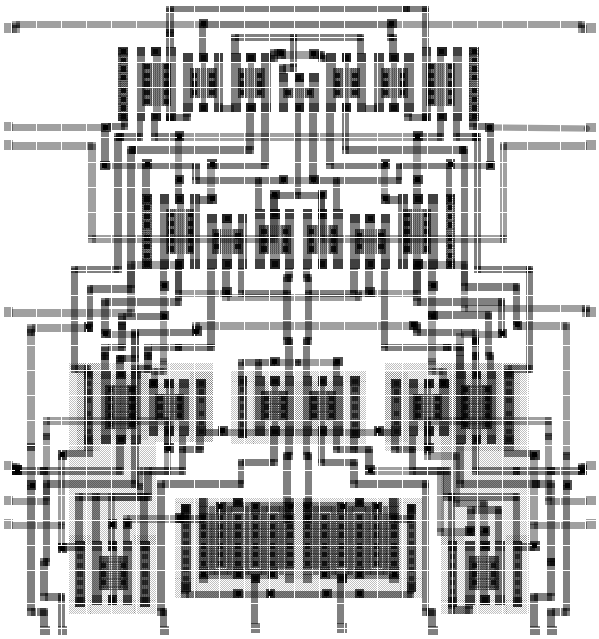

Figure 5: Final Layout for the Operational Amplifier

The results shown in Table 4 were obtained from a SPICE simulation considering all the parasitics extracted from the layout.

Specification	Min	Max	Result
Unity Gain Frequency	6 MHz	∞	6.06 MHz
Phase Margin	75^o	∞	76^o
Gain (dB)	60 dB	66 dB	63.8 dB

Table 4: Operational Amplifier Synthesis Results

In this case the placement was performed by the same module generator, CADICS, which had carried out the schematic generation. Alternatively PUPPY-A could have been used for the placement allowing for a larger search of the placement design space.

5 Conclusion

A novel design methodology has been presented for analog circuit design. The 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. This goal is achieved through the use of a hierarchical approach where early verification and constraint propagation is favored. High-level behavioral simulation is used for immediate high-level verification of circuit performances and used for supporting the constraint-driven layout tools. The tools presented are a first cut at the implementation of the design methodology advocated here. Future work includes the use of more optimization tools such as ECSTASY [21] to improve the performance of a given architecture; a new set of flexible module generators to offer a wide variety of choices in the implementation of complex systems; additional behavioral models for basic analog functions, such as filters, phase-locked loops, oversampled ADCs, and phase detectors; a complete mixed-signal, mixed-level mixed-domain simulator; and tools to support testing such as one which will evaluate a particular test set based on the probabilistic description of the circuit.

6 Acknowledgements

The authors would like to thank Micro Linear for providing design examples. This research has been partially supported by The National Science Foundation; the Swiss Science Foundation, Bern, Switzerland; SRC; and DARPA.

References

[1] J. Rijmenants et al., "ILAC: An Automated Layout Tool for Analog CMOS Circuits", *IEEE Journal of Solid State Circuits*, vol. 24, n. 2, pp. 417–425, April 1989.

[2] 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.

[3] J. M. Cohn, D. J. Garrod, R. A. Rutenbar and L. R. Carley, "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing", *IEEE Journal of Solid State Circuits*, vol. 26, n. 3, pp. 330–342, March 1991.

[4] 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.

[5] E. Liu, A. Sangiovanni-Vincentelli, G. Gielen and P. Gray, "A Behavioral Representation for Nyquist Rate A/D Converters", *Proceedings on IEEE ICCAD*, November 1991.

[6] G. Gielen, E. Liu, A. Sangiovanni-Vincentelli and P. Gray, "Analog Behavioral Models for Simulation and Synthesis of Mixed-Signal Systems", *subm. European Design Automation Conference*, March 1992.

[7] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint Generation for Routing Analog Circuits", in *Proc. Design Automation Conference*, pp. 561–566, June 1990.

[8] A. Casotto, F. Romeo and A. Sangiovanni-Vincentelli, "A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells", *IEEE Trans. on CAD*, vol. CAD-6, n. 5, pp. 838–847, Sep 1987.

[9] E. Charbon, E. Malavasi, U. Choudhury, A. Casotto and A. Sangiovanni-Vincentelli, "A Constraint-Driven Placement Methodology for Analog Integrated Circuits", in *Proc. IEEE Custom Integrated Circuits Conference*, May 1992.

[10] 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.

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

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

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

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

[15] 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.

[16] 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.

[17] G. Jusuf and P. R. Gray, "An Improved 1-Bit/Cycle Algorithmic A/D Converter Without High-Precision Comparators", Memorandum UCB/ERL M90/69, UCB, August 1990.

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

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

[20] 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.

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

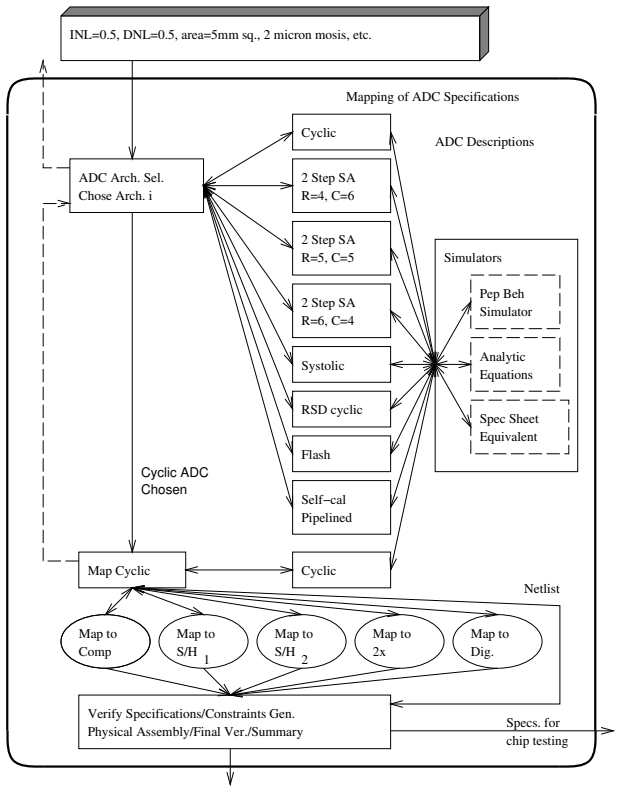

Figure 4: Example Application of the Design Methodology (ADC)

culating the results for the flash ADC, and the self-calibrating pipelined ADC was treated as a standard cell, thus having the specifications already given.

The mapping function is now performed by CADICS[3]. As shown in Figure 2 and in Figure 4, the ADC is broken down into two Sample and Holds (S/Hs), a gain of two stage (2x), a comparator (Comp), and digital circuitry (DigCon). The mapping function now generates specifications for these sub-components. These are shown in Table 2.

[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.

Type of ADC	Res.	INL LSB	DNL LSB	Area mm^2	Tool to generate specs
NEEDED SPECS	10	0.50	0.50	4.0	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	Ana. Eqns
Self-cal Pipelined	13	2.00	0.60	12	given [18]

Table 1: ADC Specifications and Simulation Results

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 2: ADC sub-block Specifications

4.2 The Architectural Selection, Optimization, and Mapping of an Interpolative 10 Bit DAC

Another example of the architectural selection and mapping step is the determination of the MSB, N_1 bits, and LSB, N_2 bits, partitioning of an interpolative 10 bit DAC (see Figure 2). The interpolative DAC architecture is chosen via an initial architectural selection step similar to the method shown in the previous example. Now, given a set of lumped process variation parameters, a set of constraints, an objective function which is to be minimized or maximized, and this particular architecture, the partitioning can be determined using a non-linear optimizer [19] and a behavioral simulator [20]. This 10 bit DAC 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. The W/L of each of the sets of transistors must be constrainted to be greater than 4 for speed concerns; the Ws must be greater than 3μm and Ls must be greater than 2μm for design rule reasons; the lumped process variations are set up to be $\sigma_L = 0.02\mu$m and $\sigma_W = 0.02\mu$m; the INL of the 10 bit DAC must be less than 2 LSB[5]; and the DNL must be less than 0.5 LSB. The objective is to minimize the total active area of the transistors. The results[6] are shown in Table 3. From this, it can be seen that the requirements can be met for all cases shown; however, the one which gives the minimum area is $N_1 = 5$.

N_1	W_1 μm	L_1 μm	W_2 μm	L_2 μm	W_M μm	L_M μm	Area mm^2	INL lsb	DNL lsb	CPU 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 3: DAC Optimization Results

4.3 Physical Assembly of an Operational Amplifier

Figure 5 shows the layout of a Folded Cascode Operational Amplifier synthesized with our methodology. Transistor elements and biasing contacts were automatically generated and placed by CADICS. The specification constraints, reported in Table 4, were used to generate a set of bounds for interconnection parasitics with PARCAR. These bounds were used to drive the area router, ROAD. After the circuit was routed all layout parasitics were extracted using our parasitics extraction package. A performance verification based on SPICE showed that all specifications were met (see Table 4). Finally, the circuit was compacted preserving symmetries and respecting all the parasitic constraints.

[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.

if the set of criteria is too difficult to meet.

The tools available in our methodology have been specifically designed to meet the constraints on parasitics, to account for symmetries necessary in the design, and to achieve device matching. Feedback is provided at all layout phases, thus providing a mechanism to transmit a failed specification mapping so that a retry can be made.

Placement is performed by PUPPY-A [8], a tool based on Simulated Annealing which has been developed [9] to take into account parasitic constraints, symmetry specifications, device matching, well separation, and device abutment. Heuristics on the parameterization of the algorithm and the enforcement of constraints at low temperature enables the tool to guarantee routability and the meeting of the specified high-level performances.

The channel router, ART [10], and the area router, ROAD [11], are two routing tools which also respect symmetries and parasitic constraints. To handle the parasitic constraints, both are driven by the parasitic constraints generator, PARCAR. Cross-coupling capacitances are controlled by limiting wire crossings and by keeping safe distances between parallel wire segments. *Shields* can be introduced to reduce cross-coupling when attempting to minimize wire spacing. Proper routing layers are chosen to control stray resistances and stray capacitances to the substrate. Analytical models for self and coupling capacitances are automatically generated by CAPMOD[12].

Mono-dimensional compaction is performed by SPARCS [13], based on the constraint-graph longest-path algorithm. Symmetries are controlled by using active constraints. All parasitics are kept within their bounds by properly spacing parallel wire segments. To continue the optimization/generation process of the layout, information from SPARCS can be fed back to the constraints generator.

3.3 Module Generation

Currently, our module generation tools are capable of generating several analog functional blocks. They can generate circuits as simple as a comparator to circuits as complicated as an ADC block.

To generate the low-level analog blocks such as Operational Amplifiers, OPASYN [14], an automatic synthesis tool for the generation of OpAmps based on analytic circuit models can be used. Given a set of specifications, this program performs the necessary optimizations to properly perform transistor sizing. It outputs this information to a layout generator, and a performance summary is also presented. Originally, this program only considered the generation of operational amplifiers. However, it has been expanded so that it is now capable of generating comparators as well as input buffers.

At the middle-level, we have developed two module generators, one capable of generating switched-capacitor filters, ADORE [15], and the other, a tool capable of generating Algorithmic ADCs, CADICS [16].

ADORE is a switched-capacitor filter module generator that places special emphasis on the physical design aspects of module generation. 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.

CADICS is a technology independent ADC module generator which generates ADC netlists and layouts from a given set of specifications. The synthesis flow in CADICS is divided into two parts: circuit synthesis and layout synthesis. The circuit synthesis consists of an architectural selector, a sub-block requirement generator, some customized routines, a behavioral simulator, an area optimizer, and a low-level synthesis block to call OPASYN. The layout synthesis conforms to our methodology for physical assembly: performing area-optimization, cell generation, placement, and routing. Currently, CADICS is only capable of generating 1-bit cyclic Algorithmic ADCs [17]. The capabilities of CADICS are also being expanded to allow coverage for a wider spectrum of resolutions and sampling rates.

4 Examples

We provide three examples to illustrate our methodology. The first example illustrates how behavioral simulation can be used in the architectural selection of a 10 bit ADC that is generated by CADICS. The second example uses an optimizer as well as a behavioral simulator to perform architectural selection and mapping. The last one demonstrates how performance specifications can be used to drive the layout assembly of an Operational Amplifier. Where these examples fit into our example design hierarchy is shown in Figure 2.

4.1 The Architectural Selection and Mapping of a 10 Bit ADC

Figure 4 shows our design methodology re-drawn for this ADC example. The specifications require 10 bits of resolution, a maximum Integral Non-Linearity (INL) of 0.5 LSB, a maximum Differential Non-Linearity (DNL) of 0.5 LSB, and a maximum area of 5 mm^2 for a 2 μm MOSIS process. Five basic ADC architectures are considered: cyclic, 2 step successive approximation (SA), systolic, and self-calibrating pipelined. Within the cyclic ADC class, two are considered- one with Opamp 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 resolution in the resistor array, and a third with 6 bits of resolution in the resistor array. Shown in Table 1 are the simulation results. The behavioral level simulations all ran under an hour except for for the Systolic ADC which required 4.5 hours[1].

Most of the simulations were accomplished with PEPADC, our behavioral simulator. Based on the technology assumed[2] statistical model parameters are derived for low-level components such as input referred offset of OpAmps and comparators. Statistical model parameters for the gain stages, resistor string DACs, and charge redistribution DACs are derived from capacitor, resistor, and input referred offset mismatches. Then, behavioral level parameters such as INL and DNL are computed by PEPADC using the component models. Analytical Equations were used in cal-

[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.

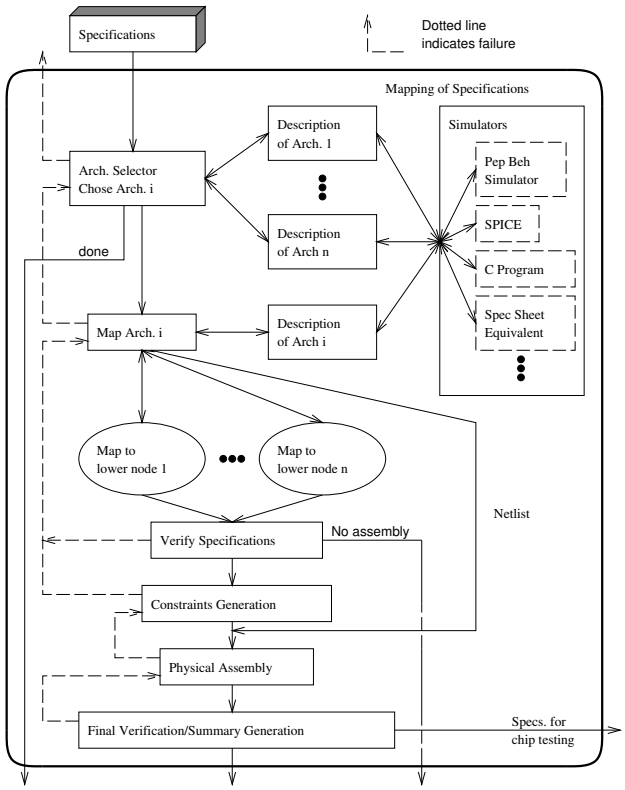

Figure 3: Design Methodology

formed. If this fails to meet the requirements then 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 plus simulation with the same simulator used in architectural selection. The "extraction" process spans the entire range from no parasitic extraction to complete extraction. 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 specifications generated.

This final block also aids in the testing by outputing information to a database to be used later by the tester. It also considers what additional hardware may be required to ease the testing. It may suggest alternative architectures which are less costly to test. This information is fed back to the mapping function, and can even be fed back to the architectural selection function.

We believe that our design methodology is significantly different from the ones currently employed by circuit designers. Today, a typical design cycle starts with a set of specifications for an integrated circuit drawn in conjunction with the customer. Then the designer takes these specifications and performs a first level decomposition with little simulation. The partitioning is accomplished based mostly on the experience of the designers. Typically, today, designers resort to a bottom-up approach. Low level circuits are built, tested, verified, and assembled hierarchically from the bottom-up until the first level decomposition blocks are reached. The main problem with this design technique is that if the final blocks do not meet the specifications, the entire circuit has to be redesigned, possibly all the way from the bottom again. This can be very time consuming and costly. A typical solution to this is designing with overconstraints on the lower blocks. This, however, is also costly, because a non-optimal solution is usually reached. Our methodology attempts to prevent these problems. We use behavioral simulations for early verification and design space exploration to guide the experience and the expertise of the designer. Each of the lower blocks is constrained as the tree is descended to match the performance specifications. Thus, at any time, we are reasonably sure that the original specifications are being met. We need not go to the bottom of our design and back up before realizing that a costly mistake has been made.

3 The Tool Set

The tool set that we present here can be divided into three categories: Behavioral Simulation Tools, Physical Assembly Tools, and Module Generation Tools. As a point of policy, we state here that all present and future tools we design will integrate into our methodology, thus providing continuity in our tool set. We hope for the adoption of our methodology by other tool writers to accelerate the process of expanding the tool set.

3.1 Behavioral Simulation

Our efforts in behavioral simulation include both the modeling of and the simulation of a wide range of mixed-signal circuits from building blocks such as operational amplifiers, comparators, voltage-controlled oscillators (VCOs), phase detectors, to Analog-to-Digital Converters (ADCs), Digital-to-Analog Converters (DACs), filters, phase-locked loops (PLLs), and systems containing these components. We emphasize the development of models for both ideal and non-ideal circuit behaviors, both of which are absolutely essential for accurate simulation of systems.

Currently, models for the building blocks are available. A high level behavioral model has been developed for Nyquist rate ADCs [5] [6] that captures all of the second order effects such as process variations and noise. Models for DACs, VCOs, oversampled converters, and phase detectors are being developed as well. To support the parameter extraction and simulation, an object-oriented library, Pep, of C++ objects which implements our behavioral models has been built. The user constructs new objects for mixed signal circuits such as ADCs by first assembling component objects (e.g. comparators, sample-and-hold). Then, parameter extraction is performed by a special simulator object. For the ADC, parameter extraction is done by PepADC which extracts model parameters for noise and process variations from the input of the user.

3.2 Physical Assembly

Our approach consists of mapping high-level constraints on electrical performances explicitly to a set of bounds on parasitics, which can then be directly controlled by the layout tools. If the parasitic bounds are respected, then the high-level performance constraints have a high probability of being met. Parasitic constraints are generated by Parcar [7] using sensitivity analysis to determine the criticality of each parasitic to performance degradation. Among the numerous possible sets of bounds, the one maximizing the flexibility of the layout tools is chosen. This choice can then be dynamically updated by the tools themselves

design-reusability in its general framework.

This paper is organized as follows: In Section 2, the proposed methodology is presented. In Section 3, the tool set is described. In Section 4, three examples are shown. And finally, these sections are followed by our conclusions in which future work is discussed.

2 The Methodology

Figure 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.

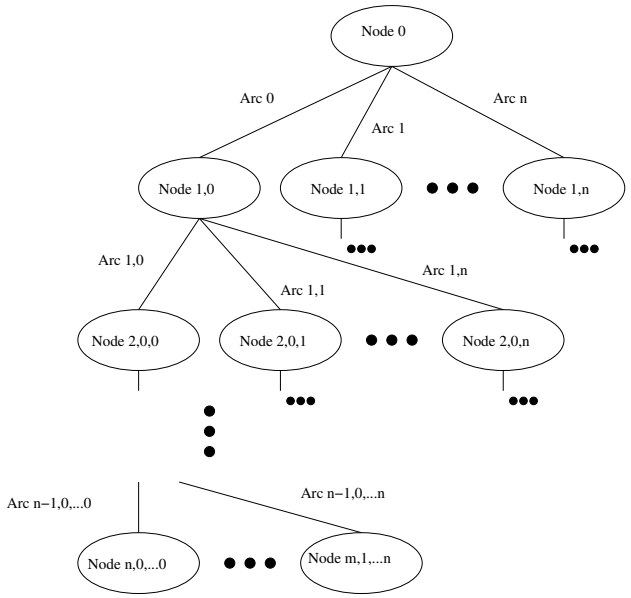

Figure 1: Design Hierarchy

An example decomposition is shown in Figure 2. Here, a mixed-mode 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.

Our methodology, illustrated in Figure 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

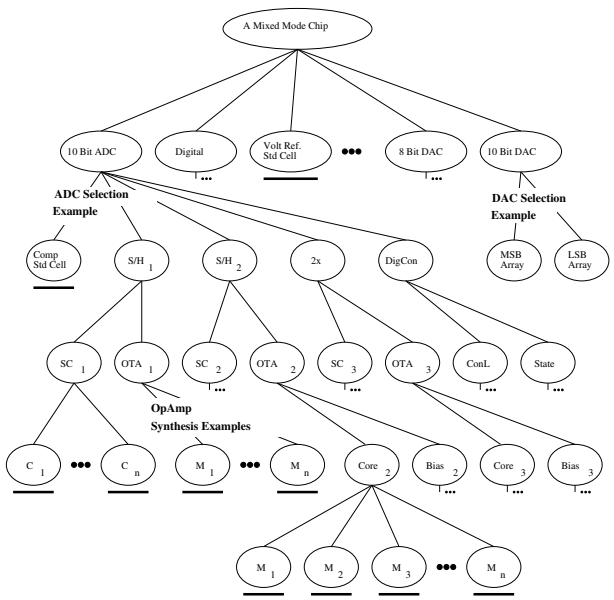

Figure 2: Example Design Hierarchy for a chip

be performed is architecture selection. Simulators and optimizers are used to aid the decision-making 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. 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 we proceed 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 non-linear 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 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, we have two options. We can 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 by the user, or it can be done automatically. As before, if this step fails, the flow control is returned to the mapping function. If it is successful, then the constraint-driven physical assembly is per-

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

Henry Chang Alberto Sangiovanni-Vincentelli Felice Balarin Edoardo Charbon Umakanta Choudhury
Gani Jusuf Edward Liu Enrico Malavasi Robert Neff
Paul R. Gray

Electronics Research Laboratory
Department of Electrical Engineering and Computer Sciences
207-13 Cory Hall
University of California, Berkeley, CA 94720, U.S.A.

Abstract

We describe a top-down, constraint-driven design methodology for Analog Circuits. We delineate some of the tools that support it. Finally, we conclude with examples to better illustrate the methodology and its integration with the tool set.

1 Introduction

The analog section of a complex mixed-signal system is often small but it is its design that often consumes the most amount of time. 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 [1] [2]. Methods for very specific mid-to-low level circuits have also been proposed [3]. More recently automatic support tools for the experienced designer have been developed [4]; however, none of these provide integration of all of these features and missing from them are some of what we believe to be very fundamental ones. In this paper, we propose a new design methodology and the necessary set of tools that supports it for effective analog design. The key points of our methodology are (1) top-down hierarchical process starting from the behavioral level based on early verification and constraint propagation; (2) bottom-up accurate extraction and verification; (3) automatic and interactive synthesis of components with specification constraint-driven layout design tools; (4) 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; (5) 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; the 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. We recognize 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, we do not believe 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. We do believe, 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. We also maintain, 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 our methodology does accommodate tools that favor