A TOP-DOWN, CONSTRAINT-DRIVEN DESIGN METHODOLOGY FOR ANALOG INTEGRATED CIRCUITS

by

Henry Chung-herng Chang

Memorandum No. UCB/ERL M95/21

3 April 1995

A TOP-DOWN, CONSTRAINT-DRIVEN DESIGN METHODOLOGY FOR ANALOG INTEGRATED CIRCUITS

by

Henry Chung-herng Chang

Memorandum No. UCB/ERL M95/21

3 April 1995

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits

Abstract

A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits

by

Henry Chung-herng Chang
Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences
University of California at Berkeley
Professor Alberto Sangiovanni-Vincentelli, Chair

Analog circuit design is often the bottleneck when designing mixed analog-digital systems. To solve this problem, we propose a new design methodology based on a top-down, constraint-driven design paradigm. This methodology has two principal advantages: (1) it provides a high probability for first silicon which meets all specifications, and (2) it shortens the design cycle. To validate these claims, we have designed, fabricated, and tested several current source digital-to-analog (D/A) converters and a Σ-Δ analog-to-digital converter following the methodology. Currently, a video driver system consisting of a frequency synthesizing phase-locked loop and three video D/A converters is being designed. Presented in this dissertation will be: (1) a historical perspective on analog design methodologies with an introduction to analog design issues, (2) the design methodology and the accompanying tool set, and (3) our three design examples.

Prof. Alberto Sangiovanni-Vincentelli
Thesis Committee Chairman

Contents

List of Figures

List of Tables

Acknowledgements

The Ph.D. research program is a long and often arduous process. I have had the great fortune of having excellent mentors to guide me through it. Professor Alberto Sangiovanni-Vincentelli, my principal research advisor, with his keen research and business insight showed me the importance of understanding the "big" picture and always having a clear goal in mind. I will always appreciate the hours of time and effort he spent improving my presentation slides and style before conferences. While most students have only one research advisor, I was fortunate to also have Professor Paul Gray to guide me. With his honest and straightforward approach, he taught me the importance of quality and attention to detail in circuit design. This philosophy greatly influenced my research. I would also like to thank Professor Bob Brayton for serving on my qualifying examination committee, and Professor Ian Adler for serving on both my qualifying examination committee and my thesis committee.

Many people have assisted me during my studies. For help with circuit design from bias circuits to A/D converters, I would like to thank Professor Bernhard Boser, Cormac Conroy, Dave Cline, Thomas Cho, Tim Hu, Ken Nishimura, and Greg Uehara. I am grateful to all of the people in the Berkeley CADgroup for providing me assistance: Andrea Casotto, for maintaining the OCTTOOLS; Edoardo Charbon, for providing quick bug fixes for the layout tools; Alper Demir, for fruitful discussions on PLL behavioral simulation; Eric Felt, for development help with MINOS, the A/D, and D/A projects; Gani Jusuf, for providing CADICS examples; Brian Lee, for his quick response when problems arose in our optimization package; Edward Liu, for excellent debates on the design methodology and D/A behavioral simulation; Enrico Malavasi, for prompt layout tool bug fixes as well as interesting business discussions; Robert Neff, for design insight and the layout generators for the D/A project; and Iasson Vassiliou, for his hard work on the video driver project. One tremendous source of learning has been from working on group projects and from studying in groups. I would like to thank Ting Kao, Diane Hoffstetter, and Kristen McNair for being excellent partners. Without the staff at EECS, research would come to a grinding halt. I would like to thank Brad Krebs, Mike Kiernan, and Judd Reiffin for vigilantly watching and improving our computer environment; Elise Mills, for all of her assistance during our MOSIS purchasing order crises; Tim Castro and Irena Stanczyk-Ng, for always making certain that I would be paid; and Flora Oviedo, for cutting through a lot of red tape and pushing the paperwork through for our reimbursements. For industrial perspectives, I would like to thank Ian Young, my Intel Fellowship Mentor, for informative discussions on campus and for setting up appointments for me to talk with Intel employees in Santa Clara; and Micro Linear, for providing design examples. I would like to give special thanks to Professor Harvey Silverman for providing me with research experience at Brown University and for pulling me aside to re-enforce the idea that I should attend graduate school. Finally, I would like to thank Mark Beardslee, Justin Chueh, Gary Jones, Desmond Kirkpatrick, Sherry Lee, Christopher Lennard, Mark Noworolski, Ravi Subramanian, Jeff Rothman, Henry Sheng, and Dawn Tilbury for interesting and useful discussions along the way.

Finally, I would like to thank my parents and my sister, Kathy, for their support and encouragement. Special thanks to my wife, Pora, for her understanding and enthusiasm for my work.

This research has been partially supported by The National Science Foundation, the Intel Foundation, SRC (93-DC-008), MICRO, and DARPA. This support is gratefully acknowledged.

Chapter 1

Introduction

Traditionally, integrated circuit design has been classified into two categories–digital and analog. While microprocessors and microcontrollers are inherently digital components, certain functions can be realized in analog or digital form. A typical example is signal processing. Signals can be manipulated as waveforms, or they can be first encoded and then manipulated in the digital domain. Noise is often the limiting factor in the quality of analog signal processing. In the digital domain, noise has much less influence. Hence there has been a strong trend towards moving computation carried out in the analog domain to its digital counterpart. However, there are still functions that have to be carried out in the analog domain. An example is signal conversion (the "real" world is analog!). In addition, there are computations that are still much more efficient in the analog domain. An example is filtering in an analog signal path (the signal conversion overhead is too expensive). Because of the continuous quest towards smaller and smaller electronic systems, many integrated circuits being designed today are mixed analog-digital. Hence, more and more chips depend upon the ability to design effectively analog components.

Because of its noise sensitivity, analog design is inherently time consuming. In mixed digital-analog systems, noise injected from digital circuits further complicates analog design. Thus, in mixed-signal systems where the analog circuits are small, analog design is often a bottleneck. Unlike its digital counterpart, fully automatic synthesis tools are not available. Several attempts have been made to speed up the process by automating parts of the design such as the synthesis of modules and the automatic layout of often used components. Methods for low-level physical layout generation and schematic synthesis using architectural selection paradigms have been proposed [27][90][66][38]. Methods for very specific mid-to-low level complexity circuits have also been proposed [50][103][46]. More recently automatic support tools for the experienced designer have been developed [71][45].

Described in this dissertation is a new design methodology and the necessary set of tools that supports it for effective analog design. It has two basic goals: (1) increasing the probability for first silicon which meets all specifications; and (2) shortening the design cycle. The key points of this methodology are:

- Top-down hierarchical process starting from the behavioral level based on early verification and constraint propagation;
- Bottom-up accurate extraction and verification;
- Automatic and interactive synthesis of components with specification constraint-driven layout design tools;
- Maximum support for automatic synthesis tools to accommodate users of different levels of expertise but not the enforcement of these tools upon the user; this is not an automatic synthesis process;
- And consideration for testability at all stages of the design.

The top-down process implies a well-defined behavioral description of the analog function. The behavioral characterization of analog circuits is quite different from the digital one; analog characterization is composed of not only the function that the circuit is to perform, but also the second order non-idealities intrinsic to analog operation. In fact, errors in the design often stem from the non-ideal behavior of the analog section, not from the selection of the "wrong" functionality. To shorten the design cycle, it is essential that design problems be discovered as early as possible. For this reason, behavioral simulation is an essential component of any methodology. This simulation can help in selecting the correct architecture to implement the analog function with bounds (constraints) on the amount of non-idealities that is allowable given a set of specifications at the system level.

Constraints on performances of the selected architecture are propagated down to the next level of the hierarchy onto the components that can be designed following the same paradigm until all the leaves of the design space are reached. These leaves can either be transistors, other atomic components, or library objects. Since models have to be estimated at high levels in the hierarchy, a bottom-up verification is also essential to fully characterize components, interconnects, and parasitics.

The physical assembly of basic blocks at all levels of the hierarchy is time-consuming and rarely very creative. This step can be effectively accomplished with automatic synthesis tools. Recognized is that the layout parasitics do affect the behavior of analog circuits and as such have to be controlled carefully. The amount of parasitics allowed on the interconnects is often estimated by the designer who usually is not able to guarantee the accuracy of the estimation when fitted with the final circuit. Often to guarantee proper functionality, designers will overconstrain the allowed parasitics. Proposed here is an approach where layout tools are directly driven by constraints on performances of the design components.

The testing of analog circuits requires a great deal of time as well as expensive equipment. This problem increases with the complexity of the circuits. It can be solved in part by taking into account the testing problem during all stages of the design, unlike the common practice of considering testing only after the design is finished.

Finally, it is not believed that full automation is achievable for all analog circuits. The amount of creativity and complexity needed to master the design of analog circuits is high. It is believed, however, that the creative task of the designer can and should be fully supported by a set of automatic and interactive tools that allow him/her to explore the design space with ease and full understanding of the trade-offs involved. Analytical tools play an important role in our methodology. It is also maintained, though, that some components of an analog design could indeed come from module generators and some from libraries both of which embody the experiences of other designers. Thus, this methodology does accommodate tools that favor design-reusability in its general framework.

This work is part of an on going research effort at the University of California at Berkeley in the Electrical Engineering and Computer Sciences Department. Many faculty and students, past and present, are working on this design methodology and its supporting tools. My research has principally consisted of: (1) developing the design methodology, (2) applying optimization tools, (3) testing the methodology using "industrial strength" design example, and (4) developing the tools that are discovered to be missing while testing the methodology. My work follows the module generation work that has been completed here. These works contain the basic elements of this methodology, but not its complete development. My successor will share my research focus. He will continue developing the design methodology while finishing and expanding the design examples. I believe that this dissertation is not a beginning nor an end in the development of a top-down, constraint-driven design methodology, but rather one chapter in a book on this topic.

This dissertation is divided into three parts. First, Chapter 2 presents foundation material. Next, Chapters 3 and 4 describe the methodology and its supporting tools. Finally, Chapters 5, 6, and 7, contain the three large scale design examples: the current source D/A converters, the Σ-Δ A/D converter, and the video driver system. This is followed by a concluding chapter in which a summary and future works are provided. Some of this work has already been presented in [12] and [11].

Chapter 2

Foundation

2.1 Design Methodologies

Many methodologies exist for analog integrated circuit design. In any design, a methodology is either explicitly stated or implied. On the surface most design paradigms are similar. All follow the general design flow illustrated in Figure 2.1.

2.1.1 Analog Design

The designer begins with a set of specifications. These have either been provided by a customer, or these are the requirements for proper operation in a larger system. Specifications generally include not only the functional and performance criteria for the product, but also data on the target process and an overall engineering objective. For example, minimizing production cost is often the objective. This could translate to minimizing area which maximizes yield which in turn minimizes cost.

Design synthesis takes two inputs: (1) the specifications and (2) an architectural library. The output is a schematic which contains not only a netlist of the basic circuits elements, transistors, resistors, capacitors, etc., but also their parametric values. The architectures in the library are either templates for the schematic, or they are starting points from which new architectures can be developed.

Analog "hand design" design up to now has been considered an art. Rigid methods based on automatic synthesis have yet to be used in practice. Most of the tools commonly used are analysis tools for the evaluation of designs. In addition, "high level" analysis is done

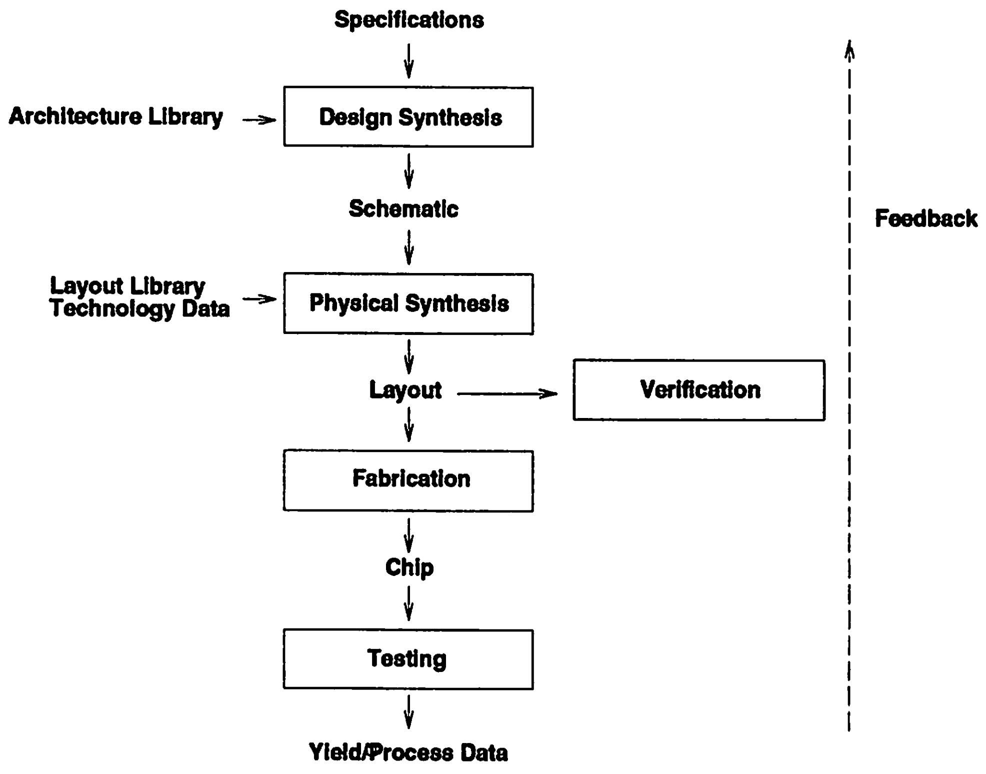

Figure 2.1: General Design Flow

on an *ad hoc* basis since a formal method for abstraction is missing. High level models are not "precise" enough for analog design where second order effects are important. Hence circuit simulators such as SPICE and prototyping on bread boards [35] is almost exclusively used for the determination of these effects. Unverifiable top-down decomposition has resulted in an unsystematic bottom-up design style. Designers overdesign low-level components to compensate for unpredicted nonidealities. This overdesign is time consuming. Bottom-up designs also imply that subsystems cannot be verified until all of their components have been designed. This results in further time consuming iterations when errors are detected. When verifying the system, circuit simulation is often too CPU expensive or infeasible. This results in systems which are not fully verified until testing.

Physical synthesis is the next phase. It takes as inputs: (1) the schematics, (2) a layout library, and (3) data on the technology. The output for this step is layout. Varieties of methods exist for layout generation. Layouts can be copied from a layout library. Layouts can be modified from an existing library entry. Layouts can be generated from scratch either manually or automatically. Although almost exclusively, designers resort to manual techniques. Automatic synthesis techniques have found little acceptance.

The next step is to verify that the layout meets specifications. Often because of the sizes of the circuits involved, verification cannot be accomplished in a single step nor by any single tool. A variety of methods have been developed. One method is simulation. In circuit simulation sets of test vectors are supplied to the system and simulated to verify functionality. Behavioral simulation verifies performances directly. However, with the use of simulation detection of differences between the schematic and the layout is not be guaranteed. An approach to solve this problem is to compare directly the schematic and the layout, element by element, connection by connection. This verification method, however, cannot find performance degradations due to layout parasitics as simulations can. A combination of the two methods is usually used.

Once the layout has passed verification, it is sent for fabrication. This process typically requires two to six weeks. When the chips are received, the parts are tested against the specifications for function and for performance. Typically, yield and process information are also gathered from the parts to determine the actual cost of the product and to collect data to improve future designs. Though rigorous methods for testing have been developed [74][73], they are also not used in practice.

Feedback is implied at any step in the design flow. At any time an output fails to meet the specifications, design steps must be re-iterated. Feedback loops can be small or large depending on how much redesign is required.

Many extensions exist to the general design flow. One extension is the use of hierarchy. This is illustrated for the design and physical synthesis phases in Figure 2.14. Systems which are too large to design as one individual unit are divided into subsystems. These smaller subsystems can then be designed independently or if they are still too large, can be further subdivided. The interactions among the subsystems are approximated to allow a certain independence of design.

In the design synthesis phase, not only is the architecture subdivided, but also the system specifications are decomposed into specific specifications for each sub-block. This process of decomposition is often defined as the "top-down" design phase. Its counterpart, the "bottom-up" design phase refers to the sub-block design process in which schematics are generated. In analog design, the top-down phase is often very difficult. Many designs are considered "bottom-up designs," because the emphasis has been placed on this step. Circuit verification can also be accomplished hierarchically. Sub-blocks are simulated and verified. They are replaced with abstracted models which characterize their function and/or performances. These models are combined and simulated to verify the overall system function and/or performance. Layouts are generated hierarchically as well. First, layout requirements for each sub-block are generated in a floorplanning step, e.g. aspect ratio, bus and power line placement, pad placement. After the sub-blocks have been generated, they are combined level by level until the layout for the system is complete. Once again, feedback is implied for all of the design steps.

Though most methodologies do follow the design flow in Figure 2.1 and consider the use of hierarchy, they differ in the details.

2.1.2 Analog Computer Aided Design

A great deal of research has focussed on solving the problems faced in analog design. Most research has emphasized specific tools and/or sets of tools. In all cases an underlying methodology is implied, but seldom discussed. Furthermore, rarely is the entire design flow considered. Many have made claims for automatic synthesis of large systems, but none have fabricated and tested these designs. Our emphasis is to explicitly develop the methodology. Then develop the tools to support it. And furthermore, we will consider the entire design flow, "proving" the methodology using industrial strength design examples.

Early work in analog design synthesis focussed solely on sizing schematics. Examples are APLSTAP [4], DELIGHT.SPICE [83], and ECSTASY [95]. These systems combined nonlinear optimization algorithms with circuit simulation.

Work which followed wrapped schematic sizing with an architectural selection phase, but because of the high CPU cost and the numerical instabilities associated with SPICE, these newer systems avoided SPICE simulation in the design loop. Knowledge based methods were developed. Complexity-wise, research was primarily focussed on operational transconductance amplifiers (OTA). IDAC [26] synthesized circuits by sizing each schematic in its architectural library using dedicated built-in analyzers for simulation. These analyzers consisted of simple equations representing circuit performances. OASYS [42] added hierarchy, macromodels, and an expert system for schematic selection and sizing. The expert system evolved from an *ad hoc* rule-based system to a strict plan-based system. One shortcoming of the IDAC and the OASYS systems was that the models they used were too simple.

OPASYN [52], a module generator for OTA's, addressed this problem by using much more accurate nonlinear equations to model circuit behavior. A nonlinear unconstrained steepest descent approach was used for schematic sizing.

STAIC [45], which followed IDAC, OASYS, and OPASYN, sought to address one of the main problems with these knowledge-based system, i.e. for each architecture in its library, a specific circuit simulator was required. It tried to combine the best of both worlds centering itself between algorithmic based methods and knowledge-based methods. It used multi-level macro-models so that different architectures could share simulation models. This simplified the development of new simulators for new architectures. To further simplify model development, STAIC incorporates a hardware description language. A successive solution refinement system is used to find the global optimum. General macro-models are used initially. When an optimal solution is found, more detailed models replace the general models, and another optimization is done. Examples illustrate automatic synthesis of BiCMOS and CMOS operational amplifiers from specifications.

During this time, work in automatic custom analog cell layout was also pursued. ILAC [87], IDAC's counterpart, takes as input a technology description, a circuit description file, and optional topological constraints description and generates layout. Leaf cells are generated by using one of two procedural languages. Block level layout is generated using placement (slicing structures), routing (global and channel), and compaction steps. ICEWATER [45], STAIC's counterpart, also uses a procedural language for layout generation. ANAGRAM [36], OASYS's counterpart, strived for the generic layout of leaf cells. Rather than using a procedural language, simulated annealing was used for placement and an area router was used. OPASYN had a built in layout generator. Like OASYS, OPASYN focussed on leaf cell layout. It used slicing structures for placement, an area router, and incorporated active constraints in the compaction step.

The next generation of work focussed on complete design systems combining the design and physical synthesis phases. They also aimed for higher circuit complexity. The IDAC/ILAC system [27] included hierarchy to address the complexity issue. However, they had no high level simulator to aid in mapping. A sigma-delta A/D converter was synthesized claiming that it requires a "trivial" mapping. However, acknowledged is that they do not have a general hierarchy mapping technique, and they cite this as an open question.

The OASYS/ANAGRAM system evolved into the ASTRX/OBLX (design synthesis)-KOAN/ANAGRAM III (physical synthesis) [90][84] system. A simulated annealing optimizer and an Asymptotic Waveform Evaluation (AWE) simulator replaced the knowledge-based system to create a fully algorithmic system like DELIGHT.SPICE. It has the advantage that AWE is much faster than SPICE, and simulated annealing can avoid local minima. Since unlike SPICE, AWE only finds the "AC" solution to a circuit, dc correctness terms with a relaxed dc-formulation and a dynamic cost function algorithm were added to satisfy Kirchoff's laws. Once again, they strived for large circuits. In [84], they claim to "have presented the largest, most complex fully automatic analog cell synthesis results obtained to date" for a circuit of 82 devices. Like the IDAC/ILAC system, they lack high level simulation tools, and as a result, hierarchical mapping is *ad hoc*. An example is the mapping for a pipelined A/D in [84]. Principal components of nonideal behavior– kT/C noise, finite OTA gain, and capacitor mismatch, are missing as performance constraints for their integrator. The layout system uses a simultaneous place and route approach. Simulated annealing and an area router are again used, but constraints on parasitics can be selected. Recently, WRIGHT [77] has been introduced to limit substrate coupling effects on critical circuits.

One other design system is the ISAID/RACHANA system [66][67][38]. This system is knowledge-based, but adds "qualitative reasoning" to provide a systematic approach to storing and using the knowledge in the system. A circuit corrector is also added to allow for topological changes in the optimization loop. RACHANA implements layout in one unified package using a depth first search to look for solutions. In keeping with the theme of expert knowledge, one of the inputs to the layout tool is the actual schematic citing that "high-quality analog layouts generated by human layout design experts very often resemble their circuit schematics"[38]. Geometric constraints such as symmetry, cross talk avoidance, minimization of wire length, minimization of wire bends and vias, device matching, and device merging are considered in layout generation. Primitives such as current mirrors, differential pairs, common gate devices, diode connected devices, and diode connected chains are programmed into the system to aid in finding solutions for the geometric constraints.

Recent proposals to address the problem of complexity have called for the inclusion of high-level description languages [24]. An example description for a sensor interface is given. Examples of abstracted components are: "amplification," "filtering," and "single-ended to differential conversion."

None of these systems address a precise use of hierarchy. Behavioral simulators are not provided, and constraint propagation is not clearly defined. For this reason, these systems have been fundamentally limited to the design of low-level circuits. This is clearly shown by the design examples: the focus is almost always OTA's or comparators. One of the major contributions of this work is to clearly define how hierarchy is used, and to use it in all of the design examples.

2.2 Analog Design Concepts

We use an example to illustrate the issues that must be addressed in analog design. Current source D/A converters have been selected because in Chapter 5, we will use this circuit to illustrate the design methodology.

2.2.1 I/O Relationship

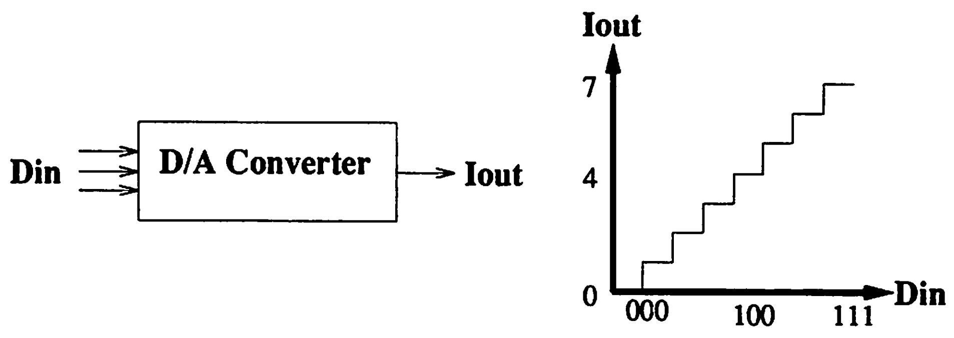

Figure 2.2: D/A Input/Output Relationship

Figure 2.2 shows the **functional** (input/output) relationship of a D/A converter. It converts an n bit digital word into a single analog signal. A three bit ($n = 3$) converter is shown in the figure. There are three input bits and one current output.[1] In the graph, the x-axis contains the 8 (2^n) possible input codes while the y-axis shows the corresponding current output.

Two important metrics in characterizing D/A converters are illustrated in Figure 2.3. One **LSB** is the ideal difference in current between any two adjacent input codes. The **full scale** range the ideal difference between highest code ($2^n - 1$) and the lowest code (0). In the figure, 1 LSB equals 1 unit of current, and the full scale range is 3 units

[1]Voltage outputs are also used.

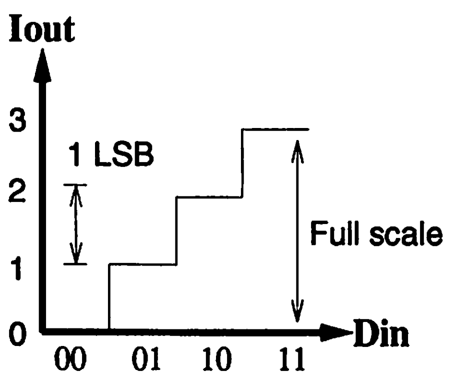

Figure 2.3: Definition of LSB and Full Scale

of current. The unit amount is usually set by a reference. Typical reference values range between 1 μA and 1 mA.

2.2.2 Linear Array Architecture

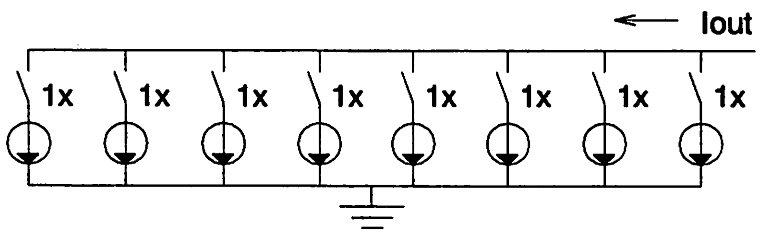

Figure 2.4: Three Bit Current Source D/A

One architecture for a D/A converter is a **linear or unit** architecture. This is shown in Figure 2.4. Identical current sources are placed in an array with their outputs connected via switches to a single output. Since current sources have high impendance outputs, the currents sum at the output node to produce the desired current. Therefore, the output is proportional to the number of switches turned on. To control the switches, the three bit input word is thermometer decoded. For example, the input code 110 translates to 11111000 turning on the first five switches.[2]

One possible implementation is shown in Figure 2.5. Each current source (unit

[2]000 converts to 00000000. The 2^nth unused current source is often kept for matching and symmetry reasons.

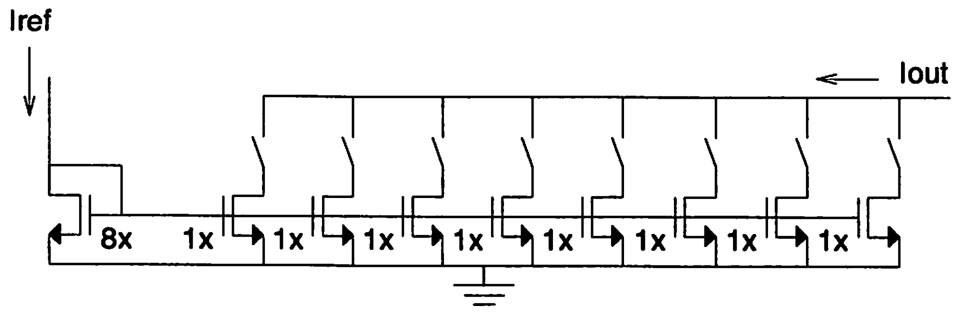

Figure 2.5: Three Bit Current Source D/A Implementation

element) is implemented by a transistor. Pulling a reference current, a diode connected transistor eight times the size of the unit elements sets up the gate voltages. Each current sources produces 1/8 of the reference current. An LSB is $1/8\,I_{ref}$. The full scale range is $7/8\,I_{ref}$.

2.2.3 Non-ideal Performance

If the transistors in Figure 2.5 were truly identical and had infinite output resistances, then this technique could be applied to build arbitrarily accurate D/A converters. Unfortunately, even when drawn identically, i.e. same gate width and length, same diffusion areas, etc., and placed within close proximity, transistors do not behave identically. There are many reasons for this; e.g. gate oxide thickness variations, lithographic variations, substrate doping variations, surface gradients, and temperature gradients [86]. This limits the number of bits that can be achieved for this type of D/A converter. Fortunately, in design, variations between transistors can be controlled using device sizing and placement techniques. Ultimately, however, typical maximum resolutions lie between 10 and 12 bits for current technologies.

There are two main lumped causes for transistor mismatch. The MOS equation for a long channel transistor in saturation is

$$I_d = \frac{1}{2}k(V_{gs} - V_T)^2 \quad \text{where} \quad k = \mu C_{ox}\frac{W}{L} \tag{2.1}$$

Taking a derivative and applying the chain rule, we have

$$dI_d = \frac{1}{2}(V_{gs} - V_T)^2 dk - k(V_{gs} - V_T)dV_T \tag{2.2}$$

Then dividing, Equation 2.2 by Equation 2.1, we obtain

$$\frac{dI_d}{I_d} = \frac{dk}{k} - 2\frac{dV_T}{V_{gs} - V_T} \tag{2.3}$$

dI_d/I_d represents the amount of mismatch between two transistors. Variations in k and V_T cause the mismatch. One method for reducing mismatch is to increase $(V_{gs} - V_T)$. However, this technique is limited by the supply voltage and does not address the k term.

An alternative technique is to control the variations in k and V_T directly. Assuming k and V_T are statistically independent [86] [72], we can rewrite Equation 2.3 as

$$\left(\frac{dI_d}{I_d}\right)^2 = \left(\frac{dk}{k}\right)^2 + 4\left(\frac{dV_T}{V_{gs} - V_T}\right)^2 \tag{2.4}$$

Proposed [86] [72] have been functional forms for the k and V_T variances. These are:

$$\sigma_k^2 = \frac{A_k^2}{WL} + S_k^2 D^2 \tag{2.5}$$

$$\sigma_{V_T}^2 = \frac{A_{V_T}^2}{WL} + S_{V_T}^2 D^2 \tag{2.6}$$

A_k, S_K, A_{V_T}, S_{V_T} are process dependent constants. W and L are the width and the length of the transistor. D is the distance between the two transistors. Thus, increasing the transistor size and placing them close together can reduce mismatch. This is an effective way for reducing mismatch.

Refinements to Equations 2.5 and 2.6 based on our experimental results (Chapter 5) have been made [33]. However, the fact that mismatch is reduced by using larger transistors and by placing them close together still remains true.

These mismatches result in non-ideal **performances**. "Non-working" parts are primarily due to these second order effects. An exaggerated possible transfer curve is shown in Figure 2.6 by the solid line. The dotted line represents the ideal behavior. Problematic points are indicated on the curve. Point A shows that there is no "off" state. Point B illustrates an extreme deviation from the ideal. In a signal processing system, this could result in large unwanted harmonics. Point C shows an area non-monotonicity. In a feedback system, this could be disastrous reversing the feedback polarity. Point D shows an extremely high output. This could overload the next stage. Methods for characterizing these non-idealities have been developed. The following definitions [58] and figures describe this.

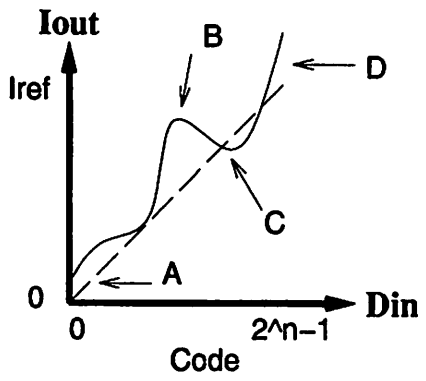

Figure 2.6: I/O Characteristics of a Typical D/A

Definition 2.2.1 *The ideal output vector, L, of a D/A converter is an N-dimensional vector, where the i_{th} component of L, $L(i)$, represents the desired output for input code, i, for $i = 0...N$-1, and $N = 2^n$, the number of codes for the D/A converter, where n is the number of bits for the converter.*

Definition 2.2.2 *The output vector, t, of a D/A converter is an N-dimensional vector, where the i_{th} component of t, $t(i)$, is the output for input code i.*

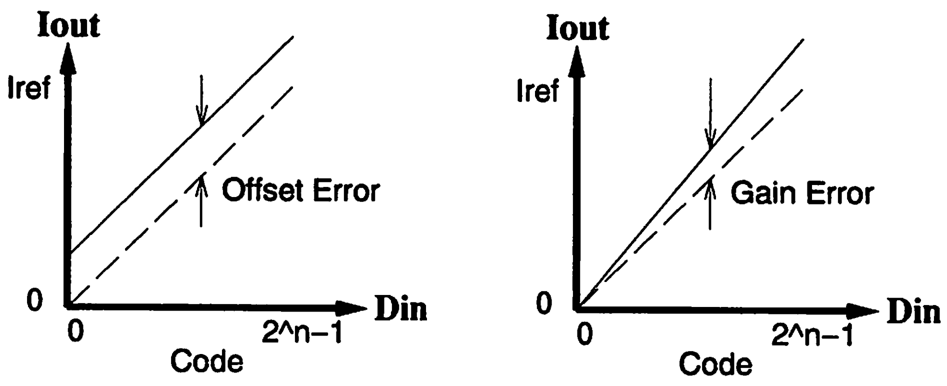

Figure 2.7: Definition of Offset and Gain

Figure 2.7 illustrates the linear components of error. **Offset** and **gain** error are defined as:

Definition 2.2.3 *Offset error is $t(0) - L(0)$.*

Definition 2.2.4 *Full scale gain error is* $[t(N-1) - L(N-1)] - [t(0) - L(0)]$.

Offset error is a constant difference between the ideal and the actual output. Gain error is a measure of the constant difference between the slopes. In general, linear errors are not critical. Preceding or following stages can often compensate for these errors.

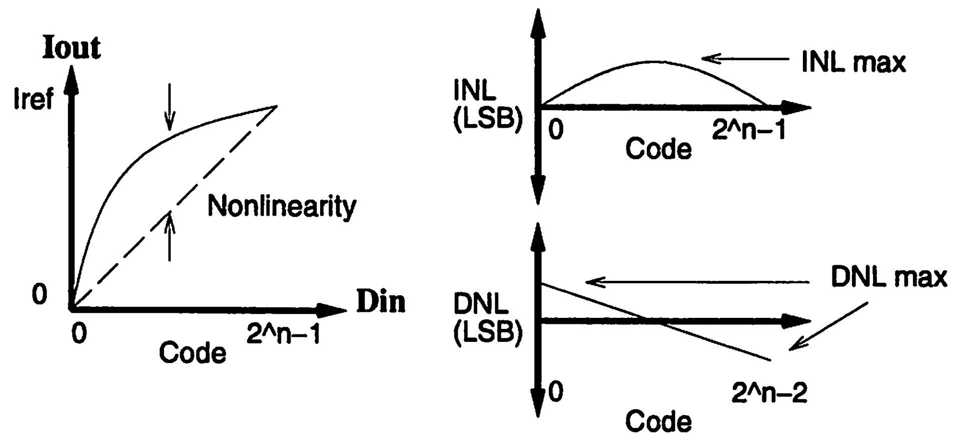

Figure 2.8: Definition of INL and DNL

Critical, however, are nonlinear errors. Two metrics for their characterization is shown in Figure 2.8 and are defined as follows:

Definition 2.2.5 *The* **integral nonlinearity** *(INL) vector, s, of a D/A converter is an N-dimensional vector*

$$s(i) = at(i) + b - L(i) \tag{2.7}$$

where a, b are constants such that $s(0) = s(N\text{-}1) = 0$. *Solving for a, b, integral nonlinearity is given by:*

$$s(i) = \left[\frac{L(N-1) - L(0)}{t(N-1) - t(0)}\right](t(i) - t(0)) - (L(i) - L(0)) \tag{2.8}$$

Definition 2.2.6 *The* **differential nonlinearity** *(DNL) vector, d, of a D/A converter is an N-1-dimensional vector obtained by a first order difference of the integral nonlinearity, s. It is given by:*

$$d(i) = s(i+1) - s(i) \tag{2.9}$$

for $i = 0 \ldots N\text{-}2$.

INL is the large signal difference between the expected and the ideal output. DNL is the small signal difference between the expected and ideal steps. Often only the maximum INL is of interest. Therefore, we define INL_{max} as, max $s(i)$, $\forall\ i = 0 \ldots N\text{-}1$, and DNL_{max} as, max $d(i)$, $\forall\ i = 0 \ldots N\text{-}2$.

2.2.4 Analysis

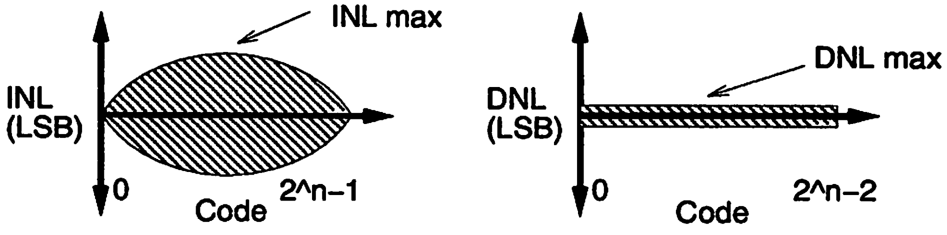

Figure 2.9: One σ bounds for INL and DNL

A simple model for the circuit is to assume that the current sources mismatch randomly with error, σ_e, and assume that errors due to finite output resistance are negligible. DNL can be represented for all input codes as

$$\sigma_{DNL} = \sigma_e \tag{2.10}$$

since all current sources are unity weighted. The DNL is graphed in Figure 2.9. The computation for INL is more complicated. It is graphed for all codes in the figure on the left. The maximum value is given by [25]:

$$\sigma_{INL_{max}} = 2^{\frac{n}{2}-1}\sigma_e \tag{2.11}$$

Typical requirements for INL and DNL range between 0.5 and 2.0 LSB. As n increases, INL requirements become increasingly difficult to achieve while there is no change in the difficulty of the DNL requirement.

2.2.5 Architectural Alternatives

An alternative architecture for current source D/A converters is a **binary** weighted architecture. An example three bit converter is shown in Figure 2.10. In CMOS, the principal advantage of this architecture over the linear architecture is that there are fewer switches. Since fewer switch control signals are required, there is a tremendous reduction in routing complexity. This translates into a huge savings in layout area. There are no savings

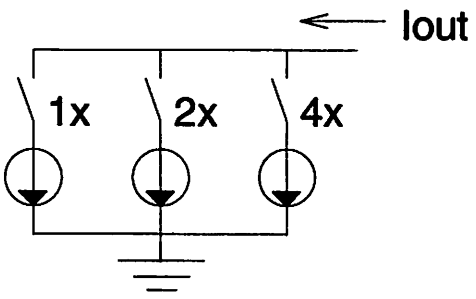

Figure 2.10: Three Bit Current Source D/A- Binary Weighted

in active transistor area since an nx sized transistor is implemented by n $1x$ transistors in parallel. Another advantage is that a thermometer decoder is not required. The input bits directly control the switches. For example, code 011, activates the 1x and 2x current source to produce a 3x output.

The INL for the binary and linear array architectures is the same [25]. The DNL, however, is much worse. The maximum DNL is

$$\sigma_{DNL_{max}} = \sqrt{2^n - 1}\,\sigma_e \tag{2.12}$$

A reduction in layout area is traded for increased DNL.

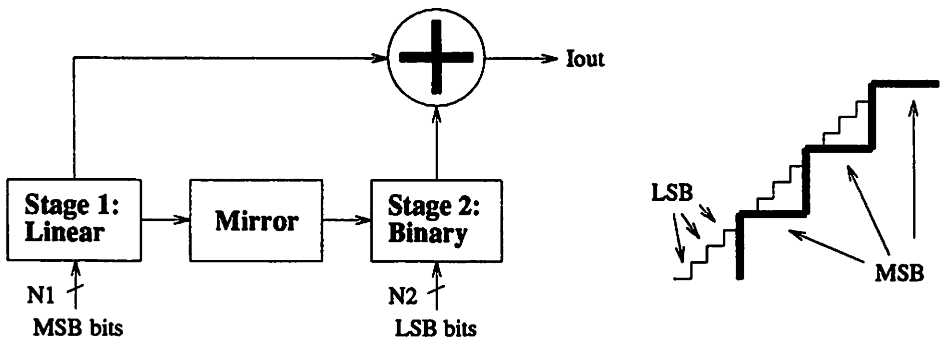

Figure 2.11: Two Stage D/A Architecture

Often this DNL penalty is too large. A compromise is to combine these two architectures. A block diagram and I/O graph for a two stage architecture are shown in Figure 2.11. The first n_1 most significant bits (MSB) are decoded in a linear first stage. The n_2 least significant bits (LSB) are decoded in a binary second stage. The first stage generates a granular staircase as shown in bold in the graph in the figure. Using one of the unit current sources from the first stage as reference, the second stage divides this current to produce the fine step sizes shown in the figure. This produces n (n_1+n_2) bits of resolution.

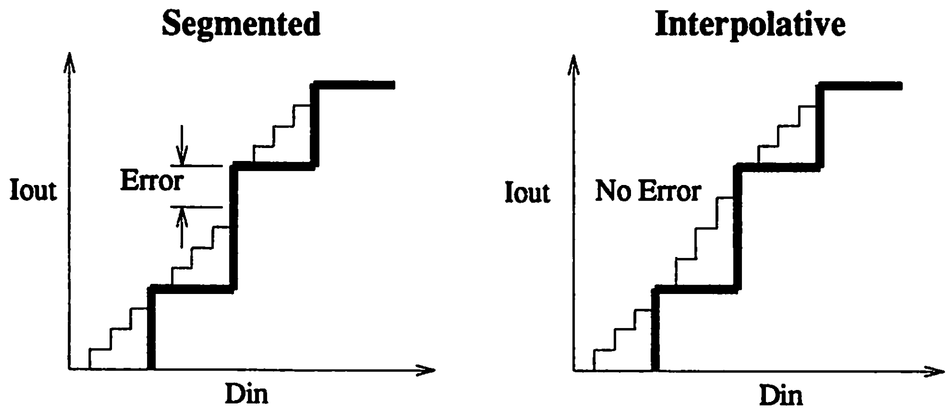

Figure 2.12: Segmented versus Interpolative Two Stage Architecture

Two stage D/A converters can be implemented in either of two basic methods. The simpler of the two is a **segmented** approach. One *dedicated* unit element of the linear array is used as reference for the binary array. As shown in Figure 2.12 on the left, if there is a large difference between the reference unit element and the other unit elements, then this results in large DNL errors when the second stage switches from all bits on to off. The second approach, an **interpolative** architecture, solves this problem. The reference element for the second stage is always the next unit element to be turned on in the first stage. In this way, the second stage transition point error is eliminated. The cost of the latter approach is more switches which results in larger layouts.

The maximum INL for both two stage architectures remains the same as before (Equation 2.11) where n is n_1+n_2. The maximum DNL for the segmented D/A is

$$\sigma_{DNL_{max}} = \sqrt{2^{n_2+1}-1}\,\sigma_e \qquad (2.13)$$

The maximum DNL for the interpolative D/A is

$$\sigma_{DNL_{max}} = \sqrt{2^{n_2}-1}\,\sigma_e \qquad (2.14)$$

2.2.6 Simulation Methods

Besides transistor mismatch, other effects such as IR drops and finite current source output resistances cause nonlinearities. To obtain more accurate INL and DNL estimates, simulation techniques can be applied.

The traditional approach is to use SPICE, a circuit simulator. Unfortunately, SPICE can only give the *functional* ($\sigma_e = 0$) characteristics. Though this is useful for verifying circuit functionality, it provides no information on INL and DNL. This simulation is also CPU intensive. 2^n operating point (".op") analyses are required. Experiments have shown that for a 10 bit converter, this requires approximately 1 CPU day. Varying transistor models in a Monte Carlo simulation to find INL and DNL is virtually impossible, because of the heavy CPU requirements.

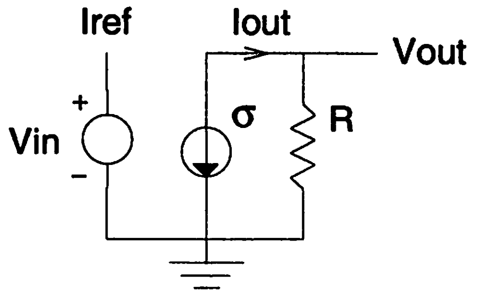

Figure 2.13: Behavioral Model for a Unit Current Source Element

A newer technique [63] based on **behavioral simulation** directly calculates the circuit *performances*, INL and DNL. The simulator, given a circuit description (Figure 2.13) and a list of random variables, can calculate the INL and DNL on the order of CPU minutes with an accuracy difference compared to SPICE of less than 0.05 LSB.

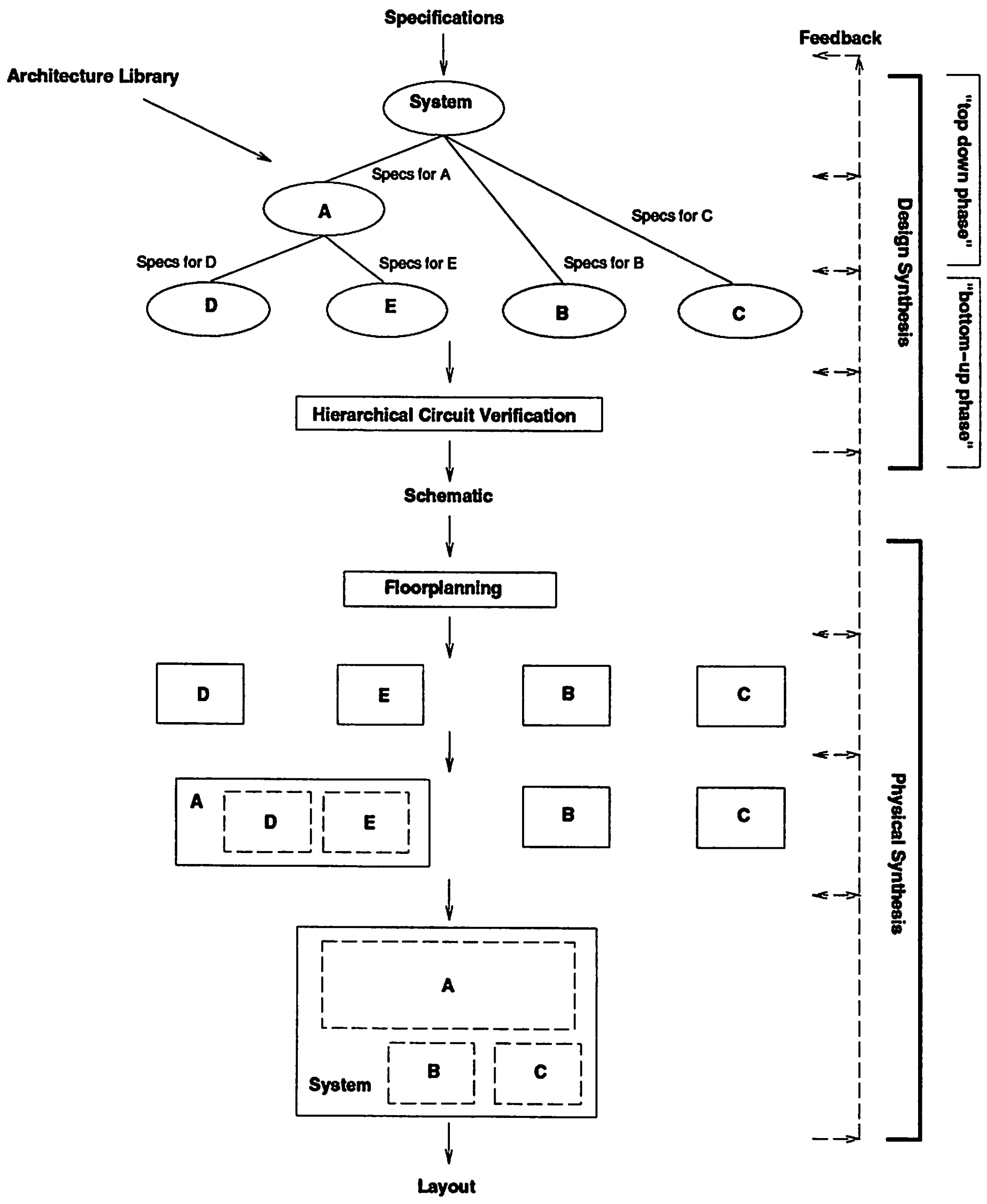

Figure 2.14: Design Using Hierarchy

Chapter 3

Design Methodology

3.1 Design Flow

Figure 3.1 shows the standard design hierarchy. At the top of the design hierarchy is a top node which could represent an entire chip or just part of a chip. From this top node there is a set of direct descendent lower nodes, representing a first level decomposition of the upper node. From these lower nodes the graph continues to be expanded. The decomposition may cease at any given node at any given level. The graph is completed with the decomposition of all of the nodes.

An example decomposition is shown in Figure 3.2. Here, a mixed-signal chip is our top node. This diagram shows a way in which this top node can be decomposed. It also shows how decomposition may cease at any level. For example, a voltage reference circuit is found in our cell library. Thus, the decomposition of this node is not necessary. A solid line under the terminating node indicates this. In some cases the decomposition does not cease until the transistor or passive element level, as indicated by the capacitors and the MOSFETs in this figure.

The methodology, illustrated in Figure 3.3, assists the hierarchical generation of the design by providing a rigorous procedure based on interactive and automatic tools. The large bubble encompassing most of the diagram represents the mapping. Given a set of circuit specifications (circuit characteristics, the design rules, the technology, and user options), a mapping is made to schematics or to layout.

Given a library of n architectures, the first operation that must be performed is architectural selection. Simulators and optimizers are used to aid the decision-making

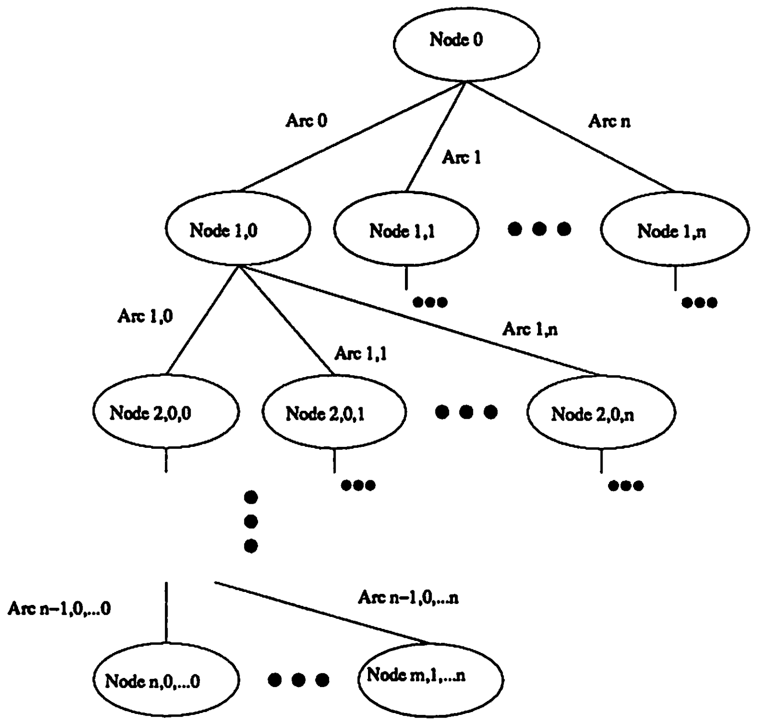

Figure 3.1: Design Hierarchy

process. For very high-level blocks, a behavioral simulator may be employed. For low-level circuits, a circuit simulator such as SPICE may be run. For an architecture where no simulators exist (e.g. a pre-made cell) the "simulator" could just be a list of performance specifications. If a suitable architecture cannot be found, then this selector must return to the upper node the fact that the selection has failed. This would mean that the specifications cannot be met; they must be changed in order to continue. If a standard cell (pre-designed cell) was chosen then a successful return to the upper node is made.

Given specifications for a particular architecture the next step is to map the chosen architecture to the detailed specifications of the component (lower) blocks. This task can be very difficult. Architectures can be mapped automatically using nonlinear optimizers; however, when this procedure fails the user must do the partitioning. If a solution to the mapping problem cannot be found, another architecture must be chosen.

The lower blocks are expanded in the same manner as this diagram depicts, thus recursively expanding the design hierarchy. From these lower blocks a set of the component specifications is returned. If the returned specifications fail to meet the criteria set by the

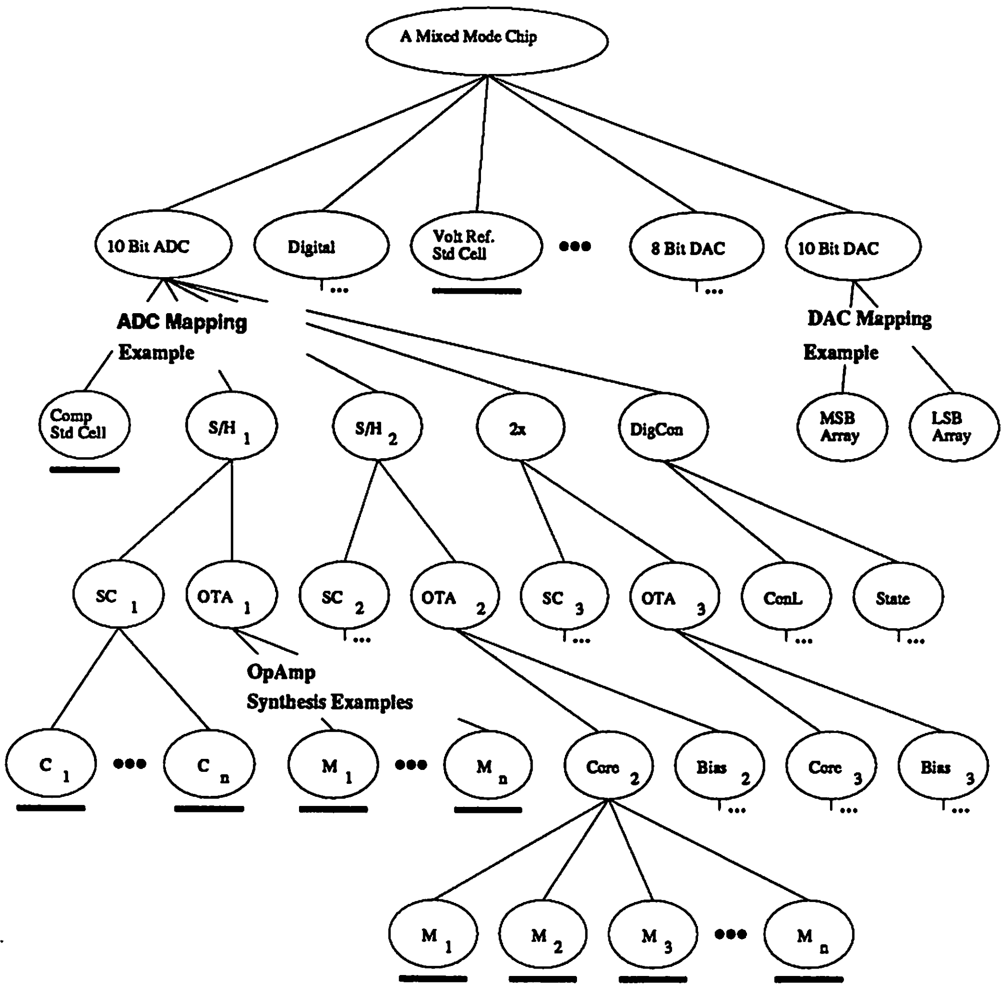

Figure 3.2: Example Design Hierarchy for a chip

mapping function, then the flow control is returned to the mapping function.

On a successful return from all of the components, the component specifications are compiled to form a list of specifications corresponding to the original list. If this compiled list fails to meet the expected specifications, then a new mapping is attempted. If it is successful, there are two options, either proceed with the layout or stop here returning only the schematics.

In creating the layout, the first step is the generation of the layout constraints for the assembly. This can be done either manually or automatically. As before, if this step fails, the flow control is returned to the mapping function. If it is successful, then a constraint-driven physical assembly is performed. If this fails to meet the requirements then

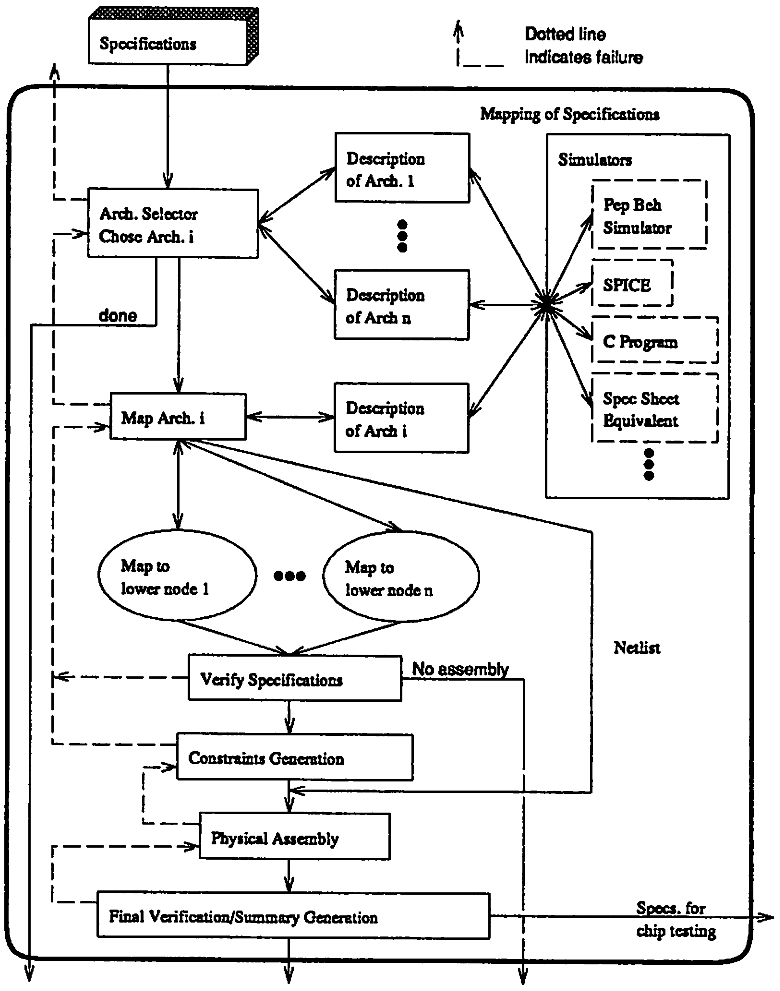

Figure 3.3: Design Methodology

an alternate set of assembly constraints is derived. If the physical assembly is successful, then the final verification step/summary generation step is required. This step includes the extraction of the circuit and simulation with the same simulator used in architectural selection. The "extraction" process spans the entire range from a simple net-list extraction to a complete extraction with parasitics. Finally, a summary of the expected performances is generated. If all of the specifications are met, the flow control is returned to the upper node with the generated specifications.

This final block also formulates the test set and methodology, based on the extracted specifications, technology considerations, and the final layout. If the desired test coverage cannot be obtained then possible hardware modifications or alternative architecture suggestions are fed back to the mapping function or the architectural selection function, as appropriate. Since the majority of analog circuit failures is due to parametric rather than catastrophic device malfunctions, determining the test set is an essential part of the design methodology which cannot be split into a separate post-processing step. Testing is based on the same functional model as the high-level behavioral simulation. This model allows the circuit to be parametrically characterized by a minimum number of well-chosen tests. The test set is derived and ordered to minimize test expense (time and equipment) while meeting the test coverage constraints. The results of the high-level behavioral simulation are used to determine the relative value and cost of testing each behavioral component of the circuit. The test coverage constraints are formulated to include both catastrophic and parametric failures. Test specifications and constraints are thus incorporated into the design of the circuit in a manner very similar to the manner in which performance constraints are accommodated. An appropriately optimized test set is part of the final chip specifications.

3.2 Use of Hierarchy

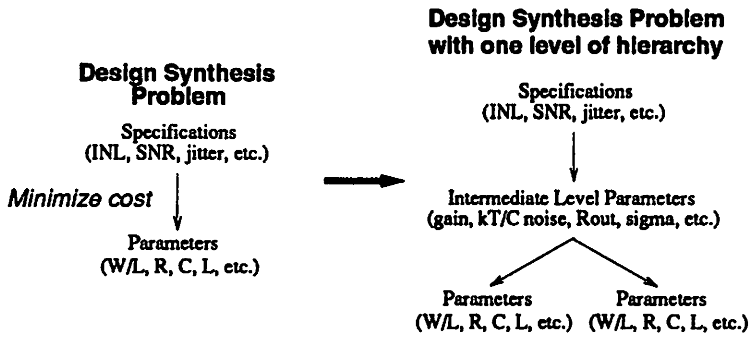
Design Synthesis Problem
Specifications (INL, SNR, jitter, etc.)
Minimize cost
Parameters (W/L, R, C, L, etc.)
Design Synthesis Problem with one level of hierarchy
Specifications (INL, SNR, jitter, etc.)
Intermediate Level Parameters (gain, kT/C noise, Rout, sigma, etc.)
Parameters (W/L, R, C, L, etc.)
Parameters (W/L, R, C, L, etc.)

Figure 3.4: Design Synthesis Problem with and without Hierarchy

One key contribution of this work is to define a mechanism by which hierarchy can be exploited successfully. The example in Figure 3.4 will be used to illustrate this.

Performances for INL, signal-to-noise ratio (SNR), and timing jitter are specified. Assuming that an architecture has been selected, component parametric values in the circuit must be found, i.e. transistor widths and lengths (W_x, L_x), resistance values (R), capacitance values (C), and inductance values (L). Without hierarchy, the problem can be formulated as

$$\begin{aligned} \textit{minimize} \quad & \text{cost}(W_x, L_x, R, C, L) \\ \textit{s.t.} \quad & \text{INL}(W_x, L_x, R, C, L) \leq \text{specified value for INL} \\ & \text{SNR}(W_x, L_x, R, C, L) \geq \text{specified value for SNR} \\ & \text{jitter}(W_x, L_x, R, C, L) \leq \text{specified value for jitter} \\ & \text{process constraints} \end{aligned}$$

where W_x, L_x, R, C, and L are vectors representing all of the passive components.

Suppose, however, that this problem contains too many design variables and cannot be solved as formulated. It is transformed from the non-hierarchical problem on the left in Figure 3.4 to the hierarchical problem on the right. The performances are first divided into a set of **intermediate level parameters**. These must characterize all of the second order performances of the low level blocks which cause performance degradation in the overall system. Satisfying this condition is often difficult. In general, they have to be selected on a design-by-design basis. If they are chosen improperly, i.e. a cause of system performance degradation is missing, when the chip is fabricated and tested, it could fail to meet specifications. In this example, the intermediate level parameters are the gain and kT/C noise for an operational amplifier and output resistance (R_{out}) and a matching term, σ, for a current source.

We use behavioral models to calculate INL, SNR, and jitter as functions of gain, kT/C noise, R_{out}, and σ. Since behavioral models are *architecturally independent*, they give no data on the design cost as a function of intermediate level parameter values. Rather than minimizing cost, since this cannot be calculated, we choose to maximize the design **flexibility** of the lower level components. For an individual variable, the goal is to choose reasonable parametric values. For example, specifying that the kT/C noise should be 0 is infeasible. We build a **flexibility function** to reflect this information. For multiple variables, the flexibility function is set up to balance design tradeoffs. For example, suppose lowering the gain, lowers the constraint on kT/C noise. Maximizing the flexibility will prevent the constraint on gain from being too low, allowing for a reasonable kT/C noise requirement.

An example flexibility function is

$$flex_P(x) = -ax^2 - \frac{b}{x} + c \tag{3.1}$$

For variables where *decreasing* values ease design difficulty, $b = 0$. For variables where *increasing* values ease design difficulty, $a = 0$. Two points are required to solve for the remaining coefficients. We use as a heuristic, a "moderate" difficulty design point, assigning it *flexibility* = 0, and a "difficult" design point, assigning it *flexibility* = -10. Table 3.1 shows an example selection. For the two examples, the variable values for the moderate

	$b = 0$ (parabolic)		$a = 0$ (hyperbolic)	
Difficulty	Parameter	Flexibility	Parameter	Flexibility
moderate	6	0	6	0
difficult	10	-10	2	-10

Table 3.1: Points on the Flexibility Curve

and difficult cases are listed in the parameter column, and the chosen values for flexibility are listed in the flexibility column.

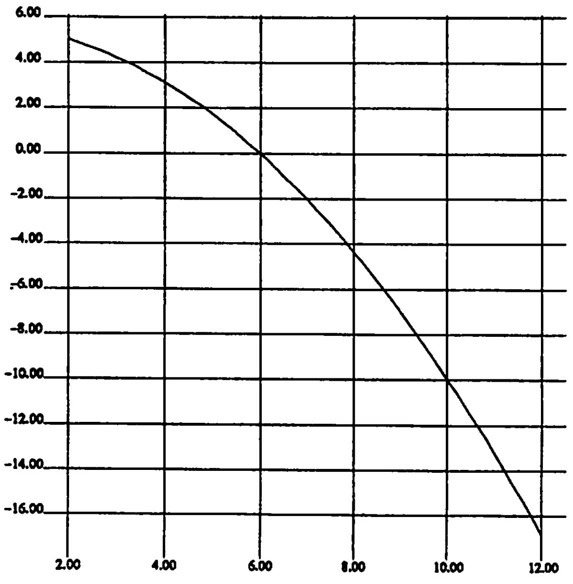

Figure 3.5: Flexibility Function- Parabolic

Solving for the coefficients for the b=0 case, we obtain

$$flex_P(x) = -\frac{5}{32}x^2 - \frac{45}{8}$$

This is graphed in Figure 3.5. The parameter value is on the x-axis. The flexibility value is on the y-axis. Marginal flexibility information is included in these curves. For variable values of high design difficulty, increasing the variable value incurs a high penalty in flexibility, while for variables values of low design difficulty, decreasing the variable value does not aid greatly in flexibility. The intuition behind this is as follows. Suppose gain is the design variable. A large value for gain is difficult to achieve. The design becomes increasingly difficult as the value of gain increases. However, small values of gain are easy to design. Asking for a reduced value in gain usually offers little advantage. This is reflected by the small change in the flexibility function around points of low gain.

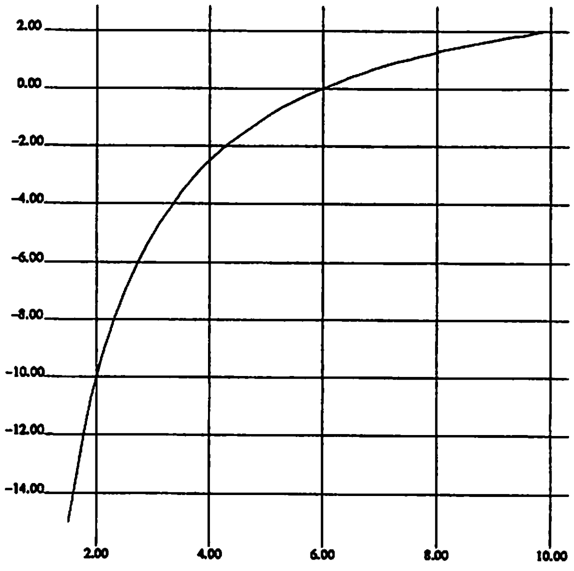

Figure 3.6: Flexibility Function- Hyperbolic

Solving for the coefficients for the a=0 case, we obtain

$$flex_P(x) = -\frac{30}{x} + 5$$

This is graphed in Figure 3.6. Once again marginal information is included. In this case for variable values of high design difficulty, *decreasing* the variable value incurs a high penalty in flexibility, while for variables values of low design difficulty, *increasing* the variable value does not aid greatly in flexibility. kT/C noise is an example.

Once flexibility functions have been developed for each of the parameters, they are summed to create the overall flexibility function. The high level problem in Figure 3.4 can now be formulated as

$$maximize \quad f_{gain}(gain) + f_{noise_{kT/C}}(noise_{KT/C}) + f_{R_{out}}(R_{out}) + f_{\sigma}(\sigma)$$

$$\begin{aligned} s.t. \quad & INL(gain, noise_{kT/C}, R_{out}, \sigma) \leq \text{specified value for INL} \\ & SNR(gain, noise_{kT/C}, R_{out}, \sigma) \geq \text{specified value for SNR} \\ & jitter(gain, noise_{kT/C}, R_{out}, \sigma) \leq \text{specified value for jitter} \end{aligned}$$

Since flexibility functions are heuristically determined, overall optimality is lost for the design. However, if a solution is found, *feasibility* is insured. If the design is found to be too suboptimal when completed, the flexibility functions can be rebuilt to reflect data from previous design passes for future iterations. [94] attempts to *a priori* build extremely accurate contours (flexibility functions) to also guarantee an optimal result. However, this is extremely expensive, since not only does the entire design space need to be simulated, but also all possible combinations of architectures must be explored *before* any designs can be obtained. Our method has the advantage that a design solution is *immediately* found, so that the designer has a feasible solution at the end of all iterations.

The low level optimization problems are directly formulated as

$$\begin{aligned} minimize \quad & \text{cost}(W_x, L_x, R, C, L) \\ s.t. \quad & gain(W_x, L_x, R, C, L) \geq gain \text{ result from high level optimization} \\ & noise_{kT/C}(W_x, L_x, R, C, L) \geq noise_{kT/C} \text{ from high level optimization} \\ & \text{process constraints} \end{aligned}$$

and

$$\begin{aligned} minimize \quad & \text{cost}(W_x, L_x, R, C, L) \\ s.t. \quad & R_{out}(W_x, L_x, R, C, L) \leq R_{out} \text{ from high level optimization} \\ & \sigma(W_x, L_x, R, C, L) \leq \sigma \text{ from high level optimization} \\ & \text{process constraints} \end{aligned}$$

Ideally, to solve these problem, we would like to use an "off-the-shelf" **optimization program** such as MINOS [76] which allows for nonlinear objective and nonlinear constraint functions. However, often the functions used are simulations which are solved using iterative techniques. These result in solutions with a large amount of numerical noise (sometimes as high as 10%), which usually caused the general purpose optimization programs to fail. Three common modes of failure are illustrated in Figure 3.7. The graph on the left shows the performance versus the input. The initial condition is the point on the

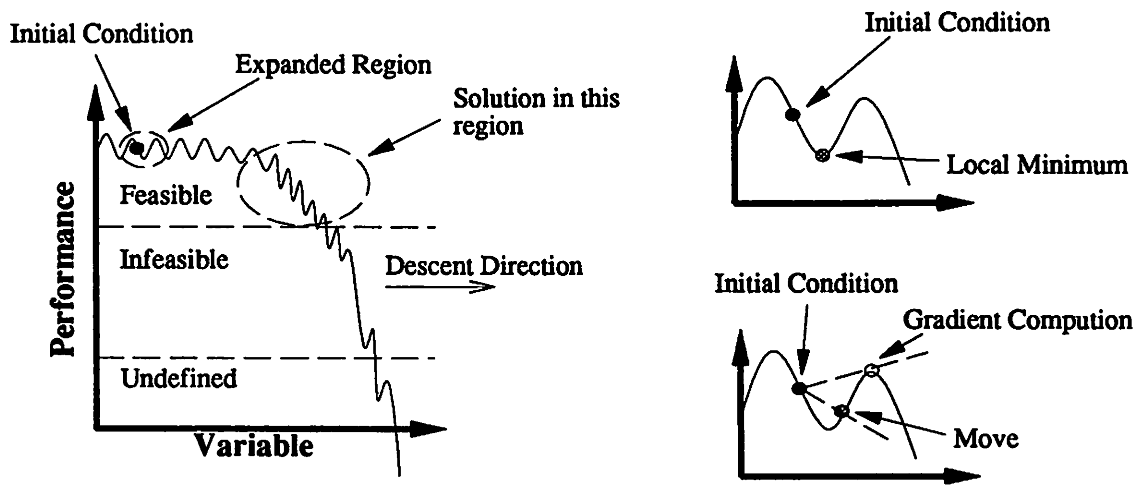

Figure 3.7: Optimization Function Surface

left. The descent direction is to the right. The feasible region is where the performance is high. In this example, the solution is in the descent direction somewhere near the boundary between feasible and infeasible region.[1] The following are the modes of failure:

1. Finding a false local minimum- The region near the initial condition is expanded and shown in the *upper right* hand corner of the figure. In this situation, because of numerical noise, a local minimum appears to exist near the initial condition. The optimizer may fall into this valley. The optimizer will terminate, and return this as the solution.

2. Iterate excessively because of gradient computation error- The area near the initial condition is expanded again and shown in the *lower right* hand corner of the figure. The gradient computation with the step size taken is shown. It indicates that the performance is increasing as the variable increases. The optimizer may make a move in that direction. The move step, however, may be smaller than the one used in the gradient computation. So, upon moving, the performance decreases. The optimizer will detect this and shrink the step sizes until a consistent solution is found. However, since it is in the noise, only random chance will allow a consistent solution to be found. The step size will continue to shrink until the iteration count is exceeded, and the

[1]This is not the only variable in the problem. If there were only one variable, the solution would be exactly at the boundary.

program will terminate with an error.

3. Wander into an undefined region- Often in the circuit design space, there are regions that are undefined, e.g. no DC convergent regions in SPICE. The graph on the left indicates an undefined region below the infeasible region. Some algorithms such as the projected augmented Lagrangian algorithm used in MINOS, allow an intermediate point to wander into the infeasible region. If it should wander into the undefined region, the simulator will not return a result, and the optimization process will terminate, or may continue but with unexpected results.

In our design examples, we tried MINOS first. It usually succeeded when we either had analytic functions or when the standard error of the simulator was low (typically less than 10^{-5}). When MINOS failed to find a solution directly, we tried to either modify the problem formulation given to MINOS or to use other optimization algorithms.

Once a solution has been found, we advocate that the designer examine the optimizer's solution. Even if there are no numerical noise problems, gradient based optimizers generally only return a local optimum, unless techniques have been implemented to search for other solutions, or the space is known to be convex. The solution, then, may be very sensitive to the initial condition. Much of the optimality obtainable depends on the designer choosing a reasonable starting point. A commonly used technique to avoid local optimum solutions is simulated annealing. However, because of function evaluation through simulation, simulated annealing techniques are usually too CPU expensive.

3.3 Testing

Analog testing is currently performed on a relatively *ad hoc* basis. A design or test engineer relies primarily upon intuition about a circuit's internal functionality to derive the circuit's test suite. This test suite frequently defaults to the complete set of circuit specifications. This approach is becoming increasingly expensive in both test development and test execution times. The specifications of mixed analog-digital circuits are usually very large (e.g. see [2]), which not only results in long manual test development, but also in prohibitive testing times on expensive (0.5M$–2M$) automated testing equipment with mixed-signal capabilities. Furthermore, the use of sophisticated computer-aided design tools has reduced the design cycle so that the influence of test on time-to-market and final cost of the circuit is more and more visible.

We propose a new methodology for testing. The produced circuit is never the same as the "ideal" (designed) one. The difference has two components: deterministic and random. The deterministic components follow from the fact that a circuit design (usually a set of mixed algebraic-differential equations) is just a simplified model of complex three-dimensional processes in the real circuit. The random component comes from the fact that even if our model were accurate, model parameters never have nominal values. This is due to instabilities of process parameters (e.g. gas flows), which can be (and usually are) both spatial and temporal, as well as other random errors: human errors, substrate inhomogeneities, mask misalignment, dust particles, etc.

Due to these random errors (faults), an analog circuit is unacceptable if it does not satisfy any of the specifications. Generally, the acceptability is determined by either a functional testing of all specifications (as is the present practice) or by checking for the existence of faults. These steps can be dramatically simplified by using behavioral models.

In this approach, behavioral models provide critical information for design engineers to evaluate the testability of the design at an early design stage and for test engineers to choose the optimum testing strategy after design. From this information, engineers can evaluate the tradeoffs between test set size, test coverage, detection thresholds, measurement noise, chip performance, and estimated yield.

3.4 Conclusion

An overview of the methodology has been presented. Examples to further illustrate the methodology are presented in Chapters 5, 6, and 7.

Chapter 4

The Tool Set

In this chapter, the most important tools that support our methodology are presented together with examples of their application to design problems. To simplify the presentation, we group them into the following categories: behavioral simulation and modeling, physical assembly, module generation, and optimization.

4.1 Analog Behavioral Simulation and Modeling

4.1.1 Discussion

The objective of behavioral modeling is to represent circuit functions with abstract mathematical models that are independent of circuit architectures or schematics. In top-down design, designers can verify system design early with behavioral models of system components before investing time in detailed circuit implementation. This enables them to explore the system design space rapidly. In bottom-up design verifications, designers can verify complex system behavior efficiently, because evaluations of behavioral models are computationally inexpensive; this results in fast system simulations. Based on the success of behavioral modeling for digital circuits, there is great interest in extending the idea of behavioral modeling to analog circuits. For digital circuits, behavioral modeling and simulation can be performed using hardware description languages such as VHDL or VERILOG. These languages do not provide the features needed for an adequate simulation and modeling of analog blocks in the system. There is an increasing need for behavioral simulation and modeling of the analog blocks. One reason for this is the tendency in ASICs and VLSICs to integrate total systems, including both analog and digital blocks, onto a single chip. Another reason is that even "simple" analog blocks are too complicated for a complete transistor-level simulation within reasonable amounts of CPU time.

For the behavioral simulation and modeling of analog circuits, the following features are essential:

- The simulator and the behavioral models have to be general. The behavioral model of a given analog block must describe the behavior of that block considered as a black box, describing its input-output behavior in terms of a set of model parameters to be supplied by the designer. It also has to hide all the internal architectural details as much as possible, resulting in generic models. The simulation engine must be independent of any particular model, so that it is possible to simulate different architectures in the same environment instead of having a different dedicated simulator for each specific architecture.
- The behavioral models for the analog circuits must include not only the first-order behavior of the circuit, but also the analog second-order effects, such as noise and distortion, in order to get a realistic idea of the performance of the overall system. Also, the statistical variation of the circuit performance has to be taken into account. In addition, analog circuits are sensitive to driving and loading impedances and may have "electrical" terminals where Kirchoff's voltage and current laws have to be satisfied.
- The behavioral simulation is to be done in time or frequency domain or a mixture of both depending on the nature of the problem. There should be methods to switch between these domains and to deal with noise, distortion and statistical parameters.

In order to realize a design environment having the features discussed above, the following research actions have been identified:

1. The development of generic behavioral models (including the second-order effects) for common analog functional blocks, implementation and testing of these models.
2. Automatic extraction of the actual parameters for the behavioral models from a given design (bottom-up extraction). This allows us to investigate the performance of different architectures, in order to be sure that the generic models cover all (or as much as possible) architectures and as much as possible second-order effects in a realistic

way. Bottom-up extraction of behavioral model parameters from a transistor-level description is essential in verifying the performance of a design.

3. The development of a behavioral simulation environment based on the behavioral models developed.

Analog circuit functions are usually simple. Designers rarely choose the wrong function; instead, circuits tend to fail due to second order effects of the functions chosen. As a result, analog behavioral models must be independent of circuit architectures, yet capture all second order effects. To meet this goal, the strategy used is to:

- Find the best mathematical representation for specific types of analog circuits, and develop realistic models by using appropriate differential equations, difference equations, transfer functions, statistical distributions, tables, or arithmetic expressions.

- Develop techniques to extract model parameters from lower level models that comprise the circuit, resulting in a model hierarchy.

According to the proposed strategies, a behavioral model for the class of Nyquist rate A/D converters [62] has been developed. The behavioral representation captures all the relevant A/D behavior without information concerning the actual implementation. The behavior of an A/D is affected by two basic statistical effects: noise and process variations. Noise can cause the same die to behave differently even when the same inputs are applied. Process variations can cause different dice to have different behavior. Circuit designers usually model noise effects by adding an input-referred noise to the input of an ideal, noiseless A/D. This approach is problematic because the A/D is nonlinear. So, noise cannot be referred to the input in general. They usually model process variation effects by several parameters such as offset error, gain error, INL, and DNL. This approach is inefficient because these parameters are not totally independent. The traditional CAD approach to A/D behavioral modeling is to model the A/D transfer function which maps a continuous input value to an output code. However, this approach is problematic because it does not represent noise effects. Furthermore, evaluating the transfer function is time consuming because it usually depends on a large number of circuit elements each with process variations. In this approach, all the statistical variations are captured in the model [62]. The variations are classified into noise and process variations according to how these non-idealities affect the A/D behavior. To describe noise effects a joint probability density function is used. To describe behavioral effects due to process variations a covariance matrix, Σ_t is used, whose rank characterizes the testability of an A/D converter; its decomposition yields efficient strategies for A/D testing.

To have a useful behavioral simulation environment, it is essential to have models for most of the high-level analog functions and develop behavioral simulators which are based on these models. To this extent, we have developed behavioral representations for voltage-controlled oscillators (VCOs) and detectors that are essential circuit components in any phase/delay-locked system [61]. Phase/delay-locked loops have many applications; they are used in receivers, clock generators [104], clock recovery [56], data synchronization [99], etc. In contrast to traditional macromodels [96], the second-order effects, distortion and phase jitter, are captured by our behavioral models. Our representations are general and independent of the circuit architectures. Moreover, we have developed parameter extraction techniques, which are described in detail in [61], to identify the models from a circuit description or from measured data. The results of parameter extraction for a sinusoidal VCO were compared with SPICE and macromodeling results. This comparison showed that the behavioral model is *more accurate* than a macromodel, as well as *faster* in terms of evaluation time. Furthermore, the behavioral representations and parameter extraction techniques developed were used to model, extract and simulate an actual phase-locked loop (PLL). The simulation time is significantly lower than an estimated SPICE simulation time, and the simulation results agree with the measured chip data.

Based on the models mentioned above, we have developed a behavioral simulator for phase/delay-locked systems [23]. Although the simulation techniques were developed in the framework of phase/delay-locked systems, the simulation methodology and the results attained are applicable to the behavioral simulation of mixed-mode nonlinear dynamic systems PLLs and delay-locked loops (DLLs) are an important part of the class of mixed-mode nonlinear dynamic systems which includes oversampling data converters, switching power supplies, etc. In general, mixed-mode nonlinear dynamic circuits are stiff systems. Traditional simulators such as SPICE spend large amounts of CPU time on computing the accurate waveforms of the digital circuits, yet the waveforms of the much slower analog circuits determine the system behavior. When a system is originally conceived in our top-down, constraint-driven hierarchical design methodology, the realization of the digital circuits, as well as the details of the analog circuits, have not been decided yet. So, an accurate circuit simulation is neither possible nor desired. Therefore, designers desire to efficiently obtain the analog waveforms and the timing of the digital waveforms rather than the actual digital waveforms.

Phase noise/jitter is a very important specification for phase/delay-locked systems. Prediction of phase noise through simulations is crucial at the early stages of the top-down design. Any modeling or design methodology, and simulation tools, which do not address this problem are not suitable for high performance phase/delay-locked system design. Phase noise simulation imposes tight restrictions on the numerical simulation algorithms to be used. Numerical noise created by the simulation algorithm should be negligible when compared with the phase noise to be computed for accurate results.

The behavioral simulator we have developed is capable of calculating the key performance measures such as acquisition time, capture range, lock range, phase noise/jitter for phase/delay-locked systems. The numerical algorithms employed in this simulator have *controllable* numerical noise which makes it possible to predict phase noise accurately. A detailed description of this simulator can be found in [23].

In addition to developing analog behavioral models, parameter extraction techniques have been investigated to identify the models from a circuit description or from measured data. For Nyquist rate A/D converters a novel noise extraction technique has been developed. Noise effects depend on parameters such as capacitor sizes and transistor thermal and flicker noise, and pose as fundamental limits on converter resolutions. For example, due to noise the probability of getting a wrong A/D output is non-zero. Currently, circuit simulators such as SPICE cannot evaluate noise effects in A/D converters because the transfer function of A/D is discontinuous. One solution is the Monte Carlo technique that injects random numbers into the signal path of an A/D during a time-domain simulation. Due to the injected random numbers in the signal path, the output code may become incorrect by chance. The drawback of such statistical technique is that the error rate of A/D converters is very low. As a result, a large number of samples is needed. A technique has been developed based on probability theory to compute the error probability directly from power spectral density descriptions of electronic noises. The advantage is that the algorithm is efficient for A/D converters with low error rate and can handle highly correlated noise such as flicker noise.

More recently, a time-domain, non-Monte Carlo method for computer simulation of electrical noise in nonlinear dynamic circuits with arbitrary excitations has been developed [22] for the extraction of parameters for PLL circuits. Available transistor-level noise simulation tools (such as the noise simulation implemented in SPICE) are not suitable for our purposes, because they cannot handle circuits with changing bias conditions, i.e., circuits in nonlinear operation. For instance, the devices in a ring-oscillator, which is composed of MOS inverter delay cells, have changing bias conditions. The timing jitter at the output of a delay cell is *nonstationary*, being highest during transitions. It is desirable to have a transistor-level noise simulation tool which can be used to simulate this timing jitter. This is a transistor-level noise simulator for nonlinear dynamic circuits. This noise simulator can simulate circuits with thermal, shot, and flicker noise sources. The algorithm for noise simulation is a direct (non Monte Carlo) method in time-domain. Results from the theory of stochastic differential equations are used to implement the simulator. It is required that the noise simulation is capable of calculating *correlations* between node voltages as well as *variances* as a function of time.

4.1.2 Examples

Two examples are presented. The first illustrates how behavioral simulation can be used as part of design space exploration. The second illustrates how behavioral simulation can be used at the circuit level to provide performances which SPICE cannot compute for architectural selection and mapping.

4.1.2.1 Design Space Exploration

In this example, architectural selection and design space exploration are performed for a 10 bit A/D converter. The results for the mapping phase are also shown.

Figure 4.1 shows our design methodology re-drawn for this example. The specifications are shown at the top of the figure. These include the requirement for 10 bits of resolution, a maximum INL of 0.5 LSB, a maximum DNL of 0.5 LSB for fabrication on a 2μm MOSIS process with an overall objective to minimize area. Five A/D architectures are considered- cyclic, 2 step successive approximation (SA), systolic, and self-calibrating pipelined. These are shown in a column of vertical boxes in the center of the figure. Within the cyclic A/D class, two are considered- one with an operational amplifier offset auto-correction and the other, a redundant significant digit (RSD) cyclic converter, with capacitor mismatch correction. Within the 2 step SA class, three are considered- one with 4 bits of resolution in the resistor array and 6 in the capacitor array, another with 5 bits of

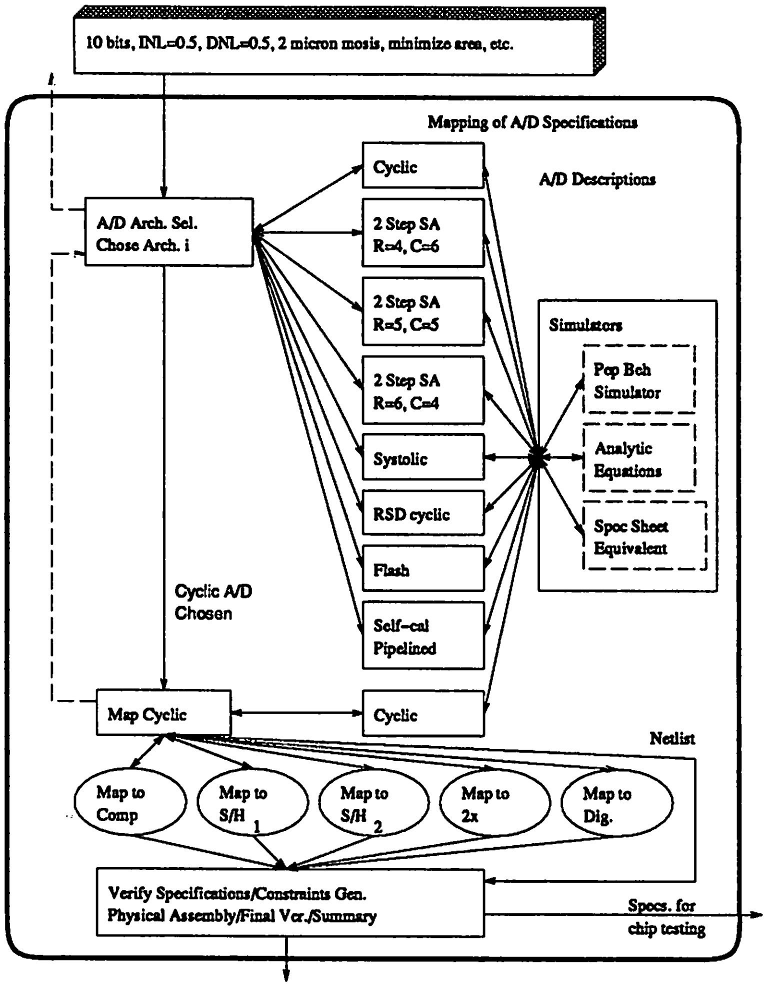

Figure 4.1: Example Application of the Design Methodology (A/D Converter)

resolution in the resistor array, and a third with 6 bits of resolution in the resistor array. The "simulators" available are shown to the right of the architectures. Table 4.1 shows the simulation results along with an *estimated* area. The required time for the simulations was under an hour for each except for the Systolic A/D which required 4.5 hours.[1]

Behavioral simulations were accomplished with PEPADC. Based on the technology assumed[2] statistical model parameters are derived for low-level components such as input referred offset of the operational amplifiers and the comparators. Statistical model

[1]Simulations were run on a DECstation 3100.

[2]2μm technology, threshold voltage offset of 5.0 mV between adjacent input MOSFET's, 0.1% mismatch for 1pF capacitors, and 1% mismatch for 100 Ω resistors.

Type of A/D	Res.	INL LSB	DNL LSB	Area mm^2	Tool to generate specs
NEEDED SPECS	10	0.50	0.50	-	Given
Cyclic	10	0.34	0.34	3.3	pepADC
RSD Cyclic	10	0.72	0.34	3.5	pepADC
2 Step SA: R=4,C=6	10	0.75	0.10	6.2	pepADC
2 Step SA: R=5,C=5	10	0.53	0.10	6.6	pepADC
2 Step SA: R=6,C=4	10	0.36	0.02	7.0	pepADC
Systolic	9	1.08	0.93	122	pepADC
Flash	10	0.54	0.71	70	Analytic Eqns
Self-cal Pipelined	13	2.00	0.60	12	given [57]

Table 4.1: A/D Design Space Exploration Results

parameters for the gain stages, resistor string D/A converters, and charge redistribution D/A converters, are derived from capacitor, resistor, and input referred offset mismatches. From these component models, PEPADC computed the INL and DNL performances. Analytical equations were used in calculating the results for the flash A/D converter, and the self-calibrating pipelined A/D converter was treated as a standard cell, thus having the specifications already given.

Based on the results, the cyclic A/D architecture is chosen (also shown in Figure 4.1). CADICS[3] performed the mapping. The results are shown in Table 4.2.

Specification	S/H_1	2x	S/H_2	Comp
Accuracy	0.997	1.9995	0.994	-
Settling Time (ns)	545	545	545	20
Max Offset voltages (mV)	3.16	3.16	12.68	0.375

Table 4.2: A/D Mapping Results

4.1.2.2 Low-level Simulation

In this example architectural selection and mapping refer to the determination of the MSB, N_1 bits and LSB, N_2 bits, in an interpolative 10 bit D/A converter. Given a set of lumped process variation parameters, a set of constraints, an objective function which is to be minimized or maximized, and an architectural model, a nonlinear optimizer [76] and

[3]Additional specifications given to CADICS included a conversion speed of 100 kHz, a power dissipation of 12 mW, a supply voltage of ± 2.5V, and a voltage reference of ± 1.5V.

a behavioral simulator [59] map the specifications to the design variables.

This 10 bit D/A can be represented by six variables. An input-referred[4] width, W_1, and length, L_1, of the first stage of non-ideal equally weighted transistors; the width, W_2, and length, L_2, of the second stage of non-ideal binary weighted transistors; and the width, W_M, and the length, L_M, of the transistors needed for the current mirror. Constraints include: (1) $W/L \geq 4$ for all transistors, because of speed concerns, (2) $W \geq 3\mu m$ and $L \geq 2\mu m$ as specified by the design rules, (3) INL $\leq$ 2 LSB[5], and (4) DNL $\leq$ 0.5 LSB. The lumped process variations are $\sigma_L = 0.02\mu$m and $\sigma_W = 0.02\mu$m. The objective is to minimize the total active area of the transistors. The results[6] are shown in Table 4.3. The requirements can be met for all cases, and the minimum area result is $N_1 = 5$.

N_1	W_1	L_1	W_2	L_2	W_M	L_M	Area	INL	DNL	CPU
	μm	μm	μm	μm	μm	μm	mm^2	lsb	lsb	hr
7	63.4	15.8	10.8	2.71	14.8	3.70	12.9	2.0	0.2	68.
6	63.6	15.9	11.1	2.77	18.3	4.60	6.54	2.0	0.4	21.
5	63.8	15.9	17.4	4.36	34.6	8.65	3.56	2.0	0.5	4.1
4	63.8	15.9	34.1	8.53	80.8	22.2	3.81	2.0	0.5	1.3
3	63.8	15.9	66.9	16.7	189.	47.4	17.0	2.0	0.5	0.9
2	63.8	15.9	132.	33.0	443.	111.	122.	2.0	0.5	0.8
1	87.6	21.9	261.	65.4	1019	255.	928.	1.5	0.5	0.5

Table 4.3: D/A Optimization Results

4.2 Physical Assembly

4.2.1 Discussion

Digitally targeted tools are often inadequate to handle the critical and specific requirements of analog layout. The performances of analog circuits are much more sensitive to the details of physical implementation than the digital ones. Custom design requires great flexibility, and layout synthesis is often a multiple-objective optimization problem, where, along with area, wiring length and delay, two groups of relevant issues must be taken into account– topological constraints (e.g. symmetries and matching) and parasitics associated with devices and interconnections. If the synthesis is performed with no explicit

[4]This is a random variable that includes mismatches and stray resistances for the non-ideal transistors.
[5]$\pm 3\sigma$ values.
[6]Optimizations and simulations were run on a DECstation 5000/125.

reference to the performance specifications of the circuit, a large number of time-consuming layout-extraction-simulation iterations may be necessary to meet the original specifications.

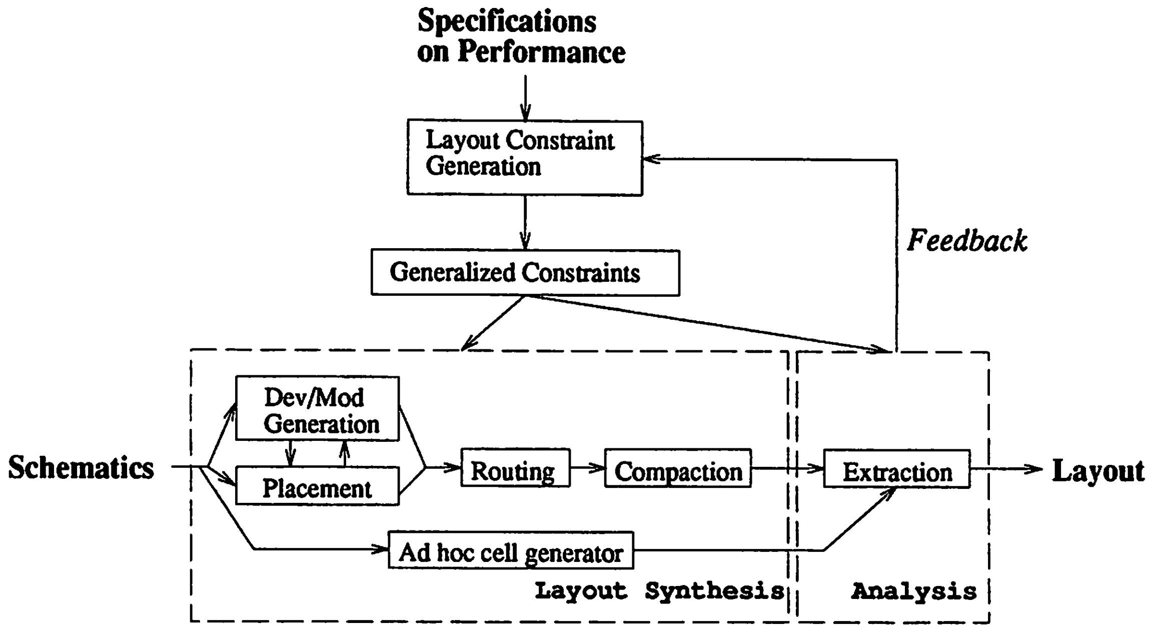

Figure 4.2: Physical Assembly Methodology

The basic premise for the constraint-driven layout of analog ICs consists of a direct mapping of performance specifications to a set of bounds on parasitics, which can then be controlled during the synthesis. This approach, shown in Figure 4.2, for the layout synthesis of analog integrated circuits consists of three phases: (1) mapping of performance specifications, (2) layout synthesis, and (3) bottom-up analysis and verification. Layout synthesis is accomplished by either placement, routing, and compaction steps, or by using an *ad hoc* cell generator. Feedback is included if the circuit fails verification. A general flow diagram valid for any of our layout tools is described in Figure 4.3.

4.2.1.1 Generalized Constraint Generation

To balance the fact that analog circuit performance is very sensitive to parasitic effects and the fact that custom design requires great flexibility, the constraint generation phase has as its main target to isolate first all parasitic effects relevant to performance and to find realistic bounds for them, without compromising the physical realizability of the circuit.

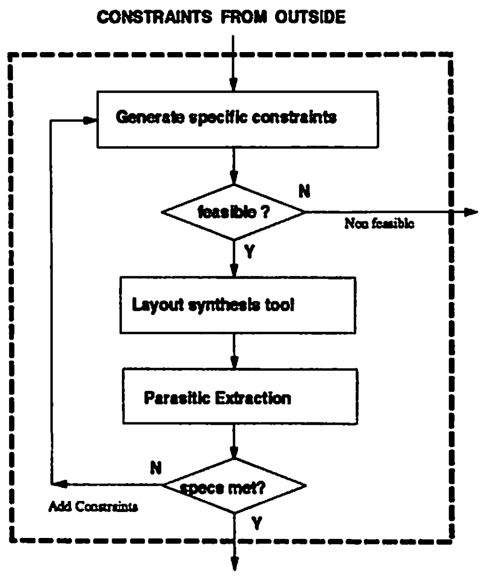

Figure 4.3: General Flow Diagram of Analog Layout Tools

Parasitic effects generally take the form of either an absolute deviation of a parameter from its nominal value or a mismatch between parameters. The parameters considered in our approach are: substrate and cross-over capacitances, stray resistances and inductances in interconnect lines, threshold voltage, and transconductance in active devices. Constrained optimization has been used to calculate bounds on parasitics and/or mismatches, based on a sensitivity analysis of the circuit [19]. The objective of the optimization is to obtain constraints that can be easily met by every phase of the layout synthesis. The tool, PARCAR, was developed to handle this.

Constraints on layout topology, i.e. interconnect symmetry and device matching, are also computed based on the sensitivity of performance with respect to the technology. Analytical models of the effect of technology gradients on various design parameters are used to calculate matching constraints on active devices, represented in terms of their relative positions and orientations [16]. Graph-based techniques are used to efficiently derive symmetry constraints from matching information.

4.2.1.2 Tools for Constraint-Driven Layout Synthesis

The layout generation process has been partitioned into its traditional components obtained from the digital world, namely module generation, placement, routing, and compaction. At each phase a sub-set of all constraints is consistently enforced, so as to minimize the number of re-design iterations for a given circuit. The extraction/verification phase checks whether all performance specifications have been met.

4.2.1.2.1 Placement and Module Optimization The realization of devices and sub-circuits can be often sped up by dedicated module generators. However, by the inherent nature of a module generator the flexibility of the design is strongly limited. To alleviate these problems, a module generator, LDO, and a simulated annealing based placer, PUPPY-A, have been combined to operate simultaneously [15]. That is, the search space of the annealing has been expanded to include all modules optimally computed by LDO. The objective of the annealing is not only area and wiring minimization, but also the elimination of parasitic constraint violations and the enforcement of topological constraints. The annealing is also responsible for the selection of the best possible set of modules in the search space. The constraint violations are evaluated at every step of the annealing through efficient parasitic estimation and use of models for the resulting performance degradation.

The optimization algorithm used in this tool is simulated annealing [49] and has been implemented in a tool called PUPPY-A [13]. The objective is not only area and wiring minimization, but also the elimination of parasitic constraint violations. The cost function to be minimized by the annealing accounts for parasitic control, symmetries, matching, and well separation as well as chip area and total wire length.

The parasitics considered are interconnect capacitances to ground, stray resistances, and crossover capacitances. Device abutment, often used by expert designers as a way of maximizing the performances of analog circuits, is performed automatically during the annealing. The decision mechanism which governs device abutment is based on estimations of diffusion capacitances of active areas and the degradation that these induce in the performances. The constraint violations are evaluated at every step of the annealing through efficient parasitic estimation and use of models for the resulting performance degradation.

Multiple symmetry axes can be managed with the algorithm of *Virtual Symmetry Axes* [68], that dynamically defines axis positions. Thus a more compact layout, less routing congestion, and a better matching between the modules is achieved. The verification phase is performed after either placement or final routing and has been made particularly accurate by use of the analytical models for interconnect proposed in [20].

4.2.1.2.2 Routing Two constraint-driven routing tools are available: SASTAR/ART [18], a global/channel router, and ROAD [69], a maze router.

In general, channel routers are the preferred tools for the routing of large mixed-mode circuits. The most important advantage of the channel routing approach is the possibility of a global overview of the communication flows in the chip [53]. Due to the increasingly greater relevance that routing area is assuming in today's VLSI circuits, global routing is a crucial phase in every framework for layout design. Channel routers however, define only the physical details of interconnect within the layout areas allocated for its implementation, the *channels*. Hence a pre-processing step called **global routing** is required. In this step all interconnect lines in the circuit are distributed among all the channels, so as to achieve uniform local congestion, to minimize net length and to control net interactions.

Two algorithms have been implemented in our analog-specific global router SASTAR: one based on the simulated annealing algorithm [98] [93] [101] and the other based on the A* algorithm [21]. Both algorithms are based on a single cost function which takes into account stray resistances, substrate and cross-over capacitances related to the interconnect. The effect of these parasitics on performance is carefully modeled using sensitivity analysis and used to drive the algorithms toward a solution which minimizes violations to the original specifications.

ART [18], is a gridless **channel router** based on the *vertical-constraint graph* (VCG) algorithm [17]. Channel routers can be efficiently used for detailed routing in the case where layout placement is generated to form a slicing structure, and where routing channels are isolated. The VCG is a graph whose nodes represent the horizontal wiring segments and whose edges represent constraint relations between the nodes. In the classical problem formulation the edges can be of two types. An undirected edge links two nodes if the associated segments have a common horizontal span. A directed edge links two nodes if one segment has to be placed above the other because of pin constraints. The weight of an edge is the minimum distance between the center lines of two adjacent segments. The routing problem consists of the minimization of the longest directed path in the VCG.

In the new constraint-driven formulation additional parasitic line coupling, i.e. crossover between orthogonal and adjacent parallel wiring lines, is controlled by means of an insertion in the VCG of additional directed edges and modification of the weights of existing edges. Unavoidable crossovers can be determined directly by the terminal placement on the top and bottom edges. The contribution of such crossovers cannot be reduced, and therefore, must be considered for the computation of lower bounds for the parasitics in the constraint generation phase. Coupling between adjacent lines that cannot be sufficiently spaced is reduced by the insertion of lateral or vertical shielding conductors.

Symmetric mirroring of interconnect nets is achieved by constraining their vertical position to the same value. Special connector configurations are used to guarantee good parasitic matching between nets crossing each other over the symmetry axis. The same number of corners and vias and the same interconnect length are maintained for both wires.

ROAD [69] is a **maze router** based on the A* algorithm [21], on a relative grid with dynamic allocation. The A* algorithm is a heuristic improvement of the Lee-Moore algorithm [55]; in the wave propagation step only one element of the wave front is propagated at a time. The propagated element is the one minimizing an estimate of the length of a wire constrained to pass through it.

A cost function is defined for each net on the edges of the grid. The path found by the maze router is the one minimizing the sum of the values of the cost function. In ROAD, the cost function is a weighted sum of several items, thus expressing a multiple target optimization problem. The items considered are the local wire crowding, stray resistances, capacitances, and cross-coupling capacitances introduced by wire segments.

Weights define the relative criticality of each item in the cost function. Hence, the quality of the layout depends dramatically on the way weights are defined. In this approach, performance sensitivities to parasitics are used to generate the weights for the cost function. After the weight-driven routing phase, parasitics are extracted and performance degradations are estimated and compared with specifications. If for some performance the specifications are not met, the weights of the most sensitive parasitics are increased and routing is repeated. If the weights of all the sensitive parasitics cannot be increased further, the tool returns an infeasibility message for the layout with the given set of constraints.

4.2.1.2.3 Compaction SPARCS-A [31][32], the mono-dimensional analog-specific compactor, combines two algorithms, one based on the *constraint graph* (CG) and a second on *linear programming* (LP), working alternatively on the same configuration.

The compaction process is partitioned into two phases. In a preliminary step the parasitic constraints are mapped onto a graph which is solved using the longest path algorithm [81]. The configuration obtained from the solution of the CG is used as starting point for the linear program. New constraints accounting for symmetries are introduced in the linear program which is solved using the simplex method.

The longest path algorithm, originally implemented for digital compaction [9] [8], has been modified to account for parasitics. Control over cross-coupling capacitances is enforced by adding directed edges to the original graph, so as to maintain minimum distances between pairs of critically coupled wiring segments. In geometrically complex wiring structures, spacing constraints are distributed among all parallel segments forming the wires according to the relative length of the segments and whether they belong to the critical path. In case of overconstraints introduced by the spacing operation, the set of minimum distance constraints are adjusted by reducing the minimum distance between overconstrained pairs and by increasing the minimum distance between non-critical pairs. The resulting set of new constraints is more likely to pose a feasible problem to the compactor. If no legal and/or feasible set of minimum distance constraints are found, additional shielding lines are introduced.

The LP approach of the second step uses the graph solution of the previous step as a starting point, but introduces a set of additional constraints to account for device and wiring symmetries. The resulting linear program is solved through the simplex algorithm and the locations of the elements not lying in the critical path are rounded off, thus eliminating the need for use of a much slower integer programming algorithm.

4.2.1.3 Parasitic Extraction

Parasitic extraction is the final phase of the layout generation process, providing means for bottom-up verification. This process has become more difficult and time consuming with the introduction of increasingly large systems. With the continuous increase of the scale of integration and die size, interconnect parasitics and technology-induced mismatches play a more significant role in influencing circuit performance and cannot be neglected. Moreover, the use of aggressive design rules, for achieving better circuit performance has made couplings between some interconnects highly critical. Hence crosstalk must be accurately calculated in order to reliably estimate possibly disastrous performance degradations.

Three-dimensional simulation of all the structures present in the layout is impractical for real circuits. In fact numerical methods based on finite-difference, finite-element, integral-equation or other techniques, generally used for computing exact electrical parameters of arbitrary structures [89] [88] [5] [48], are computationally intensive. A better approach consists of creating models of interconnect, easily obtainable through CAD model generators [1] [41]. Sophisticated analytical models for parasitic capacitances accounting for higher order non-linearities have been created using an *ad hoc* model generation tool [20]. Similar models have been created with the inductance model generator INDMOD [100]. Based on these models, ESTPAR performs physical layout extraction of parasitics and active devices, allowing greater efficiency of the extraction process, at no great expense of accuracy. Often the number of parasitic components exceeds the capabilities of today's simulators, thus making an accurate analysis a difficult task.

4.2.2 Examples

4.2.2.1 Operational Amplifier

Figure 4.4 shows the schematic of a folded cascode operational amplifier. The amplifier was placed and routed imposing constraints on three performances, *Low Frequency Gain*, *Phase Margin*, and *Unity Gain Bandwidth.* The performance constraints, listed in Table 4.4, were translated onto bounds of parasitics by PARCAR. Topological constraints were also considered during the process. The resulting layout is shown in Figure 4.5. After performing the placement, PUPPY-A predicted no violations in the imposed constraints. The circuit was then routed with ROAD using the same constraint-driven methodology, as shown in Figure 4.6. All parasitics were extracted from the routed layout and a SPICE based verification on the performance degradation confirmed that no violations occurred. Table 4.4 shows the resulting performance degradations of the circuit synthesized with and without enforcement of the specifications.

Performances	Specs.	Not Enforcing Specs.	Enforcing Specs.
Bandwidth	50 KHz	200 KHz	35.0 KHz
Phase Margin	0.5°	2.0°	0.46°
Gain	0.5 dB	0.3 dB	0.3 dB

Table 4.4: Operational Amplifier Performances

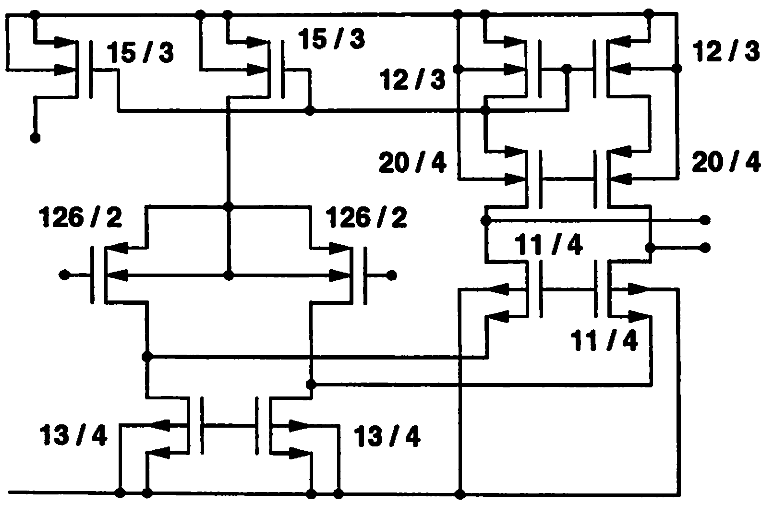

Figure 4.4: Schematics of the OpAmp

The place and route procedure required 1600 seconds of CPU time on a DECstation 3100. The results are consistent with an analysis of the layout. The six most critical nets have been carefully minimized in length when the constraints were introduced. Also no crossings were necessary to route these nets. In addition, abutment was used in four cases to remove constraint violations.

4.2.2.2 Transconductance Amplifier

Similarly, a transconductance amplifier has been synthesized using the same methodology. Transistor elements and biasing were automatically generated and placed by CADICS. Again from the performance constraints, reported in Table 4.5, a set of bounds on parasitics was generated and used to drive ROAD. All parasitics were extracted and a similar verification on the performance degradation confirmed the absence of violations. The layout was compacted using SPARCS-A and again the performance degradations were computed. Table 4.6 shows the results of the compaction process. The performance data of the synthesized layout are reported in Table 4.5. Figure 4.7 depicts the final layout.

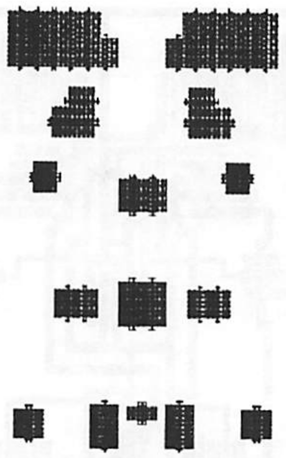

Figure 4.5: Placement with Topological and Parasitic Constraints.

Performance	Min	Max	Result
Bandwidth	6 MHz	∞	6.06 MHz
Phase Margin	75°	∞	76°
Gain	60 dB	66 dB	63.8 dB

Table 4.5: Transconductance Amplifier Synthesis Results

4.3 Module Generation

4.3.1 Discussion

Several performance-driven module generators have been developed to implement specific analog structures. Module generators can be implemented using an expert system approach or a parameterized schematic of known architectures approach. Expert system implementation for simple analog blocks such as operational amplifiers and comparators have been reported in earlier years [29]. Unfortunately, switched-capacitors and A/D converters are much more complex. No clear way exists to set the rules for their implementation.

A parameterized schematic of known architectures implementation is a better approach, because it can generate highly-effective circuits based on known architectures for complex analog functions in a flexible environment by optimizing the device sizes with respect to various performance measures [50] [28] [46]. This approach fits nicely into our methodology.

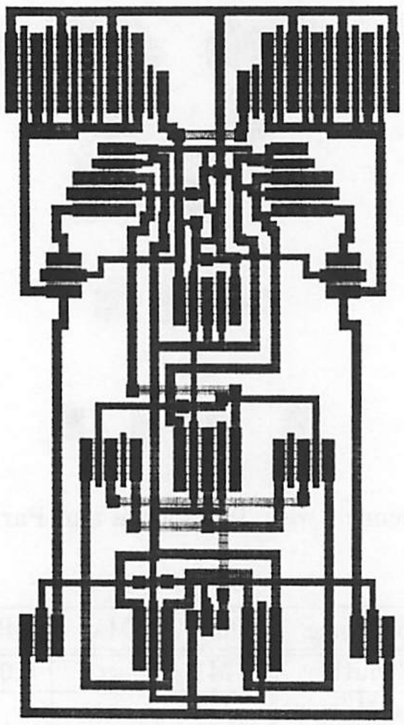

Figure 4.6: Complete Layout

In the remainder of this section, discussed briefly will be our existing module generators, OPASYN (operational amplifier synthesis) [50], ADORE (switched-capacitor synthesis) [102] [103], and CADICS [46] (A/D converter synthesis).

4.3.1.1 Opasyn

OPASYN [50] is an automatic synthesis tool for the generation of operational amplifiers. It is based on analytic circuit models. Given a set of specifications, OPASYN performs the necessary optimizations for transistor sizing. Layout is generated, and a performance summary is presented. Originally, this program only considered the generation of operational amplifiers. Recently, it has been expanded to include comparators and output buffers.

The design synthesis phase requires two steps: (1) a circuit topology is selected based on the specifications, and (2) the circuit is optimized to meet the required specifications. To simplify the optimization process, for each given topology, abstracted design parameters are selected to reduce the number of variables in the circuit optimization prob-

	original	compacted
number of instances	39	39
area	102480	79800
area compression	100%	77.9%
wiring length	7333	5883
CPU time	-	56.3 sec

Table 4.6: Transconductance Amplifier Compaction

lem. The range of possible solutions is *not* affected by this step. Such a parameter is bias current. Bias current defines the sizes of the biasing devices. Rather than having two variables (width and length) for each bias device, the problem is greatly simplified by lumping these variables into one variable representing bias current. Sets of equations representing the function and performance for each amplifier are defined using these parameters. OPASYN then uses these equations to optimize the parameters solving a nonlinear optimization problem, meeting all required specifications, and minimizing a cost function.

Physical assembly consists of three steps: (1) leaf cell generation for the layout blocks, (2) floorplanning using slicing trees, and (3) custom routing and layout spacing [51].

4.3.1.2 Adore

ADORE is a switched-capacitor filter module generator [102]. It places special emphasis on the physical design aspect. ADORE separates the switched-capacitor filter design problem into a filter synthesis problem and a layout problem. The FILSYN [97] program automates the design of a switched capacitor filters. It mimics the decision making process of a filter designer. For layout, ADORE uses a sophisticated algorithm to generate the capacitor arrays required, and then uses standard operational amplifier and switches to complete the layout. The operational amplifiers, switches, and capacitors are ordered to optimize the wiring required.

This program is currently being modified to better fit into our design methodology and to expand its capabilities such as implementing fully differential switched-capacitor filter architectures.

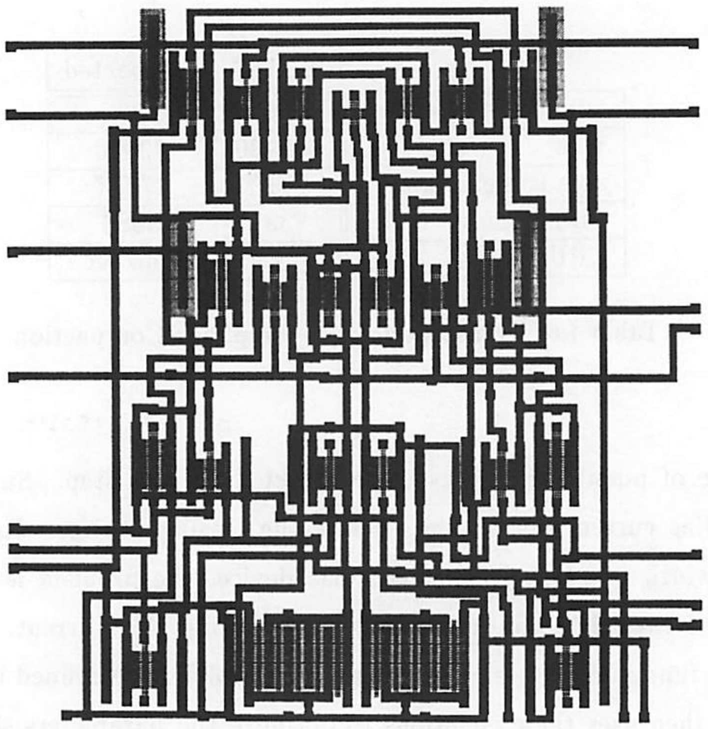

Figure 4.7: Final Layout of the Transconductance Amplifier

4.3.1.3 Cadics

A performance-driven CMOS A/D module generator, CADICS [46], has been developed. Given a set of specifications and data on the technology, the design-rule, device-matching, and the floorplan, CADICS will not only generate a complete netlist but also the layout of the A/D converter. The implementation of this A/D module generator is divided into two parts: circuit synthesis and layout synthesis.

Circuit synthesis takes as its inputs a set of specifications for an A/D converter such as resolution, conversion-speed, maximum input signal frequency, maximum differential nonlinearity (DNL), maximum integral nonlinearity (INL), total power dissipation, supply voltage, reference voltage, area, layout aspect-ratio range, and technology. In addition, statistical information such as capacitor and transistor matching can be provided.

Recognizing that A/D circuits are realized using different functional blocks such sample/hold blocks, gain-of-two circuits, comparators, digital circuitry, and others, CADICS is implemented by using a *hierarchical* optimization approach. The optimization procedure is shown in Figure 4.8. First, the A/D specifications and its respective relative priorities are

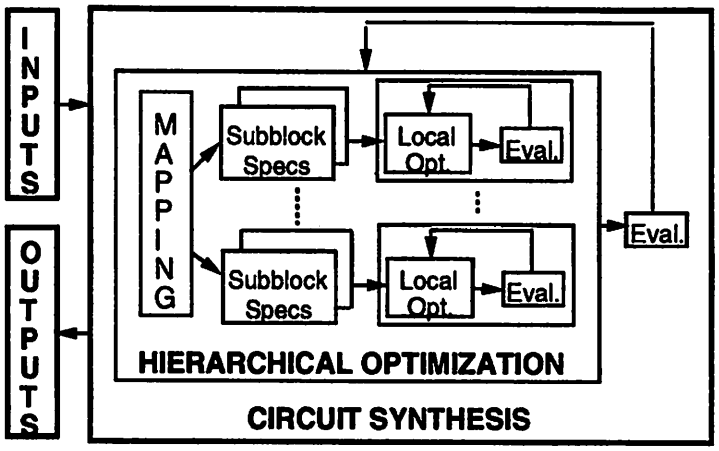

Figure 4.8: Hierarchical Optimization Procedure

mapped into sub-block specifications and their respective priorities. This mapping function can be realized by a set of rules given by experienced analog designers, by running a behavioral simulation of the A/D circuit, or by a combination of both. Then the specifications and priorities of each sub-block are fed into the local optimizer to generate the device sizes. This local optimizer can be a low-level synthesis block such as OPASYN or LAGER [30] (to generate digital circuitry), or library cells can be used. When all of the sub-blocks have been optimized locally, a behavioral simulator is used to obtain the A/D performance. If the performance is unsatisfactory, the entire operation is repeated by relaxing the given specifications until the desired performances are obtained, or the iteration limit of optimization cycles is exceeded.

In addition to the input files for circuit synthesis, layout synthesis requires: (1) a structured A/D converter netlist and (2) device structure information from the circuit synthesis phase. A mixed top-down/bottom-up approach [47] is used. This method consists of: netlist rearrangement, circuit partitioning, floorplan optimization, leaf-cells generation, sub-block place and route, and global place and route. In all steps, emphasis is placed on the *fine details of layout* which influence analog circuit performance. To isolate capacitors from noisy signals, capacitors are placed on top of wells which are connected to an analog ground line. Depending on the floorplan and netlist, well and substrate contacts are created during placement. Fully differential architectures and symmetry in layout are maintained. Sub-block routing is done with ROAD [69].

4.3.2 Example

Sample inputs and outputs for ADORE are presented in this section. Given a SPICE-like netlist for a switched capacitor filter circuit and a file containing technology information, the leaf cells are generated and placed. Routing is left as a post processing step. The circuit is composed of three primary components– the switches, the capacitors, and the operational amplifiers. The switches and capacitors are generated by ADORE. The operational amplifiers can be generated by hand, obtained from a library, or generated using OPASYN.

```
*** Low-pass Filter with 2 poles / 2 zeroes
*** Integrators and their integrating capacitors
e1 2 0 0 1 new_twost:symbolic
ca  1 2 8.0p
e2 4 0 0 3 new_twost:symbolic
cb  4 3 8.0p
*** Switched capacitors
c3 c3_a c3_b 7.0p
c1 c1_a c1_b 1.0p
c1s in  3    4.0p
c2 c1_b c2_b 7.0p
C4 c2_b c3_b 1.0p
*** Switches
sc3_1 2 phi2 c3_a phi1 0
sc3_2 3 phi2 c3_b phi1 0
sc1_1 in phi2 c1_a phi1 0
sc1_2 1 phi1 c1_b phi2 0
sout 4 phi2 ou phi1  10
sc4 0 phi1 c2_b phi2 4
*** input
vin in 0
***output
out ou 0
```

Figure 4.9: Example Input for ADORE

A sample schematic input is shown in Figure 4.9. The technology file, the other input file, contains information on capacitances, on which layers to build the capacitor, and on the sizing of nets. The layout output from ADORE is shown in Figure 4.10. At the top are the switches. The capacitor arrays are in the center. And at the bottom are the operational amplifiers. Input/output vias are located along the sides of the cell.

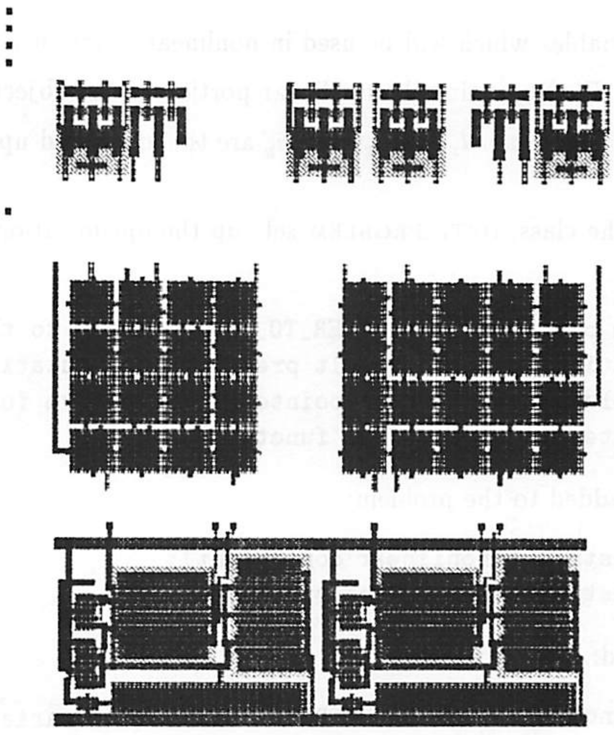

Figure 4.10: ADORE Generated Output

4.4 Optimization

4.4.1 Discussion

Within the design methodology, optimization plays an important role. Optimizers can be applied in all phases of the design. OPTZ, a generic platform for optimization algorithms, has been developed.

c^{++} classes have been implemented to allow users to specify optimization problems of the following form:

$$\begin{aligned} \text{Minimize or maximize} \quad & F(x) + c^T x + d^T y \\ \text{subject to} \quad & f(x) + A_1 y \leq, =, \geq b_1 \\ & A_2 x + A_3 y \leq, =, \geq b_2 \end{aligned}$$

$$l_x \leq x \leq u_x$$
$$l_y \leq y \leq u_y$$

x is the vector of variables which will be used in nonlinear functions. y is the vector of the remaining variables. $F(x)$ contains the nonlinear portion of the objective function. $f(x)$ is a vector of nonlinear functions. l_x, l_y, u_x, and u_y are the lower and upper bounds on x and y.

Declaring the class, OPTZ_PROBLEM sets up the optimization. An example declaration is:

```
optz_problem prob(OPTZ_OPTIMIZER_TO_USE, pointer to the objective
        function, desired result precision, computational
        resolution available, pointer to objective function,
        pointer to constraints function);
```

Constraints can be added to the problem:

```
prob.add_constraint(nonlinear constraint);
prob.add_constraint(linear constraint);
```

Bounds can be added:

```
prob.add_bound(value, OPTZ_UPPER or OPTZ_LOWER, variable name);
```

Initial conditions can be added:

```
prob.add_initial(value, variable name);
```

Once the problem is set up, asking for a result will cause the optimizer to run. For example:

```
optz_var *result = prob.get_var_term();
```

will solve the problem and return the results. Status information on whether or not the solution is optimal, feasible, or overconstrained is also returned.

An overview of the optimization process is shown in Figure 4.11. (1) The problem statement is made, and a `get_results` is requested. (2) The optimizer returns with a vector, x, and asks for function evaluation. OPTZ forwards x to the function evaluator which returns $F(x)$ and the gradients with respect to x, $G(x)$ (if it is available) to the optimizer. Most useful to us are function evaluators which deal with integrated circuit analysis such as behavioral simulators and circuit simulators. (3) The optimizer uses $F(x)$ and $G(x)$ and computes the next x vector if necessary. (4) This process continues until the optimizer finishes. The result and status are returned.

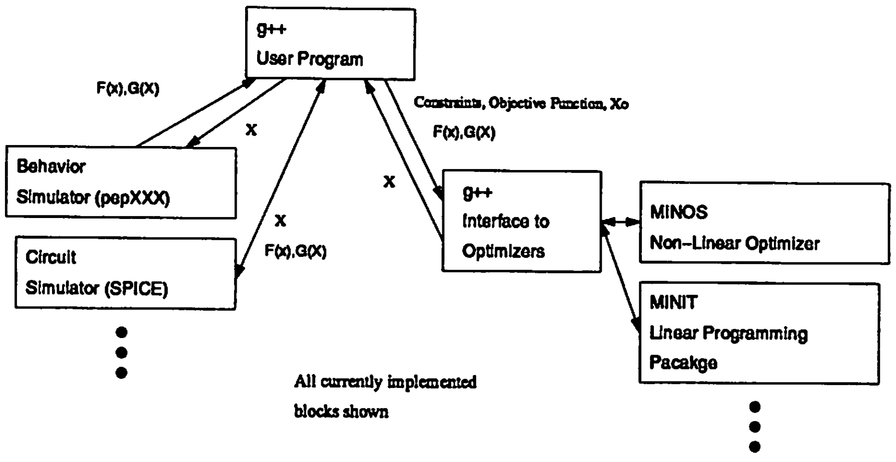

Figure 4.11: Optimizer Interface

Currently, two optimizers are supported by the interface. MINOS [78] is the first. It uses a variety of techniques to solve nonlinear programming problems. For example, if the constraints are linear, but the objective function is nonlinear, it uses a *reduced-gradient algorithm* with a *quasi-Newton algorithm.* If the constraints and the objective functions are nonlinear, then it uses a *projected augmented Lagrangian algorithm.* MINOS provides a nice user interface for the following reasons: (1) it automatically uses finite difference to compute derivatives if derivatives are not provided; (2) it can handle an arbitrary number of provided derivatives; (3) it contains many options to control its operation; and (4) it is well documented. The second optimizer that is connected to our interface is MINIT [75], a linear programming package.

4.4.2 Examples

Several examples of circuit optimizations will be presented. Both behavioral and SPICE simulations were used to evaluate circuit performance. The purpose of this section is to not only demonstrate the generality of OPTZ but also to provide insight on the CPU requirements for these problems.

4.4.2.1 Sample and Hold

This example is for the sample-and-hold (S/H) shown in Figure 4.12. The variables are: (1) C_S (C_I is set equal to C_S), bound to be between 0.25 pF and 10 pF; (2) the width

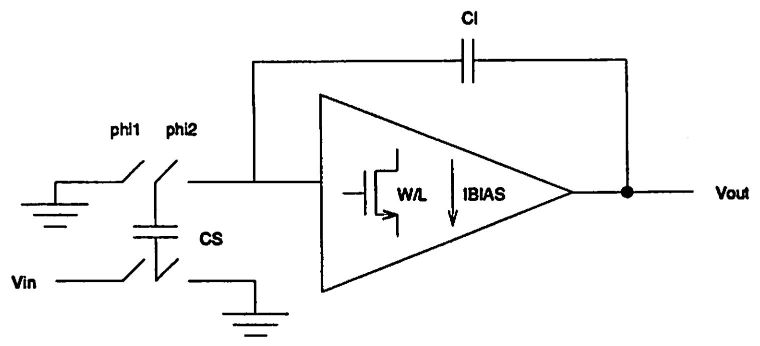

Figure 4.12: Sample-and-Hold Schematic

and the length of the input source coupled pair of the operational amplifier, bound to be between 2 μm and 3000 μm; and (3) the bias current in the input stage of the operational amplifier, bounded to be between 100 μA and 100 mA. The performance constraint is that the circuit must have 12 bits of resolution, and there may be a constraint on power (P). Initially, C_S is set to 1 pF. The width and the length of the input transistor are set at 100 μm and 20 μm respectively. And the bias current is set to 0.5 mA. The objective is to minimize area. Table 4.7 shows the optimization results. The 1.5 mW case was infeasible.

P	C_S,C_I	W_I	L_I	I_{BIAS}	RES	Area	Power	CPU	SIM
mW	pF	μm	μm	mA		mm^2	mW	min.	
2	0.264	1632	2.02	0.194	12.00	3.28	1.94	16	5744
5	0.250	622	2.25	0.338	12.00	1.49	3.38	42	8078
1.5	*	*	*	*	*	*	*	*	*

Table 4.7: Sample-and-Hold Optimization Results

4.4.2.2 Common Emitter Amplifier

Gain bandwidth trade-off is illustrated by the small-signal common emitter (CE) amplifier in Figure 4.13. C_π and C_μ are are both set to 5 pF. There is only one variable, R_L, in this circuit. It is bound to be greater than 100 Ω and is initially set at 10 $k\Omega$. The frequency response (f_{-3db}) is constrained to be greater than 1.0 MHz.

The results are shown in Table 4.8. They are optimal. Figure 4.14 shows the path of the optimizer (selected R_L values) with respect to iteration number.

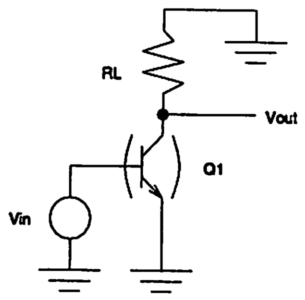

Figure 4.13: Common Emitter Amplifier

R_L $k\Omega$	f_{-3db} MHz	CPU min.	SIM
31.83	1.00	35	171

Table 4.8: Common Emitter Amplifier Optimization Results

4.4.2.3 Transimpedance Amplifier

A more complicated example, an input stage for a wide band, low-noise monolithic fiber-optic preamplifier, is shown in Figure 4.15. In this design, there are six variables, six nonlinear constraints, and a nonlinear objective function. Four types of SPICE analyses are employed- "OP," "DC," "AC," and "NOISE." The problem statement is as follows:

$$\begin{aligned}
\textit{minimize} \quad & I_{\text{NOISE}}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \times 10^8 + \\
& \text{POWER}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \times 10^3 - \\
& f_{-3dB}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \times 10^7 \\
\text{s.t} \quad & \text{GAIN}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \geq 5000\Omega \\
& \text{RIPPLE}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \leq 0.01 \\
& f_{-3dB}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \geq 150\text{MHz} \\
& \text{DC}_{\text{OUT}}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \geq 3.5V \\
& \text{POWER}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \leq 20mW \\
& \text{MAXIN}(R_F, W_{M1}, R_{L1}, R_{L3}, R_{E2}, R_{L3}) \geq 140\mu A \\
& R_F \geq 5k\Omega \\
& W_{M1} \geq 3\mu m \\
& R_{L1} \geq 1k\Omega
\end{aligned}$$

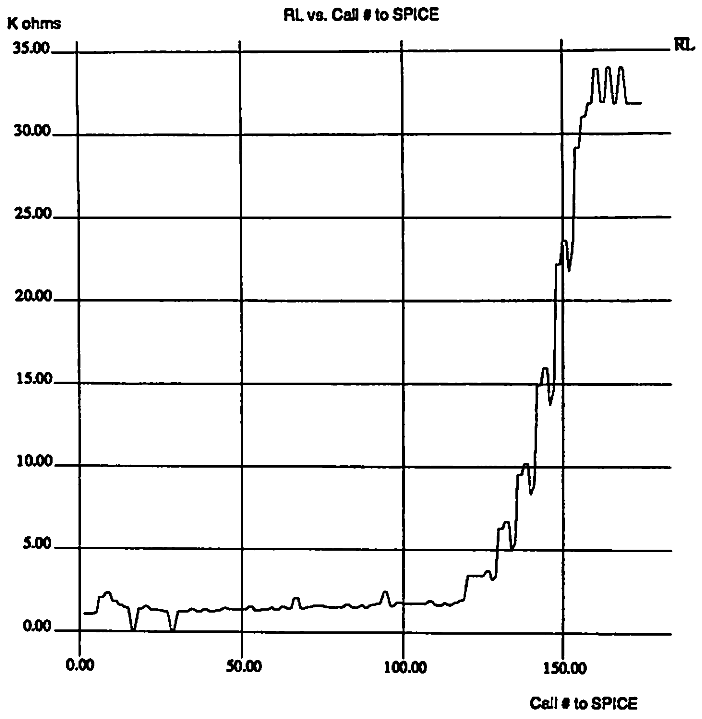

Figure 4.14: Common Emitter Optimization

$$R_{L3} \geq 100\Omega$$

$$R_{E2} \geq 100\Omega$$

$$R_{E3} \geq 100\Omega$$

I_{NOISE}, POWER, f_{-3dB}, GAIN, RIPPLE, DC_{OUT}, POWER, and MAXIN are computed by SPICE. MAXIN is defined as the point where the gain is nonlinear by 5% to the linear gain. The initial conditions were: $R_F = 6.5\ k\Omega$, $W_{M1} = 150\ \mu m$, $R_L1 = 5.2\ k\Omega$, $R_{E2} = 1.7\ k\Omega$, $R_{L3} = 500\ \Omega$, and $R_{E3} = 500\ \Omega$. The chosen values were: $R_F = 6.13\ k\Omega$, $W_{M1} = 144\ \mu m$, $R_L1 = 6.72\ k\Omega$, $R_{E2} = 1.37\ k\Omega$, $R_{L3} = 1.09\ k\Omega$, and $R_{E3} = 1.54\ k\Omega$. It required 3.8 hours of CPU time to run, and required 429 SPICE simulations. The results obtained were: GAIN = 5094Ω, RIPPLE = 0.01, f_{-3db} = 2.32 MHz, DC_{OUT} = 4.32V, POWER = 11.37mW, I_{NOISE} = 32.81nA, MAXIN = 140μm.

4.4.2.4 Inverter Chain

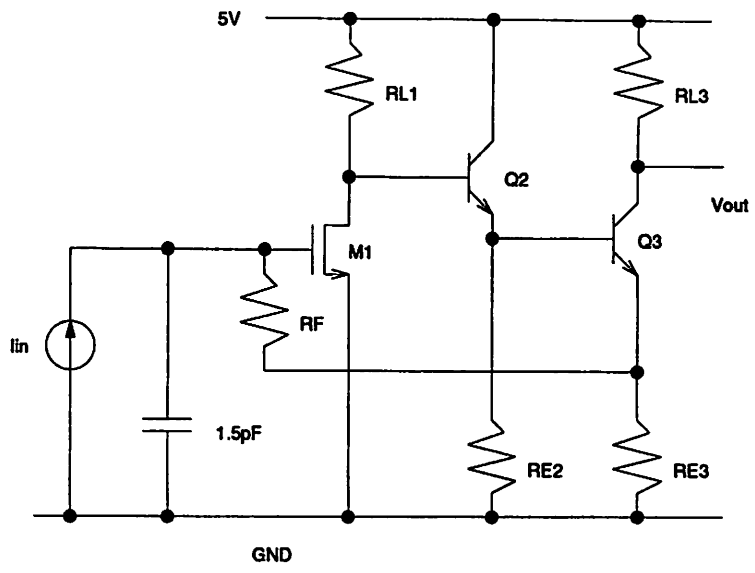

Figure 4.15: Transimpedance Optimization

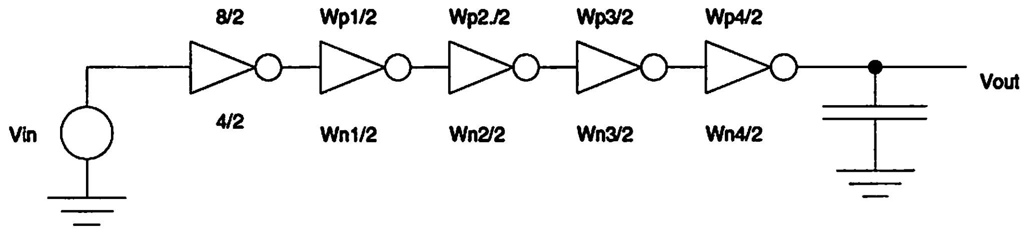

Figure 4.16: Inverter Chain Optimization

Figure 4.16 shows the classic inverter chain problem. The exact specifications for the problem were obtained from ECSTASY [95]. The lower bound and the initial condition for the widths of the transistors was 4 μm. The widths of the PMOS devices were set to be twice the size of the NMOS devices. The objective was to minimize the delay time. Table 4.9 shows the MINOS results compared ECSTASY's. The W's for the ECSTASY run and the SIM value was obtained from [95]. The delay value was obtained by simulating the result.

We present here another illustration of the optimizer's path. Figure 4.17 shows the optimizer's selection of width versus iteration number with a corresponding graph in Figure 4.18 showing the value of the objective function versus iteration number.

Optimizer	W_{n1} μm	W_{n2} μm	W_{n3} μm	W_{n4} μm	Delay ns	CPU min.	SIM
ECSTASY	14.9	56.1	231	1325	5.1	n/a	8
MINOS	15.1	65.1	516	2530	4.7	17.5m	148

Table 4.9: Inverter Chain Optimization Results

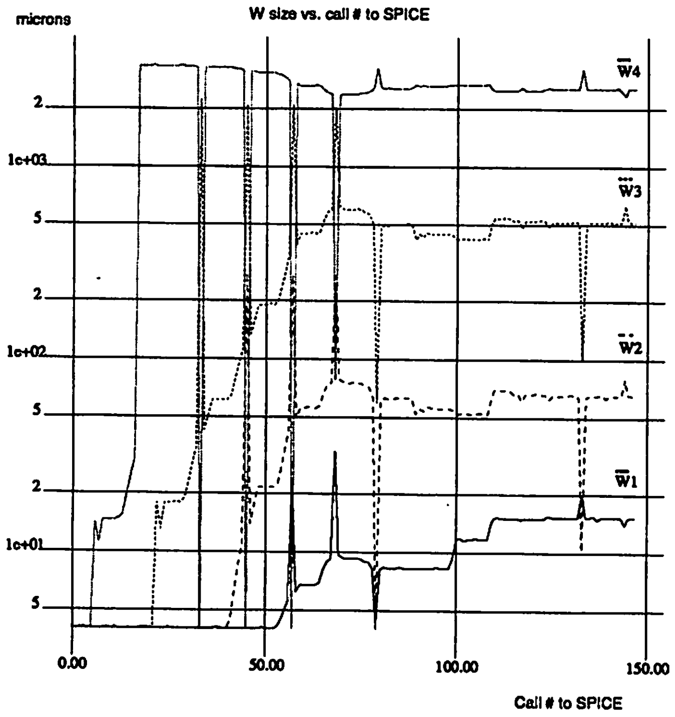

Figure 4.17: Inverter Chain Optimization (Widths)

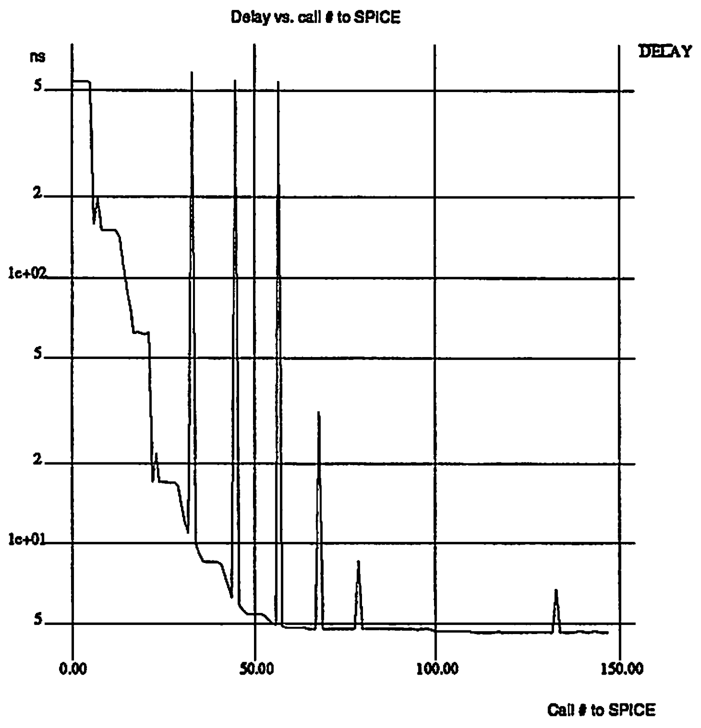

Figure 4.18: Inverter Chain Optimization (Delay)

Chapter 5

Current Source Digital-to-Analog Converter Design Example

5.1 Introduction

Presented in this chapter is the first of a series of industrial strength design examples of increasing levels of design complexity to illustrate the methodology and to show its effectiveness. **Design complexity** is difficult to quantify, because there are many factors to consider, e.g. designer level of expertise, availability of tools, and aggressiveness of performances desired. However, to show where our examples fit into the "big picture," an attempt is made to rank complexity levels in Figure 5.1. Our first example, current source D/A's, falls somewhere in the mid-range for analog circuits.

The complete design flow for an interpolative switched current source D/A converter is given. This chapter presents the following: (1) the design and physical synthesis phases; (2) data on the three fabricated 10-bit D/A converters; and (3) a testing methodology for the fabricated parts.

Using the design paradigm, the design time for this type of D/A has been greatly reduced, and a significant amount of data has been gathered for fault diagnosis. To further speed up the design process, a new module generator for this class of D/A converters has also been developed. As an example of the design speed-up, the second of the two chips fabricated required only *five days* from the time of receiving the design specifications to when it was sent out for fabrication. All three D/A's required only one fabrication run

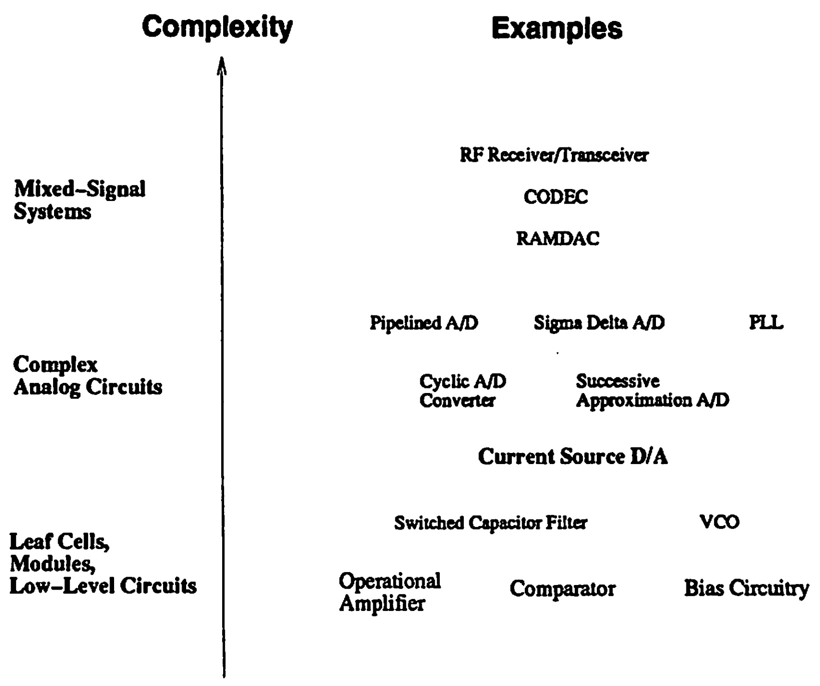

Figure 5.1: Complexity Levels

each.

5.2 Design Specifications

The architecture for the converters that were built is an interpolative switched current source D/A [91] (Figure 5.2). Given an N-bit digital word (D_{in}) and a reference current (I_{ref}), it returns an output current (I_{out}) which is proportional to D_{in}.

The converter has two stages. The first stage consists of unity or linearly weighted current sources each having a nominal value of $\frac{1}{2^{N_1}}$ I_{ref}. This stage decodes the first N_1 bits of D_{in}. Depending on these bits, the currents are either channeled to the output (I_{out}), the current mirror, or the current sink (I_{sink}). If it is determined that n current sources need to be switched to I_{out} then the first n current sources are switched to I_{out}; the $(n+1)^{th}$ current source is switched to the current mirror, and the remaining unused current sources are switched to I_{sink}. The second stage is made of binary weighted current sources with the smallest current source having a nominal value of $\frac{1}{2^N}$ I_{ref}. It decodes the remaining $(N - N_1)$ bits (N_2) of D_{in}. Depending on the remaining bits, these current sources are

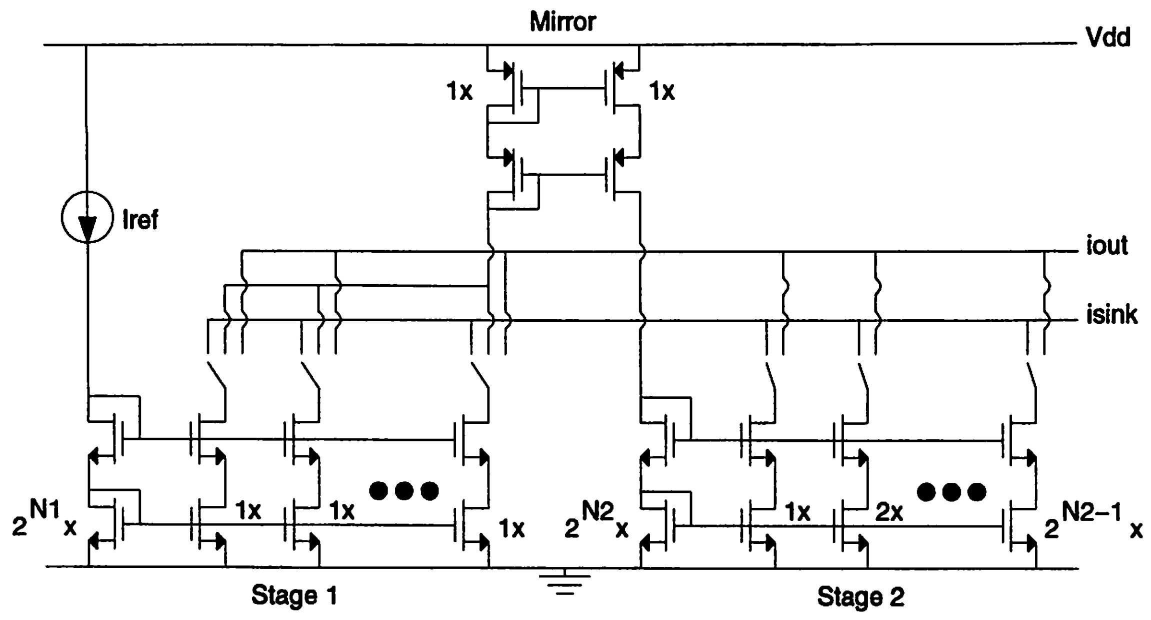

Figure 5.2: D/A Architecture

either switched to I_{sink} or to I_{out}. The currents are summed at the I_{out} node producing the analog output current.

Table 5.1 lists the set of design specifications for the D/A. They are broken down into different categories; each of which is handled differently as will be described later. The user enters these parameters to the D/A module generator.

We chose to design and fabricate three 10 bit D/A converters. Three pairs of N_1/N_2 values were selected: 4/6, 5/5, 6/4. In all cases, we specified that the maximum 3σ bound on INL and DNL to be 2.0 lsb and 0.5 lsb respectively. The output voltage was set at half the supply voltage, and the reference current was set at 1 mA. The MOSIS scalable CMOS (scmos) design rules were used. The target technologies were the MOSIS 2.0μm N-well process and the MOSIS 2.0μm P-well process. Threshold variation data were obtained from [72]. Since the 5/5 D/A converter was a stand-alone part, bonding pad locations were specified. However, the 4/6 and 6/4 D/A's were designed to be part of a larger system (a chip with both D/A's), so I/O terminal locations were specified instead. The overall objective for all of the designs was to minimize layout area.

Type	Specifications
Architecture	N_1, N_2
Performance	Max 3σ bound on INL, max 3σ bound DNL
Operation	Supply voltages, output voltage (V_{out}) range, I_{ref}
Technology	Design rules, SPICE parameters, process variation (σ_W, σ_L), threshold voltage variation ($\sqrt{a_{\mathrm{VFB}}}$) [86]
Layout	Input/output terminal locations, non-default cell placement, minimum wire widths for critical signals, use of bonding pads for a stand-alone part, etc.
Optimization	Non-default cost function(s) Objective for project: Minimize area (default)

Table 5.1: Design Specifications

5.3 Synthesis Path

Figure 5.3 shows the two-level hierarchy used in the synthesis process. The five partitions or blocks into which the D/A has been divided each contain a set of constraints and a set of parameters to be determined by the generator for that block. The synthesis begins at the top-most block and descends by choosing values for the parameters of that block. These chosen values then become constraints for the next block. This continues until all of the final low level parameters (e.g. transistor sizes) are chosen.

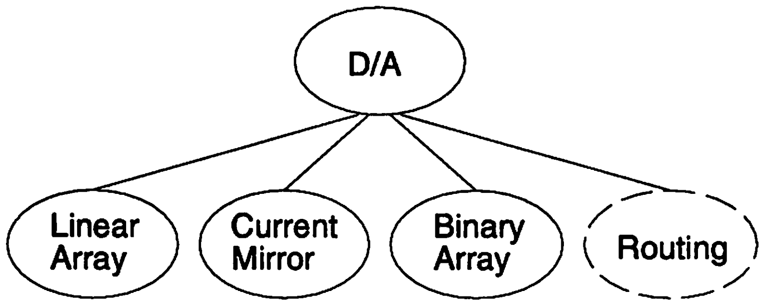

Figure 5.3: D/A Hierarchy

5.3.1 High Level Synthesis

The top block in the design as shown in Figure 5.3 is the D/A. The constraints at this level are the overall D/A performance specifications, INL and DNL. The parameters for this block are shown in Table 5.2. Two tools are applied to select values for these parameters.

Parameter	Description
σ_{iL}, σ_{iM}, σ_{iB}	Standard deviations of the current sources in the linear, mirror, and binary array respectively.
R_L, R_M, R_B	Output resistances of the current sources in the linear, mirror, and binary array respectively.
ΔINL, ΔDNL	Errors introduced from block level routing.

Table 5.2: Parameters for D/A Block

The first is the behavioral simulator [63]. It uses the **intermediate level parameters** (listed in Table 5.2) as inputs as well as some of the architectural, operational, and technological information found in Table 5.1. The simulator returns the statistical INL and DNL characteristics for the D/A converter. A sample output from the behavioral simulator is shown in Figure 5.4 for a 10-bit D/A converter. INL$_{max}$ is approximately 1.5 lsb and DNL$_{max}$ is approximately 0.3 lsb for this particular design. Recall that behavioral simulation is implementation *independent* [37]. These simulations are valid for any current source circuit that can be characterized by the parameters in Table 5.2.

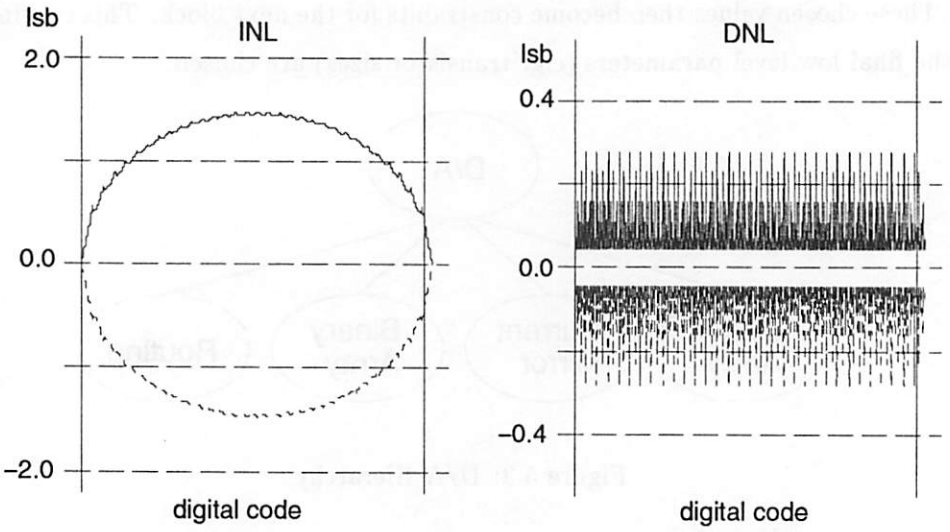

Figure 5.4: INL and DNL behavioral simulation output for a 10-bit ($N_1 = 5$, $N_2 = 5$) D/A

A nonlinear optimizer [78] is the other tool that is invoked. This tool requires the performance and optimization information found in Table 5.1. The performances are used as constraints for the optimization problem. The objective is to **maximize design**

flexibility. From Equation 3.1, the parametric form of the flexibility function for $\sigma_i$is

$$flex_{\sigma}(\sigma) = -a\sigma^2 - \frac{b}{\sigma} + c$$

where $a = 0$. Similarly, the parametric form of the flexibility function for R is:

$$flex_R(R) = -aR^2 - \frac{b}{R} + c$$

where $b = 0$. An alternative approach to the one discussed in Section 3.2 was used to calculate the coefficients for the flexibility functions. One "moderate" difficulty point, assigned a value of 0, was still used. The second point, however, was a desired slope rather than another point. For σ_i, a 1% change from the "moderate" value in mismatch was set to have a slope of +1. While for R, a pre-defined resistance change with respect to r_o, the small signal output resistance for a transistor in saturation, was set to have a slope of -1. This method was felt to be more appropriate. Ultimately, however, this design turned out to be fairly insensitive to the flexibility function. The coefficients are shown in Table 5.3 for $N_1 = 5$, $N_2 = 5$, and I_{ref}=$1mA$. Since mismatch is dependent on N_1, N_2, and I_{ref}, parametric flexibility function were built with these parameters as variables. These are contained in the D/A module generator. Figure 5.5 shows example flexibility functions for

	Coefficients		
Parameter	a	b	c
σ_{iL}	0	1.53×10^{-4}	0.7
σ_{iM}	0	3.72×10^{-3}	0.61
σ_{iB}	0	2.99×10^{-3}	2.59
R_L	8.74×10^{-18}	0	0.45
R_M	1.36×10^{-14}	0	0.45
R_B	1.52×10^{-21}	0	2.52

Table 5.3: Flexibility Coefficients for the 5/5 D/A Parameters

σ_i and R. The optimization problem can be stated as:

$$\begin{aligned} maximize \quad & flex_{\sigma_{iL}}(\sigma_{iL}) + flex_{\sigma_{iM}}(\sigma_{iM}) + flex_{\sigma_{iB}}(\sigma_{iB}) + \\ & flex_{R_L}(R_L) + flex_{R_M}(R_M) + flex_{R_B}(R_B) \\ \text{s.t.} \quad & INL(\sigma_{iL}, \sigma_{iM}, \sigma_{iB}, R_L, R_M, R_B) \leq 2.0 \text{ lsb} \\ & DNL(\sigma_{iL}, \sigma_{iM}, \sigma_{iB}, R_L, R_M, R_B) \leq 0.5 \text{ lsb} \end{aligned}$$

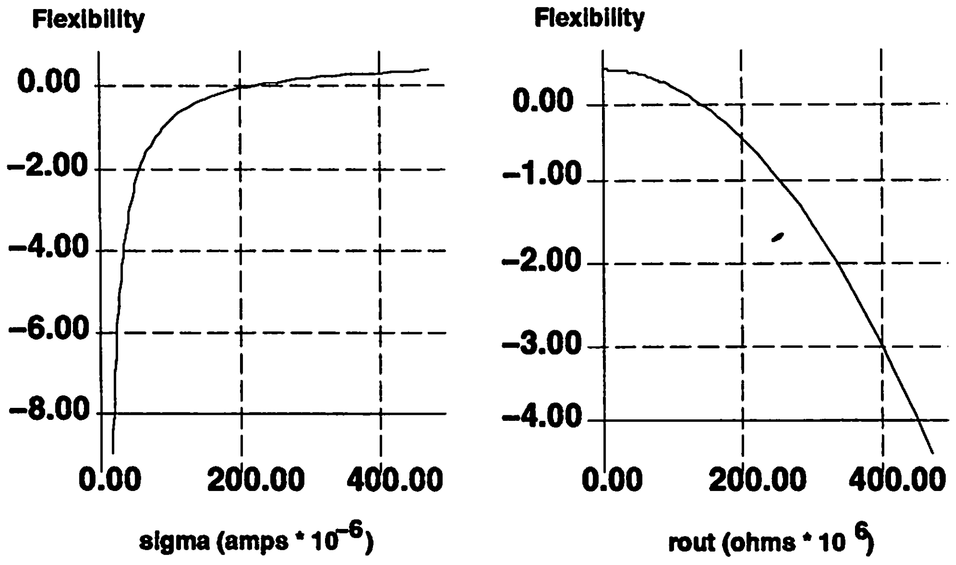

Figure 5.5: Example flexibility functions for σ_i and R.

Upper and lower bounds on the six design variables are also present but not shown in the problem statement. The behavioral simulator and the nonlinear optimizer, working together, chooses numerical values for the parameters in the D/A, solving this problem. The performance of the D/A is evaluated using the behavioral simulator, and more values are chosen. This process continues until all of the performance criteria have been met and the flexibility function has been maximized. The results of this high-level optimization are shown in Table 5.4 for the 5/5 D/A converter. An asterisk (*) denotes that they are the selected values– constraints for the next stage.

Parameter	Value
σ_{iL}^*	16×10^{-5}
σ_{iM}^*	11×10^{-4}
σ_{iB}^*	28×10^{-5}
R_L^*	8.8×10^7
R_M^*	1.0×10^7
R_B^*	2.8×10^9

Table 5.4: Parameter Selection for the 5/5 D/A Block

Extending this result, several optimization on the different N_1/N_2 values were performed to compare the results. This is shown in Table 5.5. These results were used to synthesize a 4/6 and a 6/4 D/A as well as a 5/5 D/A.

N_1/N_2	σ^*_{iL}	σ^*_{iM}	σ^*_{iB}	R^*_L	R^*_M	R^*_B
	(10^{-5})	(10^{-4})	(10^{-5})	(10^7)	(10^7)	(10^9)
3/7	33	2.7	2.9	4.2	1.0	5.3
4/6	23	5.4	9.1	6.0	1.0	3.8
5/5	16	11	28	8.8	1.0	2.8
6/4	11	23	85	13	1.0	2.0
7/3	7.5	50	260	19	1.0	1.4

Table 5.5: Parameter Selection for Various D/A Blocks

5.3.2 Low Level Synthesis

The selected values for the intermediate level parameters now become constraints for the four blocks in the next level of hierarchy. For example, σ^*_{iL} and R^*_L are the constraints for the linear array generator. As long as the linear array generator produces a current source array with $\sigma_i \leq \sigma^*_{iL}$, and an $R \geq R^*_L$, then based on the behavioral level simulation results, it can be asserted that if all of the low-level blocks meet the constraints on σ_i and R then the D/A performance constraints (i.e. INL and DNL) will be satisfied. This assertion is checked during the final extraction and verification phase (Section 5.3.4) at the end of the design cycle.

Block	Parameters
Linear Array	W_{main}, L_{main}, $W_{cascode}$, $L_{cascode}$
Current Mirror	W_{main}, L_{main}, $W_{cascode}$, $L_{cascode}$
Binary Array	W_{main}, L_{main}, $W_{cascode}$, $L_{cascode}$

Table 5.6: Parameters for Low-level Blocks

During the low-level synthesis phase, the linear, mirror, and binary array generators are invoked to select the parameters in Table 5.6, essentially the widths and the lengths of the main and cascode MOS devices for each of the arrays. As before, a simulator and an optimizer are used. SPICE enhanced with a statistical package receives as inputs a set of W's and L's corresponding to the main and cascode MOS devices along with architectural, operational, and technological information from Table 5.1, and returns σ_i and R. A cutting plane algorithm [65] is used for the optimization.[1] The optimization problem is shown below:

$$\begin{aligned} \textit{minimize} \quad & area(W_{main}, L_{main}, W_{cascode}, L_{cascode}) \\ \text{s.t.} \quad & \sigma_i(W_{main}, L_{main}, W_{cascode}, L_{cascode}) \leq \sigma_i^* \\ & R(W_{main}, L_{main}, W_{cascode}, L_{cascode}) \leq R^* \\ & W_{main}, W_{cascode} \geq 4\ \mu m \\ & L_{main}, L_{cascode} \geq 2\ \mu m \end{aligned}$$

The last two constraints are from the design rules. The optimizer, minimizes the layout area subject to the constraints on σ_i, R, and operating condition constraints from Table 5.1. As at the high-level, the optimizer chooses values the parameters, in this case the W's and L's; invokes the circuit simulator to determine whether or not the constraints are satisfied; and continues until the objective function has been minimized and the constraints have been met. The selected values, W^* and L^* are returned. These values are shown in Table 5.7 for the 4/6, 6/4, and 5/5 D/A converters. In total, the low-level optimizer was run nine times, three times for each D/A converter. The 5/5 D/A's were fabricated on a different technology and with different operating conditions as the 4/6 and 6/4. This is why no discernible pattern exists in table as might be expected.

	Linear		Mirror		Binary	
	Main	Cascode	Main	Cascode	Main	Cascode
N_1/N_2	W/L μm/μm	W/L μm/μm	W/L μm/μm	W/L μm/μm	W/L μm/μm	W/L μm/μm
4/6	121/24	118/4	402/146	285/5	48/110	22/4
5/5	21/24	12/9	179/62	42/24	15/48	6/4
6/4	28/21	26/4	48/60	30/4	22/47	16/4

Table 5.7: Low-level Optimization Solutions

The selected values, W^*'s and L^*'s, will satisfy the σ_i, R, and operating condition constraints. Therefore, with the previous assertion on the relationship between σ_i's and R's to INL and DNL, the D/A generated should meet the performance constraints on INL and DNL.

[1]The cutting plane technique was more numerically stable than the algorithm used by MINOS.

5.3.3 Physical Layout

The layout generation phase consists of two steps. First, the three arrays based on the results from the low-level synthesis phase are generated. The linear array, the binary array, and the current mirror generators use templates to create subcells, which are tiled to generate the desired blocks. A standard cell generator was used to synthesize the logic and latches needed for the D/A, and module generator was used to generate the pads. In the second step, these sub-blocks are place, routed, and compacted.

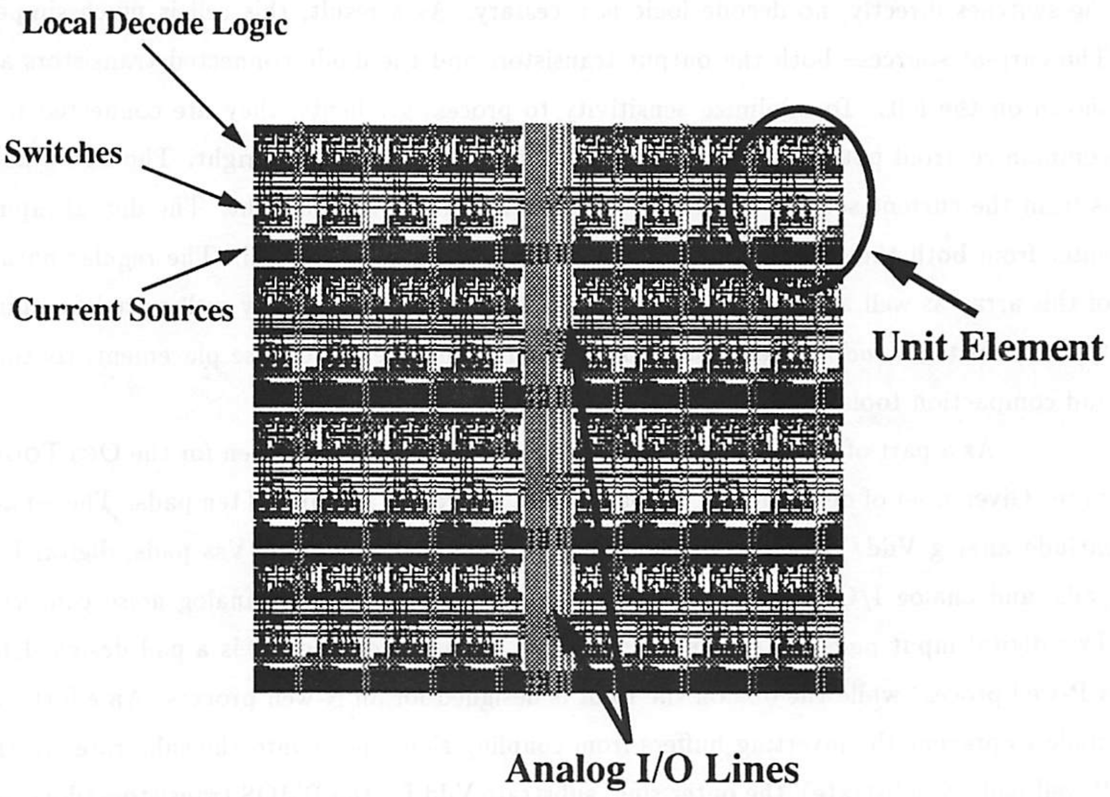

Figure 5.6: Linear Array Layout in the 5/5 D/A Converter

The linear array uses a standard two-dimensional structure [79], with the addition of new decoding circuitry to select the interpolated output. An example array for $N_1 = 5$ is shown in Figure 5.6. It consists of 32 unit elements. A sample element is circled in the upper right hand corner. There are eight unit elements in each row with four rows in the array. To minimize sensitivity to process gradients, the linear array devices are switched using a symmetrical switching pattern [79]. As illustrated in the upper left hand corner, the unit elements are broken down into three parts. The top part consists of local decode logic. This allows a tremendous savings in routing area by performing part of the decode logic locally. The three current steering switches are located in the center. And the current source transistors are located at the bottom with the output transistors one side and the diode connected transistors on the other side. The analog I/O lines run vertically in the center of the array while the digital control and power lines (not shown) are connected from the left and right sides of the array.

The binary array for $N_2 = 5$ is shown in Figure 5.7. Since the binary signals control the switches directly, no decode logic is necessary. As a result, this cell is much simper. The current sources– both the output transistors and the diode connected transistors are shown on the left. To minimize sensitivity to process gradients, they are connected in a common centroid pattern. The current steering switches are on the right. The signal path is from the current sources to the switches to the output on the right. The digital inputs enter from both the top and the bottom on the right side of the cell. The regular nature of this array as well as the linear array makes these circuits especially well suited for tiling type layout tools (*ad hoc* cell generators) rather than general purpose placement, routing, and compaction tools.

As a part of this project, a bonding pad generator was written for the OctTools suite. Given a set of design rules, the pad generator creates a family of ten pads. These pads include analog Vdd/Vss pads, digital Vdd/Vss pads, pad ring Vdd/Vss pads, digital I/O pads, and analog I/O pads. These pads were designed addressing analog noise concerns. Two digital input pads are shown in Figure 5.8. The pad on the left is a pad designed for a P-well process while the one on the right is designed for an N-well process. An effort was made to prevent the inverting buffers from coupling signal noise into the substrate. In the P-well pad (N-substrate), the outer ring, substrate Vdd for the PMOS transistors, does not connect to the power rail for the inverter but to a separate pad (not shown). To further reduce noise coupling, the power rails for the inverters use separate nets, not the internal digital Vdd/Vss lines. Only three power rings are used. As a result, the well is shorted to the power line. A further reduction in noise could have been achieved with the addition of a forth power line, but this was felt not to be necessary. The second D/A test chip fabricated uses these pads.

With complete schematics from the low-level blocks, $\Delta\mathrm{INL}^*$ and $\Delta\mathrm{DNL}^*$ constraints, and layout information from the design specifications, the three analog blocks along

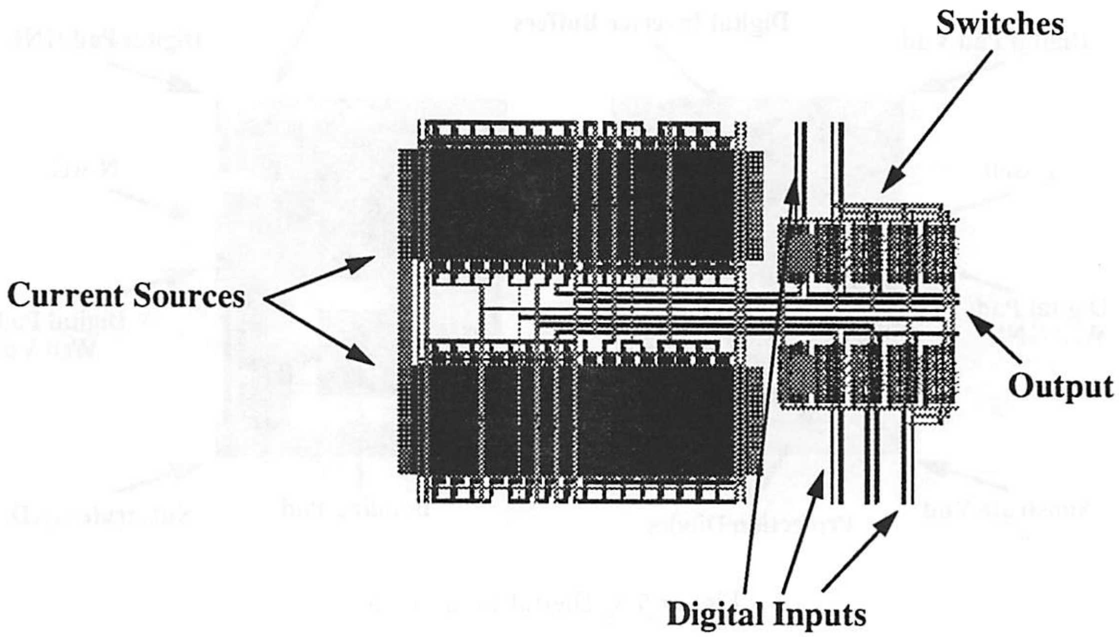

Figure 5.7: Binary Array Layout in the 5/5 D/A Converter

with the digital block, are routed. [63] describes how ΔINL and ΔDNL are calculated.

This complete layout is shown in Figure 5.9. The four main subblocks, the linear array, the binary array, the current mirror, and the control logic, are shown in the interior of the chip. This example illustrates well the point made in Section 2.2 that binary arrays consume significantly less area than linear arrays even when decoding the same number of bits. Bonding pads and bypass capacitors are shown in the exterior areas of the chip. Not necessary for the operation, but useful in debugging, test structures and probe pads are located in the unused areas of the chip.

5.3.4 Extraction and Verification

During this final phase, a full layout extraction including routing parasitics is performed. The results for each block is fed into the circuit simulator which returns σ_{iL}, σ_{iM}, σ_{iB}, R_L, R_M, and R_B. Then this plus ΔINL and ΔDNL is fed into the behavioral simulator which returns the INL and DNL statistics for the D/A generated.[2]

Sensitivity analysis at the behavioral level [63] is used to insure that extra parasitics

[2]All of the simulations required for this step consume on the order of 5 minutes of CPU time on a DECstation 5000/125.

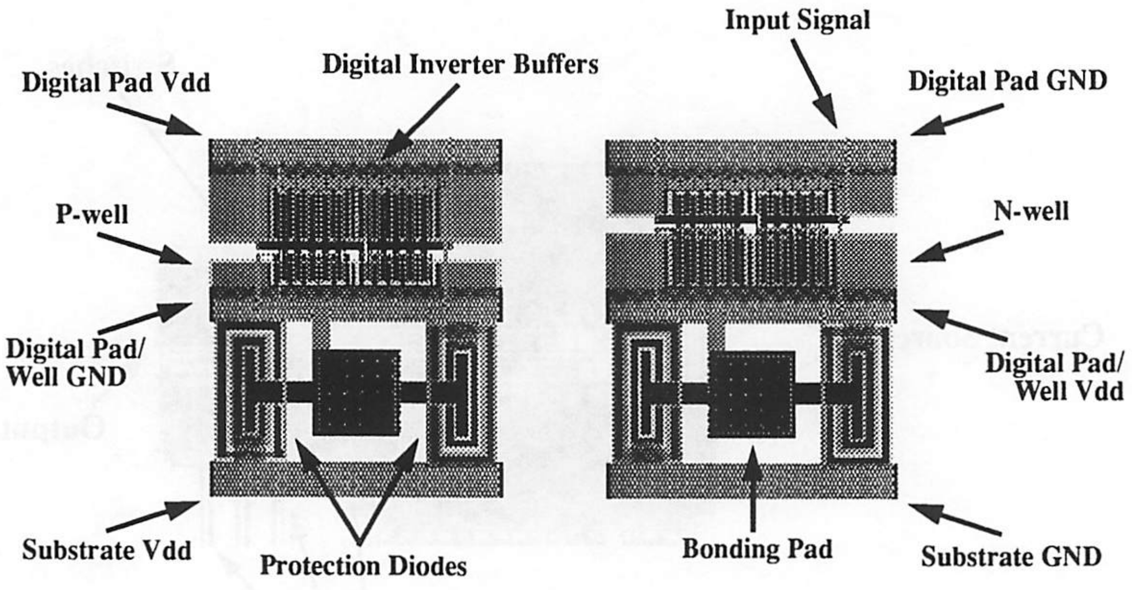

Figure 5.8: Digital Input Pads

introduced in the routing has not degraded the D/A's performance beyond the ΔINL and ΔDNL amount allocated. [63] shows the parasitic insensitive nature of this design with respect to the routing between the blocks.

5.4 Experimental Results

We have fabricated three 10-bit D/As. The first chip (die photo shown in Figure 5.10) fabricated on the MOSIS 2.0μm N-well process, contains a D/A with N_1=5, N_2=5 (5/5) partition. The active area is 1.92mm^2. The second chip (die photo shown in Figure 5.11), fabricated on the MOSIS 2.0μm P-well process, contains two D/A's one with N_1=6, N_2=4 (6/4) and the other with N_1=4, N_2=6 (4/6). The active areas are 2.86mm^2 and 3.81mm^2, respectively. The areas compare reasonably with previously published D/A's [85] [79]. Figures 5.12 and 5.13 depict the measured INL and DNL respectively. The top set of graphs show the data gathered from all of the parts[3] while the bottom set depict only the components which met the performance specifications.

Although not in the design specifications, transient testing on the D/A shows that for a full-scale switch at the output, the D/A can settle to 1 lsb for 10 bits from

[3]Twelve parts for each D/A were fabricated. Three out of the 36 total parts were not graphed due to either profound INL or DNL nonlinearities.

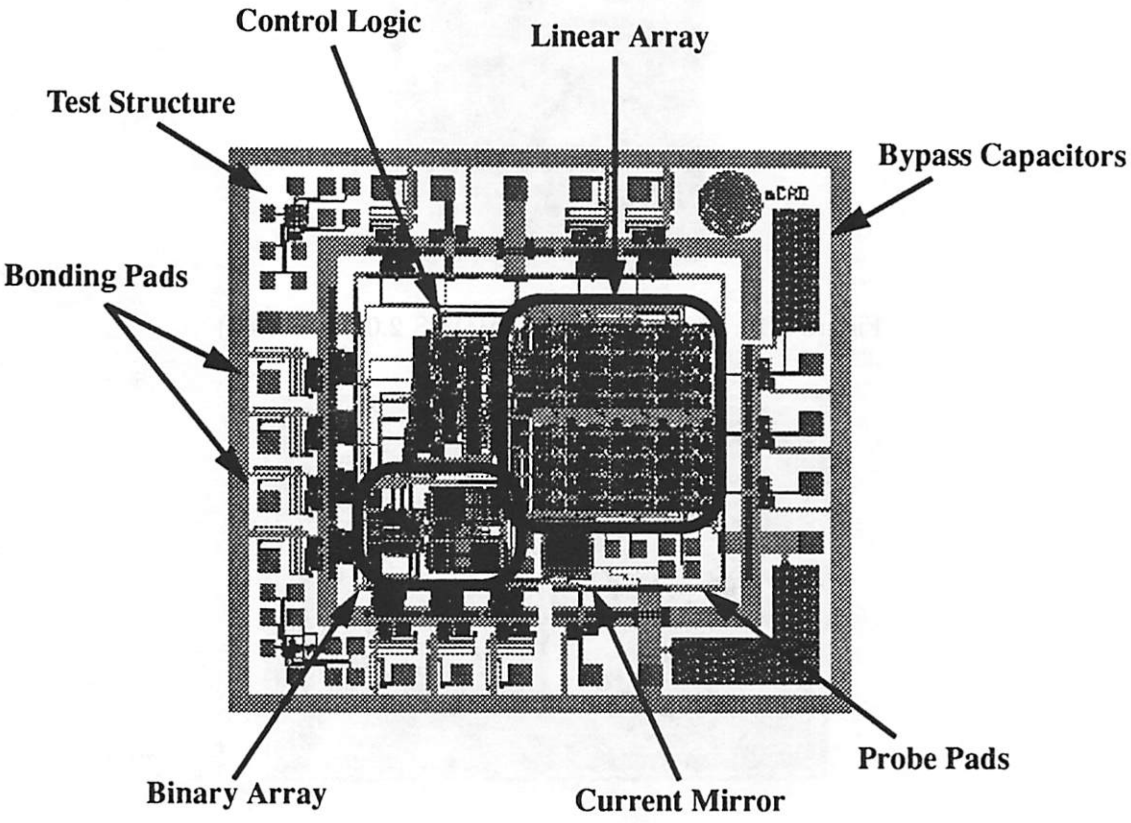

Figure 5.9: Complete Layout of the 5/5 D/A Converter

the start-of-conversion in 20ns. The delay incurred from the falling edge of the clock to the start-of-conversion is 10ns. A plot of this is shown in Figure 5.14. The first falling edge is the clock assertion, and the second edge is the current output fed into a current-to-voltage converter implemented using a 100 MHz operational amplifier with a feedback shunt resistor.

5.5 Mismatch Extraction

In addition to its use during the synthesis phase, the behavioral simulator also provides a vehicle for fault diagnosis once the design has been fabricated, since the behavioral model used captures all of the essential mismatch effects. From the measured INL curves, the mismatch for each component (i.e. current sources) can be derived for all of the experimental chips. According to our model [63], the curves were assumed to be a linear function of the

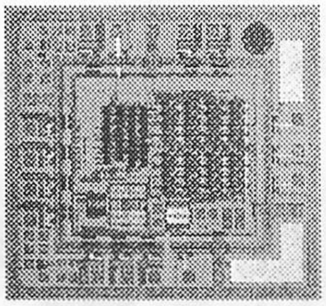

Figure 5.10: One 10-bit D/A (MOSIS 2.0μm, N-well)

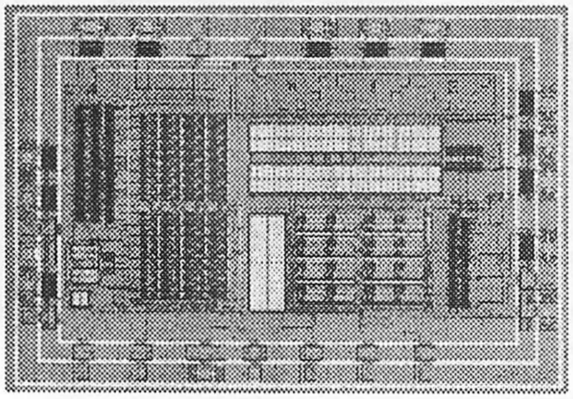

Figure 5.11: Two 10-bit D/A's (MOSIS 2.0μm, P-well)

mismatches in each component. Using linear regression, the *best* estimate can be computed for each mismatch parameter assuming that the mismatch distributions are Gaussian.

For example, shown on the left in Figures 5.15 and 5.16 are the measured INL results for two of the 5/5 D/A's. On the right, the estimated component mismatches are shown. The validity of these estimates is checked by re-computing the INL using the estimated mismatch parameters and putting these values into the behavioral simulator. The re-computed INL curve is then compared to the measured one. The graphs on the left show not only the measured INL values but also the recomputed ones are superimposed. The maximum difference between at any point on the two INL curves between the measured and re-computed values is 0.15 lsb, validating the mismatch estimates.

Having decomposed the INL into component mismatches as described above, causes for the INL characteristic can be pinpointed. For example, in Figure 5.16, the mismatch graph on the right shows that most of the nonlinearity for this chip is due to a

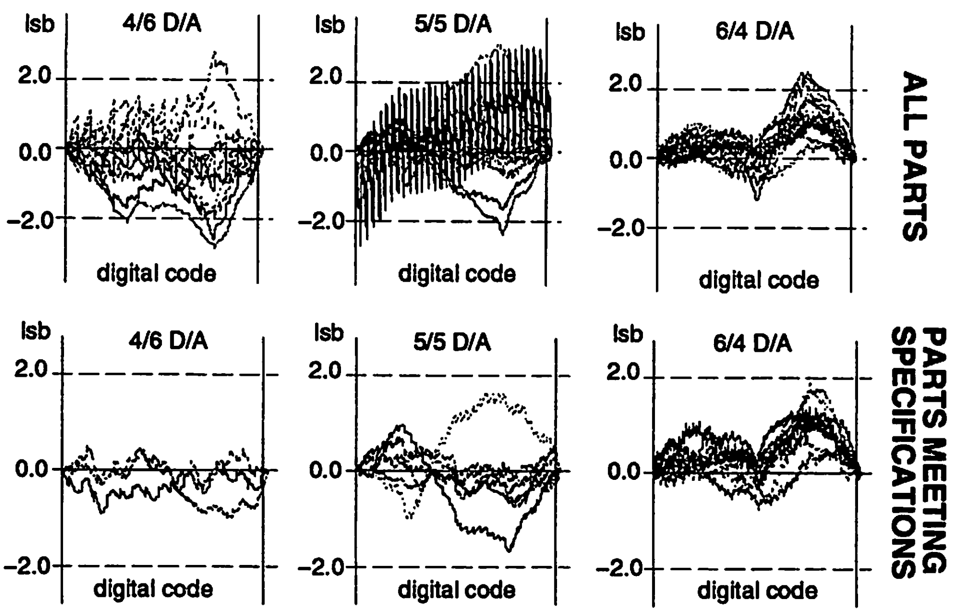

Figure 5.12: Measured INL for the three D/A's

-7% mismatch in the current mirror and a -9% mismatch in the 8x current source in the binary array. Future designs can try to improve upon this by avoiding this problem.

Component mismatches, in general, are caused by underlying basic statistical process variations. In this design, assumed were underlying process variations [86] [72] [54] such as the flat band voltage variations, a_{VFB}, and the standard deviation of the MOS transistor widths and lengths. Equation 5.1 was the expression used to model the flat band voltage.

$$\sigma_{vt}^2 = \frac{\sqrt{a_{\mathrm{VFB}}}^2}{WL} \tag{5.1}$$

Using the component mismatch data measured at two reference currents and using a basic mismatch formula [39], the process mismatches can estimated. Table 5.8 shows the results. The close agreement between the assumption and the extracted results validates the assumptions made during design.

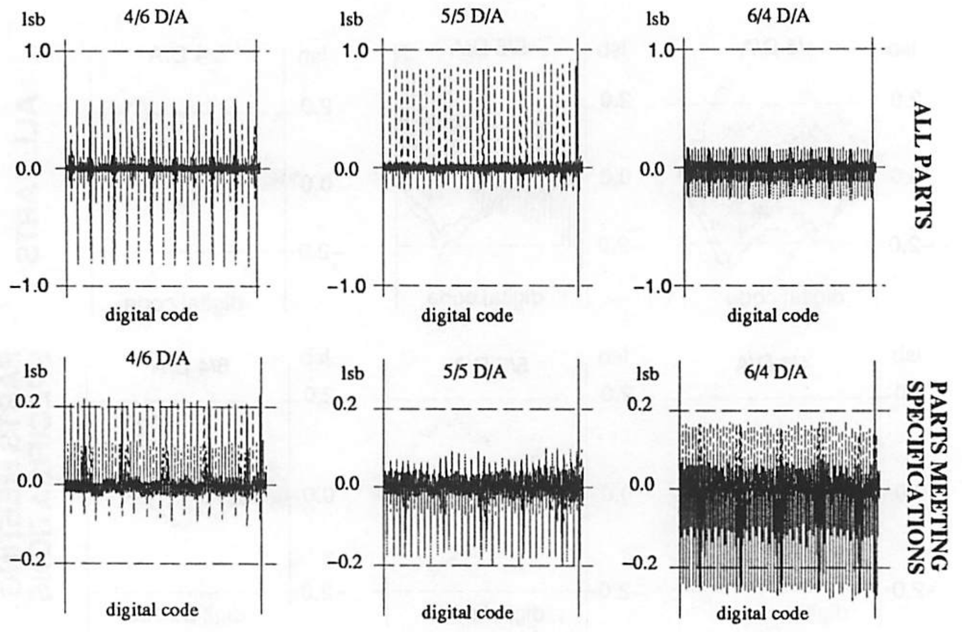

Figure 5.13: Measured DNL for the three D/A's

Constants	units	Assumed	5/5 D/A
$\sqrt{a_{\text{VFB}}}$	V μm	0.0623	0.0555
σ_W	μm	0.05	0.0437

Table 5.8: Basic Statistical Process Variations

5.6 Testing

A testing methodology was developed for testing all DC performance of these converters including offset error, full scale gain error, integral nonlinearity, and differential nonlinearity. In contrast to previous testing strategies based on linear models that require accurate measurements of circuit performance, this new strategy uses a simpler measurement to verify that a circuit performance parameter falls within certain detection thresholds in the presence of measurement noise. Using this new strategy, we evaluated tradeoffs between test set size, test coverage, detection thresholds, measurement noise, chip performance, and estimated yield [64].

The three D/A converters were fully tested against their specifications; the yields are reported in Table 5.9. Since some of the converters did not meet all of the specifications, a fault diagnosis methodology was developed to determine the cause of failure for those

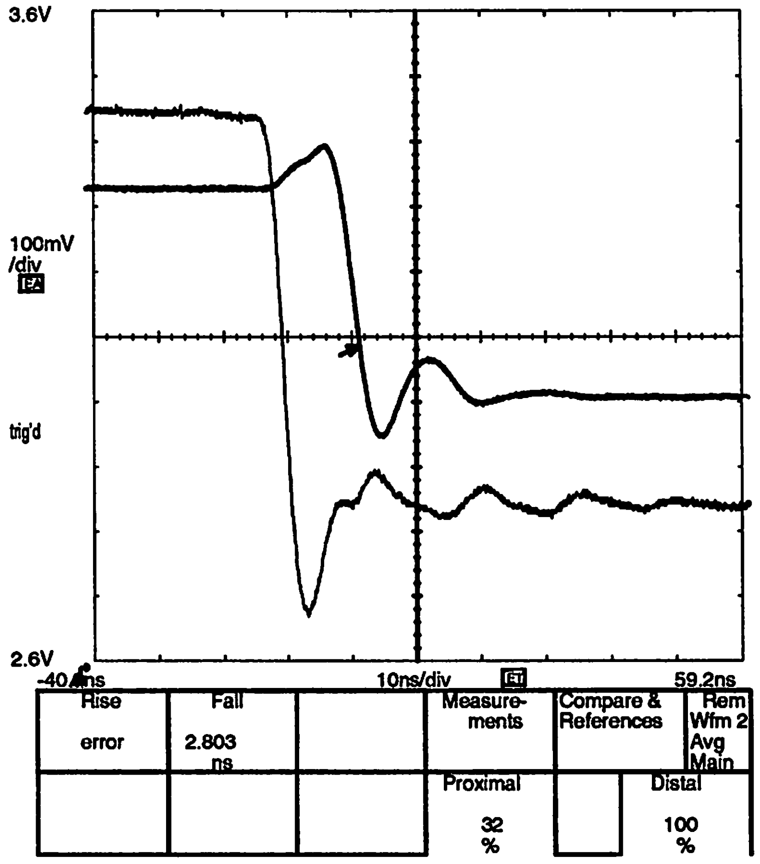

Figure 5.14: Falling Edge for the 10-bit 6/4 D/A

converters which failed one or more tests. A key contribution from this methodology is a linear-time algorithm for determining all of the untestable groups of components in a linear system. These groups are called ambiguity groups. They are untestable because the faults that occur on components within the group cannot be distinguished at the circuit output. The ambiguity groups are found in linear time by computing the null space of the circuit's sensitivity matrix [60]. A ranking of the most critical components is shown in Figure 5.17. Once the ambiguity groups were found and removed from the linear system of equations describing the D/A converter behavior, linear regression was used to find the best estimate for each behavioral model parameter. This information pinpointed the parameters which caused each of the failed chips to exceed the specifications. Using the behavioral model, automatic test patterns were generated for the D/A [34]. These results showed that the D/A's could be characterized in under 40 tests.

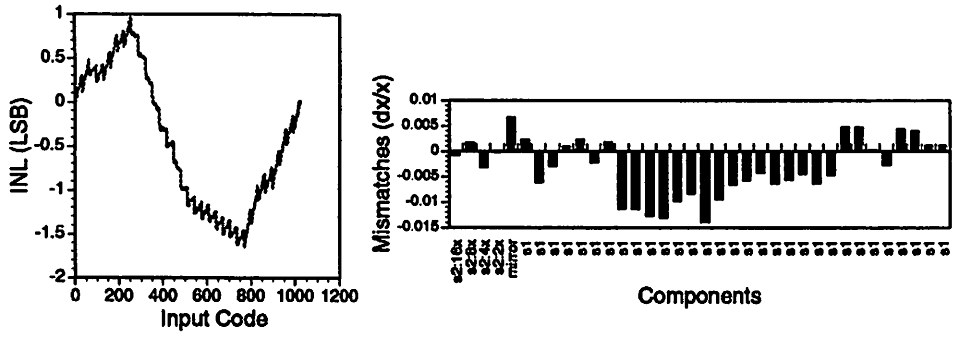

Figure 5.15: Example 1: Measured INL and Component Mismatches

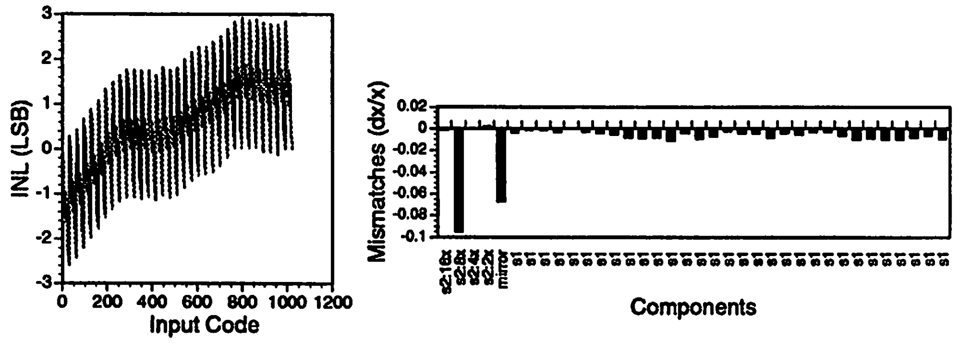

Figure 5.16: Example 2: Measured INL and Component Mismatches

5.7 Design Times

One measure of the success or failure of this methodology is the amount of *actual* design time required for the design and physical synthesis phases. This includes not only decision making times and CPU time but also the time required *to setup the tools*. This is a true measure of time spend on the project. Table 5.10 categorizes these design times for the 5/5 D/A. This shows that the first time through the design cycle required approximately 8 person months (6 months real time).

D/A Converter	# Fabricated	# Which Met Specs
6/4	12	8
5/5	12	7
4/6	12	4

Table 5.9: Yield for three fabricated D/A converters

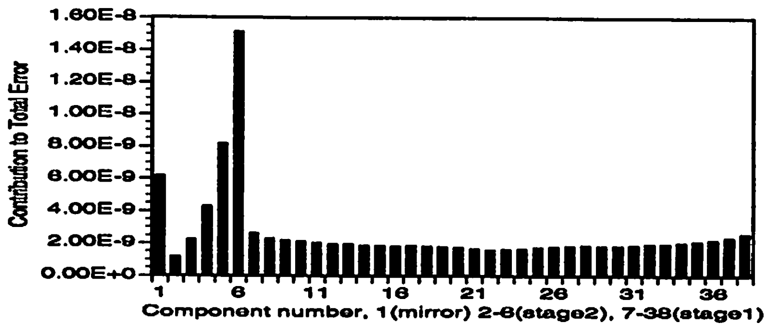

Figure 5.17: Relative Importance of Component Errors

Design Phase	Time Required
Behavioral model development	1 month
High-level optimization	2 months
Low-level synthesis	2 months
Incorporation of new unit cell to layout tool	1 month
Layout generation	2 months
Extraction/Verification	1 week

Table 5.10: Design Times for 5/5 D/A

This was our first cut through *any* serious design problem where behavioral simulation and optimization were involved. Much time was spent on the development of the methodology for the various phases of design. For example, flexibility functions were formulated. So, though this does not seem to give any advantage to hand design, this first design was expected to require more time, and six months is not unreasonable.

The design times for the second design (shown in Table 5.11), however, are much shorter. This reflects the total amount of time required for the generation of two D/A's– the 6/4 and 4/6 converters. Since a module generator had been built during the first design cycle, these times are more of a reflection of the time to set up the tools and of the CPU time consumed by the tools. They provide good estimates on how long it would take to re-do each of these phases given a completely *new* set of design specifications (those found in Table 5.1). The total person time to go from specifications to fabrication was only *five* days.

Design Phase	Time Required
Behavioral model development	4 hours
High-level optimization	10 hours
Low-level synthesis	5 hours
Incorporation of new unit cell to layout tool	2 days
Layout generation	2 hours
Extraction/Verification	5 hours

Table 5.11: Second Round Design Times

5.8 Conclusion

Following the top-down, constraint-driven paradigm, a complete design cycle for a class of D/A's has been performed. The cycle was begun with a set of high-level specifications and low-level assumptions on the technology, and was completed with the verification of both the design specifications and the gathered technology information from the finished products. Two critical measures on the quality of the methodology are shown. The fabricated parts worked the first time, and the design times are shown in Table 5.10 and 5.11.

Future work on the D/A design example include adding transient constraints into the design specifications and the introduction of new circuits such as ones for higher speed or ones with self-calibration based on test results.

Chapter 6

Σ-Δ Analog-to-Digital Converter Design Example

6.1 Introduction

This Σ-Δ A/D design example is the second in a series of design examples that we take through fabrication. We show that even though the circuits and architecture are different from the ones in the first example, the methodology is applied in the same way. The two examples significantly differ for the following reasons: (1) the subcomponents are different, (2) the performances and the reasons for their degradation are different, and (3) the sub-blocks are not in regular arrays. Its level of complexity is shown in Figure 5.1.

6.2 Design Specifications

The Σ-Δ A/D converter that has been implemented is a second order system with a 1-bit quantizer similar to the one found in [3]. The basic architecture of the Σ-Δ is shown in Figure 6.1. Figure 6.3 shows the components of the complete system, i.e. the Σ-Δ converter itself, the clock generator, the bias circuitry, etc. The input anti-aliasing filter and the decimation filter to convert the bit-stream output to data words is to be done off chip. Figure 6.2 defines the symbol used for transmission gates.

The Σ-Δ converter consists of two switched capacitor integrators, a comparator, and a 1 bit D/A. The differential signal enters the system where it is noised shaped by the two integrators. The D/A in the feedback loop drives the output bit stream to be equal to

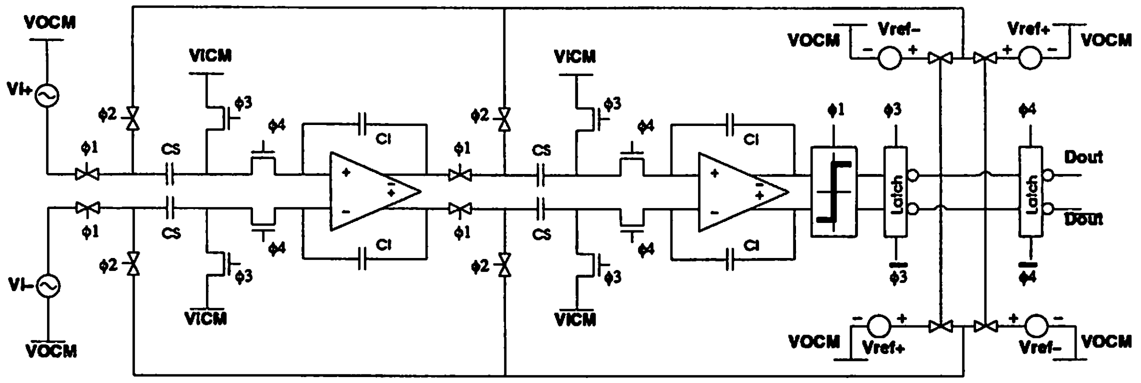

Figure 6.1: Σ-Δ A/D Architecture

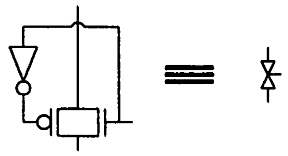

Figure 6.2: Definition of the Symbol for a Transmission Gate

the input signal thus giving the proper A/D conversion operation.

The specifications along with the specific design values used for the designed and fabricated chip are listed in Table 6.1. The performance specifications for our example design were a minimum signal-to-noise (SNR) ratio of 74 dB and a Nyquist input frequency of 250 kHz. The operating supply voltage was chosen to be 5V. The Mosis scmos design rules were used. The target technology was the HP CMOS26B (0.8 μm minimum gate length) process. The bonding pad locations were specified since the A/D was a stand-alone part. The overall objective for the project was to minimize layout area.

6.3 Synthesis Path

The two-level hierarchy used for synthesis is shown in Figure 6.4. During high-level synthesis, specifications for Σ-Δ A/D are mapped onto the integrator, comparator, D/A, digital control logic, and routing. Then in the low-level synthesis phase constraints on these sub-blocks are used to size the schematics.

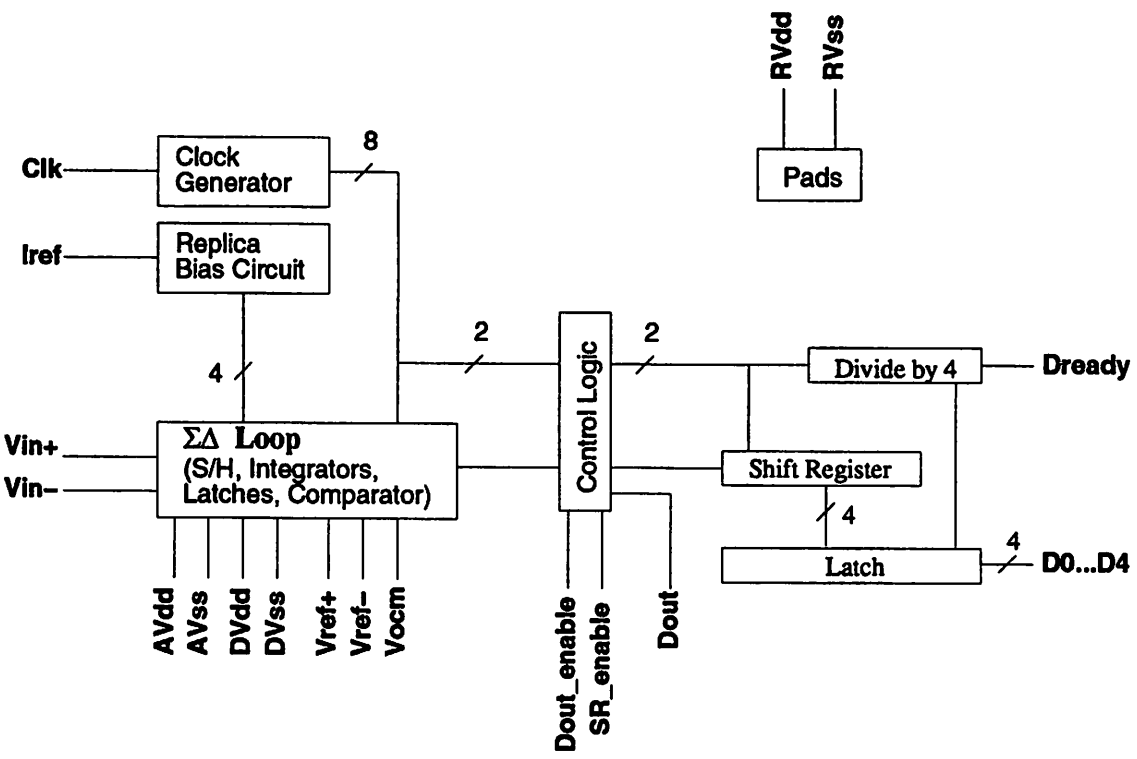

Figure 6.3: Σ-Δ system

6.3.1 High Level Synthesis

The **intermediate level parameters** for the high level synthesis phase are shown in Table 6.2. Given values for each of these parameters along with f_x from Table 6.1, the SNR for the Σ-Δ A/D can be computed using a behavioral model developed for this particular system and the behavioral simulator, MIDAS [3]. Letting x_i be the i^{th} parameter ($x_1 \equiv M$, $x_2 \equiv V_{FS}$, $\cdots$ $x_n \equiv \Delta SNR$), n be the number of parameters (n=18), and c, be a vector containing the additional parameters from Table 6.1, the behavioral simulator can

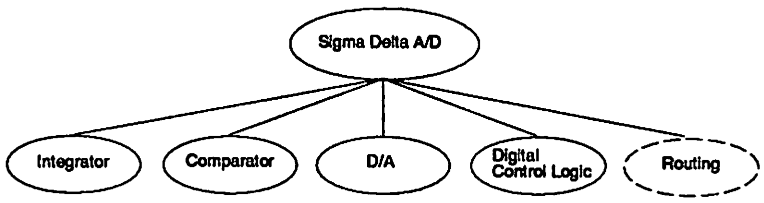

Figure 6.4: Σ-Δ Hierarchy

Type	Specifications	Value
Performance	Minimum Signal-to-Noise (SNR) ratio Nyquist Frequency (f_x)	74 dB 250 kHz
Operation	Supply voltage	5 V
Technology	Design rules SPICE parameters	scmos HP CMOS26B
Constants	Low-level circuit design constants. See Table 6.6	
Layout	Input/output terminal locations, placement, etc.	
Optimization	Non-default flexibility function Objective for project	default minimize area

Table 6.1: Σ-Δ A/D Design Specifications

be represented by the function, g, in Equation 6.1.

$$SNR = g(x_1, x_2, \cdots x_n, c) \tag{6.1}$$

To perform the design decomposition, values for each of these parameters must be chosen. To simplify the problem, reasonable values for the less critical parameters were selected by hand, rather than by optimization. This does not invalidate our approach. The values chosen for these parameters will be constraints just as the parameters chosen by the optimizer will be constraints during the low-level synthesis phase.

Table 6.3 lists the parameters that were hand chosen. To simplify the problem, M, a fairly critical parameter, was chosen *a priori*, because M usually only takes on particular quantized values. These values are determined by how difficult the decimation filter following the A/D converter is to implement. For a given SNR, a reasonable value for M is easily obtainable using Equation 6.2 and knowing the quantized values. The dynamic range (DR) bounds the SNR. L is the order of the Σ-Δ converter. Had it turned out that M was selected to be too low, M would have been increased during the following design pass.

$$DR = 1.5\frac{2L+1}{\pi^{2L}}M^{2L+1} \tag{6.2}$$

As before, the objective of high-level synthesis is to **maximize flexibility** of design. Flexibility functions of the form in Equation 3.1 were developed. The coefficients were determined using the method presented in Section 3.2. Table 6.4 lists the moderate and difficult values along with the coefficients for the flexibility functions. The overall flexibility function is shown in Equation 6.3 where m represents the number of variables

Parameter	Description
M	Oversampling ratio
V_{FS}	Full scale voltage of Σ-Δ system
O_{off}	Offset due to the OTA
O_{range}	Output range of the OTA relative to V_{FS}
O_{th}	Thermal noise due to the OTA
$O_{1/f}$	1/f noise due to the OTA
O_{gain}	Open loop gain of the OTA
I_{τ}	Time constant for the integrator
I_{SR}	Slew rate of the integrator
$I_{kT/C}$	kT/C noise of the integrator
I_{Amis}	Integrator gain mismatch
I_{gain}	Integrator gain
C_{off}	Offset due to the comparator
$C_{hysteresis}$	Hysteresis due to the comparator
C_{noise}	RMS noise due to the comparator
D_{off}	Offset in the D/A
D_{noise}	Noise contribution for the D/A
ΔSNR	Non-characterized degradations to SNR

Table 6.2: Parameters for the Σ-Δ A/D Block

actually being optimized, and where i represents the i^{th} parameter.

$$flex(x_1, x_2 \cdots x_m) = \sum_{i=1}^{m} flex_i(x_i) \qquad (6.3)$$

Equation 6.4 shows the high level optimization problem. u_i and l_i represent the upper and lower bounds on the parameters, x_i.

$$\begin{array}{ll} \max & \sum_{i=1}^{m} flex_i(x_i) \\ \text{s.t.} & \text{SNR}(x_1, x_2, \cdots x_n, c) \geq 74dB \\ & l_i \leq x_i \leq u_i \qquad \forall i = 1..m \end{array} \qquad (6.4)$$

As with other optimization problems encountered in these design examples, MINOS [78] could not solve this problem directly, so the problem had to be reformulated. The principal difficulties were (1) when computing SNR, MIDAS has a large standard error, ϵ, and (2) SNR is extremely nonlinear with respect to its input variables. The large ϵ is due in part to the fact that SNR is computed using short time domain simulations in which random number generators are used to approximate white noise sources. ϵ is approximately 2

Parameter	Value
M	100
V_{FS}	2V
O_{off}	25mV
I_{Amis}	1%
I_{gain}	0.5
C_{off}	100mV
D_{noise}	$20\mu V_{rms}$
D_{off}	100mV
ΔSNR	3.5 dB

Table 6.3: Constant Parameters chosen for the Σ-Δ A/D Block

	Design Difficulty		Coefficients		
Parameter	Moderate	Hard	a	b	c
O_{range}	1.8	2.0	13	0	43
O_{th}	$2.0\mu V_{rms}$	$0.9\ \mu V_{rms}$	0	$16\mu V_{rms}^2$	$8.2\mu V_{rms}$
$O_{1/f}$	$15\mu V_{rms}$	$6.7\ \mu V_{rms}$	0	$120\mu V_{rms}^2$	$8.1\mu V_{rms}$
O_{gain}	250 V/V	900 V/V	$1.4\times10^{-5}\ (V/V)^{-1}$	0	0.84 V/V
I_r	1 ns	0.5 ns	0	$10\ ns^2$	10 ns
I_{SR}	500 V/μs	2500 Vμs	1.7×10^{-6} (V/μs)$^{-1}$	0	0.42 V/μs
$I_{kT/C}$	$20\ \mu V_{rms}$	$6.5\ \mu V_{rms}$	0	$96\ \mu V_{rms}^2$	$4.8\ \mu V_{rms}$
$C_{hysteresis}$	50 mV	20 mV	0	$330\ mV^2$	6.7 mV
C_{noise}	$300\ \mu V_{rms}$	$134\ \mu V_{rms}$	0	$2400\ \mu V_{rms}^2$	$8.1\ \mu V_{rms}$

Table 6.4: Flexibility Coefficients for the Σ-Δ A/D Parameters

dB, significant compared to the SNR requirement of 74 dB. A typical plot of SNR versus an input variable, in this case O_{range}, is shown in Figure 6.5. Because of the unsmooth response surface, MINOS would either get trapped in local minima or iterate excessively while it reduced the step size. The step size can be adjusted to a fixed value in MINOS, but because of the nonlinear nature, the appropriate step size changed depending on the region. Averaging multiple simulations with a varying sine wave input amplitude between 0.37-0.005 to 0.37+0.005, we reduced ϵ to approximately 0.5 dB.[1]

The following is the optimization algorithm that we used:

1. Start with feasible solution, x^*. Let $x_o = x^*, j = 1$.

[1]This reduction was not enough to prevent MINOS from getting trapped.

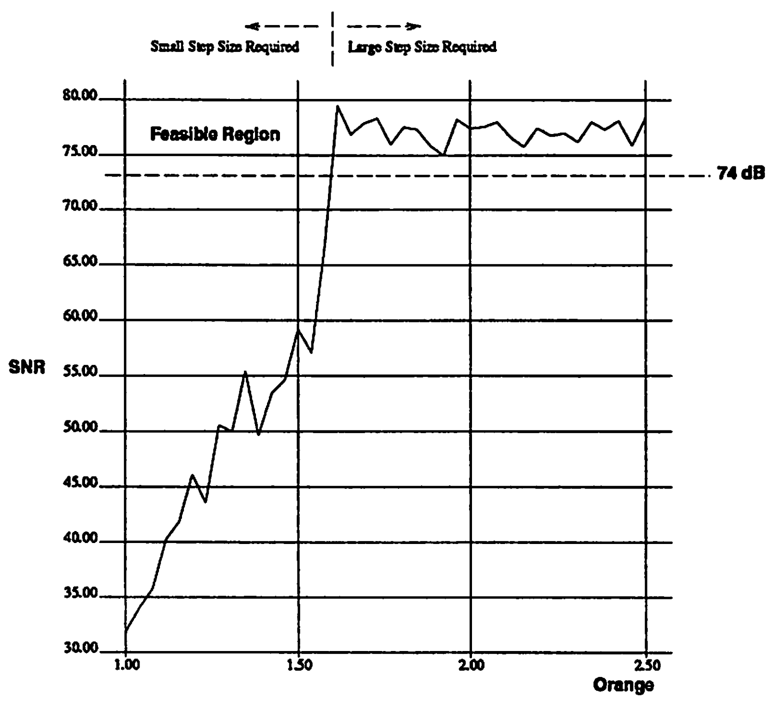

Figure 6.5: SNR versus O_{range}

2. Replace the SNR constraint function with a linearized SNR constraint around x^*.

$$SNR(x) = SNR(x^*) + \left[\frac{\partial SNR(x^*)}{\partial x_1}, \frac{\partial SNR(x^*)}{\partial x_2}, \cdots \frac{\partial SNR(x^*)}{\partial x_m}\right](x - x^*)$$

3. Impose bounds

$$(1 - d_i)x_i^* \leq x_i \leq (1 + d_i)x_i^* \quad \forall\, i = 1...m$$

where d_i is a preset value set by the user (typically 0.1) depending on how nonlinear SNR is.

4. Solve the problem using MINOS.[2] Let the solution be x'_j.

5. For each i, if $x'_{i,j}$ is at the upper bound and $x'_{i,j-1}$ was at the lower bound, or if $x'_{i,j}$ is at the lower bound and $x'_{i,j-1}$ was at the upper bound, then reduce d_i by a factor of two.

6. If $||x_j - x_{j-1}|| \leq \epsilon$, exit with x'_j as the solution.

[2]Since the objective and constraints are now analytic and C^2 smooth within the bounds of interest, MINOS easily solves this problem.

7. Set $j = j + 1$. Goto 2.

The derivatives in Step 2 are computed using finite differences. However, because of ϵ, a derivative was only considered valid when the finite difference caused a 1 dB change (with a certain tolerance) in the SNR. Binary search was used to find this point. 1 dB was chosen, because Δx_i still had to be small, but the change in SNR had to be greater than ϵ. The idea behind this approach is to dynamically control the step size, using the performance change as the measure of whether or not the step size is valid. This allows the optimizer to see overall shape of the curve rather than the shape of the noise.

The solution is shown in Table 6.5. The value for the flexibility function is 28.5 indicating that the overall difficulty of this design should be easier than what was described to be "moderate" in Table 6.4.

Parameter	Value
O_{range}	1.9
O_{th}	87.4 μV_{rms}
$O_{1/f}$	1.05 mV_{rms}
O_{gain}	250 (V/V)
I_{τ}	1.5 ns
I_{SR}	1045 V/μs
$I_{kT/C}$	191 μV_{rms}
$C_{hysteresis}$	101 mV
C_{noise}	9.35 mV_{rms}

Table 6.5: Parameter Selection for the Σ-Δ A/D Block

6.3.2 Low Level Synthesis

With the parameters for the high level design having been chosen, these now become constraints for the low-level synthesis phase. The following sections describe the design synthesis of the basic building blocks.

6.3.2.1 Integrator/OTA

Figure 6.6 shows the schematic of the switched capacitor integrator. Bottom plate sampling is employed to reduce the charge injection from the switches into the signal. This arrangement also allows for level shifting from the output common mode voltage to the input

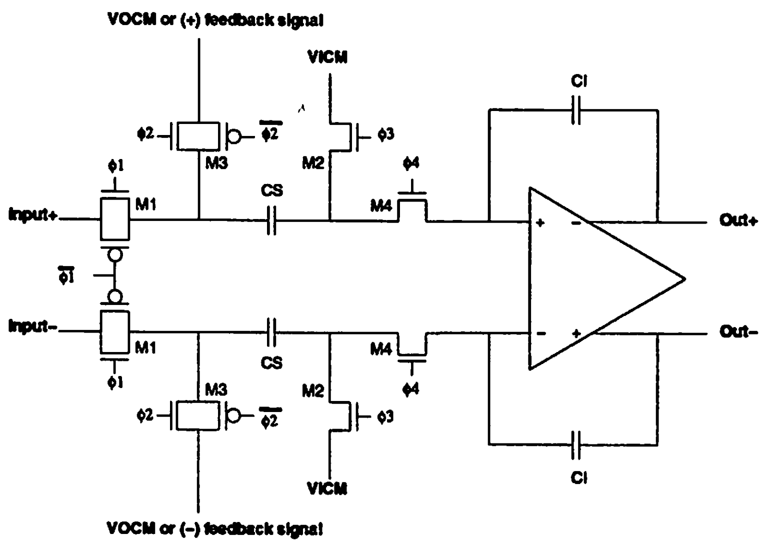

Figure 6.6: Schematic of the Integrator

common mode voltage. C_S is the input sampling capacitor. C_I is the integrating capacitor. This integrator works in two phases. During the first phase, ϕ_1 and ϕ_3 are asserted. The input signal is sampled onto C_S. ϕ_3 is de-asserted two inverter delays before ϕ_1 to allow for the bottom plate sampling. The clock phases are shown in Figure 6.10. Following the de-assertion of ϕ_1, ϕ_2 and ϕ_4 are asserted. The feedback signal is subtracted from C_S and the remaining charge is integrated onto C_I. Again, ϕ_4 is de-asserted two inverter delays before ϕ_2.

A telescopic or unfolded cascode operational transconductance amplifier (OTA) was selected for the integrator. The schematic is shown in Figure 6.7. Its design along with the bias circuitry is similar to the ones in [82]. In this differential OTA, M_1 and M_2 are the input devices. They provide the gain. M_3 through M_6 function as cascode devices to boost the output resistance. M_7 and M_8 in conjunction with M_9 and M_{10} set up the bias current for the OTA based on external bias voltages generated from a bias generator. M_{50} through M_{57} provide a fixed voltage drop to set up the gate voltage for M_3 and M_4 in order to keep M_1 and M_2 saturated. M_{11}, M_{12}, and M_{30}-M_{33} are sized to allow a certain percentage of the tail current to flow down these two bias paths. The transistors driven by the clock lines, ϕ_1, $\overline{\phi_1}$, ϕ_2, $\overline{\phi_2}$, and the capacitors, C_1 and C_2 form the dynamic common mode feedback (CMFB) circuitry [10]. These switches and capacitors set up the output common mode

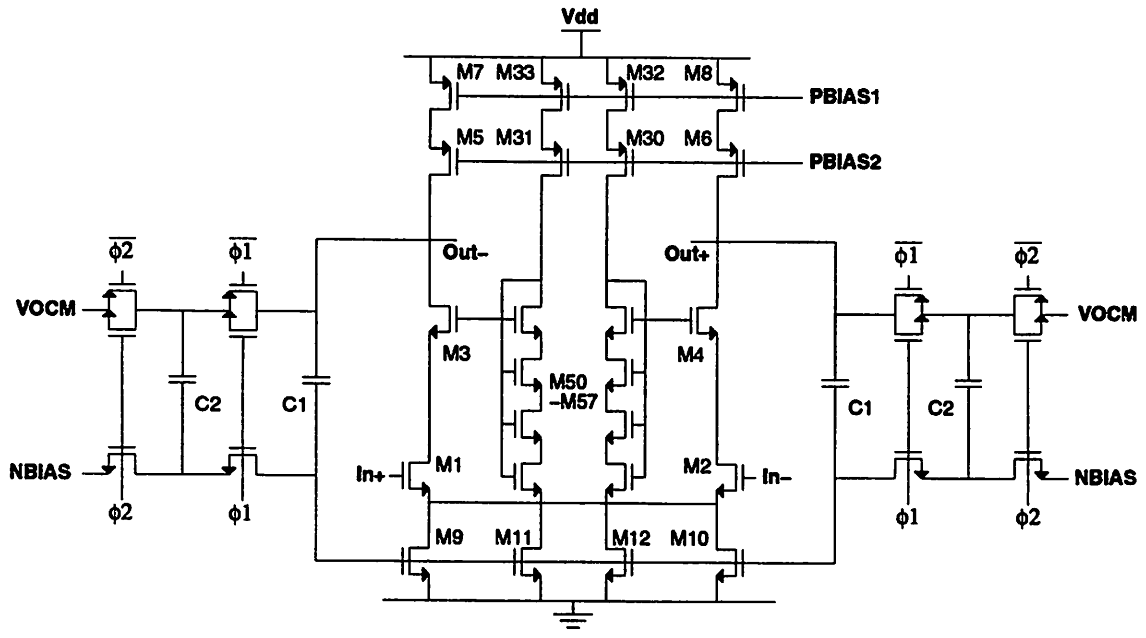

Figure 6.7: Schematic of the OTA

voltage as well as gate voltage for M_9 through M_{12}

A replica bias circuit, shown in Figure 6.8 was used to provide reference and bias voltages for the OTA and integrator. Given an external reference current, the bias circuit generates the gate voltage (NBIAS) for the NMOS tail current devices, the gate voltages (PBIAS1, PBIAS2) for the PMOS current sources in the OTA, and the input common mode voltage (VICM) for the OTA. I_{ref} pulls current from M_0 and M_1 and sets up the voltage, PBIAS1. $M_2 - M_5$ mirror the current for the NMOS devices, $M_{40} - M_{43}$, to allow them to set up the voltage, NBIAS. $M_{10} - M_{13}$ set up the proper gate voltage, PBIAS2, in order to keep the top PMOS transistors driven by PBIAS1 in saturation. Likewise, $M_{30} - M_{33}$ keep M_{40} and M_{42} saturated by setting up a proper gate voltage for M_{41} and M_{43}. M_{21} and M_{22} provide a diode drop up followed by a diode drop down to VICM to give it a low impedance output. Finally, since there are two quiescent solutions for this circuit, M_{20} provides a leak current from PBIAS2 to insure that the proper state is selected. For this prototype design, the leakage current device is left on. This alters the reference current by a small percentage. Since the reference current is controlled externally, the offset caused by M_{20} can be compensated externally.

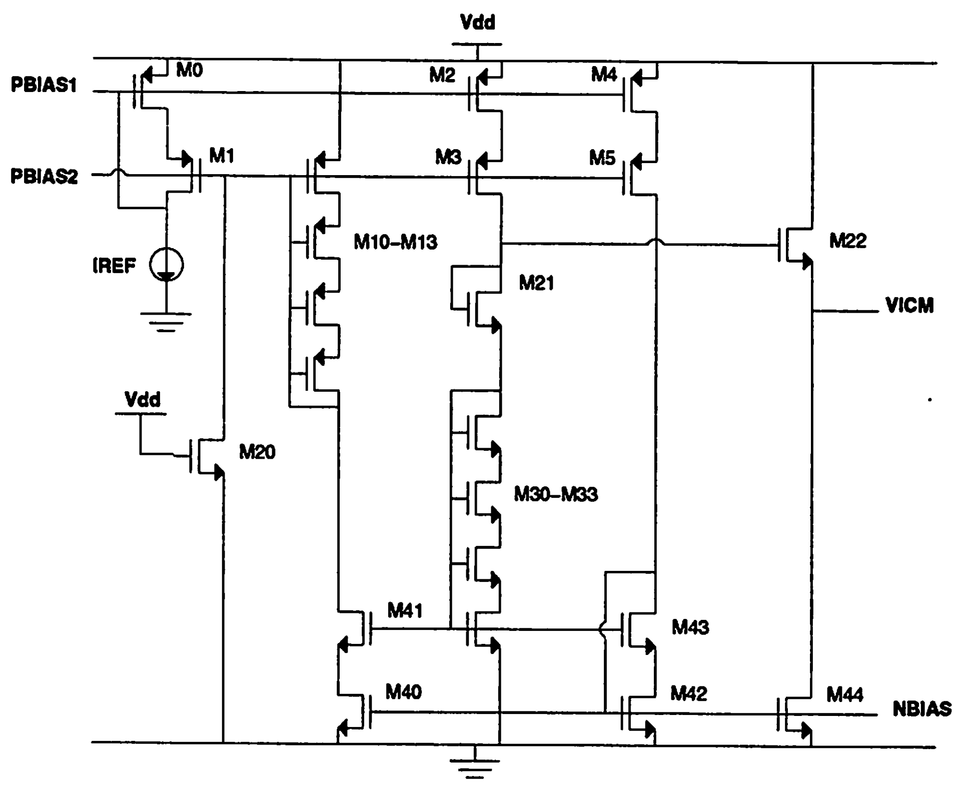

Figure 6.8: Schematic for the bias circuitry

Having chosen the architecture for the integrator and OTA, the task now is to choose component values for the transistors and capacitors in these circuits subject to the constraints in Table 6.5 and some design constants. These design constants are shown in Table 6.6.

A set of design **constraints** are specified for the integrator. These are derived from the high level synthesis results (Tables 6.3 and 6.5). Because analytic methods [25] for determining these constraints differed from MIDAS results, we chose to tighten some of the constraints. This is an example of why we do not propose a fully automatic synthesis process. By allowing us to intervene, we incorporated the value-added of the designer. Three constraints were changed– the thermal noise bound was lowered, kT/C noise bounds was lowered, and the OTA output range bound was raised. Except for the O_{range}, whose flexibility was lowered from -3.9 to -9.0, the other two constraints' flexibilities remained greater than or equal to 0. Thus, we should not have increased the design difficulty by much,

Parameter	Description	Value
V_{safety}	Voltage above V_{ds} where cascode device will be biased	250mV
f_{min}	Minimum frequency for flicker noise calculation	1 Hz
I_{ratio}	Current ratio between $M_1 1$ to M_9 of OTA	0.1
$R_{c1,m9gate}$	Ratio between C_1 and M_9 gate capacitance of OTA	0.3
$R_{c1,c2}$	Ratio between C_1 and C_2 of CMFB circuit	5
p_{leak}	Percent leakage current M_{20} of the bias circuit	0.01

Table 6.6: Design constants for the OTA

but by taking this more conservative step, we better insured a working chip. Table 6.7 lists the complete set of constraints for the integrator. The only constraint not found in either Table 6.3 or 6.5 is common-mode feedback time constant constraint, $O_{\tau,CMFB}$. This was arbitrarily set at 4/3 the value of I_τ. Since it is derived from the high level results, the constraint-driven nature of this design is maintained.

Parameter	Value
O_{off}	25mV
O_{range}	2.0
O_{th}	20.0 μV_{rms}
$O_{1/f}$	1.05 mV_{rms}
O_{gain}	250 (V/V)
I_τ	1.5 ns
I_{SR}	1045 V/μs
$I_{kT/C}$	20 μV_{rms}
$O_{\tau,CMFB}$	2.0 ns

Table 6.7: Design Constraints for the Integrator

Seven variables were selected to describe the design of the integrator. They are listed is Table 6.8.[3] This set is sufficient to generate all component transistor and capacitor values, and is sufficient to characterize each of the design constraints listed in Table 6.7, but is only one of many possible sets. Section 6.7 shows how these variables translate to devices sizes and performances.

In keeping with the Σ-Δ A/D design specifications, the **objective** during this phase is to minimize the active area of the integrator. Active area is computed using

[3]Note that in hindsight, $M_{30} - M_{33}$ probably should have been set to scale with L_{bias} rather than L_{nmos} since it is actually setting up the V_{ds} for the bias transistors.

Parameter	Description
L_{nmos}	Length of transistors $M_1 - M_4$, $M_{50} - M_{57}$ of OTA Length of transistors $M_{21} - M_{22}$, $M_{30} - M_{33}$
L_{pmos}	Length of transistors $M_5 - M_8$, $M_{30} - M_{33}$ of OTA Length of transistors $M_0 - M_5$, $M_{10} - M_{13}$ of the bias circuit
L_{bias}	Length of transistors $M_9 - M_{12}$ of the OTA Length of transistors $M_{40} - M_{43}$ of the bias circuit
L_{M44}	Length of transistor M_{44} of the bias circuit
$V_{gs} - V_T$	$V_{gs} - V_T$ voltage for the transistors in the gain stage
I_{leg}	Current flowing through the transistors in the gain stage
C_S	Size of the sampling capacitor

Table 6.8: Design Parameters for the Integrator

Equation 6.24. It turned out that for this circuit, minimizing area is equivalent to minizing power.

To simplify the integer nature of determining L, the problem was solved in two parts. First, the values of L were determined. L_{nmos}, L_{pmos}, and L_{bias} were minimized subject to the constraint on gain and on common-mode rejection ratio (CMRR) issues. Note that it is critical that L_{nmos} be as small as possible to maximize the speed of the OTA. Enumerating the L's for L_{nmos} and L_{pmos} for $L = 0.8\mu m, 1.3\mu m, 1.8\mu m, 2.3\mu m$ it was determined that in order to satisfy the gain requirements with allowance for a safety margin that $L_{nmos} = 1.3\mu m$ and $L_{pmos} = 1.8\mu m$. L_{bias} was chosen to have a transistor length of 1.8 μm based on CMRR issues, and L_{M44} was determined to require an even longer channel length, 2.3 μm, to boost its output resistance because of the larger V_{ds} across M_{44}.

$$\begin{aligned}
\text{min} \quad & area(V_{gs} - V_T, I_{leg}, C_S) \qquad (6.5)\\
\text{s.t.} \quad & O_{off}(V_{gs} - V_T, I_{leg}, C_S) \leq 25\ mV\\
& O_{range}(V_{gs} - V_T, I_{leg}, C_S) \geq 2.0\\
& O_{th}(V_{gs} - V_T, I_{leg}, C_S) \leq 20.0\ \mu V_{rms}\\
& O_{1/f}(V_{gs} - V_T, I_{leg}, C_S) \leq 1.05\ mV_{rms}\\
& O_{gain}(V_{gs} - V_T, I_{leg}, C_S) \geq 250\ (V/V)\\
& I_{\tau}(V_{gs} - V_T, I_{leg}, C_S) \leq 1.5\ ns\\
& I_{SR}(V_{gs} - V_T, I_{leg}, C_S) \geq 1045\ V/\mu s
\end{aligned}$$

$$
\begin{gathered}
I_{kT/C}(V_{gs} - V_T, I_{leg}, C_S) \leq 20\ \mu V_{rms} \\
O_{\tau,CMFB}(V_{gs} - V_T, I_{leg}, C_S) \leq 2.0\ ns \\
V_{gs} - V_T \geq 100\ mV \\
I_{leg} \geq 100\ \mu A \\
C_S \geq 100\ fF
\end{gathered}
\tag{6.6}
$$

Equation 6.6 shows the **optimization problem** for the integrator. All performances are evaluated analytically (see Section 6.7), resulting in a problem easily solved by standard optimization techniques. MINOS solved the problem in less than 1 CPU second. The solution is shown in Figure 6.9. Table 6.10 shows the performances for the solution

Parameter	value
$V_{gs} - V_T$	330 mV
I_{leg}	1.22 mA
C_S	207 fF

Table 6.9: Low-level optimization problem solution

indicating which of the constraints were active (limiting factors). The active constraints are highly dependent on the high level specifications and the flexibility function.

Parameter	Value	Active
O_{off}	20 mV	
O_{range}	2.0	√
O_{th}	1.2 μV_{rms}	
$O_{1/f}$	9.9 μV_{rms}	
O_{gain}	1336 (V/V)	
I_τ	0.63 ns	
I_{SR}	1045 V/μs	√
$I_{kT/C}$	20 μV_{rms}	√
$O_{\tau,CMFB}$	0.98 ns	

Table 6.10: Constraint values at low-level solution

Using the values determined for $(V_{gs} - V_T)$, I_{leg}, C_S, gate lengths, and the design constants found in Table 6.6, the component values for the OTA, integrator, and bias circuitry are computed. A program was written to generate a complete SPICE deck for the integrator, OTA, and bias circuit once the component values were calculated. Tables 6.11, 6.12, and 6.13 list the results.[4]

Component	Value
$M_1 - M_4$	W=225 μm, L=1.3 μm
$M_5 - M_8$	W=935 μm, L=1.8 μm
$M_9 - M_{10}$	W=310 μm, L=1.8 μm
$M_{11} - M_{12}$	W=31 μm, L=1.8 μm
$M_{30} - M_{33}$	W=93.5 μm, L=1.8 μm
$M_{50} - M_{57}$	W=12 μm, L=1.3 μm
All Switches	W=3 μm, L=0.8 μm
C_1	213 fF
C_2	43 fF

Table 6.11: Component values for the OTA

Component	Value
NMOS Switches	W=4 μm, L=0.8 μm
PMOS Switches	W=9 μm, L=0.8 μm
C_S	207 fF
C_I	414 fF

Table 6.12: Component values for the Integrator

Component	Value
$M_0 - M_5$	W=935 μm, L=1.8 μm
$M_{10} - M_{13}$	W=459 μm, L=1.8 μm
M_{20}	W=2 μm, L=163 μm
$M_{21} - M_{22}$	W=225 μm, L=1.3 μm
$M_{30} - M_{33}$	W=120 μm, L=1.3 μm
$M_{40} - M_{43}$	W=309 μm, L=1.8 μm
M_{44}	W=396 μm, L=2.3 μm

Table 6.13: Component values for the Bias Circuit

[4]The size of transistors $M_{40} - M_{43}$ of the bias circuitry should be but are not identical to the size of transistors M_9 and M_{10} of the OTA. They was due to a layout generation error. $M_{40} - M_{43}$ were divided into stacks of six each while M_9 and M_{10} were divided into stacks of ten. This was caught too late in the design cycle to fix, but it should not have a very large effect. The CMFB circuitry compensates for this small difference.

6.3.2.2 Comparator

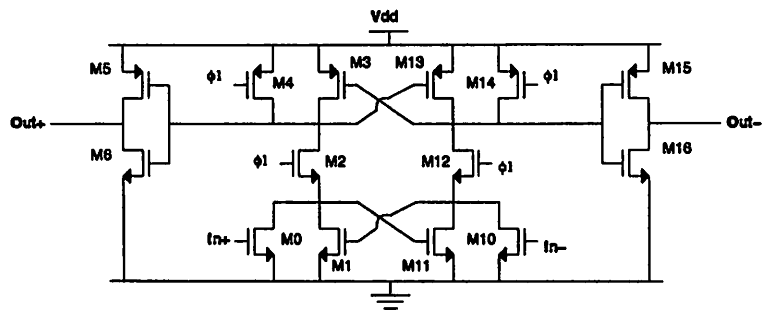

Figure 6.9: Schematic of the comparator used in the Σ-Δ A/D

Figure 6.9 shows the design of the comparator used in this design [3]. During the non-latching phase, when ϕ_1 is low, transistors M_4 and M_{14} are used to reset the comparator while a voltage difference is set up by M_0 and M_{10}. When ϕ_1 goes high, M_1, M_{11}, M_3, and M_{13} form a positive feedback gain stage and boosts the input to full logic levels. M_5, M_6, M_{15}, and M_{16} form inverting output buffers.

The set of **constraints** or specifications required for the design of the comparator is specified in Table 6.14. As with the constraints generated for the integrator and OTA, these are either from Table 6.3 or from 6.5.

Parameter	Value
C_{off}	100 mV
$C_{hysteresis}$	100 mV
C_{noise}	9.35 mV_{rms}

Table 6.14: Design Constraints for the Comparator

Since the design constraints are fairly lax for this comparator, device sizes similar to the switch sizes were chosen for the design. Table 6.15 shows the transistor sizes used in the comparator. These sizes were verified with SPICE and by analytic equations to meet the design constraints.

Component	Value
M_0, M_2, M_{10}, M_{12}	W=6 μm, L=0.8 μm
M_1, M_{11}	W=4 μm, L=0.8 μm
M_3, M_4, M_{13}, M_{14}	W=12 μm, L=0.8 μm
M_5, M_{15}	W=9 μm, L=0.8 μm
M_6, M_{16}	W=3 μm, L=0.8 μm

Table 6.15: Component values for the Comparator

6.3.2.3 D/A

The only components that have to be sized for the 1 bit D/A converter are the switches. These are selected to be the same as the integrator.

6.3.2.4 Digital

There are three digital blocks in this circuit. They are the clock generator, the two latches which follows the comparators, and the shift register at the output for the Σ-Δ A/D.

Four phases and their inverses are required for proper operation of the Σ-Δ system. They are shown in Figure 6.10. ϕ_1 and ϕ_2 are non-overlapping clocks, and ϕ_3 and ϕ_4 are

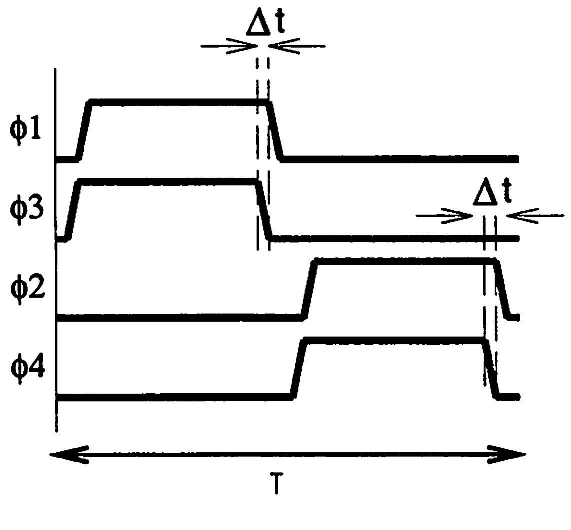

Figure 6.10: Clock Timing Diagram

early versions of ϕ_1 and ϕ_2 respectively. ϕ_3 and ϕ_4 are required for the bottom-plate sampling.

The circuit used for the clock generator is shown in Figure 6.11. Given a single phase clock at the desired frequency at node "CLK," the appropriate phases are produced by this circuit. The "1x" sized inverters provide the delays between ϕ_3 and ϕ_1 and between ϕ_4 and ϕ_2. It also provides for the non-overlap time between ϕ_1 and ϕ_2.

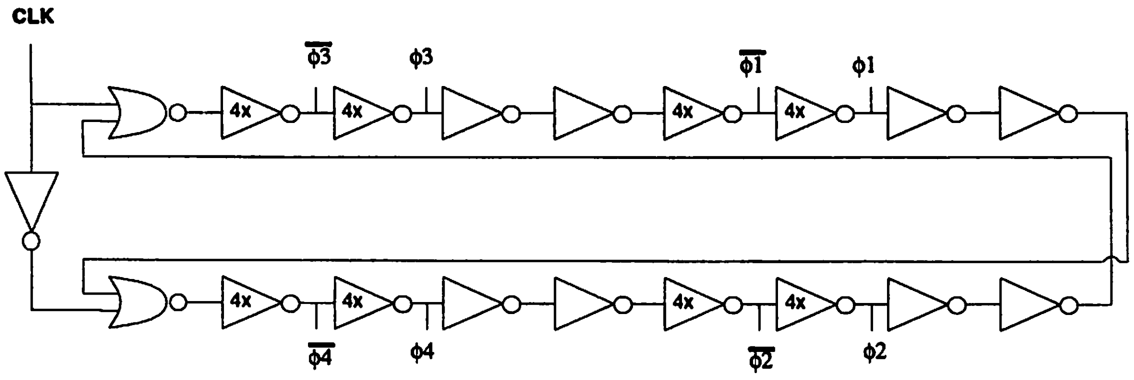

Figure 6.11: Schematic of clock generator

The basic elements of the clock circuit are shown in Figure 6.12. The "1x" sized inverter is shown on the left, and the "NOR" gate is shown on the right. The NMOS device size for the inverters were sized in such a way that the "4x" inverter is the same size as the sum of the sizes of all of the switches that the clock phase with the maximum load drives. The PMOS device size is then scaled by k'_p/k'_n. The NOR gates were sized to have the same driving capacity as the "1x" sized inverter.

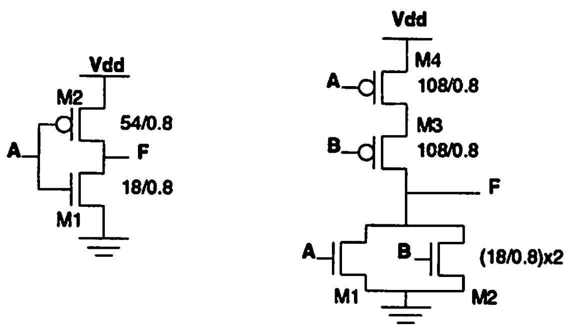

Figure 6.12: Schematic of clock generator elements

The second of the three digital elements is the latch. It is shown in Figure 6.13. It is a basic eight transistor latch. The transmission gate formed by M_1 and M_2 allow the new input to enter. The other transmission gate formed by M_3 and M_4 closes the two inverter feedback loop which holds the input.

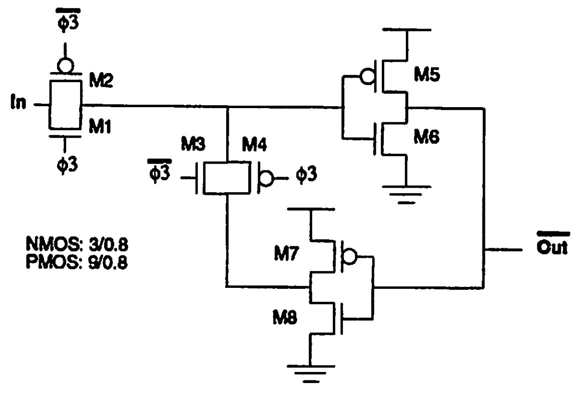

Figure 6.13: Schematic of Latch

The last of the digital elements is the shift register. This was added so that the output would not have to be read at the full speed of the Σ-Δ system. A four bit shift register was used. Because the chip was pad limited, adding any more bits would have increased the layout area significantly. This gave a $4x$ lower frequency requirement at the output. The clock divider and shift register system is shown in Figure 6.14.

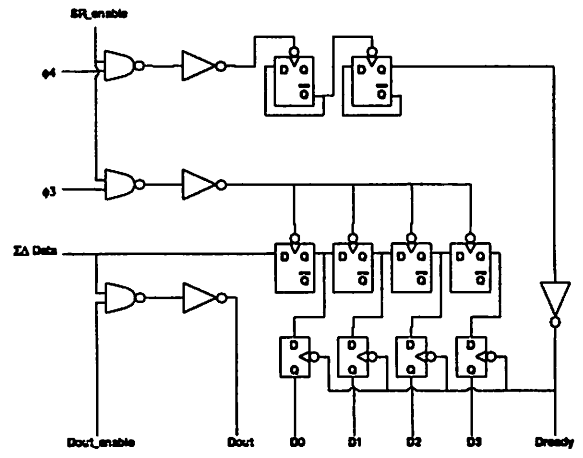

Figure 6.14: Schematic of the Shift Register and Clock Divider

In the schematic, "Dout" represents the actual data coming out of the Σ-Δ A/D at the full clock rate. It can be shut off by de-asserting "Dout_enable." Because, it was unclear how this digital system would affect substrate noise, a shift-register enable line, "SR_enable," was added to turn off the shift register. The shift register latches its input on the de-asserting edge of ϕ_3. The output buffer, the set of four flip-flops directly underneath the shift-register, latches its output on a clock at $f_{clk}/4$ derived from ϕ_4. The shift register data outputs are D_0 through D_3. The assertion of "D_ready" indicates that the data outputs are valid. This cell was implemented with standard cells and was verified to meet timing requirements.

6.3.3 Physical Design

Physical assembly for the A/D converter will be discussed in this section emphasizing the design flow used for leaf (transistor level) cell generation. Most of the layout was automatically synthesized. Issues involving why we had to resort to manual layout for some of the circuits will also be discussed.

We describe the steps necessary for leaf cell generation. A consistent example for an OTA synthesis is presented throughout the steps to better illustrate the concepts.

1. Enter the initial schematic in the form of a SPICE deck. A partial SPICE deck with the input transistors for the OTA is shown below (schematic in Figure 6.7; component sizes in Table 6.11):

```
* OTA
...
M1 n304 inp n308 0 CMOSN W=181u L=1.2u
M3 outm n305 n304 0 CMOSN W=181u L=1.2u
...
```

2. Add analog constraints to the SPICE deck either automatically using PARCAR or manually. The extra lines added: (i) indicate transistor matching, (ii) indicate transistor symmetry, and (iii) split transistors into smaller parallel transistors for decreasing drain capacitance and for canceling linear gradient effects. An example for the OTA is below:

```
...
*SYM M1 M2
*SYM M3 M4
```

```
...
*SPLIT M1 6
*SPLIT M3 6
...
*MATCH M1 M2 M3 M4
...
```

The "SPLIT" commands ask that transistors M_1 and M_3 be "split" into 6 parallel W=30.2μm transistors. The "SYM" commands ask that the two halves of the differential circuit be made symmetric. M_1 will be placed symmetrically to M_2 and M_3 will be placed symmetrically to M_4. The "MATCH" command asks that the transistors listed be placed close together.

3. Generate the **transistor stacks** using MKSTACK [14]. If transistors have the same gate width and share diffusion contacts, they are combined to form stacks of parallel transistors. The M_1/M_3 stack is shown in Figure 6.15.

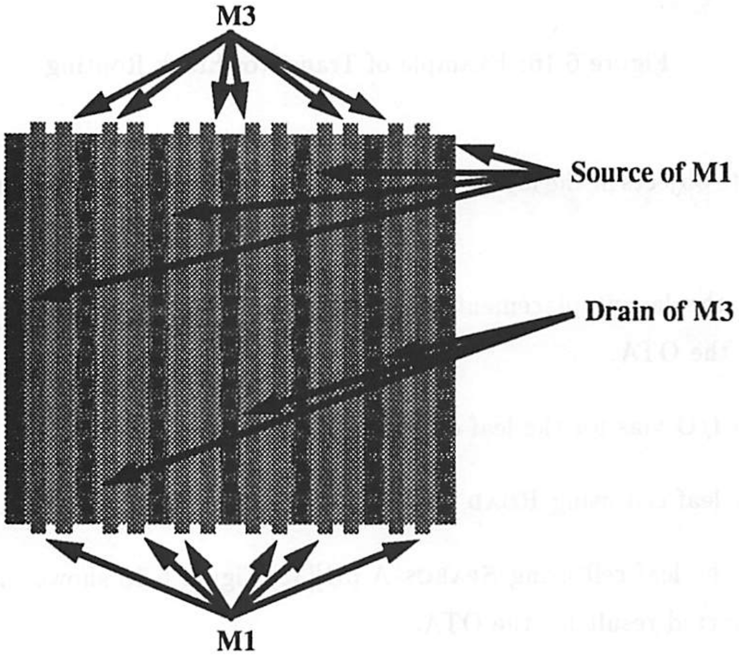

Figure 6.15: Example of a Transistor Stack

4. Generate the capacitors using an available layout tool.

5. Implement the nets of the parallel transistors which are in stacks using another available layout tool. This routing is highly organized. Figure 6.16 shows this for the M_1/M_3 stack. This extra routing step is necessary, because ROAD cannot find this solution.

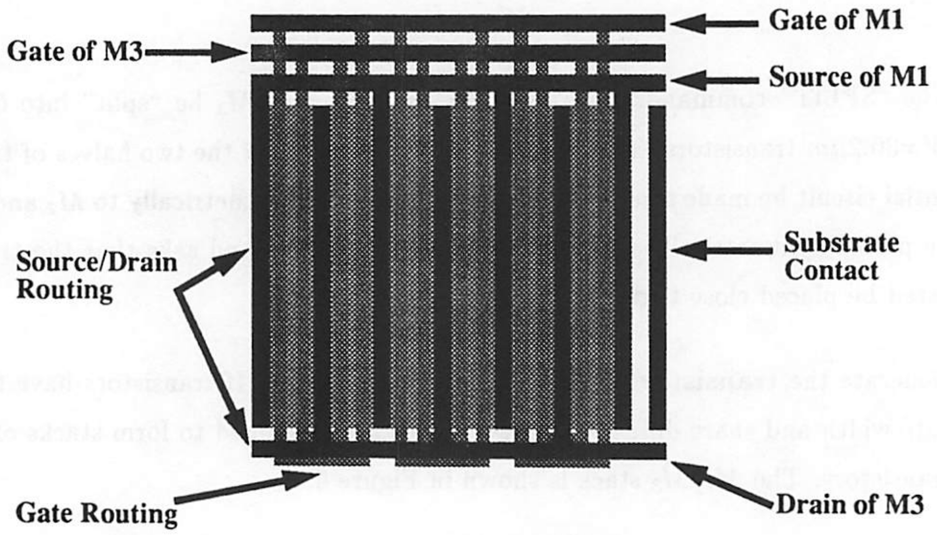

Figure 6.16: Example of Transistor Stack Routing

6. Instantiate objects in the layout and connect the nets symbolically using BDNET [43][44].

7. Generate the layout placement using PUPPY-A. Figure 6.17 shows the placement result for the OTA.

8. Put down I/O vias for the leaf cell using PADPLACE.

9. Route the leaf cell using ROAD [69][70].

10. Compact the leaf cell using SPARCS-A [31][32] Figure 6.18 shows the placed, routed and compacted result for the OTA.

The level of automation possible depends greatly on the level of organization in the circuits. The automatic tools often fail to find desirable solutions for **highly organized circuits**. Once such circuit is the clock generator. Its schematic is shown in Figure 6.11, and the hand layout is shown in Figure 6.21. The layout matches the schematic closely with

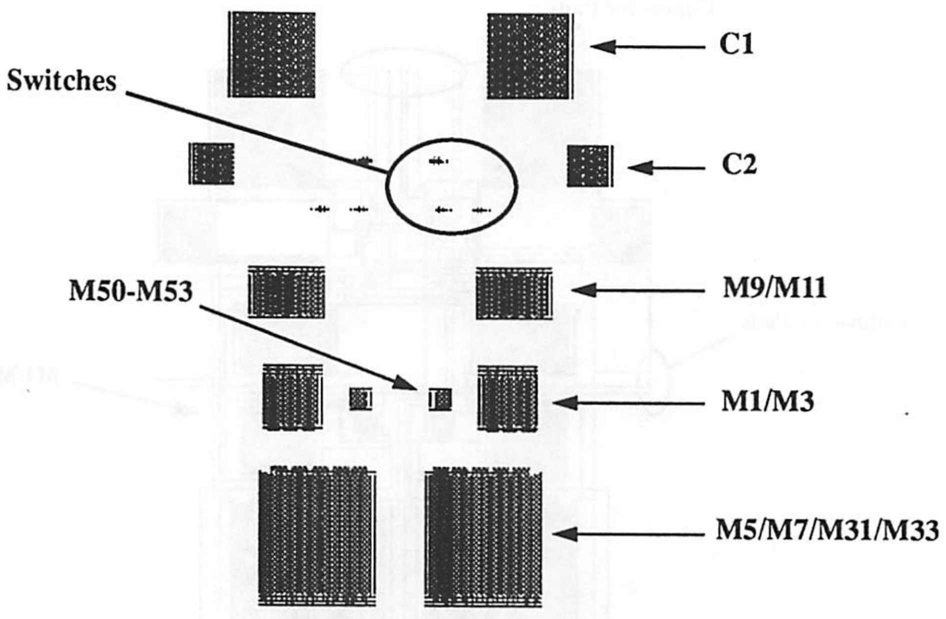

Figure 6.17: Placement by PUPPY-A of the Transistors in the OTA

a great deal of organization in the placement. Each inverter is placed one after the other in the chain in almost the exact manner as was drawn in the schematic. The routing is also very organized. The four power lines run horizontally across the cell so that the inverters and NOR gates can be directly tiled. The four power lines are a result of separating the substrate and well contacts from the supply and ground lines to reduce noise coupling.

Figure 6.19 shows the result from automatic placement. Since ROAD failed to route this cell, an exact area comparison could not be made. However, assuming that it can be routed with *no* expansion (the transistor stacks seem too close together), this cell is already 42% larger than the hand layout. A second attempt was made in which hand placement was followed by ROAD. Figure 6.20 shows the result. SPARCS-A was unable to compact this due to overconstraints. However, even if SPARCS-A were to succeed, there are portions which are not compressible. These are areas where ROAD created a non-optimal topology where an obvious optimal existed. For example, rather than routing two parallel wires which turn a corner in parallel, vias and extra jogs have been inserted, so that one wire overlaps the other. SPARCS-A can only push the vias closer to the other wire. It cannot remove them. These sections will cause the layout to be larger and more irregular than necessary. Without compression, the layout is 250% larger than the hand layout.

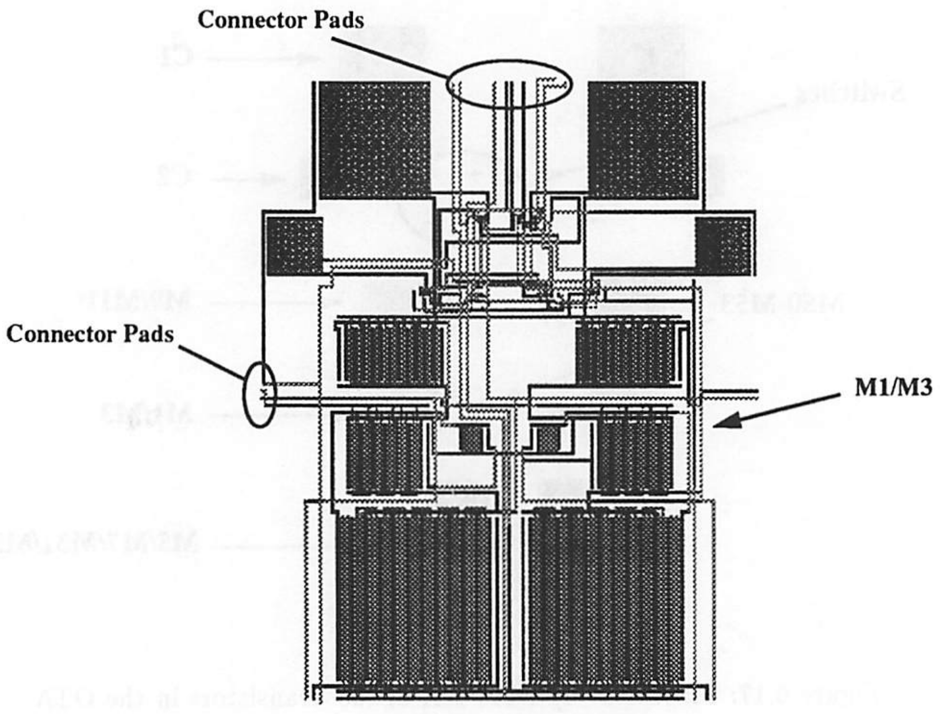

Figure 6.18: Placed, Routed, Compacted OTA

A layout for a circuit which requires **little organization** is the latch layout shown in Figure 6.22. The schematic is shown in Figure 6.13.

A circuit with a **medium level of organization** is the OTA shown in Figure 6.18. The transistors have certain matching and symmetry requirements, however, these can be satisfied using a variety of placements. In this layout, we did intervene by hand during the placement phase. For aesthetic reasons, a couple of the transistor stacks were swapped.

Table 6.16 lists the various circuits in the Σ-Δ converter with the various layout steps enumerated. The tool required for each layout step for each layout block is indicated. An asterisk (*) indicates that slight hand modification were required after the tool was invoked. Custom tools were developed for this project. This is indicated by a $\surd$. "Step" refers the step in the layout flow described previously. "n/a" indicates that for the circuit in question, that step is not applicable. The last column gives the approximate *total* design time required for that layout. The D/A is a combination of the comparator and the latch plus the switches necessary for selecting the proper reference voltages.

The layout for the final chip is shown in Figure 6.23. The major components of the

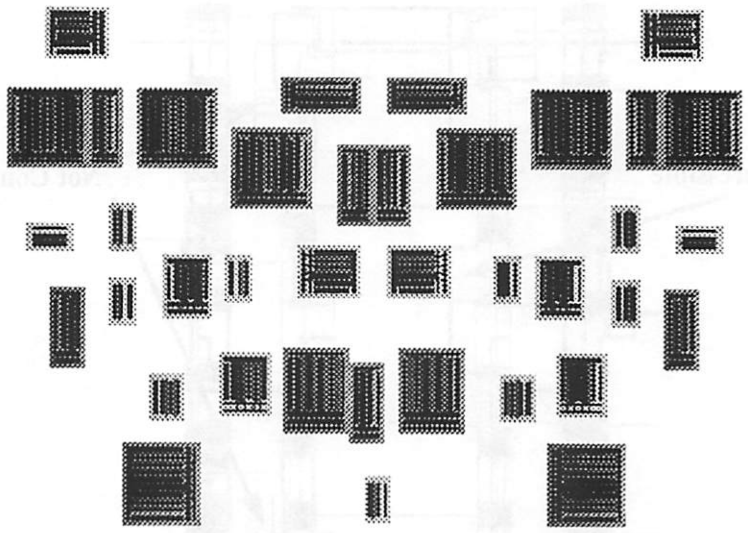

Figure 6.19: Clock Layout: Automatic Placement

integrated circuit are labeled. At the top is the clock generator and the shift register. On the right is the bias circuitry along with the latches and the D/A. The two integrators are in the lower left hand corner of the chip. The area not including the bonding pads (active area), is fairly small, 1.1 mm^2. The total area is approximately 3.1 mm^2.

6.4 Extraction and Verification

We used hierarchical simulation to verify circuit performance, and we used non-hierarchical simulations to verify circuit functionality. In the hierarchical method, each sub-block was extracted. Using SPICE simulations, each sub-block's performances were determined. These performances were fed into MIDAS which returned the SNR for the Σ-Δ A/D. This verified the performance.

In the non-hierarchical method, we extracted the entire circuit, and performed pad-to-pad SPICE transient simulations for a number of clock periods, given a fixed amplitude sine wave at the input. The bit stream result was fed into MIDAS which decimated the output and returned an SNR value which was greater than the one returned from the hierarchical simulations. This was expected, since there were no noise sources in the flat SPICE simulation. The higher SNR value verified the circuit's functionality.

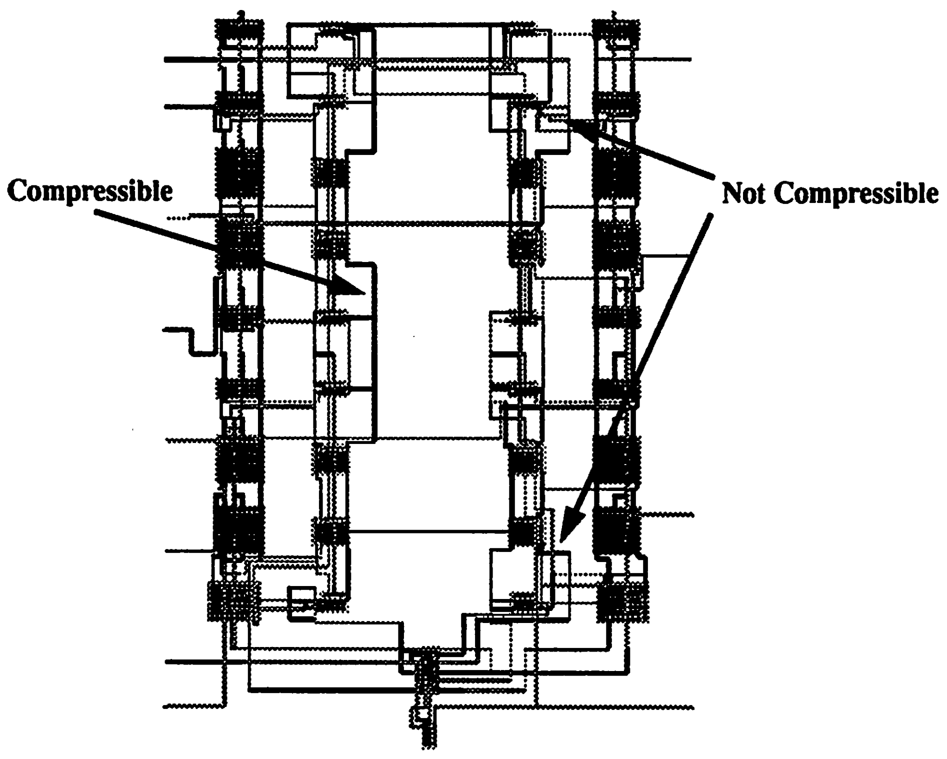

Figure 6.20: Clock Layout: Manual Placement/Automatic Routing

6.5 Experimental Results

This chip was fabricated on a MOSIS 0.8 μm process. A die photo is shown in Figure 6.24. A test circuit board was built consisting of coaxial cable inputs, voltage regulators, by-pass capacitors, etc. Using a sine wave generator fed to a transformer to create a differential signal, a logic analyzer is used to measure the bit stream output from the Σ-Δ A/D converter. This data is then sent to MIDAS which performs the decimation and returns the SNR. Preliminary results are shown in Figure 6.17. It is believed that these results are pessimistic, because (1) we have had trouble increasing the analog input signal's SNR above 73 dB, and (2) we cannot acquire enough sample points using the logic analyzer that we have to get a precise FFT plot. Further tests will be completed.

6.6 Design Times

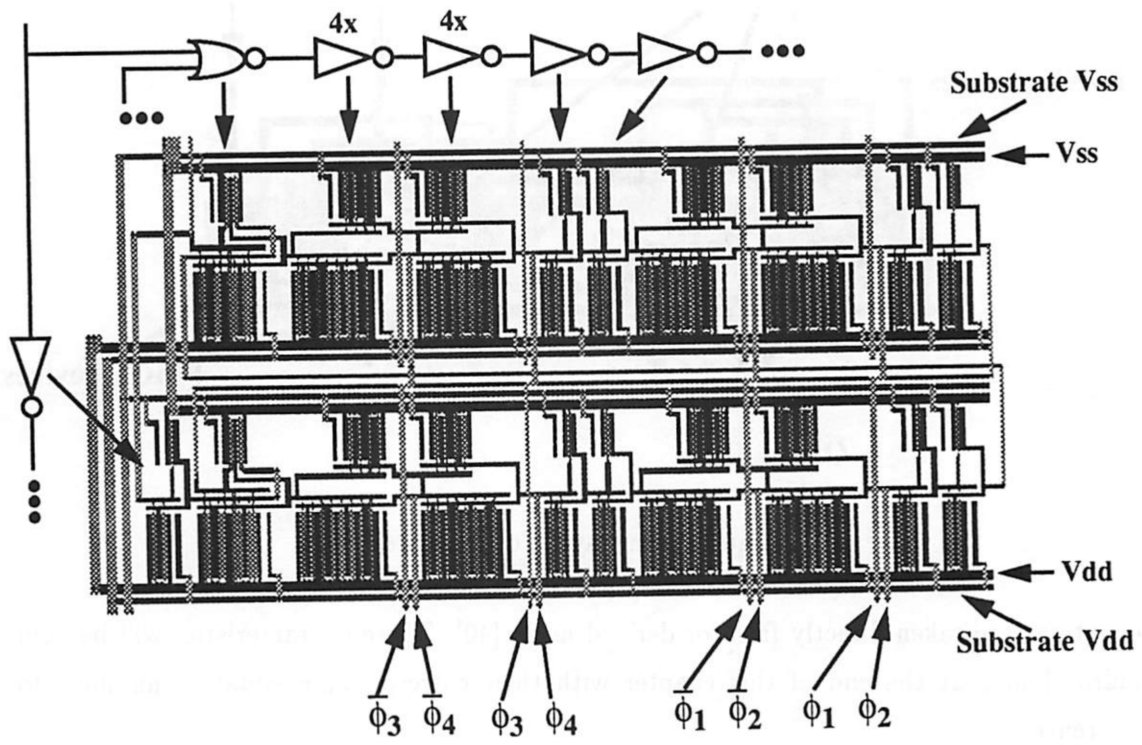
4x
4x
Substrate Vss
Vss
Vdd
Substrate Vdd
$\overline{\phi_3}$ ϕ_4 ϕ_3 $\overline{\phi_4}$
$\overline{\phi_1}$ ϕ_2 ϕ_1 $\overline{\phi_2}$

Figure 6.21: Layout of the Clock Generator

The total elapsed time for this project, design synthesis and physical synthesis phase, was approximately two months. Table 6.18 shows how this time is broken down. These times are intended to give an accurate account of the length of this project from its inception to "taping-out" to illustrate the *real time required* for design when using this methodology.

6.7 Analytic Equations for Σ-Δ A/D

Given the values for the parameters found in Tables 6.19, 6.20, 6.22, and 6.21, a variety of performance approximations and design characteristics can be determined for the integrator described in Section 6.3.2 using analytic equations. These need to be solved during the design process of the integrator/OTA. The flicker noise parameters, K_{fn}, K_{fp}, A_{fn}, and A_{fp}, were determined by fitting curves to device measurements of flicker noise obtained from HP. $\Delta k_n'$, $\Delta k_p'$, and a_{VFB} are estimated values.

This section will show how these characteristics can be calculated. Most of the

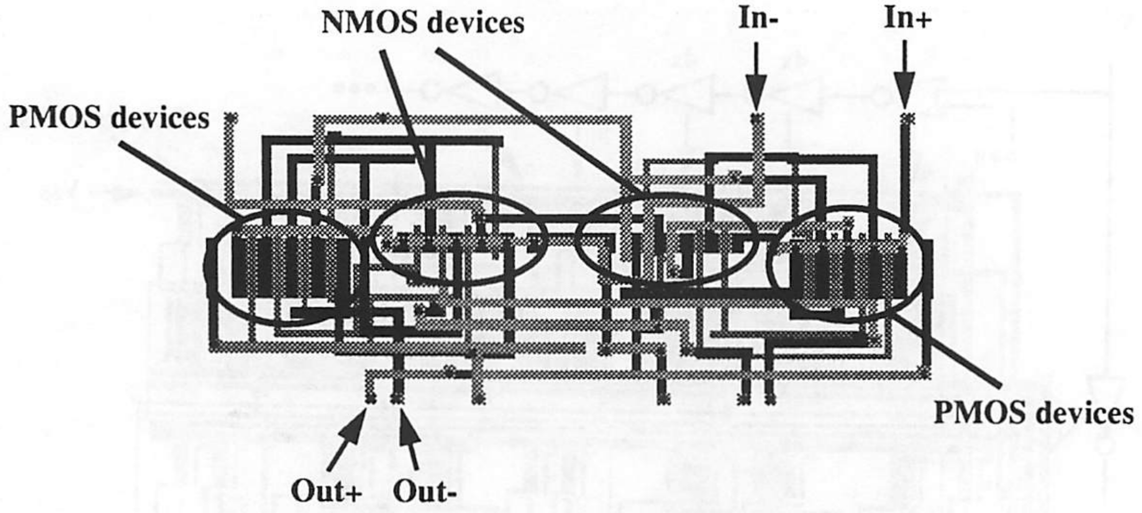

Figure 6.22: Layout of the Latch

equations are taken directly from or derived using [40]. These characteristics will be summarized in a at the end of this chapter with their corresponding equation numbers for reference.

There are nine basic performances that need to be calculated. These are listed in Table 6.7.

The **open-loop gain**, O_{gain}, of the OTA is

$$O_{gain} = g_m \left[\frac{1}{r_{on}(1 + g_m r_{on})} + \frac{1}{r_{op}(1 + g_m r_{op})} \right]^{-1} \tag{6.7}$$

where g_m, the transconductance, is $\frac{2I_{leg}}{V_{gs}-V_T}$, $r_{on} = \frac{1}{I_{leg}\lambda_n}$, and $r_{op} = \frac{1}{I_{leg}\lambda_p}$. The channel length modulation parameters, λ_n and λ_p, were determined by SPICE simulation using transistors with L_{nmos} and L_{pmos} for transistor lengths respectively. λ_n and λ_p are extremely strong functions of L_{nmos} and L_{pmos} for short channels.

The **kT/C noise** of the integrator, $I_{kT/C}$, is

$$I_{kT/C} = \left(\frac{2kT}{MC_S} \right)^{\frac{1}{2}} \tag{6.8}$$

The factor of 2 is present because the circuit is differential with two sampling capacitors.

The equation for **thermal noise** for this OTA (shown in Figure 6.7), O_{th}, is

$$O_{th} = \left[2 \cdot 4kT \frac{2}{3g_{m1}} f_x 2 \cdot \left(\frac{g_{m7}}{g_{m1}} \right)^2 \cdot 4kT \frac{2}{3g_{m7}} f_x \right]^{\frac{1}{2}} \tag{6.9}$$

Circuit	Element Generation Step 3,4	Stack Routing Step 5	Placement Step 7	Routing Step 9	Compaction Step 10	Time (days)
Bias	MKSTACK	$\surd$	PUPPY-A	ROAD	SPARCS-A	0.5
OTA	MKSTACK	$\surd$	PUPPY-A*	ROAD	SPARCS-A	1.0
Integrator	MKSTACK	$\surd$	PUPPY-A	ROAD	SPARCS-A	0.5
Comparator	MKSTACK	$\surd$	PUPPY-A	hand done		0.5
Latch	MKSTACK	$\surd$	PUPPY-A	ROAD	SPARCS-A	0.5
D/A	MKSTACK	$\surd$	PUPPY-A	ROAD	SPARCS-A	0.5
Clock	hand done					0.5
Shift Reg.	digital standard cells		WOLFE [92]			0.5
Pads	<Module Generator from D/A project>					0.1
Chip	n/a	n/a	hand done	MOSAICO [7]	SPARCS	1.5

Table 6.16: Level of Layout Automatic Σ-Δ A/D Converter

Specifications	Value
SNR	69 dB
f_x	165 kHz

Table 6.17: Σ-Δ A/D Experiment Results

In this design since $V_{gs} - V_T$ is the same for all of transistors in the main current legs, $g_{m1} = g_{m7} = g_m$, thus Equation 6.9 reduces to:

$$O_{th} = \left(4 \cdot 4kT\frac{2}{3g_m}f_x\right)^{\frac{1}{2}} \tag{6.10}$$

The **output voltage range** of the OTA is

$$V_{range} = 2\left(V_{o,max} - V_{o,min}\right)$$

where $V_{o,max} = V_{dd} - 2(V_{gs} - V_T) - 2V_{safety}$ and $V_{o,min} = 3(V_{gs} - V_T) + 3V_{safety}$. The factor

Design Phase	Time Required
Behavioral model development	4 days
High-level optimization	15 days
Low-level synthesis	5 days
Layout generation	10 days
Extraction/Verification	5 days

Table 6.18: Design Times

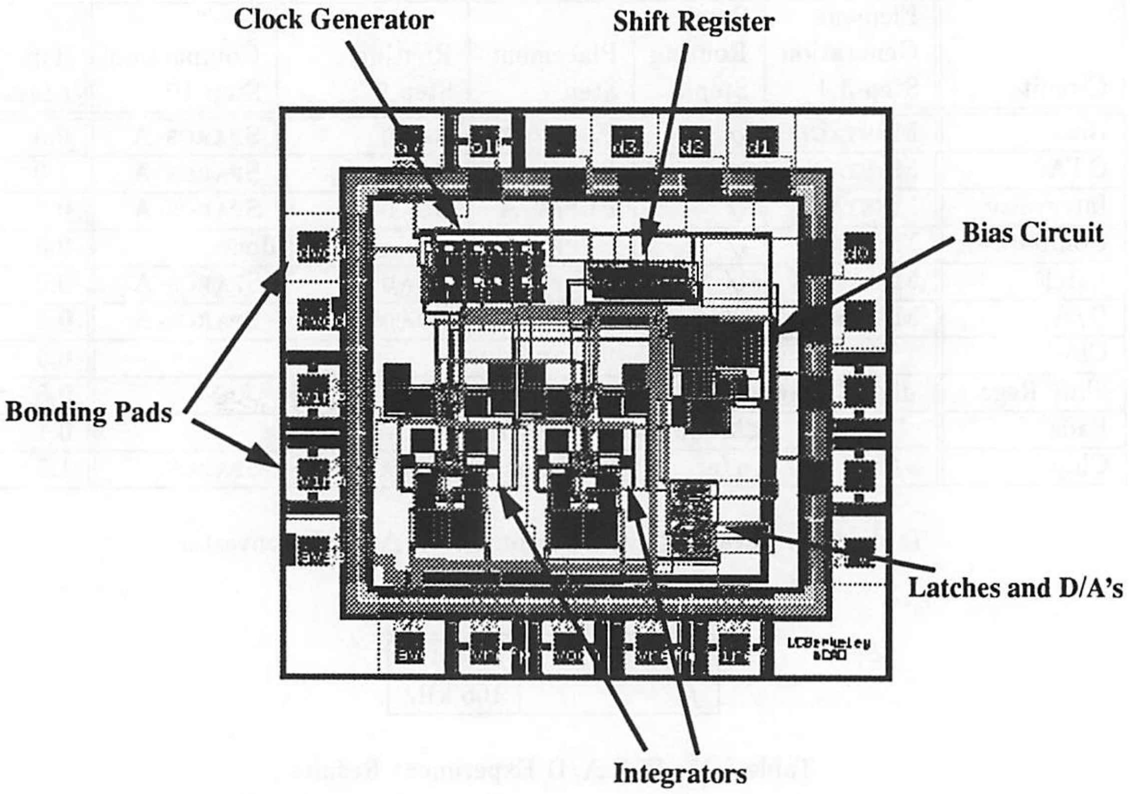

Figure 6.23: Layout of Σ-Δ A/D Converter

of 2 is because the system is differential. The output voltage range relative to V_{FS}, then, is

$$O_{range} = \frac{V_{range}}{V_{FS}} \tag{6.11}$$

Also, the output common voltage is

$$V_{ocm} = \frac{V_{o,max} + V_{o,min}}{2} \tag{6.12}$$

In order to calculate the remaining performances, some device sizes must be calculated using

$$W_{M1-M4} = \frac{2I_{leg}(L_{nmos} - L_{Dn})}{k_n'(V_{gs} - V_T)^2} + \Delta W_n \tag{6.13}$$

$$W_{M5-M8} = \frac{2I_{leg}(L_{pmos} - L_{Dp})}{k_p'(V_{gs} - V_T)^2} + \Delta W_p \tag{6.14}$$

$$W_{M9-M10} = \frac{2I_{leg}(L_{bias} - L_{Dn})}{k_n'(V_{gs} - V_T)^2} + \Delta W_n \tag{6.15}$$

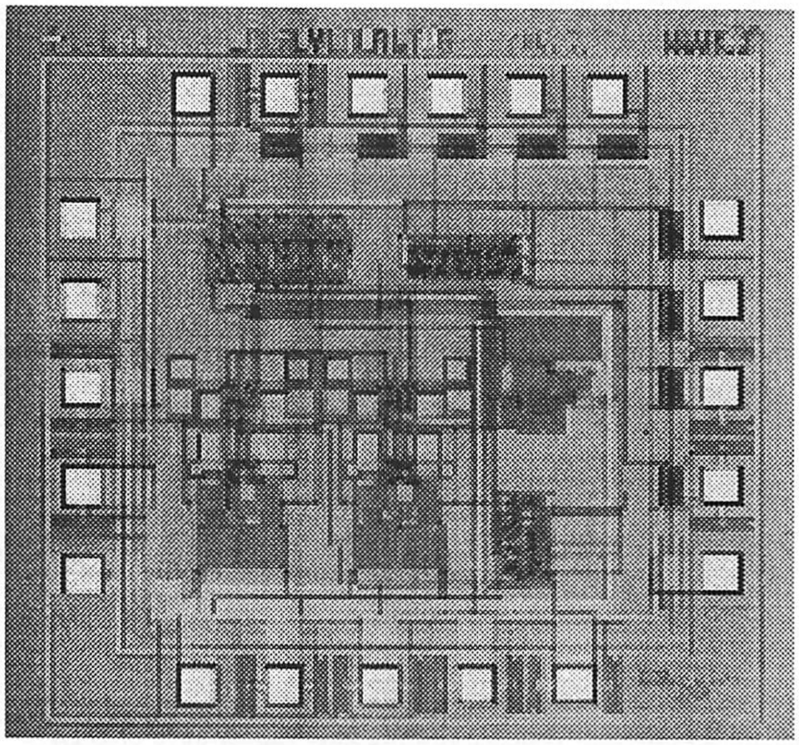
Figure 6.24: Σ-Δ A/D Converter Die Photo

The **flicker noise** for this type of OTA is

$$
\begin{aligned}
v_{1/f,n}^2 &= \begin{cases} 2 \cdot \frac{K_{fn}}{W_{M1}(L_{nmos}-L_{Dn})C_{ox}} \ln \frac{f_x}{f_{min}} & \text{if } A_{fn} = 1 \\ 2 \cdot \frac{K_{fn}}{W_{M1}(L_{nmos}-L_{Dn})C_{ox}} \cdot \frac{f_x^{1-A_{fn}} - f_{min}^{1-A_{fn}}}{1-A_{fn}} & \text{otherwise} \end{cases} \\
v_{1/f,p}^2 &= \begin{cases} 2 \cdot \left(\frac{g_{m7}}{g_{m1}}\right)^2 \cdot \frac{K_{fp}}{W_{M7}(L_{pmos}-L_{Dp})C_{ox}} \ln \frac{f_x}{f_{min}} & \text{if } A_{fp} = 1 \\ 2 \cdot \left(\frac{g_{m7}}{g_{m1}}\right)^2 \cdot \frac{K_{fp}}{W_{M7}(L_{pmos}-L_{Dp})C_{ox}} \cdot \frac{f_x^{1-A_{fp}} - f_{min}^{1-A_{fp}}}{1-A_{fp}} & \text{otherwise} \end{cases} \\
O_{1/f} &= (v_{1/f,n}^2 + v_{1/f,p}^2)^{\frac{1}{2}}
\end{aligned} \tag{6.16}
$$

where $C_{ox} = \frac{\kappa_{ox}\epsilon_o}{t_{ox}}$. Again, since the g_m for all of the transistors are the same, $\frac{g_{m7}}{g_{m1}} = 1$ and the equations are slightly simplified.

To calculate the **slew rate**, I_{SR}, the capacitance at the output must be computed. One component is the drain capacitances of M_3-M_6. These computations assume that the transistors are interleaved in the layout, reducing the junction capacitance by approximately a factor of two and virtually eliminating the sidewall capacitance. The design rule, L_{drain}

Parameter	Description
V_{dd}	Supply Voltage
f_x	Nyquist Frequency
M	Oversampling ratio
V_{FS}	Full scale voltage of Σ-Δ system
I_{gain}	Integrator gain
L_{nmos}	Length of transistors $M_1 - M_4$, $M_{50} - M_{57}$ of OTA Length of transistors $M_{21} - M_{22}$, $M_{30} - M_{33}$
L_{pmos}	Length of transistors $M_5 - M_8$, $M_{30} - M_{33}$ of OTA Length of transistors $M_0 - M_5$, $M_{10} - M_{13}$ of the bias circuit
L_{bias}	Length of transistors $M_9 - M_{12}$ of the OTA Length of transistors $M_{40} - M_{43}$ of the bias circuit
L_{M44}	Length of transistor M_{44} of the bias circuit
W_{M20}	Width of transistor M_{20} of the bias circuit
$V_{gs} - V_T$	$V_{gs} - V_T$ voltage for the transistors in the gain stage
I_{leg}	Current flowing through the transistors in the gain stage
C_S	Size of the sampling capacitor
V_{safety}	Voltage above V_{ds} where cascode device will be biased
f_{min}	Minimum frequency for flicker noise calculation
I_{ratio}	Current ratio between M_11 to M_9 of OTA
$R_{c1,m9gate}$	Ratio between C_1 and M_9 gate capacitance of OTA
$R_{c1,c2}$	Ratio between C_1 and C_2 of CMFB circuit
p_{leak}	Percent leakage current M_{20} of the bias circuit

Table 6.19: Integrator Design Parameters

is the gate-to-gate distance of interleaved transistors. The drain capacitances are

$$\begin{aligned} C_{d,M3} &= \frac{W_{M3}}{2} L_{drain} \cdot \frac{C_{jo,n}}{(1 + \frac{V_{ocm}}{\phi})^{m_{j,n}}} \\ C_{d,M5} &= \frac{W_{M5}}{2} L_{drain} \cdot \frac{C_{jo,p}}{(1 + \frac{V_{ocm}}{\phi})^{m_{j,p}}} \end{aligned}$$

Another component is the series capacitance of C_1 and the capacitance at the gate of M_9. The capacitance of at the gate of M_9 is $C_{g,M9} = \frac{2}{3} C_{ox} W_{M9} (L_{bias} - L_{Dn})$. C_1 is taken as

$$C_1 = R_{c1,m9gate} C_{g,M9} \tag{6.17}$$

There is also a fairly large parasitic capacitance associated with C_1. Since this is only a single poly process, the parasitic capacitance between the bottom plate of the capacitor to the substrate is approximately the same as the capacitor itself. To reduce the drain capacitance, the bottom plate for C_1 was selected to be the node at the gate of M_9. Thus,

Parameter	Description	NMOS device	PMOS device
$\Delta W_n, \Delta W_p$	Lithography width variation	0.25 μm	0.39 μm
L_{Dn}, L_{Dp}	Gate Lateral Diffusion	0.10 μm	0.02 μm
V_{Tn}, V_{Tp}	Zero-bias Threshold Voltage	0.608V	0.838V
k'_n, k'_p	Transconductance Parameters	130 $\mu A/V^2$	43 $\mu A/V^2$
γ_n, γ_p	Bulk Threshold Parameter	0.448 $V^{\frac{1}{2}}$	0.497 $V^{\frac{1}{2}}$
t_{ox}	Oxide Thickness	16 nm	16 nm
$C_{jo,n}, C_{jo,p}$	Zero Bias Bulk Junction Cap.	$0.11962 \cdot 10^{-3}$F/m^2	$0.53093 \cdot 10^{-3}$F/m^2
$m_{j,n}, m_{j,p}$	Bulk Junction Exponential	0.4398	0.5074
phi	Surface Potential	0.6 V	0.6 V
K_{fn}, K_{fp}	Flicker Noise Coefficient	$6.5 \cdot 10^{-24}V^2$F	$2.1 \cdot 10^{-24}V^2$F
A_{fn}, A_{fp}	Flicker Noise Exponential	1.2	1.2
$\Delta k'_n, \Delta k'_p$	Transconductance Matching	4%	4%
a_{VFB}	Threshold Voltage Matching	0.0623 μVm	0.0623 μVm

Table 6.20: Relevant HP CMOS26B Process Parameters

Parameter	Description	value
L_{drain}	Length of Drain	3.0 μm
L_{source}	Length of Source	2.5 μm

Table 6.21: Relevant MOSIS SCMOS Design Rules

$C_{g,M9}$ has in parallel with it approximately C_1. The total series capacitance then is

$$C_{series} = \left[\frac{1}{C_1} + \frac{1}{C_{g,M9}(1 + I_{ratio}) + C_1}\right]^{\frac{1}{2}}$$

The $(1 + I_{ratio})$ scaling term is necessary to include the gate capacitance of M_{11}. The total output capacitance is

$$C_{out} = C_{d,M3} + C_{d,M5} + C_{series} + C_I + C_S$$

Parameter	Description	Value
k	Boltzmann's constant	$1.38 \times 10^{-23} J/K$
T	Temperature	300 K
ϵ_o	Dielectric constant of vacuum	$8.85 \times 10^{-12} F/m$
κ_{ox}	Silicon Oxide dielectric ratio	3.97

Table 6.22: Physical Constants

The slew rate, then, is finally

$$I_{SR} = \frac{I_{leg}}{C_{out}} \tag{6.18}$$

where

$$C_I = \frac{C_S}{I_{gain}} \tag{6.19}$$

Because each capacitor carries with it a parasitic capacitance of almost the same value as the capacitor itself, this must be considered when choosing which side (top or bottom) to connect to the OTA. Traditionally, top plates are connected to the inputs of the OTA (a sensitive input node). This minimizes the capacitance at the gate and improves the feedback factor. The bottom plate also acts as a shield from substrate noise for the sensitive node. However, in this design, as an experiment, since the optimization at the low-level showed that slew rate was a limiting factor (see Table 6.10) and not I_T, the bottom plates were connected to the input of the OTA, thus minimizing the drain capacitance, and as such, improving the slew rate. Also, consider that the gain of this integrator is only 1/2. For this low amount of closed loop gain, an increase in the parasitics at the input of the OTA, does not affect the system too greatly (see Equation 6.20).

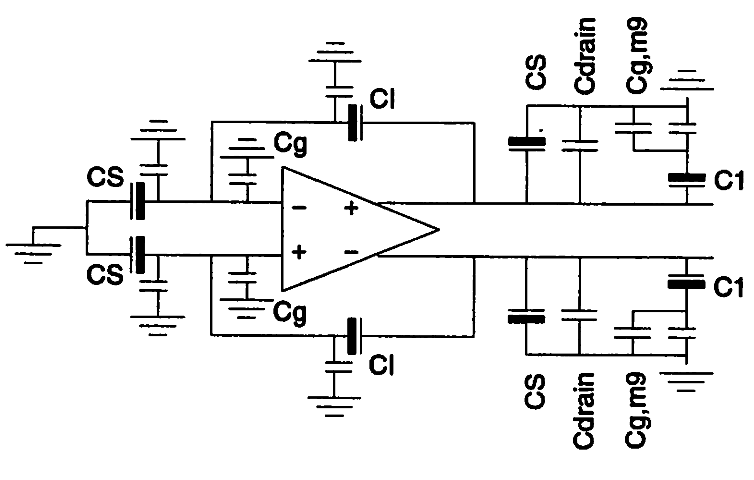

Figure 6.25: Location of Parasitics introduced by capacitors, C_S and C_I

Figure 6.25 shows the position of the parasitics. The side in bold on the capacitors indicates the bottom plate. The extra capacitor adjacent to C_S and C_I depicts where the parasitic capacitance is.

To calculate the **time constant for the integrator**, I_T, first, the feedback factor is computed. It is

$$f = \frac{C_I}{C_I + C_{I,par} + C_S + C_{S,par} + C_{g,M3}} \tag{6.20}$$

where $C_{g,M3} = 2/3 \cdot C_{ox} W_{M3}(L_{nmos} - L_{Dn})$ and $C_{I,par}$ and $C_{S,par}$ are the parasitic capacitances from C_I and C_S respectively. Assuming that $C_{I,par} \approx C_I$ and $C_{S,par} \approx C_S$, then

$$f = \frac{C_I}{2C_I + 2C_S + C_{g,M3}}$$

From this,

$$I_\tau = \frac{C_{out}}{g_m} \cdot \frac{1}{f} \tag{6.21}$$

The feedback factor for the common mode feedback circuit can be calculated in a similar fashion. Again assuming the parasitic capacitor associated with C_1 is about the same as C_1, the feedback factor is

$$f_{cmfb} = \frac{C_1}{2C_1 + C_{g,M9}(1 + I_{ratio})}$$

The **time constant for the common mode feedback circuit** is

$$O_{\tau,CMFB} = \frac{C_{out}}{g_m} \cdot \frac{1}{f_{cmfb}} \tag{6.22}$$

The final of the nine performance parameters is the OTA **input offset voltage,** O_{off}. From Equation 5.1,

$$\begin{aligned} \sigma_{vt,n} &= \frac{\sqrt{a_{\mathrm{VFB}}}}{W_{M1}(L_{nmos} - L_{Dn})} \\ \sigma_{vt,p} &= \frac{\sqrt{a_{\mathrm{VFB}}}}{W_{M7}(L_{pmos} - L_{Dp})} \end{aligned}$$

From this

$$O_{off} = \sigma_{vt,n} + \frac{V_{gs} - V_T}{2\Delta k'_n} + \frac{g_{m7}}{g_{m1}}\left(\sigma_{vt,p} + \frac{V_{gs} - V_T}{2\Delta k'_p}\right) \tag{6.23}$$

To obtain a rough approximation of the **active area** suited for use as the objective function in the low-level optimization step in Section 6.3.2, the area is taken as

$$area = W_{M1} L_{nmos} + W_{M7} L_{pmos} \tag{6.24}$$

All of the capacitors scale with this area.

Enough has been developed to solve the low-level optimization problem. However, in order to generate the full schematic/spice deck for the OTA, integrator, and bias circuit, there are a few more expressions that need to be evaluated. Most of these device sizes are direct consequences of the formulas above. For example,

$$W_{M11-M12} = I_{ratio} W_{M9} \tag{6.25}$$

$$W_{M30-M33} = I_{ratio} W_{M5} \tag{6.26}$$

$$W_{M0-M5,bias} = W_{M5} \tag{6.27}$$

$$W_{M21-M22,bias} = W_{M1} \tag{6.28}$$

$$W_{M40-M43,bias} = W_{M9} \tag{6.29}$$

$$W_{M44} = \frac{2I_{leg}L_{M44}}{k_n'(V_{gs} - V_T)^2} + \Delta W_n \tag{6.30}$$

The other capacitor in the CMFB circuit, C_2 was chosen to be

$$C_2 = \frac{C_1}{R_{c1,c2}} \tag{6.31}$$

The value for $R_{c1,c2}$ of was simply chosen as a rule of thumb.

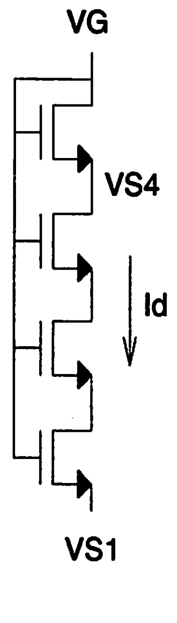

Figure 6.26: Diode bias chain

Finally, the widths of the diode connected transistors need to be determined. Figure 6.26 shows the schematic with node names and currents for the diode chain. The transistor widths can be approximated by solving simultaneously, Equations 6.32 through 6.35

for W and V_{S4}.

$$\begin{aligned}
V_{t1} &= V_{T0} + \gamma(\sqrt{\phi + V_{S1}} - \sqrt{\phi}) && (6.32)\\
V_{t4} &= V_{T0} + \gamma(\sqrt{\phi + V_{S4}} - \sqrt{\phi}) && (6.33)\\
I_D &= \frac{k'(W - \Delta W)}{2(n-1)(L - L_D)}[2(V_G - V_{S1} - V_{t1})(V_{S4} - V_{S1}) - (V_{S4} - V_{S1})^2] && (6.34)\\
I_D &= \frac{k'(W - \Delta W)}{2(L - L_D)}(V_G - V_{S4} - V_{t4})^2 && (6.35)
\end{aligned}$$

n is the number of diodes in the chain. $n = 4$ in this case. For the NMOS chain, $W_{M50-M57}$, let

$$\begin{aligned}
I_D &= I_{leg} I_{ratio}\\
\gamma &= \gamma_n\\
k' &= k'_n\\
L &= L_{nmos}\\
V_{T0} &= V_{Tn}\\
V_{S1} &= V_{gs} - V_T + V_{safety}\\
V_G &= 3(V_{gs} - V_T) + 2V_{safety} + V_{tx}\\
\text{where } V_{tx} &= V_{Tn} + \gamma_n(\sqrt{\phi + 2(V_{gs} - V_T) + 2V_{safety}} - \sqrt{\phi})
\end{aligned}$$

For the PMOS chain, $W_{M10-M13}$, let

$$\begin{aligned}
I_D &= I_{leg}\\
\gamma &= \gamma_p\\
k' &= k'_p\\
L &= L_{pmos}\\
V_{T0} &= V_{Tp}\\
V_{S1} &= 0\\
V_G &= 2(V_{gs} - V_T) + V_{safety} + V_{Tp}
\end{aligned}$$

Solving twice, then, Equations 6.32 to 6.35, one can obtain $W_{M50-M57}$ and $W_{M10-M13}$. Finally,

$$W_{M30-M33} = \frac{W_{M50-M57}}{I_{ratio}} \qquad (6.36)$$

Note that this only yields an approximate solution. These values had to be increased 10% to 20% to achieve the exact headroom voltage. This was verified through SPICE simulation.

The last transistor length, L_{M20} to be determined is the one for M_{20}, the leak transistor in the bias circuit. For this,

$$L_{M20} = \frac{(W_{M20} - \Delta W)k_n'}{2p_{leak}I_{leg}}(V_{dd} - V_{Tn})^2 + L_{Dn} \tag{6.37}$$

Performance	Description	Equation
O_{off}	Offset voltage due to the OTA	6.23
O_{range}	Output range of the OTA relative to V_{FS}	6.11
O_{th}	Thermal noise due to the OTA	6.10
$O_{1/f}$	Flicker noise for the OTA	6.16
O_{gain}	Open loop gain of the OTA	6.7
I_{τ}	Time constant for the integrator	6.21
I_{SR}	Slew Rate of the integrator	6.18
$I_{kT/C}$	kT/C noise of the integrator	6.8
$O_{\tau,CMFB}$	Time constant for the CMFB circuit of the OTA	6.22
area	Active Area approximation	6.24

Table 6.23: Calculated Integrator/OTA Performances

These equations characterize the critical performances for integrator and OTA in the Σ-Δ A/D design example. They also define all of the device sizes. Tables 6.23 and 6.24 summarize the two sets of results.

6.8 Conclusion

During the span of two months a Σ-Δ A/D was synthesized following the methodology and sent for fabrication. The total design time was reasonable, and preliminary tests indicate that the A/D converter is close to meeting performance specifications. Future work for the Σ-Δ A/D consists of further tests and the development of a testing methodology which will attempt to speed up the testing process. One possible approach is to find an optimal set of sine wave input amplitudes and frequencies in order to quickly characterize SNR.

Parameter	Description	Equation
V_{ocm}	Output common mode voltage	6.12
W_{M1-M4}	Width of transistors $M_1 - M_4$	6.13
W_{M5-M8}	Width of transistors $M_5 - M_8$	6.14
W_{M9-M10}	Width of transistors $M_9 - M_{10}$	6.15
$W_{M11-M12}$	Width of transistors $M_{11} - M_{12}$	6.25
$W_{M30-M33}$	Width of transistors $M_{30} - M_{33}$	6.26
W_{M44}	Width of transistor M_{44}	6.30
$W_{M50-M57}$	Width of transistors $M_{30} - M_{33}$	6.32-6.35
$W_{M0-M5,bias}$	Width of transistors $M_0 - M_5$ in the bias circuit	6.27
$W_{M10-M13,bias}$	Width of transistors $M_{10} - M_{13}$ in the bias circuit	6.32-6.35
L_{M20}	Length of transistor M_{20} in the bias circuit	6.37
$W_{M21-M22,bias}$	Width of transistors $M_{21} - M_{22}$ in the bias circuit	6.28
$W_{M30-M33,bias}$	Width of transistors $M_{30} - M_{33}$ in the bias circuit	6.36
$W_{M40-M43,bias}$	Width of transistors $M_{40} - M_{43}$ in the bias circuit	6.29
C_1	First of two CMFB capacitors	6.17
C_2	Second of two CMFB capacitors	6.31
C_I	Integrating capacitor	6.19

Table 6.24: Determined Integrator/OTA Voltages and Device Sizing

Chapter 7

Video Driver Design Example

7.1 Introduction

The design of a video driver system is the last example in our current set. This system includes two major analog subsystems: a frequency synthesizing phase-locked loop (PLL) and three current source D/A converters. This system is intended to be equivalent to a RAMDAC [6] system except that we have chosen not to implement the static RAM lookup table. This greatly reduces the cost, and should not degrade the worth of this example as an experiment for the design methodology. In this system, the PLL functions as a clock generator, synthesizing the various dot clocks required for different screen resolutions. The D/A converters generate the output currents required for the red, green, and blue signals to the monitor.

This system is more complex than the previous design examples. It contains two large analog components and requires an extra layer of hierarchy. It also more closely resemble large commercial systems. Its level of complexity is shown in Figure 5.1.

This is work currently in progress. We present it, because enough has been done to show that hierarchy can be successfully used in this design, a key point to illustrating that this methodology does work. The high-level synthesis phase is complete, showing the use of **intermediate level parameters** and **flexibility functions** for the PLL, and the use of a constraint-driven **module generator** for the synthesis of the D/A converters. The low level optimization and physical assembly are only partially completed. What has been completed will be discussed, and what still remains to be done will be discussed at the end of this chapter, in "future works."

7.2 Design Specifications

A block diagram of the overall system is shown in Figure 7.11. The specifications for this system are broken down into several categories. This is shown in Table 7.1. The

Type	Specifications	Value
Performance	Timing Jitter Phase Margin Frequency Range INL of video signal DNL of video signal	$\leq 1\%$ $\leq 45^o$ 10 MHz to 130 MHz $\leq$ 1 LSB $\leq$ 0.5 LSB
	Miscellaneous constraints on D/A output behavior.	
Operation	Supply voltage	5 V
Technology	Design rules SPICE parameters	scmos HP CMOS34
Layout	Input/output terminal locations, placement, etc.	
Optimization	Non-default flexibility function	default
Objective for project	Minimize Power	

Table 7.1: Video Driver Design Specifications

critical performances that have to be met in this design are timing jitter, phase margin, frequency range of operation, and INL and DNL of the video signals. A module generator [80] is used to synthesize the D/A converters. It is from here that we obtain the "miscellaneous constraints." These constraints are not listed here. [80] contains a complete list. Examples are constraints on glitch energy for one *lsb* code change and glitch energy for one *msb* code change.

7.3 Synthesis Path

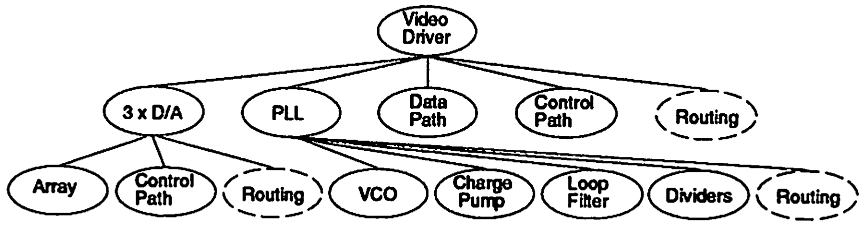

Figure 7.1: Video Driver System Hierarchy

The design hierarchy for the system is shown in Figure 7.1. The first level of hierarchy divides the video driver into its basic system components- three D/A converters, a PLL, and a data and control path to handle the incoming pixel data as well as registers to control the frequency of the PLL output. The D/A converter is broken down into two parts: a current source array which contains both a linear array and binary array and a control path for the decode logic and latches that are necessary for the operation of the D/A converter. The PLL is broken down into a voltage-controlled oscillator (VCO), a charge pump (CP), a loop filter, and several dividers.

7.3.1 High Level Synthesis

Unlike the previous design examples, three high level decompositions are required in this problem. (1) The video driver system is broken down its subcomponents. (2) The PLL system is broken down into its subcomponents. And (3) the D/A converters are mapped into their subcomponents.

7.3.1.1 Video Driver System

The mapping is accomplished trivially at this level. Since the PLL and the D/A's are fairly decoupled, performances map directly from the high level specifications to constraints on the PLL and the D/A's. The constraints for the PLL are the timing jitter, phase margin, and the frequency range. The constraints for the D/A are the frequency range, the INL, the DNL, the miscellaneous specifications.

7.3.1.2 Phase-Locked Loop

The architecture for the PLL is shown in Figure 7.2. It consists of a phase frequency detector (PFD) which compares the input reference frequency (F_{ref}) that has been divided by m and the output from the voltage controlled oscillator (VCO) which has been divided by n. The PFD controls the charge pump. The current from the charge pump goes through a low pass filter ("loop filter"). The output from the loop filter is the controlling voltage for the VCO. Finally, the output signal is the VCO output divided by k.

The **intermediate level parameters** for the PLL are shown in Figure 7.2. Given a value for each of these parameter, a behavioral simulator is used to evaluate the performance of the PLL [23]. Letting x_i be the i^{th} parameter ($x_1 \equiv K_o$, $x_2 \equiv \sigma^2_{\Delta\tau}$, $\cdots x_n \equiv \Delta\tau_{div}$),

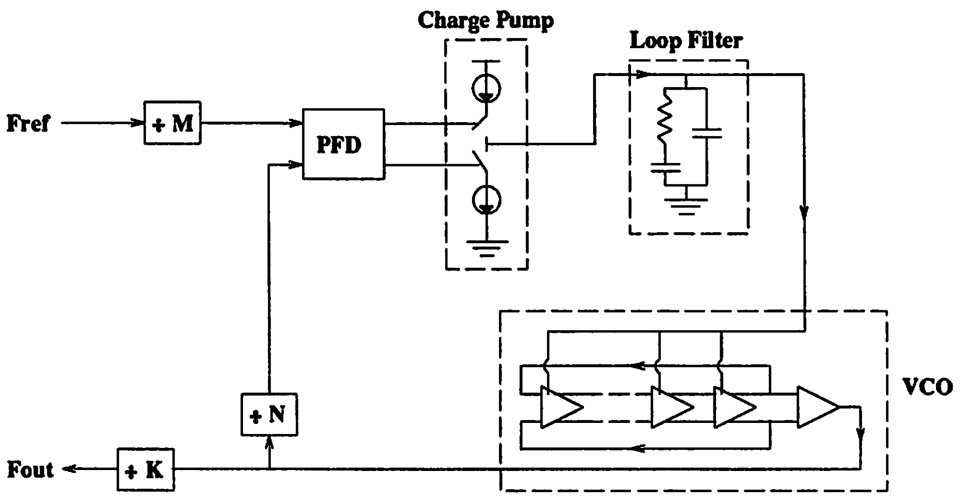

Figure 7.2: PLL Architecture

and n be the number of parameters (n=14), and c, a list containing the additional parameters from Table 7.1, the behavioral simulator is evaluates the timing jitter $((\sigma^2_{\Delta_T})_{system})$, g, and the phase margin (θ), h, as shown in in Equations 7.2.

$$\begin{aligned}(\sigma^2_{\Delta_T})_{system} &= g(x_1, x_2, \cdots x_n, c) \\ \theta &= h(x_1, x_2, \cdots x_n, c)\end{aligned} \tag{7.1}$$

A set of constant non-critical parameters were chosen *a priori* to reduce the complexity of the problem. These are shown with their values in Table 7.3.

Flexibility functions following the general form in Equation 3.1 were built for the parameters which are variables in the optimization problem. The "moderate" and "hard" values are shown in Figure 7.4, along with the coefficients used for each of the optimization variables.

The optimization problem is shown by Equations 7.3 where m is the number of optimization variables, and where l_i and u_i represent lower and upper bounds for those variables.

$$\begin{aligned}\max \quad & \textstyle\sum_{i=1}^{m} flex_i(x_i) \\ \text{s.t.} \quad & (\sigma^2_{\Delta_T})_{system}(x_1, x_2, \cdots x_n, c) \leq (50ps)^2 \text{ @ 140 MHz} \\ & \theta(x_1, x_2, \cdots x_n, c) \leq 45^o \\ & l_i \leq x_i \leq u_i \qquad \forall i = 1..m\end{aligned} \tag{7.2}$$

Parameter	Description
K_o	VCO gain
$\sigma^2_{\Delta T}$	VCO timing jitter
$V_{vco,sat_{up}}$	VCO upper saturation voltage
$V_{vco,sat_{lo}}$	VCO lower saturation voltage
STT	PFD State Transition Table
R	Loop Filter resistor
C	Loop Filter Capacitor
C_1	Loop Filter Capacitor
I_p	Charge pump reference current
$V_{cp,sat_{up}}$	Charge pump upper saturation voltage
$V_{cp,sat_{lo}}$	Charge pump lower saturation voltage
$\Delta I_p/I_p$	Charge pump reference current mismatch
R_{out}	Charge pump output resistance
m	Divider constant

Table 7.2: Parameters for the PLL Block

Parameter	Value
$V_{vco,sat_{up}}$	4.8V
$V_{vco,sat_{lo}}$	1.8V
STT	table
$V_{cp,sat_{up}}$	4.8V
$V_{cp,sat_{lo}}$	0.2V
$\Delta I_p/I_p$	10%
R_{out}	100 kΩ
m	100 - 250

Table 7.3: Constant Parameters chosen for the PLL Block

As in the high level optimization for the Σ-Δ A/D converter (Section 6.3.1), the general purpose optimizer, MINOS [76], was not suitable. In this case, in addition to the problem of "noisy" calculations, part of the infeasibility region in the optimization problem is where the PLL is unstable. This is a region where the timing jitter is undefined. Since MINOS does not restrict itself to the feasible region while searching for the solution, it may wander into a region where one of its constraint functions is undefined; this results in a failure. The Supporting Hyperplane Algorithm [65] was used to solve the optimization problem. In this approach, the search remains in the feasible region. Great care, however, had to be taken when computing the planes. Since the feasible region boundary is often near an undefined

Parameter	Design Difficulty		Coefficients		
	Moderate	Hard	a	b	c
K_o	50 MHz	80 MHz	2.6×10^{-15}	0	6.4
$\sigma^2_{\Delta T}$	3.5 ps	1.4 ps	0	-2.5×10^8	0.10
R	434 kΩ	868k Ω	1.8×10^{-11}	0	3.3
C	63 pF	126 pF	8.4×10^{20}	0	3.3
C_1	63 pF	126 pF	8.4×10^{20}	0	3.3
I_p	20 μA	200 μA	2.5×10^8	0	6.4

Table 7.4: Flexibility Coefficients for the PLL Block

region, finite difference points for computing the planes are chosen to be within the feasible region.

Parameter	Value
K_o	$40MHz/V$
$\sigma^2_{\Delta T}$	$(3.5psec)^2$
R	116 kΩ
C	39 pF
C_1	5 pF
I_p	11.2 μA

Table 7.5: Parameter Selection for the PLL Block

The results from the optimization problem are shown in Figure 7.5. The resulting value of the flexibility function was 4.16. Thus, this design is considered just a bit easier the moderate (flexibility = 0).

7.3.1.3 Digital-to-Analog Converters

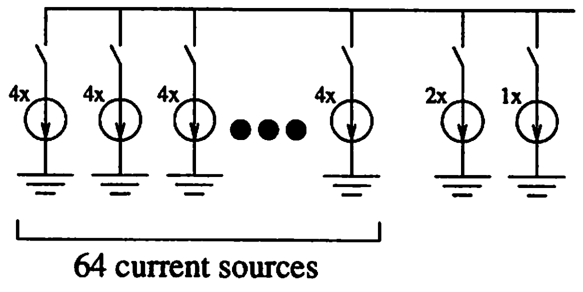

Figure 7.3: Architecture for Video D/A Converters

A **module generator** for video D/A converters [80] is used to generate the D/A's. This was chosen over the one developed in 5 because of its suitability for video, e.g. the generator considers the high current requirements for video. The 8 bit D/A converters have segmented architectures consisting of six bits in the linear array and two bits in the binary array. This architecture is shown in Figure 7.3.

7.3.2 Low Level Synthesis

The low level synthesis phase consists eight parts: D/A array, D/A control path, PLL VCO, PLL CP, PLL Loop Filter, PLL dividers, video driver datapath, and the video driver control path. Components of the D/A are not discussed, because they are generated within the module generator. The digital video driver data and control paths are uninteresting from a synthesis point of view. The PLL CP and loop filter have not been developed. The remaining two, the VCO and the PLL dividers, will be discussed.

7.3.2.1 VCO

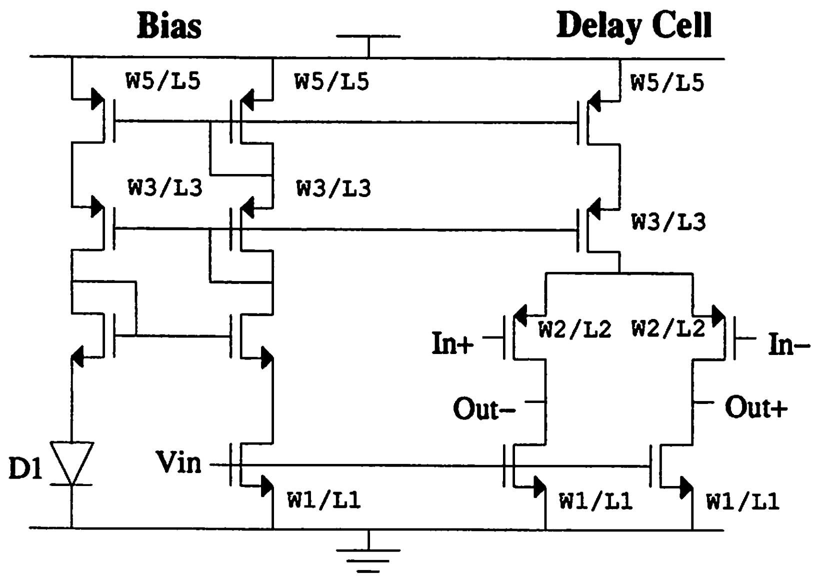

Figure 7.4: Schematic for the Delay Cell in the VCO

Probably the most significant component of the PLL is the VCO. The VCO is comprised of a series of cascaded delay cells. The schematic for the delay cells is shown in Figure 7.4 along with its bias circuitry.

The design **constraints** for the VCO are a subset of the constraints from high level optimization results in Tables 7.1, 7.3, and 7.5. These are listed in Table 7.6.

Performance	Value
Frequency Range	10 MHz to 130 MHz
$V_{vco,sat_{up}}$	≥ 4.8 V
$V_{vco,sat_{lo}}$	≤ 1.8 V
K_o	$= 40$ MHz/V $\pm$ 10%
$\sigma^2_{\Delta_T}$	≤ 3.5 ps (@ 135 MHz)

Table 7.6: Design Constraints for the VCO

For this circuit, the most critical devices sizes are W_1, L_1, W_2, and L_2. These will be the design **variables**. The **objective** for this problem is to minimize power. Equation 7.4 shows the optimization problem statement.

$$\begin{array}{ll} \text{min} & power(W_1, L_1, W_2, L_2) \\ \text{s.t.} & V_{vco,sat_{lo}}(W_1, L_1, W_2, L_2) \leq 1.8V \\ & V_{vco,sat_{up}}(W_1, L_1, W_2, L_2) \geq 4.8V \\ & K_o(W_1, L_1, W_2, L_2) = 40MHz/V \\ & \sigma^2_{\Delta_T}(W_1, L_1, W_2, L_2) \leq (3.5ps)^2 \text{ @ 140 MHz} \end{array} \tag{7.3}$$

7.3.2.2 PLL Dividers

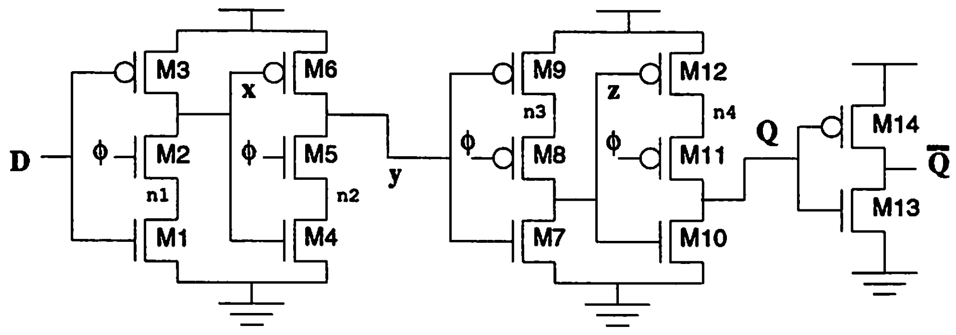

Figure 7.5: TSPC Negative Edge Master Slave Flip-Flop

A **performance** requirement of 130MHz is required for the two dividers in the PLL loop. They are two identical programmable 8 bit dividers. To meet the performance requirement, a faster family of logic was employed for the flip-flops. True single phase clocking (TSPC) logic [105] was used. A example of a falling edge master slave flip-flop is shown in Figure 7.5. This type of logic is also used in the video D/A digital control path. A schematic for the programmable 8 bit divider is shown in Figure 7.6.

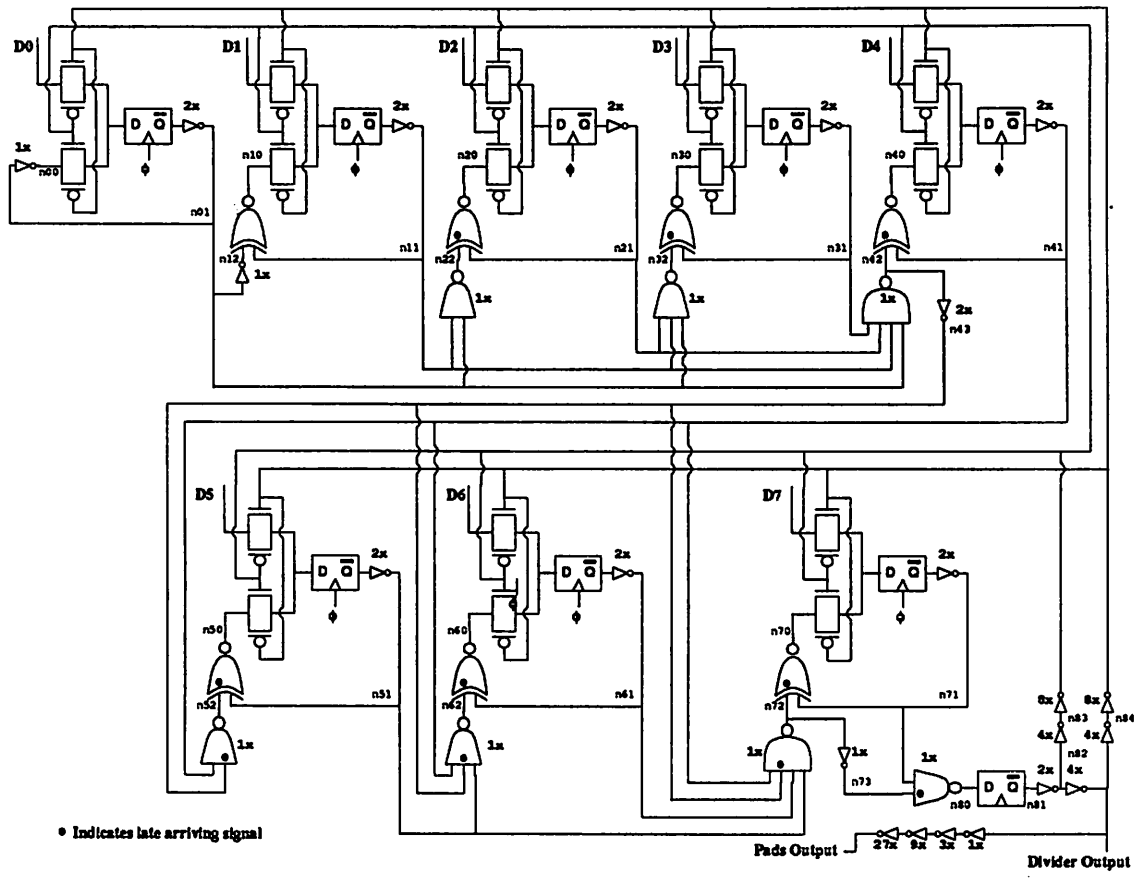

Figure 7.6: Programmable 8 bit Divider Schematic

7.3.3 Physical Synthesis

Currently, all of the layout has been completed except for the analog portions of the PLL. A VCO cell generator has been written, but it cannot be used until the low-level optimization phase for the PLL is finished. This section contains discussion on the layout interesting sub-blocks.

7.3.3.1 VCO

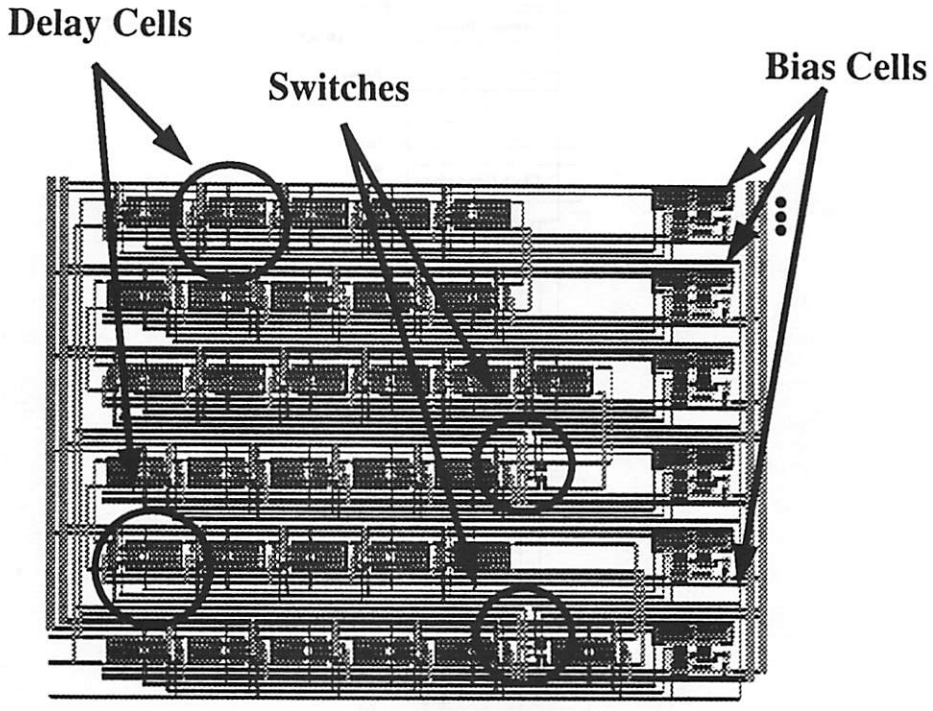

Figure 7.7: Layout for VCO

The layout for the VCO is critical. Much of the nonidealities result from this stage. A layout cell generator has been written the VCO. The program gives the user control over the sizes of the various power busses, the number of components per row, the location of the switches, and the various transistor sizes for the input, triode, mirror, and switch transistors. A sample output is shown in Figure 7.7.

7.3.3.2 Digital Datapath

The layout for the register file (digital datapath) was done manually, because it is wire limited making it unsuitable for standard cells. An 8 bit slice of the floorplan is shown in Figure 7.8. Nine slices are tiled horizontally to form the complete layout shown in Figure 7.9. At the top is the address decoder and the clock generator. The address lines run horizontally across the cell. The two phases of the clock and their inverses run over the flipflops vertically. Also vertically running over the flipflops are the 8 bit output lines for the register. There are separate substrate and power lines to limit the amount of coupling from the digital switching to the substrate. A 4-to-1 multiplexor is shown at the bottom of

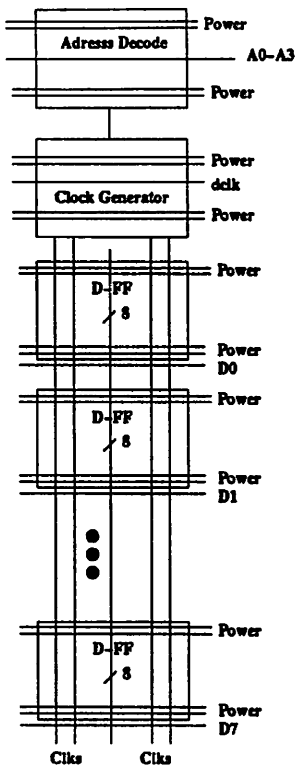

Figure 7.8: Register File Floorplan

the cell to switch between two registers, a blanking signal, and the data input lines. Output buffers have also been placed on all of the digital outputs so that the signals can be driven across the chip to the D/A converters and the PLL.

7.3.3.3 Video Driver System

A preliminary layout for the chip is shown in Figure 7.10. The diagram shows the locations of the major components. The future location for the phase frequency detector, charge pump, low pass filter, and VCO is shown in the lower left hand corner. The active area for this chip is much larger than the active areas for the first two design examples; it is 8.0 mm^2. The entire chip area is approximately 12.5 mm^2.

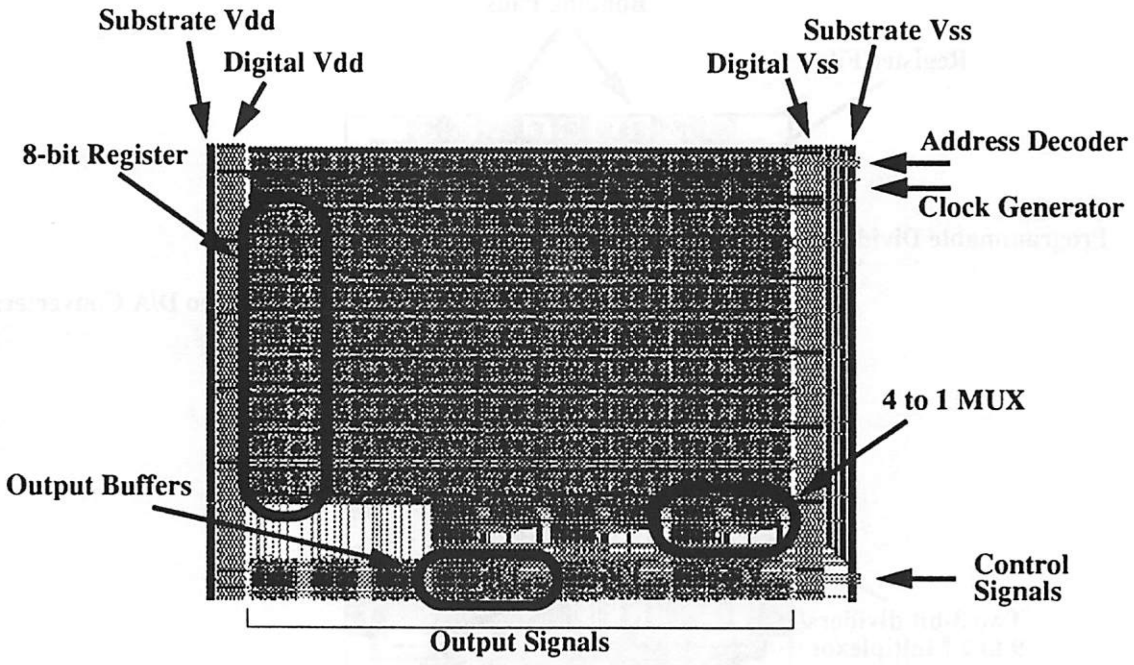

Figure 7.9: Register File Layout

7.4 Testing

The D/A converters will be tested using the test methodology developed in Section 5.6. As before, process mismatch will influence the INL and DNL of the D/A converters. These will again be extracted.

7.5 Conclusion/Future Work

The general design flow for the video driver has been presented. With respect to design and physical synthesis, this project is close to completion. The following steps are required: (1) perform low-level optimization for the VCO; (2) synthesize the layouts for the phase frequency detector, charge pump, low pass filter, and VCO; (3) add the PLL to the chip and re-route; and (4) extract and verify. The two more interesting points in these steps will be the optimization strategy used for the VCO and the use of performance constraints in the automated layout generation of the PLL. The chip will be fabricated, and a test circuit board will be built. With the fabricated parts, measurements will be taken to verify performances. Research will also be conducted to expand the D/A test strategy to

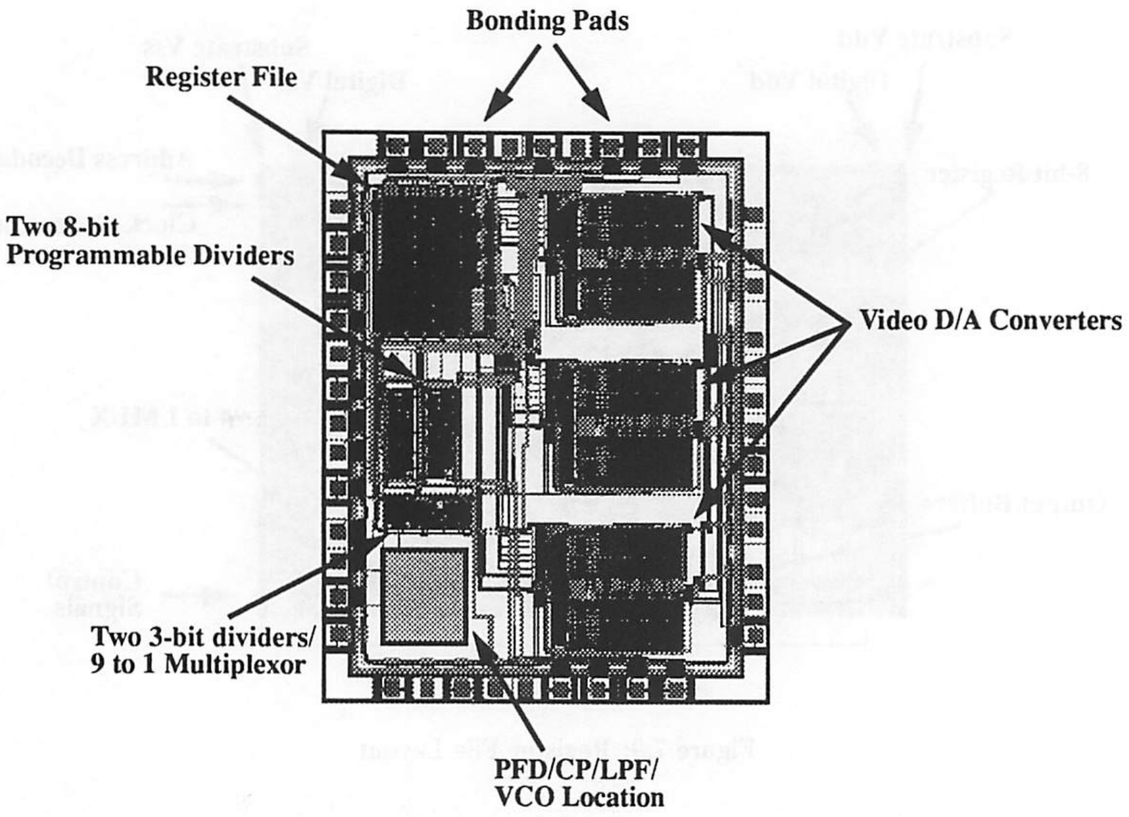
Bonding Pads
Register File
Two 8-bit Programmable Dividers
Video D/A Converters
Two 3-bit dividers/ 9 to 1 Multiplexor
PFD/CP/LPF/ VCO Location

Figure 7.10: Preliminary Video Driver Layout

incorporate the complete video driver.

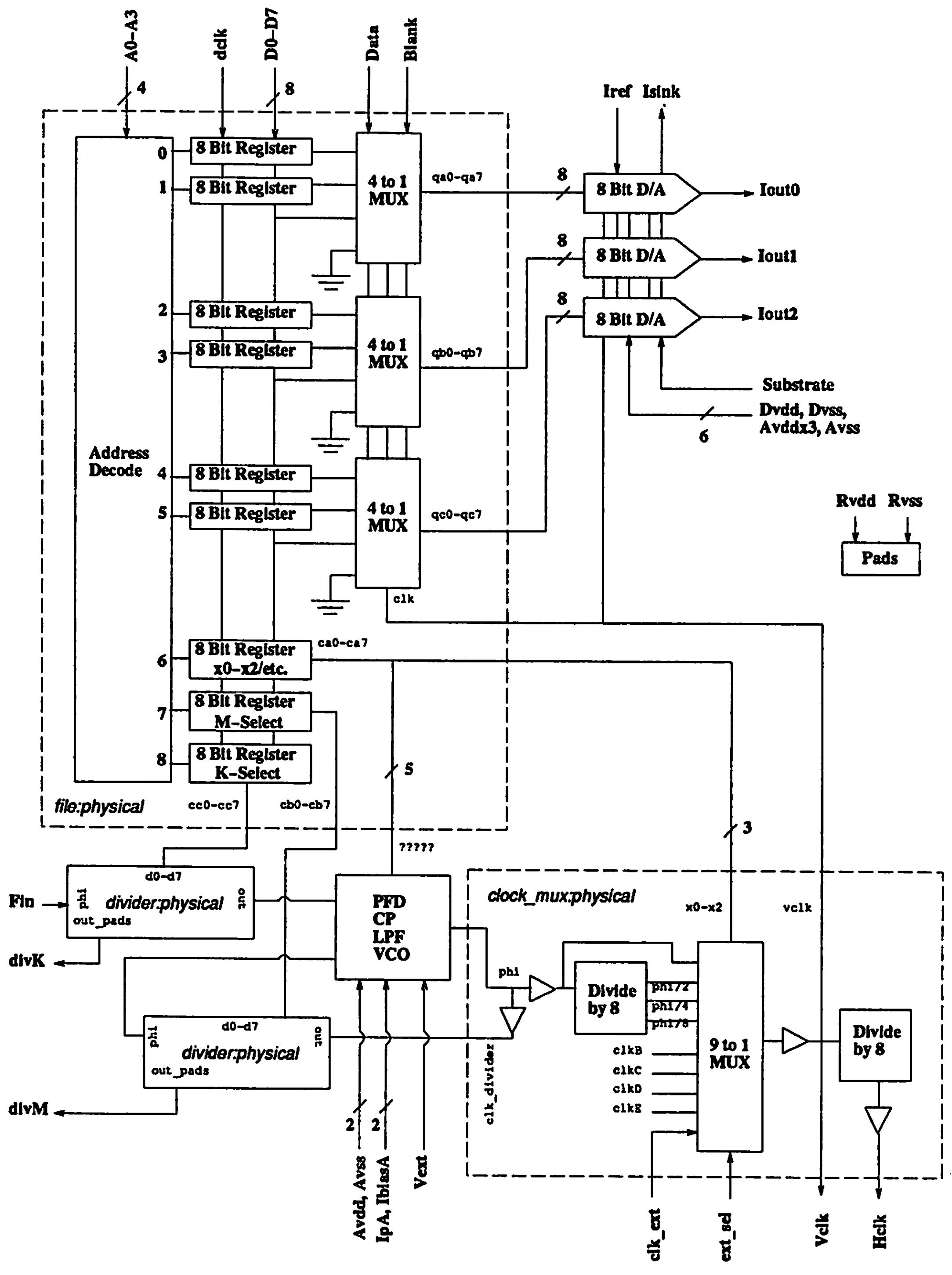

Figure 7.11: Video Driver System

Chapter 8

Conclusion and Future Directions

8.1 Conclusion

Presented has been a new design methodology for analog integrated circuit design. This methodology is aimed at reducing substantially the design cycle of complex analog and mixed analog-digital systems while maintaining or even improving the performance of the final design. A hierarchical approach with early verification and constraint propagation is favored as a way of achieving this goal. The tools, developed within the top-down, constraint-driven paradigm, are the product of the continuing long term effort to meet the design needs of analog integrated circuit design. To show the feasibility of such an approach to doing design, large design examples have been presented. Unlike many other methodologies that have been proposed in the past, we have taken our design examples all the way from the design synthesis path to testing to show that our methodology and tools can actually be used in practice. Most other methodologies have stopped short of fabrication, leaving it up to the end user who wants to use their tools to prove that it can actually be used in practice. One of our goals has been to eliminate this uncertainty.

This work has shown the methodology's potential in its ability to systematize design and speed up the design process. It is hoped from this work with its two completed design examples and the one in progress, from the discussion of the methodology, and from the discussion on its tool set, that this methodology will gain acceptance and be adopted by the design community.

8.2 Future Directions

Future work in the the development of this methodology will undoubtly include refinements to this approach. One way to make these discoveries is through the pursuit of even larger design examples. Not only should they be larger, but a wide variety of circuits should be examined. If all goes well, it should be easier to demonstrate the effectiveness of this methodology in larger systems. These are systems in which the complexity level can overwhelm traditional hand design, and, therefore, will absolutely require computer-aided designs tools for design management.

Immediate work, of course, is the completion of the Video Driver system. One possible area for the future is in RF circuits. The share of the RF circuits for mobile communications in the semiconductor market has dramatically increased over the past years. The design of RF systems is being done at high level, where decisions have to be made about the partitioning of the constraints among the various high level blocks, mostly heuristically, using designer's expertise. We are proposing to fit the design methodology into the requirements of such RF systems, and as an example we can start with a direct conversion CMOS transceiver. The methodology should accelerate and make design more systematic to handle the complexities involved. The tool set will also grow, because in order to work on this RF system, behavioral models will have to be built, and chances are new optimization algorithms will have to be developed as was necessary for the A/D and video driver design.

This methodology will only evolve from doing new designs to constantly challenge it. The field of analog integrated circuit design is constantly changing. In order for the tool set and the methodology to not become obsolete, the designs chosen must follow the state-of-the-art in analog circuit design. Our research group is dedicated to doing this. The primary goal of research in analog design methodologies will be to do design, design, and more designs.

Bibliography

[1] E. Barke, "Line-to-Ground Capacitance Calculation for VLSI: A Comparison," *IEEE Trans. on CAD*, vol. 7, n. 2, pp. 295–298, February 1988.

[2] L. Bonet, J. Ganger, J. Girardeu, C. Greaves, M. Pendelton and D. Yatim, "Test features of the MC145472 ISDN U-transceiver," *Proc. 1990 International Test Conf.*, pp. 68–79, 1990.

[3] B. E. Boser, *Design and Implementation of Oversampled Analog-to-Digital Converters*, Ph.D. Thesis, Integrated Circuits Laboratory, Stanford Univeristy, Stanford, Oct 1988.

[4] R. Brayton, G. Hachtel and A. Sangiovanni-Vincentelli, "A Survey of Optimization Techniques for Integrated-Circuit Design," *Proc. of the IEEE*, vol. 69 n.10, pp. 1334–1364, 1981.

[5] P. A. Brennan, N. Raver and A. E. Ruehli, "Three Dimensional Inductance Computations with Partial Element Equivalent Circuits," *IBM Journal of Research and Development*, vol. 23, pp. 661–668, November 1979.

[6] Brooktree Graphics and Imaging Product Databook, Brooktree Corporation, 1993.

[7] J. Burns, A. Casotto, M. Igusa, F. Marron, F. Romeo, A. Sangiovanni-Vincentelli, C. Sechen, H. Shin, G. Srinath and H. Yaghutiel, "MOSAICO: An integrated Macro-cell Layout System," in *VLSI '87, Vancouver, Canada*, Aug 1987.

[8] J. L. Burns, *Techniques for IC Symbolic Layout and Compaction*, Memorandum UCB/ERL M90/103, UCB, November 1990.

[9] J. R. Burns and A. R. Newton, "SPARCS: A New Constraint-Based IC Symbolic Layout Spacer," in *Proc. IEEE CICC*, pp. 534–539, 1986.

[10] R. Castello and P. Gray, "A High-Performance Micropower Switched-Capacitor Filer," *IEEE Journal of Solid State Circuits*, vol. SC-20, pp. 1122–1132, December 1985.

[11] H. Chang, E. Liu, R. Neff, E. Felt, E. Malavasi, E. Charbon, A. Sangiovanni-Vincentelli and P. R. Gray, "Top-Down, Constraint-Driven Methodology Based Generation of n-bit Interpolative Current Source D/A Converters," in *Proc. IEEE CICC*, pp. 369–372, May 1994.

[12] H. Chang, A. Sangiovanni-Vincentelli, F. Balarin, E. Charbon, U. Choudhury, G. Jusuf, E. Liu, E. Malavasi, R. Neff and P. Gray, "A Top-down, Constraint-Driven Design Methodology for Analog Integrated Circuits," in *Proc. IEEE CICC*, pp. 841–846, May 1992.

[13] E. Charbon, E. Malavasi, U. Choudhury, A. Casotto and A. Sangiovanni-Vincentelli, "A Constraint-Driven Placement Methodology for Analog Integrated Circuits," in *Proc. IEEE CICC*, pp. 2821–2824, May 1992.

[14] E. Charbon, E. Malavasi, D. Pandini and A. Sangiovanni-Vincentelli, "Imposing Tight Specifications on Analog IC's through Simultaneous Placement and Module Optimization," in *Proc. IEEE CICC*, pp. 525–528, May 1994.

[15] E. Charbon, E. Malavasi, D. Pandini and A. Sangiovanni-Vincentelli, "Simultaneous Placement and Module Optimization of Analog IC's," in *Proc. IEEE/ACM DAC*, pp. 31–35, June 1994.

[16] E. Charbon, E. Malavasi and A. Sangiovanni-Vincentelli, "Generalized Constraint Generation for Analog Circuit Design," in *Proc. IEEE ICCAD*, pp. 408–414, November 1993.

[17] H. H. Chen and E. S. Kuh, "Glitter. A Gridless Variable-Width Channel Router," *IEEE Trans. on CAD*, vol. CAD-5, n. 4, pp. 459–465, October 1986.

[18] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint-Based Channel Routing for Analog and Mixed-Analog Digital Circuits," in *Proc. IEEE ICCAD*, pp. 198–201, November 1990.

[19] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint Generation for Routing Analog Circuits," in *Proc. IEEE/ACM DAC*, pp. 561–566, June 1990.

[20] U. Choudhury and A. Sangiovanni-Vincentelli, "An Analytical-Model Generator for Interconnect Capacitances," in *Proc. IEEE CICC*, pp. 861–864, May 1991.

[21] G. W. Clow, "A Global Routing Algorithm for General Cells," in *Proc. IEEE/ACM DAC*, pp. 45–51, 1984.

[22] A. Demir, E. Liu and A .Sangiovanni-Vincentelli, "Time-Domain non-Monte Carlo Noise Simulation for Nonlinear Dynamic Circuits with Arbitrary Excitations," in *Proc. IEEE ICCAD*, pp. 598–603, November 1994.

[23] A. Demir, E. Liu, A. Sangiovanni-Vincentelli and I. Vassiliou, "Behavioral Simulation Techniques for Phase/Delay-Locked Systems," in *Proc. IEEE CICC*, pp. 453–456, May 1994.

[24] S. Donnay, K. Swings, G. Gielen and W. Sansen, "A Methodology for Analog High-Level Synthesis," in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 373–376, May 1994.

[25] EECS290Y class, Professor Bernhard Boser, Electrical Engineering and Computer Sciences, University of California at Berkeley, Fall, 1993.

[26] M. Degrauwe et al., "IDAC: An Interactive Design Tool for Analog CMOS Circuits," *IEEE Journal of Solid State Circuits*, vol. SC-22, No.6, pp. 1106–1116, December 1987.

[27] M. G. R. Degrauwe et al., "Towards an Analog System Design Environment," *IEEE Journal of Solid State Circuits*, vol. 24, n. 3, pp. 659–671, June 1989.

[28] M.G. DeGrauwe et al, "An Analog Expert Design of Analog Integrated Circuits," in *Proc. IEEE International Solid-State Circuits Conference*, pp. 212–213, 1987.

[29] R. Harjani et al, "A Prototype Framework for Knowledge-Based Analog Circuit Synthesis," in *Proc. Design Automation Conference*, pp. 42–49, 1987.

[30] R.W. Brodersen et al., *Users' Manual*, Dept. of EECS, University of California at Berkeley, 1990.

[31] E. Felt, E. Charbon, E. Malavasi and A. Sangiovanni-Vincentelli, "An Efficient Methodology for Symbolic Compaction of Analog IC's with Multiple Symmetry Constraints," in *Proc. European Design Automation Conference*, pp. 148–153, September 1992.

[32] E. Felt, E. Malavasi, E. Charbon, R. Totaro and A. Sangiovanni-Vincentelli, "Performance-Driven Compaction for Analog Integrated Circuits," in *Proc. IEEE CICC*, pp. 1731–1735, May 1993.

[33] E. Felt, A. Narayan and A .Sangiovanni-Vincentelli, "Measurement and Modeling of MOS Transistor Current Mismatch in Analog IC's," in *Proc. IEEE ICCAD*, pp. 272–277, November 1994.

[34] E. Felt and A .Sangiovanni-Vincentelli, "Testing of Analog Systems Using Behavioral Models and Optimal Experimental Design Techniques," in *Proc. IEEE ICCAD*, pp. 672–678, November 1994.

[35] "The Future Role of the Analog Designer", Discussion session at *Solid-State Circuits Conference*, Februrary, 24, 1994.

[36] D. J. Garrod, R. A. Rutenbar and L. R. Carley, "Automatic Layout of Custom Analog Cells in ANAGRAM," in *Proc. IEEE ICCAD*, pp. 544–547, November 1988.

[37] G. Gielen, E. Liu, A. Sangiovanni-Vincentelli and P. Gray, "Analog Behavioral Models for Simulation and Synthesis of Mixed-Signal Systems," in *Proc. EDAC*, March 1992.

[38] N. Gohar, P. Cheung and C. Pun, "RACHANA: An Integrated Placement and Routing Approach to CMOS Analog Cells," in *Proc. IEEE Int. Symposium on Circuits and Systems*, 1992.

[39] P. R. Gray and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, pp. 717–718, J. Wiley & Sons, 1977.

[40] P. R. Gray and R. G. Meyer, *Analysis and design of analog integrated circuits*, J. Wiley & Sons, New York, 1977.

[41] K. C. Gupta, R. Garg and R. Chardha, *Computer Aided Design for Microwave Circuits*, Technical report, Norwood, MA: Artech House, 1981.

[42] R. Harjani, R. A. Rutenbar and L. R. Carley, "Analog Circuit Synthesis for Performance in OASYS," in *Proc. IEEE ICCAD*, pp. 492–495, November 1988.

[43] D. Harrison, P. Moore, R. L. Spickelmier and A. R. Newton, "Data Management and Graphics Editing in the Berkeley Design Environment," in *Proc. IEEE ICCAD*, pp. 24–27, November 1986.

[44] D. S. Harrison, A. R. Newton, R. L. Spickelmier and T. J. Barnes, "Electronic CAD Frameworks," *Proc. of the IEEE*, vol. 78, n. 2, pp. 393–417, February 1990.

[45] P. Harvey J, M. I. Elmasry and B. Leung, "STAIC: An Interactive Framework for Synthesizing CMOS and BiCMOS Analog Circuits," *IEEE Trans. on CAD*, vol. 11, n. 11, pp. 1402–1417, November 1992.

[46] G. Jusuf, P. R. Gray and A. Sangiovanni-Vincentelli, "CADICS - Cyclic Analog-To-Digital Converter Synthesis," in *Proc. IEEE ICCAD*, pp. 286–289, November 1990.

[47] G. Jusuf, P. R. Gray and A. Sangiovanni-Vincentelli, "A Compiler for CMOS Analog-To-Digital Converters," in *TECHCON*, pp. 347–350, October 1990.

[48] M. Kamon, M. Tsuk, C. Smithhisler and J. White, "Efficient Techniques for Inductance Extraction of Complex 3-D Geometries," in *Proc. IEEE ICCAD*, pp. 438–442, November 1992.

[49] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, n. 4598, pp. 671–680, May 1983.

[50] H. Y. Koh, C. H. Sèquin and P. R. Gray, "Automatic synthesis of operational amplifiers based on analytic circuit models," in *Proc. IEEE ICCAD*, pp. 502–505, 1987.

[51] H. Y. Koh, C. H. Sèquin and P. R. Gray, "Automatic Layout Generation for CMOS Operational Amplifiers," in *Proc. IEEE ICCAD*, pp. 548–551, November 1988.

[52] H. Y. Koh, C. H. Sèquin and P. R. Gray, "OPASYN: A compiler for CMOS operational amplifiers," *IEEE Trans. on CAD*, vol. 9, n. 2, pp. 113–126, February 1990.

[53] E. S. Kuh and M. Marek-Sadowska, "Global Routing", in *Layout Design and Verification*, ch. 5, pp. 169–198. T.Ohtsuki Ed., North Holland, 1986.

[54] K. R. Lakshmikumar, R. A. Hadaway and M. A. Copeland, "Characterization and modeling of Mismatch in MOS Transistors for Precision Analog Design," *IEEE Journal of Solid State Circuits*, vol. SC-21, n. 6, pp. 1057–1066, December 1986.

[55] C. Lee, "An algorithm for path connections and applications," *IRE Trans. Electron. Computer*, vol. EC-10, pp. 346–365, September 1961.

[56] T.H. Lee and J.F. Bulzacchelli, "A 155-MHz Clock Recover Delay- and Phase-Locked loop," *IEEE Journal of Solid State Circuits*, vol. 27, No. 12, December 1992.

[57] Yuh-Min Lin, *Performance Limitations on High-Resolution Video-Rate Analog-Digital Interfaces*, Memorandum UCB/ERL M90/55, UCB, June 1990.

[58] E. Liu, *Analog Behavioral Simulation and Modeling*, Technical report, Ph.D. Thesis, University of California at Berkeley, 1993.

[59] E. Liu, G. Gielen, H. Chang, A. Sangiovanni-Vincentelli and P. Gray, "Behavioral Modeling and Simulation of Data Converters," in *Proc. IEEE Int. Symposium on Circuits and Systems*, 1992.

[60] E. Liu, W. Kao, E. Felt and A. Sangiovanni-Vincentelli, "Analog Testability Analysis and Fault Diagnosis using Behavioral Modeling," in *Proc. IEEE CICC*, pp. 413–416, May 1994.

[61] E. Liu and A. Sangiovanni-Vincentelli, "Behavioral Representations for VCO and Detectors in Phase-Lock Systems," in *Proc. IEEE Custom Integrated Circuits Conference*, 1992.

[62] E. Liu, A. Sangiovanni-Vincentelli, G. Gielen and P. Gray, "A Behavioral Representation for Nyquist Rate A/D Converters," in *Proc. IEEE ICCAD*, pp. 386–389, November 1991.

[63] E. W. Y. Liu, H. C. Chang and A. L. Sangiovanni-Vincentelli, "Analog System Verification in the Presence of Parasitics using Behavioral Simulation," in *Proc. IEEE/ACM DAC*, pp. 159–163, June 1993.

[64] E. W. Y. Liu and A. L. Sangiovanni-Vincentelli, "Nyquist Data Converter Testing and Yield Analysis using Behavioral Simulation," in *Proc. IEEE ICCAD*, pp. 341–348, November 1993.

[65] D. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley Publishing Company, 2nd Ed., 1984.

[66] C. Makris and C. Toumazou, "ISAID: Qualitative Reasoning and Trade-off Analysis in Analog IC Design Automation," in *Proc. IEEE Int. Symposium on Circuits and Systems*, 1992.

[67] C. Makris and C. Toumazou, "Qualitative Reasoning in Analog IC Design Automation," in *Proc. IEEE Custom Integrated Circuits Conference*, 1992.

[68] E. Malavasi, E. Charbon, G. Jusuf, R. Totaro and A. Sangiovanni-Vincentelli, "Virtual Symmetry Axes for the Layout of Analog IC's," in *Proc.* 2^{nd} *ICVC, Seoul, Korea*, pp. 195–198, October 1991.

[69] E. Malavasi, U. Choudhury and A. Sangiovanni-Vincentelli, "A Routing Methodology for Analog Integrated Circuits," in *Proc. IEEE ICCAD*, pp. 202–205, November 1990.

[70] E. Malavasi and A. Sangiovanni-Vincentelli, "Area Routing for Analog Layout," *IEEE Trans. on CAD*, vol. 12, n. 8, pp. 1186–1197, August 1993.

[71] S. W. Mehranfar, "A Technology-Independent Approach to Custom Analog Cell Generation," *IEEE Journal of Solid State Circuits*, vol. 26, n. 3, pp. 386–393, March 1991.

[72] C. Michael and M. Ismail, "Statistical Modeling of Device Mismatch for Analog MOS Integra ted Circuits," *IEEE Journal of Solid State Circuits*, vol. 27 no. 2, pp. 154–166, Feb 1992.

[73] L. Milor and A. Sangiovanni-Vincentelli, "Optimal Test Set Design for Analog Circuits," in *Proc. IEEE ICCAD*, pp. 294–297, November 1990.

[74] L. Milor and V. Visvanathan, "Detection of Catastrophic Faults in Analog Integrated Circuits," *IEEE Trans. on CAD*, vol. CAD-8, n. 2, pp. 114–130, February 1989.

[75] "MINimum Iterations", B. Lokanathan, Department of Electrical Engineering at the University of Rochester, adapted from R. Salazar and S. Sen, "MINIT algorithm for Linear Programming," *Collected Algorithms from CACM*, algorithm #333,1968.

[76] "Modular In-core Nonlinear Optimization System (MINOS 5.3)", Systems Optimization Laboratory, Department of Operations Research, Stanford University.

[77] S. Mitra, R. Rutenbar, L. Carley and D. J. Allstot, "Substrate-Aware Mixed-Signal Macro-cell Placement in WRIGHT," in *Proc. IEEE CICC*, pp. 529–532, May 1994.

[78] B. A. Murtagh and M. A. Saunders, *MINOS 5.1 User's Guide*, Technical Report Rep. SOL 83-20R, Dept. of Operations Research, Stanford University, Stanford, CA, January 1987.

[79] Y .Nakamura, T. Miki, A. Maeda, H. Kondoh and N. Yazawa, "A 10-b 70-MS/s CMOS D/A Converter," *IEEE Journal of Solid State Circuits*, vol. 26, No. 4, pp. 637–642, April 1991.

[80] R. Neff. Ph.D. Thesis. Electrical Engineering and Computer Sciences, University of California at Berkeley.

[81] A. R. Newton, "Symbolic Layout and Procedural Design", in *Design Systems for VLSI Circuits*, pp. 65–112. De Micheli et al. Eds., Martinus Nijhoff, 1987.

[82] K. A. Nishimura, *Optimum Partitioning of Analog and Digital Circuitry in Mixed-Signal Circuits for Signal Processing*, Technical report, Ph.D. Thesis, University of California at Berkeley, July 1993.

[83] W. Nye, D. C. Riley, A. Sangiovanni-Vincentelli and A. L. Tits, "DELIGHT-SPICE: An Optimization-Based System for the Design of Integrated Circuits," *IEEE Trans. on CAD*, vol. 7, n. 4, pp. 501–519, April 1988.

[84] E. Ochotta, L Carley and R. Rutenbar, "Analog Circuit Synthesis for Large, Realistic Cells: Designing a Pipelined A/D Converter with ASTRX/OBLX," in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 365–368, May 1994.

[85] M. J. M. Pelgrom, "A 10-b 50-Mhz CMOS D/A Converter with 75-Ω Buffer," *IEEE Journal of Solid State Circuits*, vol. 25, pp. 1347–1352, December 1990.

[86] M. J. M. Pelgrom, A. C. J. Duinmaijer and A. P. G. Welbers, "Matching Properties of MOS Transistors," *IEEE Journal of Solid State Circuits*, vol. 24, pp. 1433–1440, October 1989.

[87] J. Rijmenants, J. B. Litsios, T. R. Schwarz and M. G. R. Degrauwe, "ILAC: An Automated Layout Tool for Analog CMOS Circuits," *IEEE Journal of Solid State Circuits*, vol. 24, n. 2, pp. 417–425, April 1989.

[88] A. E. Ruehli, "Survey of Computer-Aided Electrical Analysis of Integrated Circuit Interconnections," *IBM Journal of Research and Development*, vol. 23, n. 6, pp. 626–639, November 1979.

[89] A. E. Ruehli and P. A. Brennan, "Efficient Capacitance Calculations for three-Dimensional Multiconductor Systems," in *IEEE Trans. on Microwave Theory and Techniques*, volume 21, pp. 76–82, February 1973.

[90] R. A. Rutenbar, "Analog Design Automation: Where are we? Where are we going?," in *Proc. IEEE CICC*, pp. 1311–1318, May 1993.

[91] H. Schouwenaars, D. Groeneveld and H. Termeer, "A Low-Power Stereo 16-bit CMOS D/A Converter for Digital Audio," in *Proc. IEEE International Solid-State Circuits Conference*, 1988.

[92] C. Sechen, *VLSI Placement and Routing Using Simulated Annealing*, Kluwer Academic, Boston, MA, 1988.

[93] C. Sechen and A. Sangiovanni-Vincentelli, "Chip-Planning, Placement and Global Routing of Macro/Custom Cell IC's using Simulated Annealing," in *Proc. IEEE/ACM DAC*, pp. 73–80, June 1988.

[94] J. Shao and R. Harjani, "Macromodeling of Analog Circuits for Hierarchical Circuit Design," in *Proc. IEEE ICCAD*, pp. 656–663, November 1994.

[95] J. M. Shyu and A. Sangiovanni-Vincentelli, "ECSTASY: A new Environment for IC Design Optimization," in *Proc. IEEE ICCAD*, pp. 484–487, November 1988.

[96] Mark Sitkowski, "The Macro Modeling of Phase Locked Loops for the Spice Simulator," *IEEE Circuits and Devices Magazine*, vol. 7, No. 2, pp. 11–15, March 1991.

[97] G. Szentirmai, "FILSYN - A General Purpose Filter Synthesis Program," in *Proc. of the IEEE*, pp. 65:1443–1458, October 1977.

[98] M. P. Vecchi and S. Kirkpatrick, "Global Wiring by Simulated Annealing," *IEEE Trans. on CAD*, vol. CAD-2, n. 4, pp. 215–222, October 1983.

[99] K.M. Ware, H-S. Lee and C.G. Sodini, "A 200-MHz CMOS Phase-Locked Loop with Dual Phase Detectors," *IEEE Journal of Solid State Circuits*, vol. 24, No. 6, December 1989.

[100] P. H. Xiao, E. Charbon, T. Van Duzer and S. R. Whiteley, "INDEX: An Inductance Extractor for Superconducting Circuits," in *1992 Applied Superconductivity Conference*, August 1992.

[101] J. G. Xiong, "Algorithms for Global Routing," in *Proc. IEEE/ACM DAC*, 1986.

[102] H. Yaghutiel, A. Sangiovanni-Vincentelli and P. R. Gray, "A Methodology for Automated Layout of Switched-Capacitor Filters," in *Proc. IEEE ICCAD*, pp. 444–447, 1986.

[103] H. Yaghutiel, S. Shen, P. R. Gray and A. Sangiovanni-Vincentelli, "Automatic Layout of Switched-Capacitor Filters for Custom Applications," in *Proc. IEEE International Solid-State Circuits Conference*, pp. 170–171, February 1988.

[104] I.A. Young, J.K. Greason and K.L. Wong, "A PLL Clock Generator with 5 to 110 MHz of Lock Range for Microprocessors," *IEEE Journal of Solid State Circuits*, vol. 27, No. 11, November 1992.

[105] J. Yuan and C. Svensson, "High-Speed CMOS Circuit Technique," *IEEE Journal of Solid State Circuits*, pp. 62–70, February 1989.