluation of substrate noise effect on analog circuits in mixed-signal designs," *2000 Southwest Symposium on Mixed-Signal Design*, February 2000, pp. 131-134.

[21] A. Sharma, "Predictive Methodologies for Substrate Parasitic Extraction and Modeling in Heavily doped CMOS Substrates," M.S. Thesis, Oregon State University, July 2003.

APPENDICES

APPENDIX A - Silencer! Software Documentation

The substrate noise coupling analysis tool Silencer! consists of either SKILL- or C-functions. SKILL functions are mainly used to communicate with the CADENCE DFII database and for the entire graphical user interface (Silencer! GUI). These functions can be found in the folder ‘Silencer/Skill’. Computationally more expensive algorithms, however, e.g., substrate network extraction and matrix operations, have been programmed in C language. They are compiled to an executable child-process ‘sncCC.exe’ and will be invoked by the SKILL parent process. The communication between the two processes is done through the I/O stream.

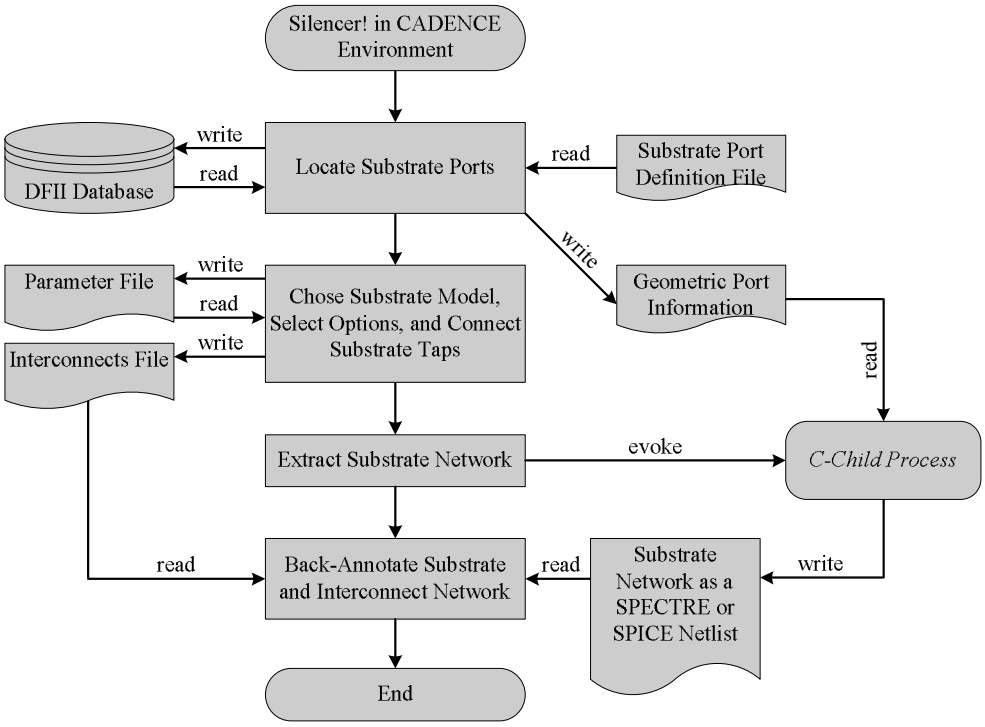

Figure A.1 The Silencer! software flow

First, Silencer! has to know the different types of substrate ports it needs to locate in the layout. The ports available for a specific technology are defined in the substrate port definition file. Silencer! reads these definitions and then accesses the DFII database to search for such objects (layers with certain attributes). Located substrate ports (marked with a new layer and a label) will be stored in both, the database and the geometric port information file. Second, the parameters have to be read and eventually edited in the parameter file. Some substrate ports, e.g., p-taps, have to be connected to a specific voltage potential through interconnects. These interconnects need to be defined by the user are saved in the interconnects file. Third, after a model for the substrate resistor extraction has been chosen, the

substrate network can be extracted. In that step, the SKILL parent process evokes a C child-process. This process reads the geometric port information (available in two different file formats) and calculates the substrate network. It saves the network as a netlist in a file (SPECTRE or SPICE format). If the child-process is finished, it will send a message to the parent. The parent reads the substrate network and eventually back annotates it into the schematic view. Not only the substrate network, but also the interconnect network in the interconnects file will be back annotated. The schematic is then ready for simulation.

Appendix A contains a description of all Silencer! functions including software flow diagrams that explain some of the function flows in greater detail. Furthermore, all Silencer! parameters are listed in table in the same order they are stored in the parameter data structure.

A.1 Software Flow Diagrams

The following flow charts show how the various functions get invoked either during CADENCE startup or the user interface:

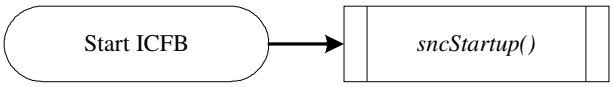

Figure A.1.1 CADENCE startup function

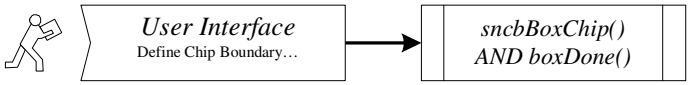

Figure A.1.2 User Interface: Define Chip Boundary…

Figure A.1.3 User Interface: Select New Sensor/Injector Region…

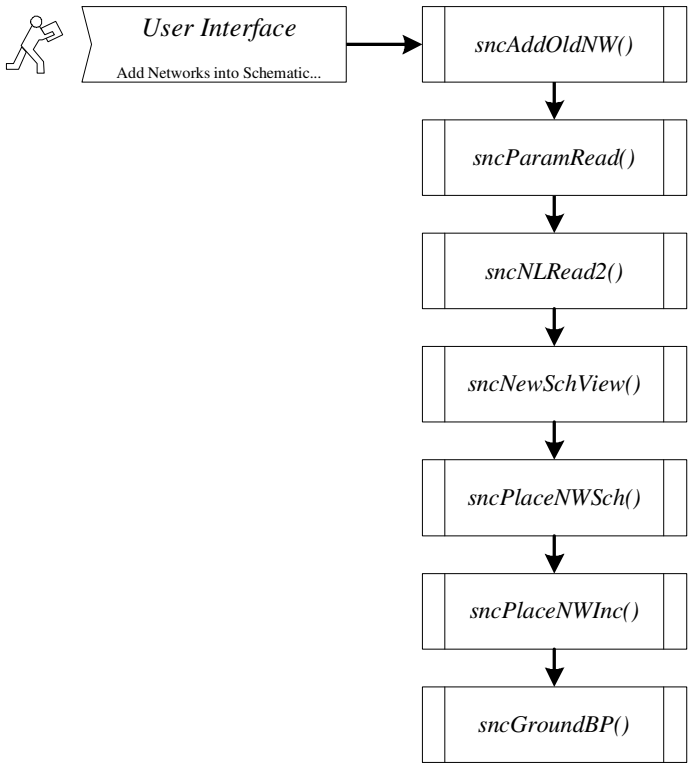

Figure A.1.4 User Interface: Add Networks into Schematic…

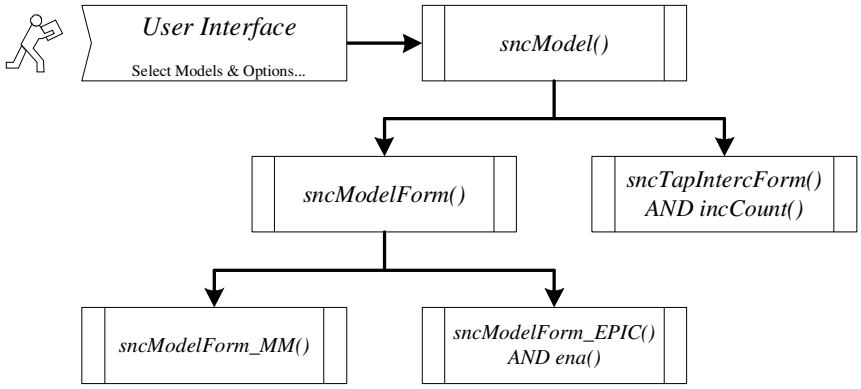

Figure A.1.5 User Interface: Select Models & Options…

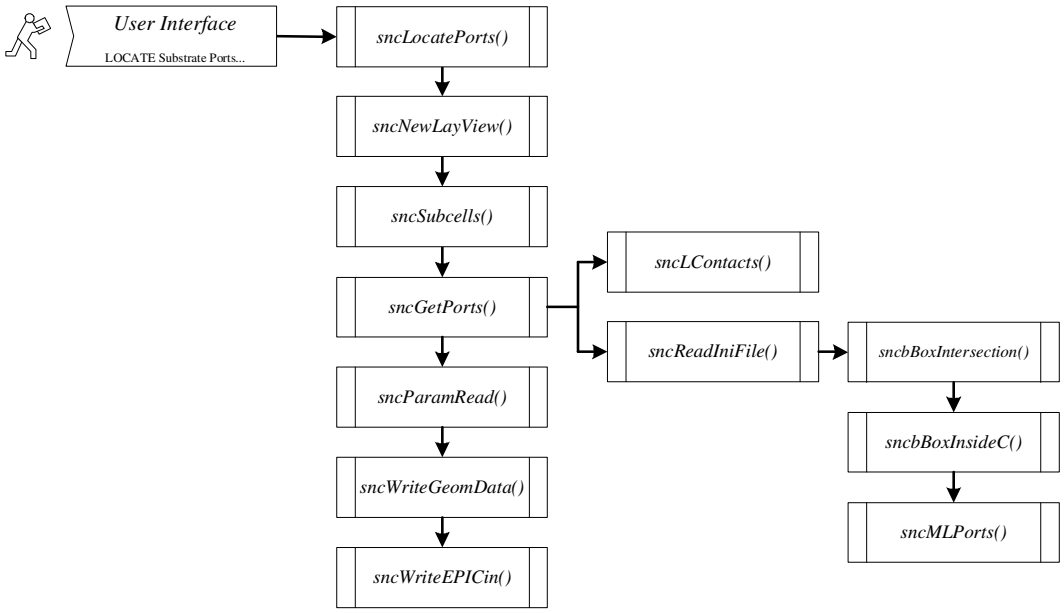

Figure A.1.6 User Interface: LOCATE Substrate Ports…

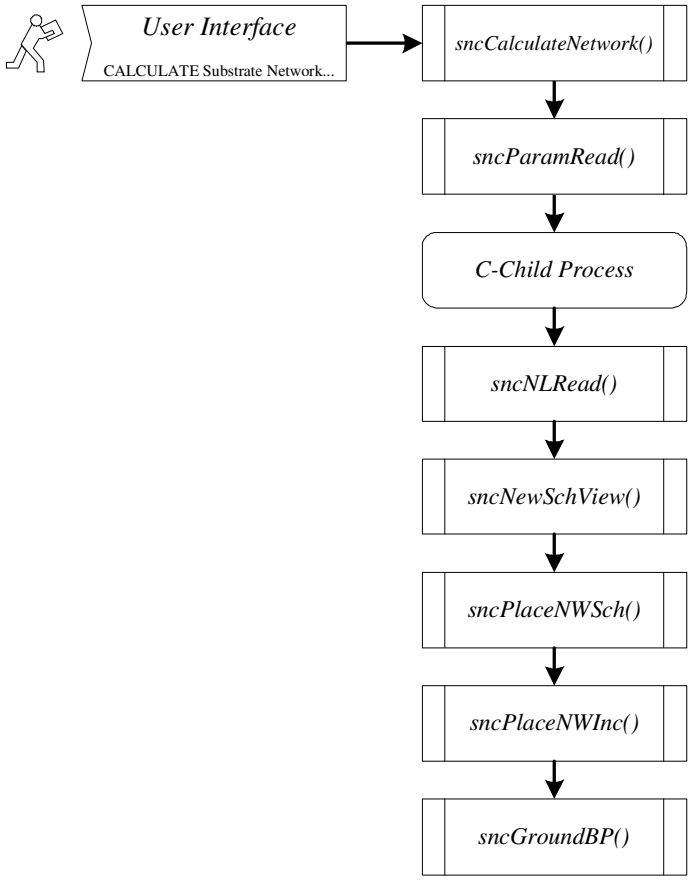

Figure A.1.7 User Interface: CALCULATE Substrate Network…

A.2 SKILL Functions Overview

The following Tables lists all Silencer! SKILL files and its functions with a brief description:

Table A.2.1 SKILL Functions in File sncMain.il.

File	Function	Description
sncMain.il	sncStartup()	Initialize SKILL constants and variables, setup symbols, and define path- and file names of the Silencer! I/O- files.
	sncLocatePorts()	Copy layout into a new layout cell view, locate substrate ports and mark them with injector/sensor layers. If they haven't been labeled previously, label them and store geometric port data in data structure and as an ASCII file.
	sncCalculateNetwork()	Calculate a substrate network from the geometric port data using one particular substrate model and create a SPICE or SPECTRE netlist. If wished, display substrate network, tap interconnects network, and/or network for grounding the backplane in a new schematic cell view.

Table A.2.2 SKILL Functions in File sncGUI.il.

sncGUI.il	sncModel()	Create a 2-D from to choose models & options for substrate network calculation.
	sncModelForm()	Choose which model form to call.
	sncModelFormMM()	Create a 2-D form to view parameters of macro-models.
	sncModelForm_EPIC()	Create a 2-D form to view and edit parameters of EPIC.
	ena()	Enable or disable menu options in EPIC form.
	sncTapIntercForm()	Create a 2-D form to enter p-tap and other interconnects.
	incCount()	Update interconnect counter and add new values in list.
	sncbBInjSen()	Select injector and sensor circuits in layout view.
	sncbBChip()	Define chip boundary in layout view.
	sncInfoDialog()	Display an information message on the screen.

Table A.2.3 SKILL Functions in File sncPortData.il.

sncPortData.il	sncSubCells()	Flatten sub-cells and merge layers specified in file <Silencer.ini>.
	sncNewLayView()	Open a new layout cell view with suffix '_snc', copy the entire layout into the new cell view, and return its ID.
	sncGetPorts()	Recognize substrate ports underneath injector and/or sensor regions. Recognized ports will be marked labeled (if not previously labeled). If no injector and/or sensor regions, but contacts are present in the layout (i.e. for pre-layout analysis), the function recognizes them as such. Names and coordinates of ports or contacts will be returned in a data structure.
	sncMLPorts()	Mark ports, label ports (if not previously labeled), and return coordinates and port names.
	sncLContacts()	Label contacts (if not previously labeled), and return coordinates and contact names.
	sncbBoxInsideC()	Return bounding boxes that are either enclosed (flag = 0) or not enclosed (flag = 1) in bounding box C.
	sncbBoxInterSection()	Return interSections of bounding boxes in list A with bounding boxes in list B.

Table A.2.4 KILL Functions in File sncFiles.il.

sncFiles.il	sncParamRead()	Read parameter file <parameters.txt> and return structure with parameters. Flag = 0, all parameters, flag = 1, general parameters, flag = 2, macro-model parameters, flag = 3, EPIC parameters.
	sncParamWrite()	Write updated parameters into parameter file <parameters.txt>.
	sncWriteGeomData()	Write port coordinates, area, and perimeter into ASCII file <subsPorts.txt>.
	sncWriteEPICin()	Generate EPIC input file <EPICinput.dat>.
	sncNLRead()	Read netlist file <SCnetlist.txt> and return a structure with resistor names and values.
	sncNLRead2()	Read netlist file <SCnetlist.txt> and return a structure with resistor names and values → same as sncNLRead(), but number of ports is unknown.
	sncReadIncFile()	Read interconnect file <ptapsItco.txt>.
	sncWriteIncFile()	Write interconnect file <ptapsItco.txt>.
	sncReadIniFile()	Read file <Silencer.ini>.

Table A.2.5 SKILL Functions in File sncPlaceNW.il.

sncPlaceNW.il	sncNewSchView()	Open a new schematic cell view, copy circuit schematic into the new cell view, and return its ID.
	sncPlaceNWSch()	Place substrate network in schematic view.
	sncGroundBP()	Ground backplane if option was enabled.
	sncPlaceNWInc()	Place interconnect network in schematic view, if any were defined.
	sncAddOldNW()	Add previously generated substrate network and interconnect network into schematic view without recalculating the substrate network.

A.3 SKILL Functions Description

The following paragraphs describe the most important Silencer! SKILL functions in greater detail and include some Jackson flow diagrams:

A.3.1 sncMain.il

The SKILL file sncMain.il contains three Silencer! main functions: sncStartup(), sncLocatePorts(), and sncCalculateNW().

The function sncStartup() will automatically be called during CADENCE startup. Its purpose is not only to initialize constants and variables, but also to set up symbols and define path- and file names of the Silencer! I/O- files. All symbols used in the program, such as resistors, inductors, ground symbols, and pins, are either taken from the analog or basic library. sncStartup() has to be invoked from the .cdsinit.user file.

The function sncLocatePorts() will be called after selecting the menu option 'SILENCER!/LOCATE Substrate Ports…'. It locates substrate ports in a layout (substrate ports are physical locations in the layout where noise could be coupled into the substrate or sensed from the substrate, such as transistors, substrate taps, and wells), marks them with either an injector or sensor layer and eventually labels them automatically. However, in some cases, it may be more convenient to label some ports manually before locating the substrate ports. In such a case, the function will recognize a label and use its name as a substrate port name. The entire layout including the located substrate ports will be copied into a new cell view with suffix '_snc'. After localizing the substrate ports, the function stores the geometric port data extracted from the layout (coordinates, port names) in both, a data structure and the file 'subsPorts.txt'. Moreover, an EPIC input file 'EPICinput.dat' will be

created. If no error occurred while locating the substrate ports, the function will display the following message: ‘New port files <subsPorts.txt> and <EPICinput.dat> have been generated!’.

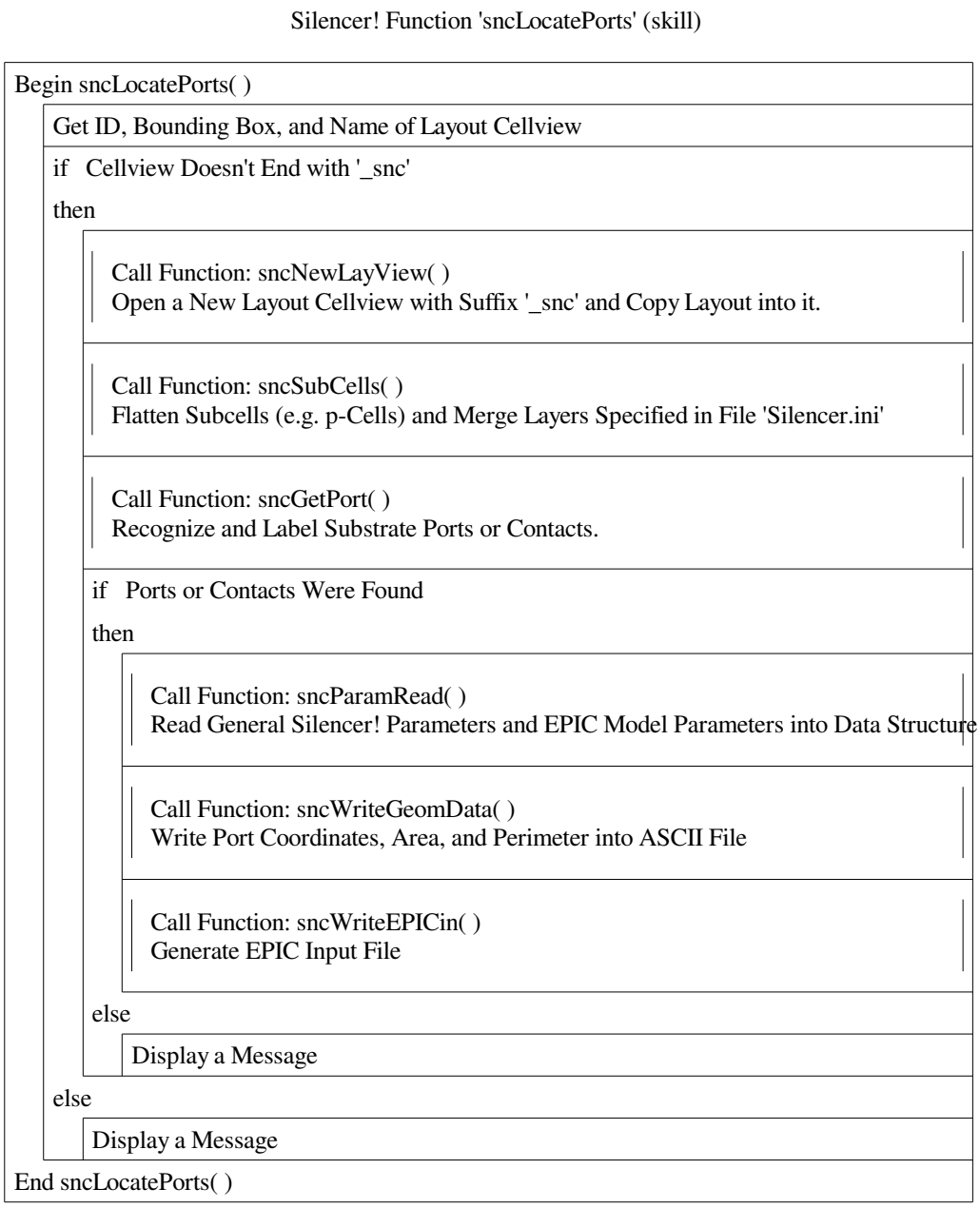

Figure A.3.1.1 Jackson flow diagram of the function sncLocatePorts().

The function sncCalculateNW() calculates a substrate network from the geometric port data. If any substrate ports are present in the layout, it will be invoked by choosing the menu option 'SILENCER!/CALCULATE Substrate Network…'. After a substrate network was calculated using one particular substrate model, the function will create a netlist in a SPICE or SPECTRE format and save it in the file 'SCnetlist.txt'. Silencer! allows to display the substrate network in the schematic view, to include a network for the substrate tap interconnects, and/or ground the backplane. If the option 'Display Substrate and Interconnect Network in Schematic…' has been enabled, the function will display the networks automatically in a new schematic view with suffix '_snc'.

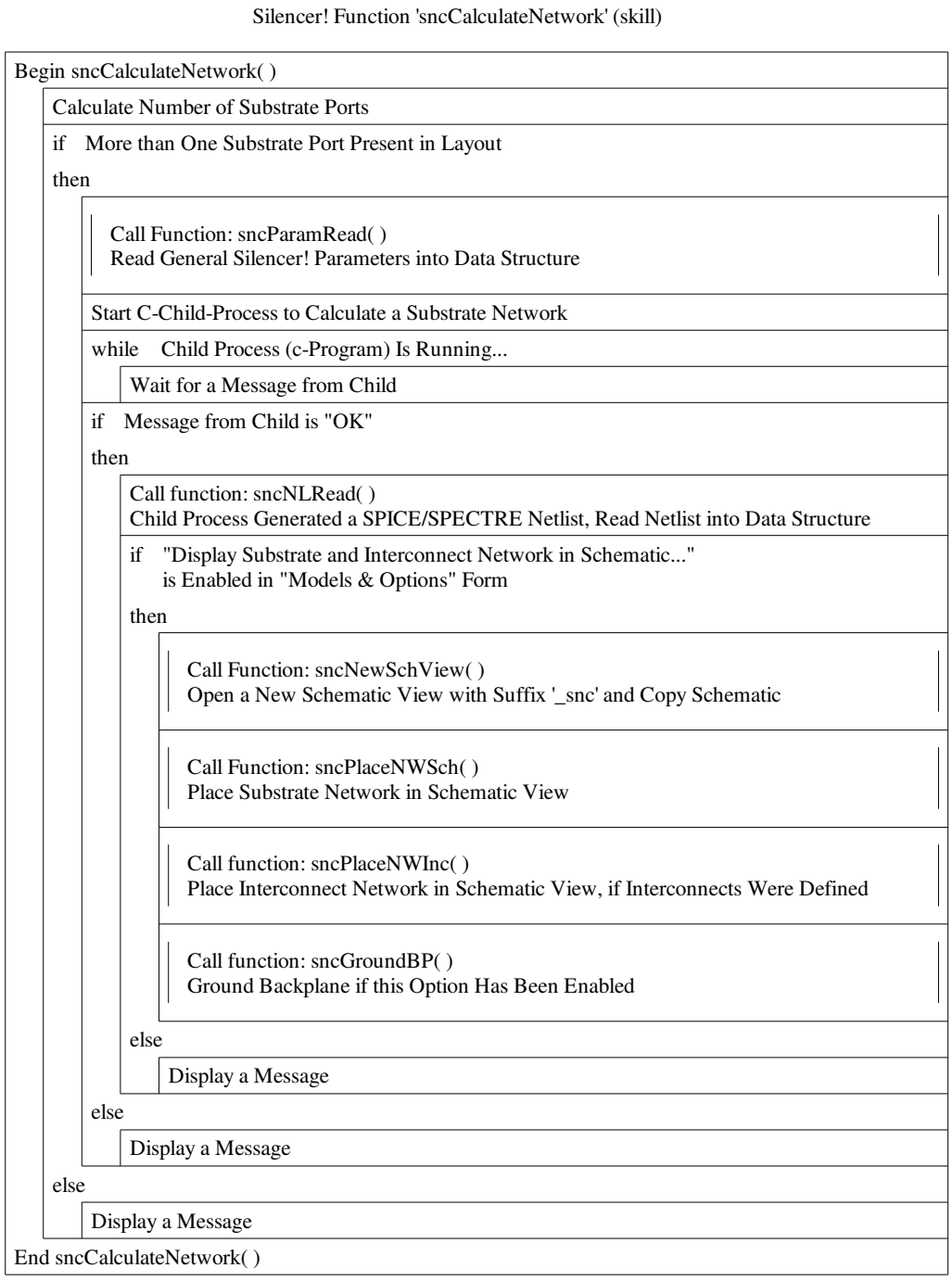

Figure A.3.1.2 Jackson flow diagram of the function sncCalculateNW().

A.3.2 sncPorts.il

The SKILL file sncPorts.il contains the following Silencer! functions to identify substrate ports and contacts: sncSubCells(), sncNewLayView(), sncGetPorts(), sncMLPorts(), sncLContacts(), sncbBoxInsideC(), sncbBoxInterSection(), and sncbBoxChip().

The function sncSubCells() flattens all sub-cells (e.g. p-cells) and sub-hierarchies of the duplicated layout view with suffix '_snc'. In addition, layers specified in the file <Silencer.ini> will be merged. The function sncSubCells() will be invoked by function sncLocatePorts().

The function sncNewLayView() opens a new layout cell view with suffix '_snc', copies the entire layout into the new cell view, and returns its ID. It will be called from sncLocatePorts().

The function sncGetPorts() detects if there are injector/sensor contacts or regions present in the schematic. Contacts can be used for pre-layout analysis or validating a substrate model. Injector and/or sensor regions will be used to simulate entire circuits and substrate ports underneath an injector/sensor region layer will automatically be recognized, marked, and labeled. However, it is possible to manually label some substrate ports by putting an injector/sensor label inside the port. In such a case, the function will recognize a label and use its name as a substrate port name. The kinds of substrate ports to be detected are specified in the file <Silencer.ini>. Consequently, any substrate ports of interest for any specific technology can be programmed. As an example: A particular substrate port, let's call it port_type1 would be recognized in the layout whenever layer1 overlaps with layer2, but is not surrounded by layer3. All substrate ports that were detected will be stored with name and coordinates in a data structure. The data structure will be

returned as a function value. The function sncGetPorts() is invoked by sncLocatePorts().

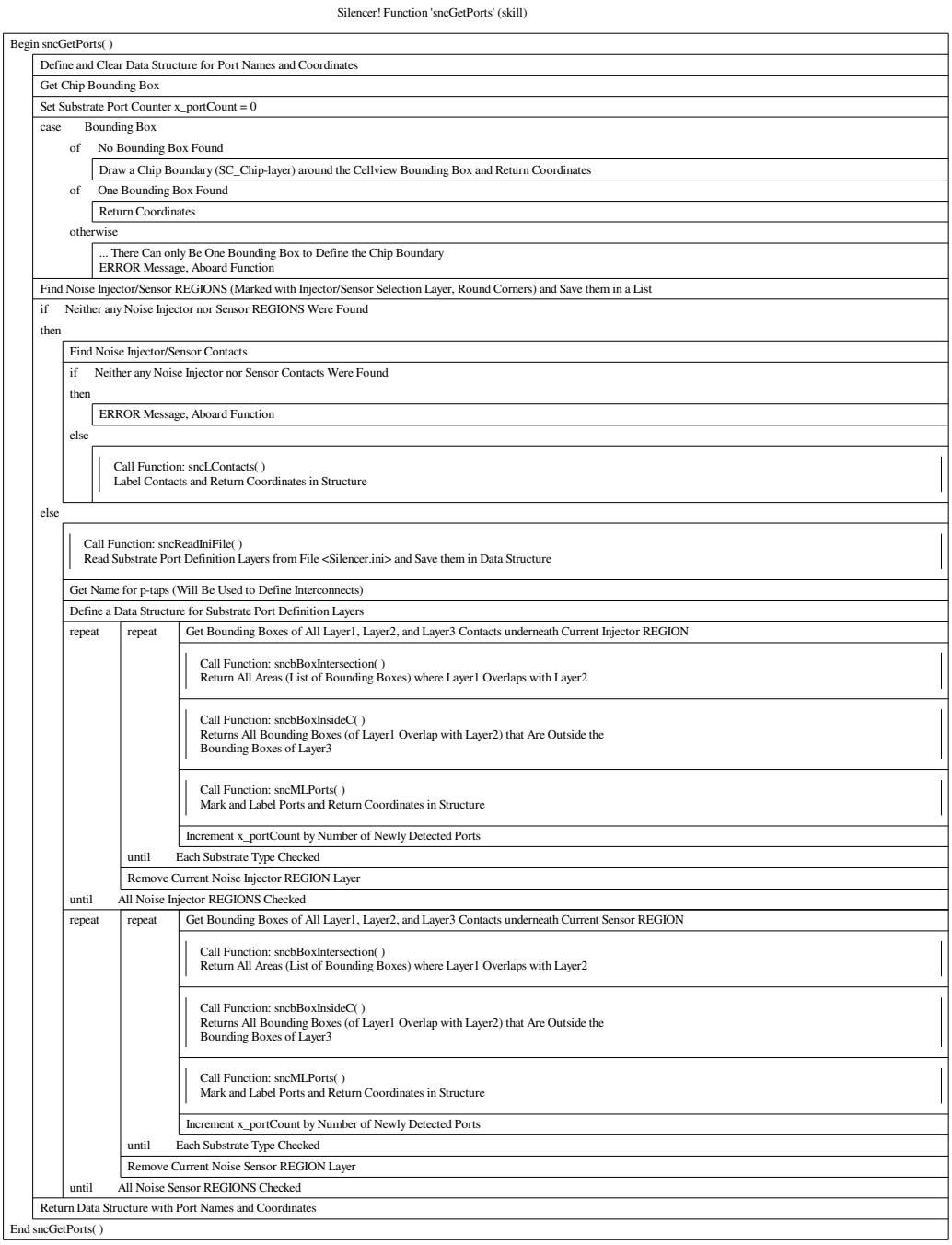

Figure A.3.2.1 Jackson flow diagram of the function sncLocatePorts().

A.3.3 sncLocatePorts.il

The function sncMLPorts() will be called from function sncGetPorts() for each substrate injector/sensor region and each substrate port type. It checks for each port, if there is already a label inside. If so, it will use that label as a port name, otherwise, it will create a new name. Such a new name always consists of the substrate port type plus a number. Labels will be placed at the left hand side, outside the port. In the layout, the ports will be marked with a sensor or injector layer (depending on whether the circuit is a noise injector or sensor. Finally names and coordinates of the ports will be returned in a data structure.

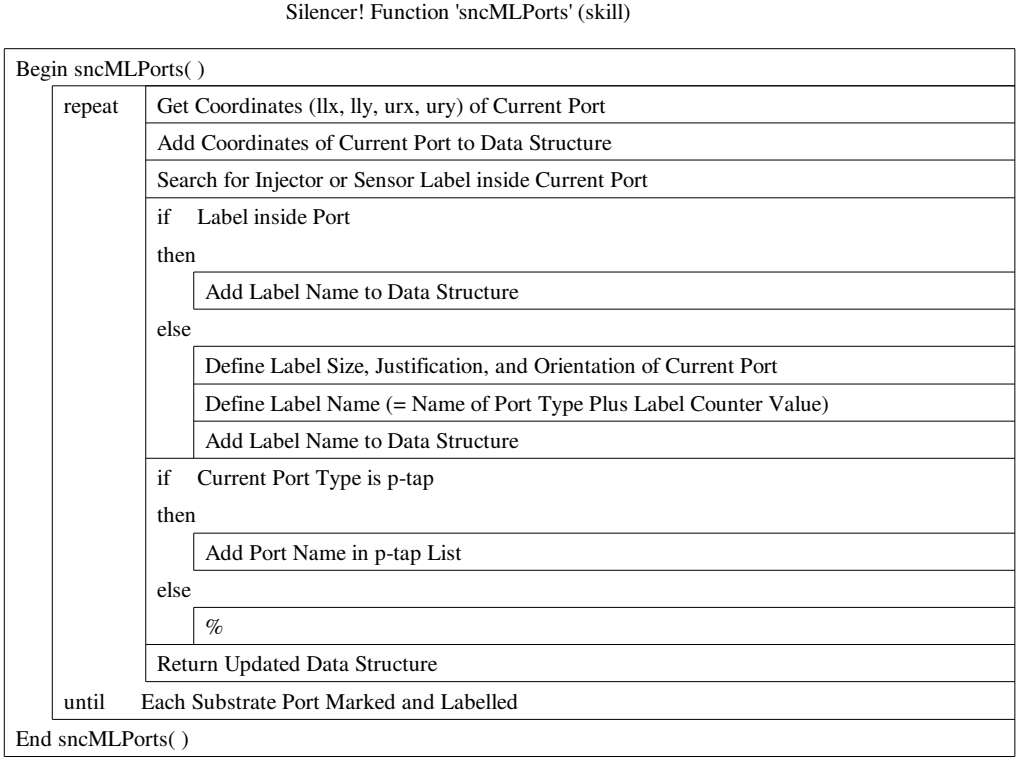

Figure A.3.3.1 Jackson flow diagram of the function sncMLPorts().

The function sncLContacts() is very similar to function sncMLPorts() and will also be invoked by function sncGetPorts(). It checks for each contact, if there is already a label inside. If so, it will use that label as a contact name, otherwise, it will create a new name. A new name will be 'cont_' and a number. Labels will be placed at the left hand side, outside the contact and contact names and coordinates will be returned in a data structure.

The function sncbBoxInsideC() goes through a list of bounding boxes, lets call them bounding boxes C. It will check if any bounding box of another list of bounding boxes is inside (or not inside) the bounding boxes C. It will return a list of bounding boxes that are inside (or not inside) the bounding boxes C. This function will be called from function sncGetPorts().

The function sncbBoxInterSection() finds in two lists (let's call them A and B) any bounding box in A that intersects with another bounding box in B. It will return a list with all the interSection bounding boxes. This function will be called from function sncGetPorts().

A.3.4 sncFiles.il

The SKILL file sncFiles.il contains the following Silencer! functions used for file handling: sncParamRead(), sncParamWrite(), sncWriteGeomData(), sncWriteEPICin(), sncNLRead(), sncNLRead2(), sncReadIncFile(), sncWriteIncFile(), and sncReadIniFile().

The functions sncParamRead() and sncParamWrite() are used to read the Silencer! parameters from a file into a data structure or to write parameters into a file - the parameter file is called <parameters.txt>. The parameters are grouped in general Silencer! parameters, macro-model parameters, and EPIC parameters. By using flag = 0, all parameters are read, flag = 1, general parameters are read, flag = 2, macro-model parameters are read, and flag = 3, EPIC parameters are read. sncParamRead() is called from the functions sncLocatePorts(),

sncCalculateNetwork(), and sncModel(). sncParamWrite() is only invoked by the function sncModel() and always all parameters are written into the file. The first few lines of the parameter file tell when the parameters have been changed the last time and by whom.

The functions sncWriteGeomData() and sncWriteEPICin() write the geometric information about the substrate ports into an ASCII file format. The files created are <subsPorts.txt> and <EPICinput.txt>. They will be used by the macro-models and EPIC to calculate the substrate network. The functions will be called from the function sncLocatePorts().

The functions sncNLRead() and sncNLRead2() are used to read the substrate network (SPICE or SPECTRE netlist) into a data structure. Saved will be resistor name, terminal1, terminal2, and resistor value. They will be called from the functions sncCalculateNetwork() and sncAddOldNW().

The functions sncReadIncFile() and sncWriteIncFile() are used to read the Silencer! p-tap interconnects from a file into a data structure or to generate a new interconnects file. The tap-interconnect file is called <ptapsItco.txt>. The functions sncReadIncFile() and sncWriteIncFile() are invoked by the functions sncPlaceNWInc() and sncTapIntercForm().

The function sncReadIniFile() reads definitions from the file <Silencer.ini>. This can be definitions of substrate ports or layers in subcells that have to be merged. It is called from the functions sncGetPorts() and sncSubCells().

A.3.5 sncPlaceNW.il

The SKILL file sncPlaceNW.il contains Silencer! functions used to display the substrate and interconnect network in the circuit schematic. These functions are: sncNewSchView(), sncPlaceNWSch(), sncGroundBP(), sncPlaceNWInc(), and sncAddOldNW().

The function sncNewSchView() creates a new schematic view with suffix '_snc' and copies the circuit schematic into the new view. If such a schematic already exists, it will be overwritten. The old schematic will be closed, if it was open. The function is called from the function sncCalculateNetwork() and returns the new ID of the schematic.

The function sncPlaceNWSch() places the substrate network at the right hand side of the schematic, labeles the terminals, and adds pins. It is called from the functions sncCalculateNetwork() and sncAddOldNW().

The function sncGroundBP() will be invoked by the function sncCalculateNetwork() if the backplane needs to be grounded. The function displays a series R-L circuit connected from the backplane to ground in the schematic.

The function sncPlaceNWInc() will be invoked by the function sncCalculateNetwork() if any p-tap interconnect network has been defined. The function displays R-L series circuits connected from one p-tap to another p-tap or to ground in the schematic.

The function sncAddOldNW() will be called after selecting the menu option 'SILENCER!/Add Networks into Schematic…'. It reads the parameter file <parameters.txt> and the netlist file <SCnetlist.txt>. If a netlist is present, it opens a new schematic cell view with suffix '_snc' and copies the old schematic into that new cell view. Finally, it connects both, substrate and interconnect network to the schematic view.

A.3.6 sncGUI.il

The SKILL file sncGUI.il contains the graphical user inerface functions of Silencer! such as sncModel(), sncModelForm(), sncModelFormMM(), sncModelForm_EPIC(), ena(), sncTapIntercForm(), sncbBInjSen(), sncbBChip(), and sncInfoDialog().

The functions sncModel(), sncModelForm(), sncModelFormMM(), sncModelForm_EPIC(), and ena() display the Models & Options forms for the general Silencer! options, and the EPIC parameters. They are invoked by the menu option 'SILENCER!/Model & Options…'. Any changes in these forms will be saved in the file <parameters.txt>.

The function sncTapIntercForm() will be called after pressing the button 'Tap Interconnects…' in the 'Models & Options' form. A form will be displayed in which p-tap interconnects can be defined as an R-L series circuit. Interconnects will be saved in the file < ptapsItco.txt >.

The functions sncbBInjSen(), sncbBChip() are used to draw injector/sensor regions and a chip boundary, respectively. They will be called by selecting the corresponding menu items.

The function sncInfoDialog() displays a dialog window with information about Silencer! Dialogs are used to guide the user in the substrate coupling analysis flow.

A.4 C Child-Process

The following table lists the Silencer! C child-process functions, not including EPIC. The software flow of the child-process is described in the Jackson diagram.

Table A.4.1 Silencer! C child-process functions.

gFile_r.c	modelPar()	Read model parameters from file <t_paraFile>
	z_matrixMM()	Get z-matrix from port data for either macro-model1 or macro-model2
matrixInv.c	matrixInv()	Code taken from Chenggang Xu to calculate y-matrix from z-matrix
rCalc.c	rNWcalc()	Calculate r-matrix from y-matrix
	rWrite()	Write substrate network in either SPECTRE or SPICE format into file <t_netlFile>
	getPortNames()	Get names of substrate ports from port data file <t_portFile>

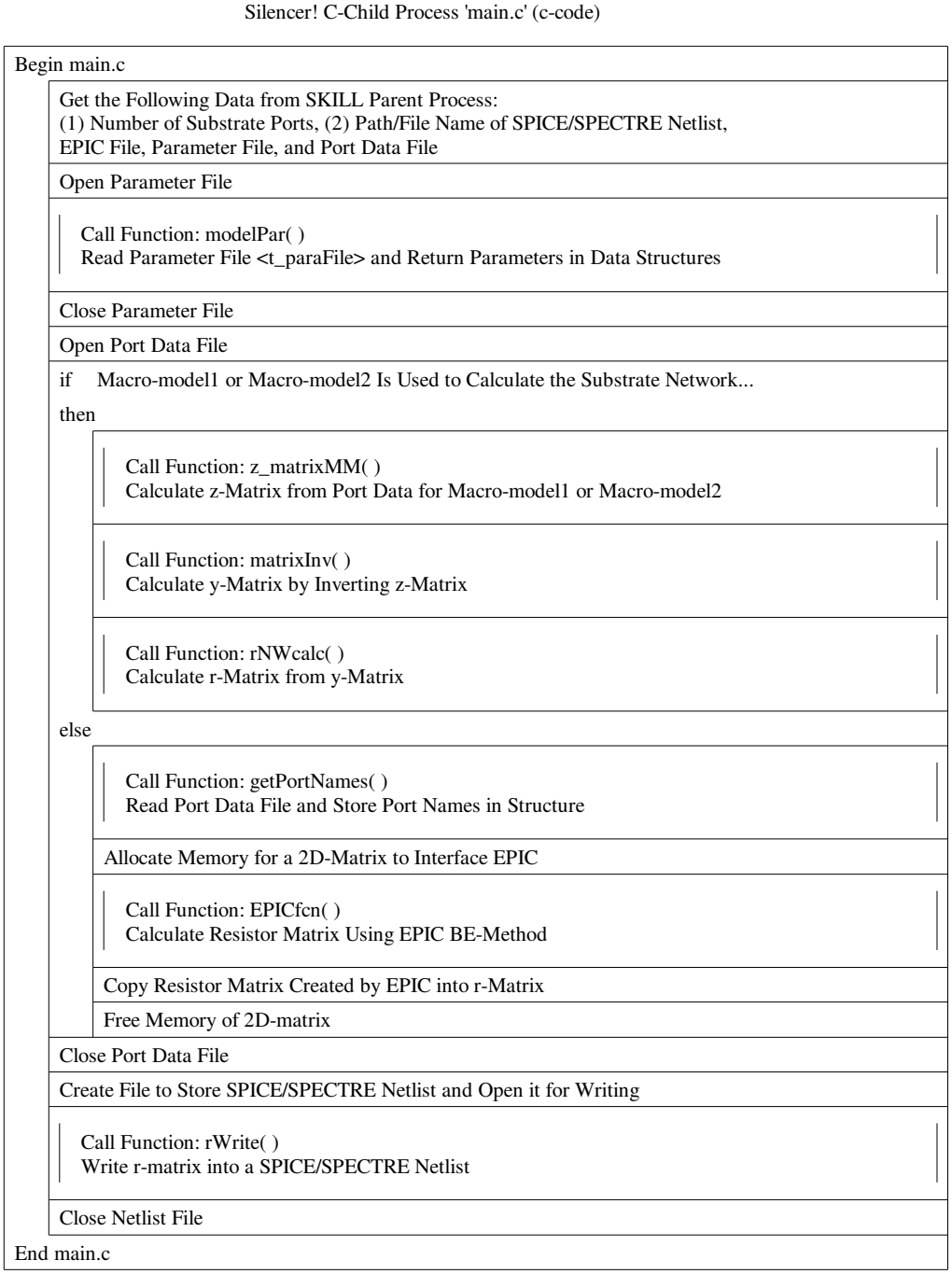

Figure A.4.1 Jackson flow diagram of the function main.c.

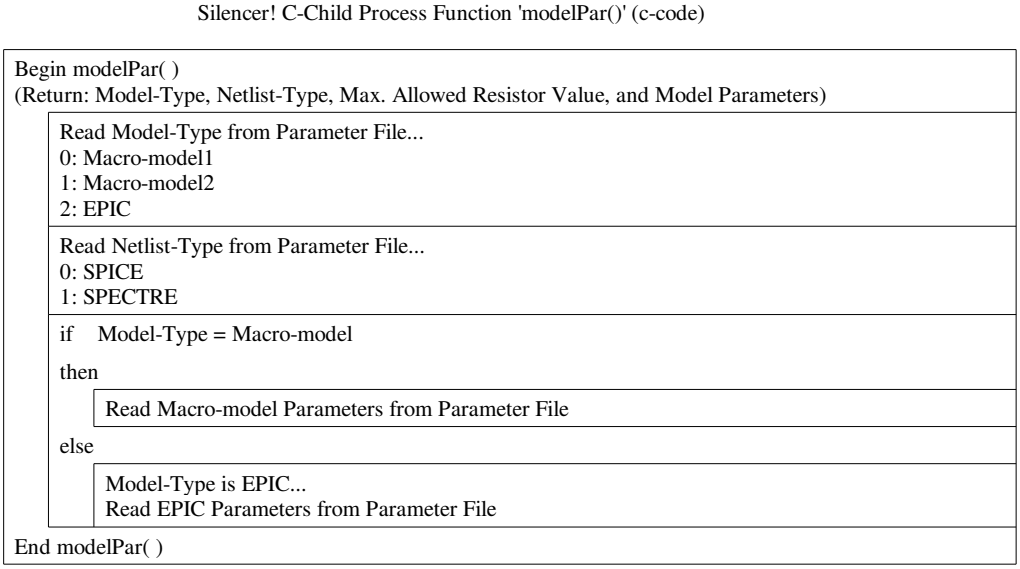

Figure A.4.2 Jackson flow diagram of the function modelPar().

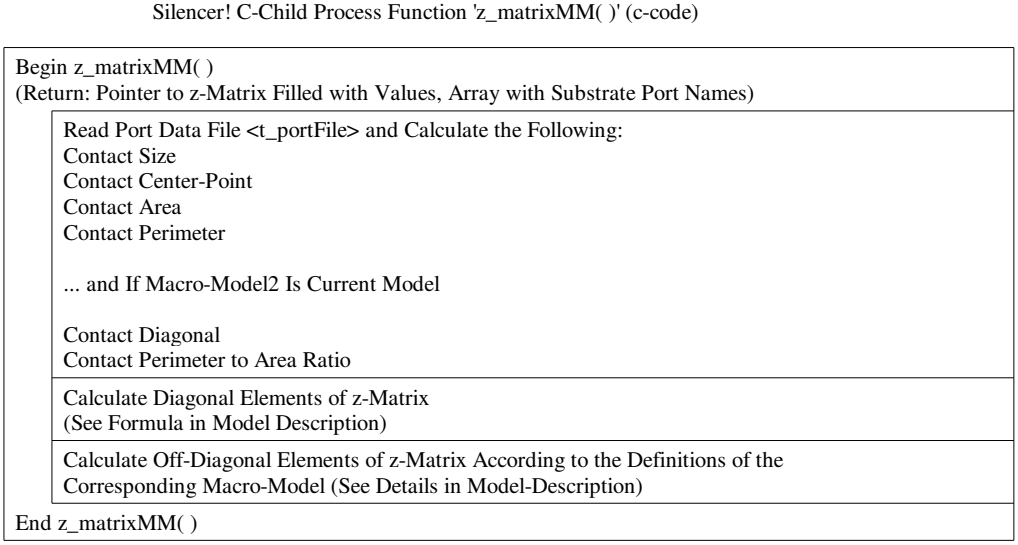

Figure A.4.3 Jackson flow diagram of the function z_matrixMM.c.

A.5 Silencer! Model Parameters

The following table describes the Silencer! model parameters and how they are organized inside the data structure 'a_param'. They are grouped in three sets of parameters: general Silencer! parameters, macro-model parameters, and EPIC parameters.

Table A.5.1 Silencer! model parameters and their organization within the data structure.

Array Location	***Parameter Name***	***Description***
a_param[0]	a_para_general	General parameters
a_param[1]	a_para_MM	Macro-model parameters
a_param[2]	a_para_EPIC	EPIC parameters
Array Location	***Parameter Name***	***Description***
a_para_general[0]	modelType (v)	Model type value and default value, 0: Macro-model1, 1: Macro-model2, 2: EPIC
a_para_general[1]	modelType (d)	
a_para_general[2]	networkType (v)	Network type value and default value, 0: SPICE, 1: SPECTRE
a_para_general[3]	networkType (d)	
a_para_general[4]	simView (v)	Simulation from 0: layout view, 1: extracted view
a_para_general[5]	simView (d)	
a_para_general[6]	maxRes	Discard large resistors in network
a_para_general[7]	maxResVal	Max. resistor value in network
a_para_general[8]	bpGnd	0: floating backplane, 1: grounded backplane
a_para_general[9]	bpR	R, L series circuit for grounding backplane
a_para_general[10]	bpL	
a_para_general[11]	flattenCV	Flatten all hierarchies of subcells
a_para_general[12]	inSchem	Place network in schematic

Array Location	***Parameter Name***	***Description***
a_para_EPIC[0]	numOfLay	Number of layers in substrate
a_para_EPIC[1]	DCT	Discrete-Fourier-Transform size
a_para_EPIC[2]	contS	Contact size
a_para_EPIC[3]	blk1	Bulk layer resistivity
a_para_EPIC[4]	blk2	Bulk layer thickness
a_para_EPIC[5]	epi1	Epi layer resistivity
a_para_EPIC[6]	epi2	Epi layer thickness
a_para_EPIC[7]	chStp1	Channel stop resistivity
a_para_EPIC[8]	chStp2	Channel stop thickness
a_para_EPIC[9]	method	Simulation method, 0: direct, 1: iterative
a_para_EPIC[10]	methodTol	Tolerance for iterative method
a_para_EPIC[11]	maxDiv	Max. division of contact

A.6 Accessing the DFII Database

All Cadence tools use the Design Framework II unified database; a binary database that stores data as objects. There are many types of objects, each representing a distinct concept in the world of electronic design automation. Examples of object types include rectangles, terminals, instances, and cell views. The Design Framework II database can store both physical and logical information about a design. Physical information is stored in objects such as geometrical shapes and an IC layout. Logical information is stored in objects such as nets or schematics, which can exist without any corresponding physical realizations.

To access an object in the database, a label known as the object identifier or ID has to be used. Each ID uniquely identifies a database object and is represented by the special type dbObject. Only database routines can create IDs, consequently the user cannot alter IDs directly. When a function that operates on a database object is used, the ID as an argument to the function has to be given as well.

In programming terms, a variable of dbObject type represents the ID of an object in the database. If the database object identified by a dbObject variable is deleted, that dbObject variable is invalid and cannot be used to do any database operation.

Each object is associated with a type, and each object type has a set of attributes that describe the object. An object class is a data-type abstraction that groups related object types into a class. Object classes are created when a number of distinct object types share enough attributes that they can be discussed as a single class, without being concerned about the exact object type. The different types and classes of objects form a class hierarchy. At the top of the hierarchy is a class containing all types. At the bottom of the hierarchy, each leaf node represents an object type that can be created, deleted, and saved on disk. Each node in the

hierarchy has attributes that are common to all of its children. Geometrical shapes, for example, have the common attributes of layer, purpose, and bounding box. Each object type has a predefined set of attributes that you can retrieve using type specific access functions. There are three types of attributes: mandatory, optional, and derived attributes.

Mandatory attributes must be specified at the time an object is created. Most attributes fall into this category. Optional attributes may or may not exist for a particular object. These attributes do not need to be specified at the time an object is created, they can be added to the object at any time. Derived attributes are derived from the object's other attributes. Only read-only access exists to derived attributes and they cannot be stored in the database. However they play an important role in speeding up repetitive computation.

An object property has a name and a value, such as a Boolean value, an integer, a floating point number, a string, or a representation of time (see Appendix A.7). An arbitrary number of properties can be attached to an object used to store different kinds of information such as mechanisms for extending the basic schema of the database.

The most important distinction between attributes and properties is that attributes are predefined and managed by the database, while properties are defined and managed by applications built on top of the database. This distinction is often blurred because some information currently stored as properties is also essential to the system's operation. Properties are not mandatory as far as the database is concerned; however, some properties are essential to the operation of certain applications (for example, net listing, place and route, etc). Within these applications, such properties can be considered mandatory.

The database access operator -> works on both attributes and properties associated with a given database object. Because the name of an attribute or a property can be specified on the right-hand side of the operator, attributes and properties share a single name space. For this reason, in earlier documentation

attributes and properties were lumped together and referred to as properties. For the ~> operator, attributes always have precedence over properties, which means that one cannot access a property that has the same name as an attribute using the ~> operator.

The CADENCE database stores many predefined relationships among different types of objects. Some relationships are automatically kept by the database, while other inter-object relationships must be explicitly created and maintained by applications. Some relationships in the database are mandatory, for example, each terminal must be attached to a net. Application programs cannot, therefore, create a terminal without specifying the net it connects to. Mandatory relationships are very often unidirectional. For example, it is acceptable to have a net that does not connect to a terminal. Some relationships in the database are optional. Objects can be created independently now with a relationship created later by an application. When a predefined relationship is one-to-one or many-to-one, there is an access function (in the form of a pseudo-attribute) to go from the source object to the object it relates to. When a predefined relationship is a one-to-many relationship, there is an access function (in the form of a pseudo-attribute) that returns a list of the objects contained in the relationship.

Before the database can be accessed, the ‘cds.lib’ file has to be set up so the database can locate the cells in the design. Then various cell views in the design in different libraries located in directories listed in the cds.lib file can be opened. When a cell view is open, other database functions can be used to examine and modify the contents of the cell view.

A.7 Terminology of Newly Defined SKILL Functions

The SKILL functions of CADENCE tool typically use prefixes in front of the function names that identify for which tool the function has been developed. For example, all Layout Editor functions have the prefix ‘le’, or the Schematic Composer functions have the prefix ‘sc’, etc. Consequently, it makes sense to continue with that terminology and all Silencer! functions have a prefix ‘snc’, which stands for substrate noise coupling. Furthermore, the data type of symbols (variables) used in the Silencer! SKILL code can be recognized by the Single Character Mnemonic in the symbol name.

Table A.7.1 Data types supported by SKILL and used in the Silencer! functions:

Data Type	Internal Name	Single Character Mnemonic	Data Type	Internal Name	Single Character Mnemonic
Array	array	a	Defstruct		r
Database object	dbobject	d	Symbol	symbol	s
Floating-point number	flonum	f	Symbol or character string		S
Any data type	general	g	Character string (text)	string	t
Linked list	list	l	Function object		u
Integer or floating-point number		n	Window type		w
User-defined type		o	Integer number	fixnum	x
I/O port	port	p	Binary function		y

APPENDIX B - Installing Silencer!

B.1 Installing Silencer! for the TSMC 0.35µm Heavily Doped Process

Before Silencer! can be installed, the process has to be installed. The following Section describes first, how to set up the TSMC 0.35µm heavily doped process in the CADENCE DFII environment. Second, it describes the Silencer! installation process.

First, a new folder for the project directory has to be created. For example, '~/cadence/SC_Tool' was chosen as the new project directory. Once the directory has been created, the path has to be changed to access that directory, e.g., >cd SC_Tool in Unix. The command '~cdsmgr/process/tsmc0.35.3/setup' will set up a new TSMC 0.35µm heavily doped process and the path '~/cadence/SC_Tool'. That path will become the new CADENCE startup directory for that process. If no automatic setup for the technology is provided, the required files have to be copied into the directory manually.

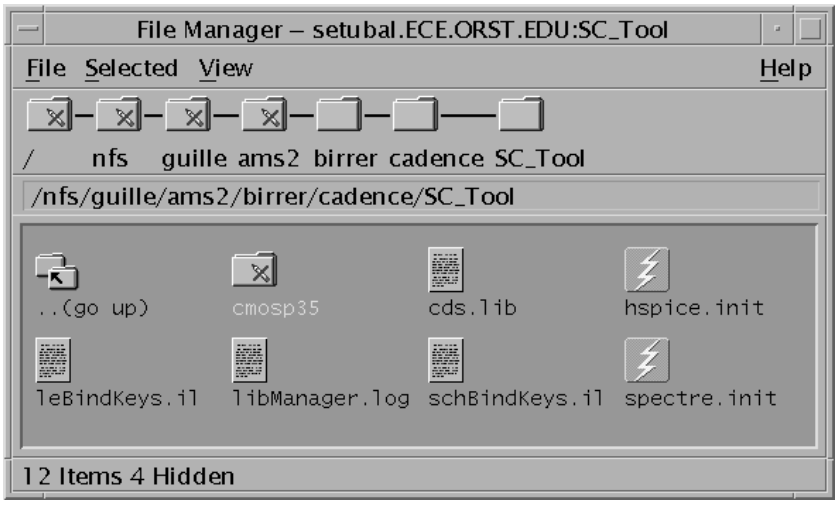

Figure B.1.1 The files created in the CADENCE startup directory for the TSMC 0.35µm heavily doped process.

Once the setup new of the process is done, Silencer! can be installed. This can be done simply by copying the folder 'Silencer' into the startup directory. The folder contains totally three directories and six files.

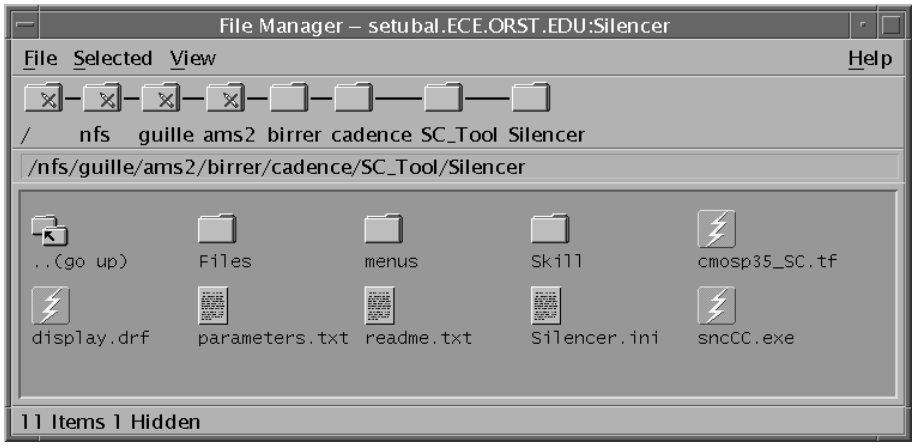

Figure B.1.2 The folder 'Silencer' contains three directories and six files.

The folder ‘Files’ is empty and will be used by Silencer! during the process of localizing substrate ports and extracting the substrate network to save the created output files.

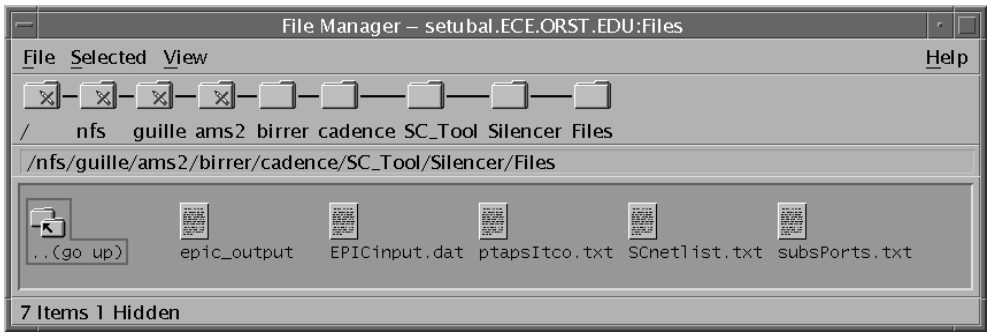

Figure B.1.3 The folder ‘Files’ is used during the process of localizing substrate ports and extracting the substrate network to save the created output files.

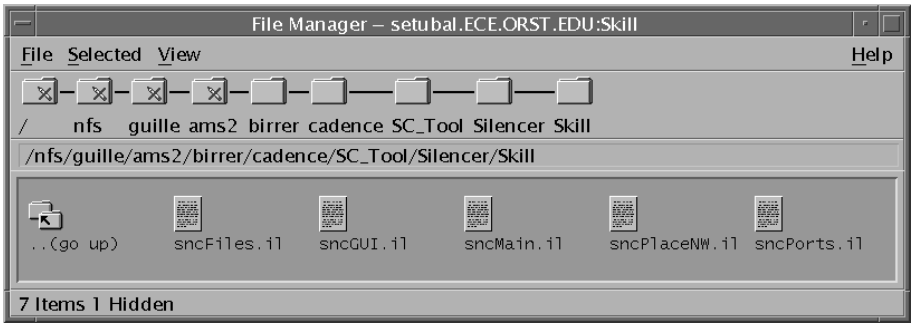

Figure B.1.4 The folder ‘Skill’ contains all the SKILL functions used by the tool, as shown in the following figure.

Further information about the SKILL functions and SKILL programming in general can be found in Appendix A. In addition, the folder ‘menus’ contains the file ‘layEdit.menu’ in which the CADENCE and Silencer! menus and menu items are defined. The file cmosp35_SC.tf contains the modified technology file for the

TSMC 0.35µm process. Three more layers were previously added to the technology file, which are 'SC_Inj', 'SC_Sen', and 'SC_Chip'. Moreover, the file 'display.drf' contains the colors and patterns of these three layers. For any new process, these three layers have to be added. Further information about adding layers to a technology file and define patterns and colors can be found in Appendix B.1. The parameter file 'parameters.txt' contains all Silencer! parameters

```
// Last update of ~/cadence/SC_Tool/Silencer/parameters.txt on Dec  8 15:43:31 2003 by
birrer
// Model type (Macro-model1, Macro-model2, EPIC)
modelType=
     EPIC     Macro-model2
// Network type (SPECTRE, SPICE)
networkType=
     SPICE     SPICE
// Simulation (Layout-view, Extracted-view)
simView=
     Layout-view Layout-view
// Max. resistor in network discarded (t, nil), resistor values
maxRes=
     t     500000
// Grounded backplane (t, nil), R, L values
bpGnd=
     nil     0.2     1.5e-08
// Include extracted LVS information (t, nil)
flattenCV=
     nil
// Place network in schematic (t, nil)
inSchem=
     nil
endGeneral

// Parameters for macro-models
// Zii = a1, a2, a3
// Zij macro-model1 = b
// Zij macro-model2 < 12um = m1_A, m1_B, m1_C, m2_A, m2_B, m2_C, m3_A, m3_B, m3_C
// Zij macro-model2 = m4_A, m4_B, m5_A, m5_B, m5_C
// Zij macro-model2 > 12um = m1_Al, m1_Bl, m1_Cl, m2_Al, m2_Bl, m2_Cl, m3_Al, m3_Bl, m3_Cl
param_MM=
 3.201e+06   67.862   0.0004099
 154000
 758.521 17996.9 4.46326e+10 0.249083 -0.00579473 0.00161499 -0.0100331 -5.02842e-07
         52039.9
 -0.0210468 4605.18 -0.057346 0.00625791 -6.6367e-09
 343.591 8339.21 1.15093e+10 0.119094 -0.000151257 1.85493e-06 -0.0603908 -4.54979e-07
         31087.2
endMM

// Parameters for EPIC for 2 or 3 layer substrate
// numOfLay, DCT, contS
// layer1_res, layer1_th, layer2_res, layer2_th, layer3_res, layer3_th
// method, methTol
param_EPIC=
     3   4096   0.5
     0.0230956   194.598   4.12237   4.637   0.187458   0.765272
     1   1e-08   6
endEPIC
```

Figure B.1.5 Parameter file for the TSMC 0.35µm process.

The parameters are grouped in three sets: general Silencer! parameters, macro-model parameters, and EPIC parameters. The first line shows time, date, and username of the last update of the file. These parameters are stored in a data structure during Silencer! runtime. The data structure is described in Appendix A.5.

The file 'sncCC.exe' is the compiled, executable child-process that will be called from the SKILL parent process to extract a substrate network. The file 'Silencer.ini' contains the definitions of the substrate ports for the technology that Silencer! is being installed for, in this example for the TSMC 0.35µm process. Furthermore, the layers may have to be merged to get the correct substrate ports. More information about how to set up the technology file can be found in Appendix B.2. The file 'readme.txt' contains a brief description how to install Silencer! Further, it contains the paths to the different SKILL functions that have to be included in the file '.cds.ini' or '.cds.ini.user'.

```
loadi "Silencer/Skill/sncMain.il"
loadi "Silencer/Skill/sncPortData.il"
loadi "Silencer/Skill/sncMessage.il"
loadi "Silencer/Skill/sncModels.il"
loadi "Silencer/Skill/sncFiles.il"
loadi "Silencer/Skill/sncPlaceNW.il"
sncStartup()
```

Figure B.1.6 Paths to access the Silencer! SKILL functions.

After the 'Silencer' folder has been copied into the startup directory of the newly setup TSMC 0.35µm process and the paths added in the '.cds.ini' or '.cds.ini.user' file, CADENCE can be started from the startup directory by typing >icfb in the command line. After creating a new library and layout cell view, not only the three layers defined for Silencer!, but also the menus and menu items to run Silencer! will appear in the layout editor.

B.2 Adding the Noise-Injector and Noise_Sensor Layers

In Appendix B.2, there was a description of how Silencer! has to be installed for the TSMC 0.35µm process. For that particular process, the three new layers used by Silencer! have already been included in both, the technology file and the display.drf file. However, for a completely new process, these three layers have to be added first.

The three layers will be used not only to define the substrate ports (locations where noise gets injected to the substrate and where noise will be picked up through the substrate), but also a layer to define the chip boundary (used by the BE substrate network extractor). The layers have to be named 'SC_ Inj', 'SC_Sen', and 'SC_ Chip'.

In order to add layers into the CADENCE environment, one have to make the following changes in the technology file. First, the technology file 'cmosp35.tf' has to be copied from the folder 'cmosp35' into the startup directory (where the process has been set up). For the TSMC 0.35µm process example, the startup directory would is '~/cadence/SC_Tool'. Second, once the file has been copied, it has to be renamed, e.g., 'cmosp35_SC.tf'. Finally, the new technology file needs to be edited.

```
;*******************************
; LAYER DEFINITION
;*******************************
layerDefinitions(

techPurposes(
 ;( PurposeName               Purpose#   Abbreviation )
 ;( -----------               --------   ------------ )
 ;User-Defined Purposes:
 ;System-Reserved Purposes:
  ( warning                   234        wng          )
  ( tool1                     235        tl1          )
  ( tool0                     236        tl0          )
  ( label                     237        lbl          )
  ( flight                    238        flt          )
        .
        .
        .
techLayers(
 ;( LayerName                 Layer#     Abbreviation )
 ;( ---------                 ------     ------------ )
 ;User-Defined Layers:
  ( nwell                     1          nwell        )
  ( active                    3          active       )
  ( poly1                     5          poly1        )
        .
        .
        .
  ( drcex                     88         drcex        )
  ( pkg                       90         pkg          )
 ;Substrate port layers for substrate coupling analysis
  ( SC_Inj                    91         SC_Inj       )
  ( SC_Sen                    92         SC_Sen       )
  ( SC_Chip                   93         SC_Chip      )

 ;System-Reserved Layers:
  ( Unrouted                  200        Unroute      )
  ( Row                       201        Row          )
  ( Group                     202        Group        )
        .
        .
        .
techLayerPurposePriorities(
 ;layers are ordered from lowest to highest priority
 ;( LayerName                 Purpose    )
 ;( ---------                 -------    )
  ( background                drawing    )
  ( nwell                     drawing    )
  ( nwell                     net        )
```

```
  ( nwellres                  drawing    )
  ( nwellres                  net        )
  ( active                    drawing    )
  ( poly1                     drawing    )
        .
        .
        .
  ( m4res                     drawing    ) ;MOSIS add
  ( m4res                     net        ) ;MOSIS add
  ( SC_Inj                    drawing    ) ;Added substrate ports
  ( SC_Sen                    drawing    ) ;Added substrate ports
  ( SC_Chip                   drawing    ) ;Added substrate ports
  ( specres                   drawing    )
  ( tight                     drawing    )
        .
        .
        .
techDisplays(
        .
        .
        .
  ( m4res        drawing      met4R           t t t t t )   ;MOSIS add
  ( m4res        net          m4resNet        t t t t nil ) ;MOSIS add
  ( SC_Inj       drawing      SC_Inj          t t t t t )   ;Added substrate ports
  ( SC_Sen       drawing      SC_Sen          t t t t t )   ;Added substrate ports
  ( SC_Chip      drawing      SC_Chip         t t t t t )   ;Added substrate ports
  ( tight        drawing      tight           t t t t nil )
  ( analog       drawing      analog          t t t t nil )
```

Figure B.2.1 Code to define the three new added layers (highlighted in bold).

It has to be noted that the layer number ‘Layer#’ should not be the same as for any already used layer. In this case, one may use 91, 92, and 93.

These layers are now defined in the new technology file, however, they need some patterns and colors to be recognizable if they are displayed. In order to give them some patterns and colors, some changes in the file ‘display.drf’ have to be made. First, the display.drf file in the folder ‘cmosp35’ has to be copied into the startup directory.

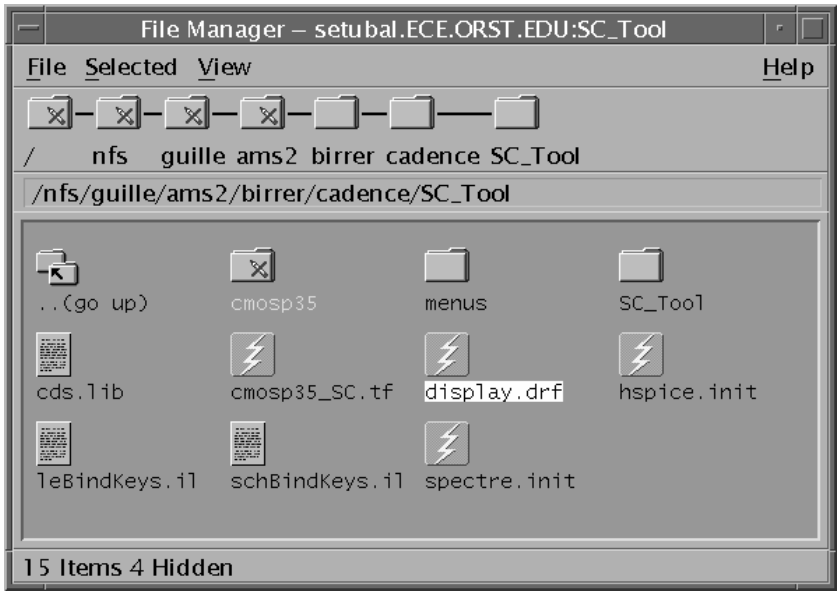

Figure B.2.2 The files currently in the startup directory.

```
drDefinePacket(
;(DisplayName   PacketName   Stipple   LineStyle   Fill   Outline    )
        .
        .
        .
( display        pplus    backslash        solid    gray     slate  )
( display        SC_Inj   contp            solid    cyan     cyan   ) ;Added substrate ports
( display        SC_Sen   contp            solid    green    green  ) ;Added substrate ports
( display        SC_Chip  contp            solid    purple   purple ) ;Added substrate ports
)
drDefineColor(
;( DisplayName   ColorsName   Red      Green    Blue     )
 ( psc           white        255      255      255      )
 ( psc           silver       217      230      255      )
 ( psc           cream        255      255      204      )
```

Figure B.2.3 Code (highlighted in bold) added to the 'display.drf' file (remark: we can define any color and pattern as we wish, e.g. cyan/dotted for injector, purple/dotted for sensor, etc.).

B.3 Loading the Newly Created Technology File

In order to use the modified technology file 'cmosp35_SC.tf' that includes the Silencer! layers for a new layout, it has to be loaded first. Once the CADENCE DFII environment is started up, a new library, e.g., 'SC_Tool' with the new technology file can be created.

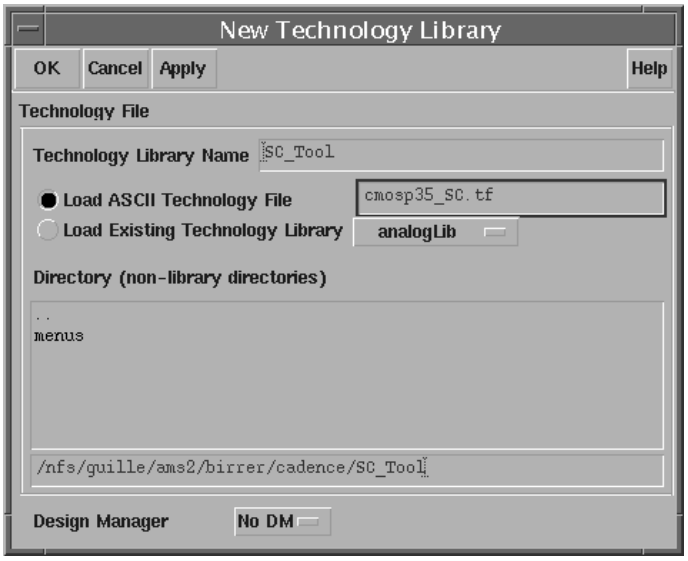

Figure B.3.1 Loading a technology file (CIW-window, Technology File / New…).

If a new layout cell view is created in the 'Library Manager', the new layers used for Silencer! appear in the LSW window.

Figure B.3.2 New layers in the LSW window.

In the 'Display Resource Editor', one can easily change the properties for the new created layers. For instance, the pattern of the injector/sensor layers can be changed by selecting the menu item 'Edit / Display Resource Editor…' in the LSW window.

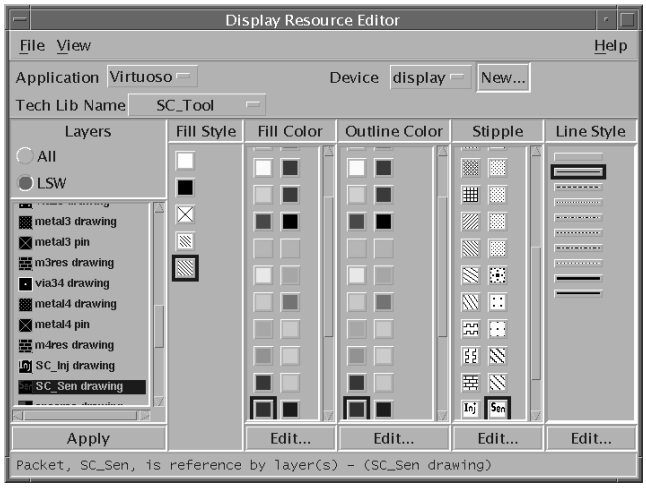

Figure B.3.3 The 'Display Resource Editor' window.

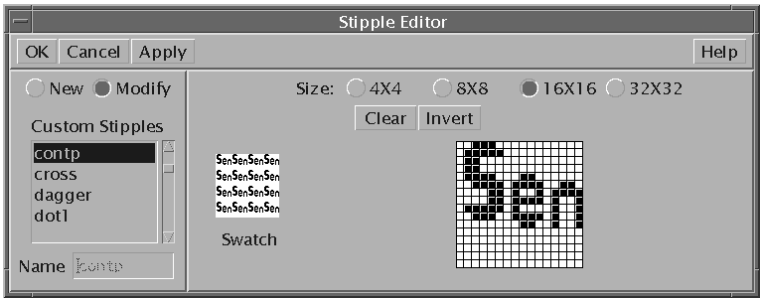

Figure B.3.4 The ‘Stripple Editor’ window. The changes, e.g., writing the pattern ‘Sen’, have to be saved in the ‘display.drf’ file in the startup directory, otherwise they will get lost!

B.4 The 'Silencer.ini' File

The file 'Silencer.ini' has to be set up for any new technology to define the substrate ports in that technology. Substrate ports are locations in the layout where substrate noise gets injected or picked up. For example, two important substrate ports in the TSMC 0.35µm heavily doped process for low frequencies are p-plus diffusion regions (p-taps) and the NMOS junction capacitances. However, depending on the layout, more substrate ports may have to be included, such as n-plus diffusion regions inside n-wells, fluctuating interconnects in large digital circuits, etc.

```
;--------------------------------------------------------------------
; Setup file for substrate coupling analysis tool version 1.2
;--------------------------------------------------------------------

;--------------------------------------------------------------------
; Substrate port definitions (all different port types are listed
; here. Remark: The FIRST line defines the p-substrate-taps )
; portName (max. 8 characters) layer1 OVERLAPS layer2 OUTSIDE layer3
;--------------------------------------------------------------------
ports=
        ptap_           active          pplus           nwell
        nblk_           active          nplus           nwell
end
;--------------------------------------------------------------------
; All layers that need to be merged before a substrate port is de-
; fined must be listed here, e.g. p-cells or user defined subcells
;--------------------------------------------------------------------
cellsLayMerge=
        active
        nplus
        pplus
end
```

Figure B.4.1 The file 'Silencer.ini' for the for the TSMC 0.35µm process.

The p-taps in that process are used to ground the substrate. However, due to the non-ideal ground (resistive metal traces, package line inductances), the voltage potential at these p-taps is bouncing, which causes noise coupling into the substrate. P-taps have to be defined in the 'Silencer.ini' file as the first substrate port between the key-words 'ports=' and 'end'[3]. P-taps for the TSMC 0.35µm heavily doped process are locations where an 'active' layers and a 'pplus' layers overlap, but are not located inside an n-well. If Silencer! locates such ports, it will label them with 'ptap_' (unless a label is already inside the port bounding box). The name can be defined by the user, but should not exceed eight characters.

Similarly to the p-taps, the ports to the NMOS junction capacitances can be defined. Substrate noise can get picked up through these capacitances (change in the transistor threshold voltage). Furthermore, switching circuits will inject substrate noise due to charge and discharge of the junction capacitances. These ports are recognized in the TSMC 0.35µm process as 'active' layers overlapping with an 'nplus' layers, but not located inside an n-well.

As another example, if there was a substrate port underneath each gate region of an NMOS transistor, the following line could be added to the Silencer.ini file:

```
ports=
        .
        .
        .
        nblk_          active          poly1          nwell
end
```

[3] Silencer! assumes that the first line in the 'Silencer.ini' file defines the p-taps. It adds them in a list that can be used to define the interconnect routing late on, after the substrate ports have been located.

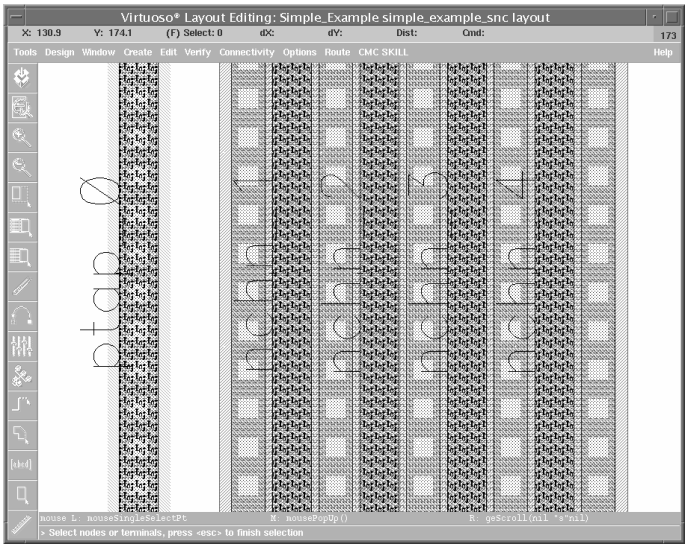

Figure B.4.2 An NMOS transistor with four fingers for which Silencer! recognized four substrate ports underneath the gate regions.

The layers in the 'Silencer.ini' file between the key-words 'cellsLayMerge=' and 'end' define all the layers that will be merged before Silencer! locates the substrate ports. Depending on the layout, transistor and taps may be defined as parameterized cells (p-cells) and consist of not only one layer, but several small pieces of layers touching each other. In such a case, Silencer! would recognize each piece as an individual substrate port, which is incorrect and would at least for some substrate network extractors cause an inaccurate result. For that reason, layers that are defined as substrate ports, may be listed (for the TSMC 0.35µm process these layers would be 'active', 'nplus', and 'pplus') in the file to be merged before any ports will be located.

APPENDIX C - Example for Macro-Model1 Implementation

The following example shows how macro-model1 was implemented into Silencer! for n=3 contacts c1, c2, and c3:

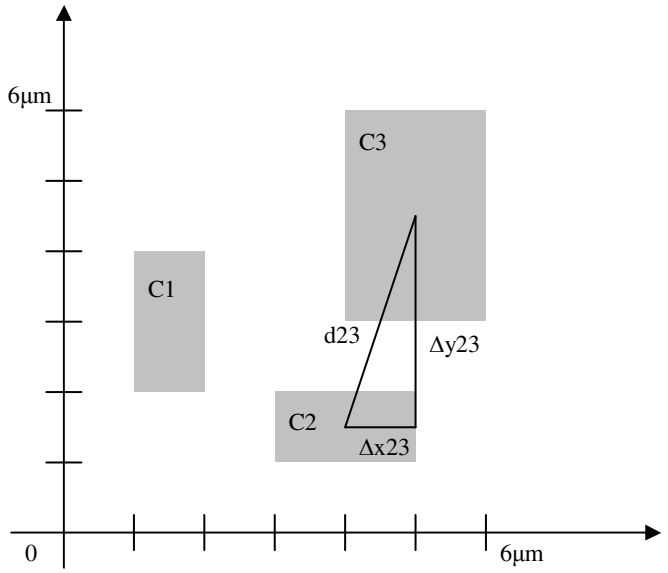

Figure C.1 Macro-model1 implementation example for three contacts.

The coordinates of c1, c2, and c3 (rows) are defined as left, right, top, bottom (columns) in the following matrix:

coord = 1e-6 *

1	2	4	2
3	5	2	1
4	6	6	3

From the coordinates, Side lengths x / y and center points x / y of c1, c2, and c3 (rows) are:

cont_size = 1e-6 *

1	2
2	1
2	3

cont_center = 1e-6 *

1.5000	3.0000
4.0000	1.5000
5.0000	4.5000

The distances between the contact center points can be calculated (using Pythagoras) and saved in a n x n matrix:

$$\Delta xij = \left(cxj - cxi\right)$$
$$\Delta yij = \left(cyj - cyi\right)$$
$$dij = \sqrt{\Delta xij^2 + \Delta yij^2}$$

For this example, the distance matrix has the following entries (center point to center point):

cont_dist = 1e-6 *

0.0000	2.9155	3.8079
2.9155	0.0000	3.1623
3.8079	3.1623	0.0000

Before calculating the merged perimeter of two contacts, the following matrix for the contact orientation is needed:

cont_orit = 1e-6 *

0	1	2
1	0	1
2	1	0

This matrix has been filled according to the following rules: If the side y of contact ci is smaller than the side y of contact cj, then cont_orit(ij) is the side y of ci, otherwise it is the side y of cj. In other words, the smaller side y of the two contacts ci and cj will be stored in the matrix. The diagonals will be zero.

The area of two merged contacts can be calculated simply by adding the areas of two contacts ci and cj. The diagonal of the matrix contains the area of the contact itself.

merged_area = 1.0e-011 *

0.2000	0.4000	0.8000
0.4000	0.2000	0.8000
0.8000	0.8000	0.6000

The merged perimeter for each contact c1, c2, and c3 (rows) needs to be calculated next. Using the equation

$$merged\_peri(ij) = perimeter(ci) + perimeter(cj) - cont\_orit(ij),$$

this matrix will be:

merged_peri = 1e-6 *

6	10	12
10	6	14
12	14	10

This calculation may seem a bit confusing, but it is not. The way the perimeters of two contacts are merged is quite simple: First, the sum of the perimeters of both contacts is calculated, and then the shortest side y of either contact ci or cj will be subtracted twice. The diagonal contains perimeter of the contact itself.

Before the z-parameters can be determined, a matrix for the alpha-parameters has to be calculated using the equation:

$$a\_par(ij) = \frac{1}{a1 \cdot merged\_area(ij) + a2 \cdot merged\_peri(ij) + a3}$$

a_par= 1.0e+003 *

0.0000	2.0423	1.6732
0.0000	0.0000	1.4432
0.0000	0.0000	0.0000

These parameters are used to calculate the z-parameters zii and zij using the following equations:

$$zii = \frac{1}{a1 \cdot merged\_area(ii) + a2 \cdot merged\_peri(ii) + a3}$$

$$zij = a\_par(ij) \cdot e^{-b \cdot cont\_dist(ij)}$$

z_par = 1.0e+003 *

3.4154	1.5258	1.1434
1.5258	3.4154	1.0520
1.1434	1.0520	2.0160

The y-parameter matrix can be calculated by inverting the z-parameter matrix:

y_par = 1.0e-003 *

0.4056	-0.1315	-0.1614
-0.1315	0.3915	-0.1297
-0.1614	-0.1297	0.6553

Finally, using the y-parameter matrix, the resistor matrix can be calculated the following way:

$$rii = \frac{1}{yi1 + yi2 + ... + yin}$$

$$rij = -\frac{1}{yij}$$

r_matrix = 1.0e+003 *

8.8742	7.6064	6.1955
7.6064	7.6749	7.7090
6.1955	7.7090	2.7463

APPENDIX D - Including a New Menu Item into CADENCE

To include a menu item into the CADENCE Virtuoso™ layout editor, the process has to be set up first (create a new project directory). First, a new folder ‘menus’ has to be created in that project directory.

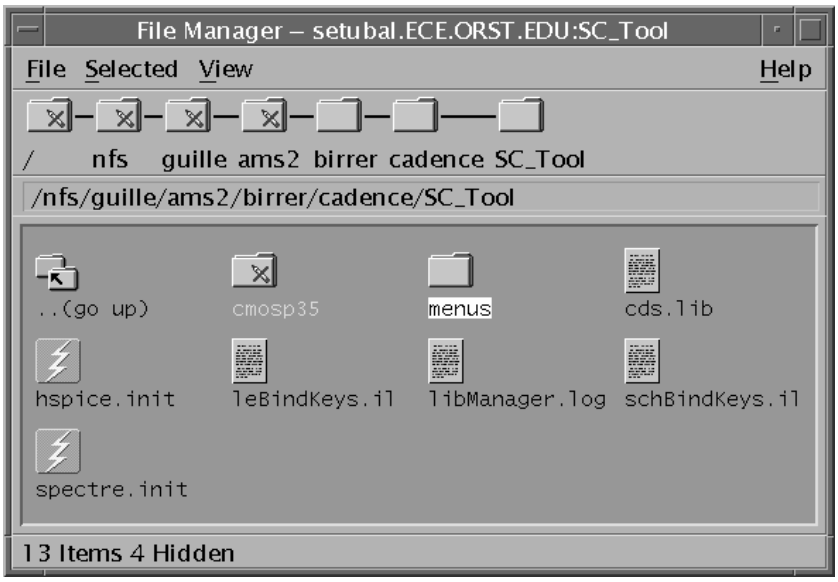

Figure D.1 The SC_tool project directory and the folder created folder ‘menus’.

Second, the file ‘layEdit.menu’ has to be copied into the folder ‘menus’. If a folder ‘menus’ is present anytime CADENCE gets started, this file will be executed. Depending on the CADENCE version and setup, this file is available in a specific CADENCE directory. For the current setup at Oregon State University, the path would be:

/nfs/guile/a1/sunapps/cadence/IC446/tools.sun4v/dfII/etc/tools/menus

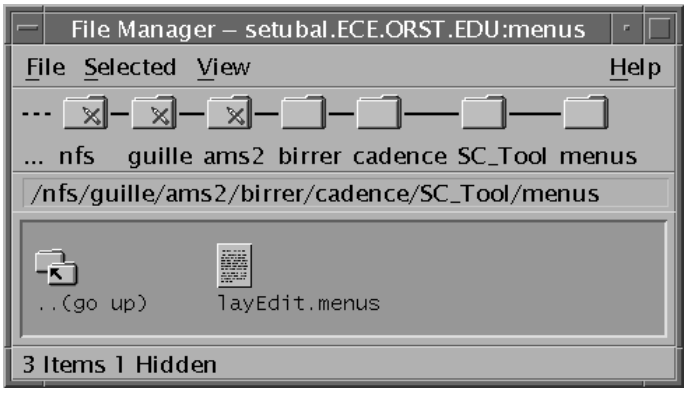

Figure D.2 The file 'layEdit.menu' copied into the folder 'menus'.

Finally, changes in the file 'layEdit.menu' have to be made in order to define any new menu items. Menus for the layout editor can very conveniently be customized. Specifically, new menu items, menu texts and callback functions can be defined, modified, and hierarchically organized without having to rebuild the executable or SKILL context in order to see the change.

D.1 Changing the Text of a Pulldown Menu Item

For example, the text of the pulldown menu item 'Save' can be changed by replacing the 'Save' with 'Hello Menu':

```
lecDesignSaveItem =    '(SaveItem
                       "Save"
                       "geSave()")
```

```
lecDesignSaveItem =    '(SaveItem
                       "Hello Menu"
                       "geSave()")
```

D.2 Callback Function of a Pulldown Menu Item

The callback function of a pulldown menu item is usually a SKILL function. For the 'Save' item, the SKILL function 'geSave()' will be called after the user has selected the 'Save' item (e.g. per mouse click). Further, the SKILL

function can be user defined. Thus, the user can customize what happens off a menu item.

D.3 Adding a New Menu Item to an Existing Pulldown Menu

In order to implement a new menu item, the following code needs to be added in the file 'layEdit.menu':

```
lecMenuTypeItem =      '(itemSymbol
                       "itemText"
                       "callback()")
```

'lecMenuTypeItem' consists of 'itemSymbol', 'itemText', and 'callback()' function. The 'itemSymbol' is the symbol (a symbol in SKILL is like a variable) to use when creating the menu item. The 'itemText' is the text to appear in the menu and 'callback()' is the string containing the callback function invoked when the user selects the menu item.

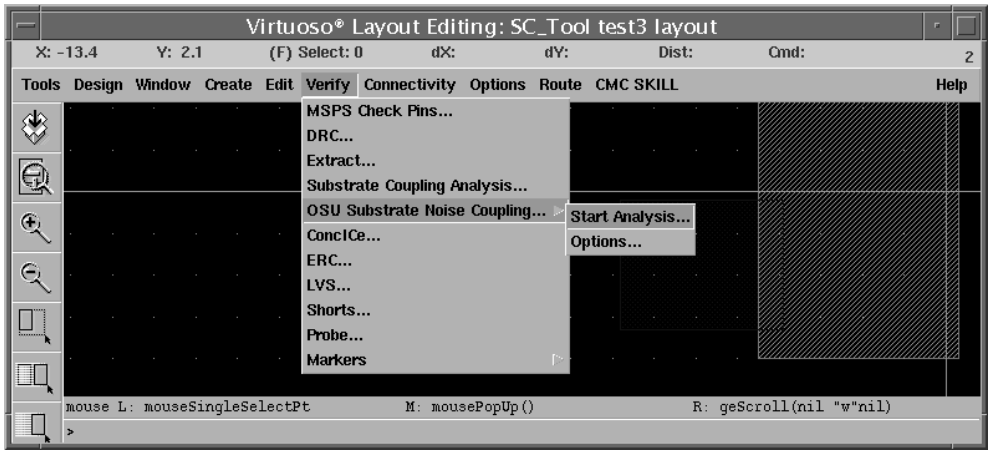

Figure D.3 adding the menu items 'Start Analysis…' and 'Options…' to the menu 'Verify'.

For example, in order to add the menu items ‘Start Analysis…’ and ‘Options…’ to the menu ‘Verify’ - illustrated in the figure above - the following code needs to be added:

```
lecVerifyCouplingItem =       '(VerifyCouplingItem
                              "Start Analysis..."
                              "printf(\"Start Substrate Coupling callback
                              function\n\")")

lecVerifyCouplingOptionsItem =        '(VerifyCouplingOptionsItem
                                      "Options..."
                                      "printf(\"Menu for Substrate Coupling
                                      OPTIONS\n\")")
```

The items ‘Start Analysis…’ and ‘Options…’ are defined as a pulldown list. Such a list can be defined by adding the following code:

```
pulldownList = '(symbol "name" (
               item
               item
               ...
               ))
```

The variable ‘symbol’ defines the pulldown list and ‘name’ is the title of the pulldown list. An ‘item’ can be another pulldown list, a slider item, etc. In the example, ‘item’ is the slider item ‘OSU Substrate Noise Coupling…’ used to select ‘Start Analysis…’ and ‘Options…’. The code for the slider item is:

```
lecVerifyCouplingSliderItem = '(CouplingSliderItem "OSU Substrate Noise
Coupling..."   (
    lecVerifyCouplingItem
    lecVerifyCouplingOptionsItem
))
```

The ‘pulldownList’ for the ‘Verify’ menu with all listed items in the order as required looks like the following:

```
lecVerifyMenu = '(leVerifyMenu "Verify" (
    lecVerifyDRCItem
    lecVerifyExtractItem
    lecVerifyCouplingSliderItem
    lecVerifyConcICeItem
    lecVerifyERCItem
```

```
    lecVerifyLVSItem
    lecVerifyShortsItem
    lecVerifyProbeItem
    lecVerifyMarkersSliderItem
))
```

Further, the ‘pulldownList’ for the slider item can now be implemented the following way:

```
lecLayoutOnlyVerifyMenu = '(leLayoutOnlyVerifyMenu "Verify" (
    lecVerifyCouplingSliderItem
    lecVerifyDRCItem
    lecVerifyMarkersSliderItem
))
```

All the changes made in the file layEdit.menu for that example are listed below and marked in bold:

```
;*****************************************************************************
; layEdit.menus - Copyright (C) 1997 Cadence Design Systems, Inc.
;                          All Rights Reserved.
;
        .
        .
        .
;*****************************************************************************
; Design Menu Items
;*****************************************************************************

lecDesignSaveItem = '(SaveItem
                     "Save"
                     "geSave()")
        .
        .
        .
;*****************************************************************************
; Common Verify Menu Items
;*****************************************************************************

lecVerifyDRCItem = '(DRCItem
                   "DRC..."
                   "ivHiDRC()")

lecVerifyExtractItem = '(ExtractItem
                        "Extract..."
                        "ivHiExtract()")

lecVerifyCouplingItem = '(VerifyCouplingItem
                        "Start Analysis..."
                        "printf(\"Start Substrate Coupling callback function\n\")")


lecVerifyCouplingOptionsItem = '(VerifyCouplingOptionsItem
                        "Options..."
                        "printf(\"Menu  for  Substrate  Coupling  OPTIONS  will  be  implemented
here\n\")")

lecVerifyCouplingSliderItem = '(CouplingSliderItem "OSU Substrate Noise Coupling..." (
    lecVerifyCouplingItem
    lecVerifyCouplingOptionsItem
))

lecVerifyConcICeItem = '(ConcICeItem
```

```
                        "ConcICe..."
                        "ivHiConcICe()")
        .
        .
        .
;*****************************************************************************
; Define the rest of the pulldowns for layout windows.
;*****************************************************************************

lecVerifyMenu = '(leVerifyMenu "Verify" (
    lecVerifyDRCItem
    lecVerifyExtractItem
    lecVerifyCouplingSliderItem
    lecVerifyConcICeItem
    lecVerifyERCItem
    lecVerifyLVSItem
    lecVerifyShortsItem
    lecVerifyProbeItem
    lecVerifyMarkersSliderItem
))

lecLayoutOnlyVerifyMenu = '(leLayoutOnlyVerifyMenu "Verify" (
    lecVerifyCouplingSliderItem
    lecVerifyDRCItem
    lecVerifyMarkersSliderItem
))
        .
        .
        .
;*****************************************************************************
; Define list of all pulldowns, used to get them built by ciw code.
;*****************************************************************************

lecAllPulldownMenus = '(
    lecDesignMenuR
    lecDesignMenuW
    lecDesignMenuV
    lecWindowMenu
    lecCreateMenu
    lecEditMenu
    lecLayoutOnlyVerifyMenu
    lecVerifyMenu
    lecMiscMenu
    lecMiscMenuV
    lecUwCreateMenu
    lecConnMenu
    lecOptionsMenu
    lecRouteMenu
    lecHelpMenu
)
        .
        .
        .
```