Structured Simulation-Based Analog Design Synthesis

Rob A. Rutenbar
Department of Electrical and Computer Engineering
Carnegie Melon University
Pittsburgh, Pennsylvania 15213 USA

Abstract

Early generations of analog synthesis tools failed to migrate into mainstream use primarily because of difficulties in reconciling the simplified models required for synthesis with the industrial-strength simulation environments required for validation. We have recently seen the emergence of *simulation-based synthesis* tools that can size/bias a fixed circuit topology by exploiting the *same* simulation environment created to validate the sized circuit. These methods work remarkably well across a range of difficult analog circuits, and augmented with suitable macromodeling, have also been applied successfully to system-level designs. In this paper we review the motivation and architecture of simulation-based analog synthesis tools, and survey a few recent results from industrial designs.

1. Introduction

Mixed-signal design starts will shortly outnumber purely digital starts. The reason is simple: new communication-oriented ICs require an interface to the external, continuous-valued world. The digital portion of these designs can be attacked with modern cell-based tools for synthesis, mapping, and physical design. The analog portion, however, is still routinely designed by hand. Although it is typically a small fraction of the overall device count (*e.g.,* 10,000 to 20,000 analog transistors), the analog partition in these designs is often the bottleneck because of the lack of automation tools.

The situation worsens as we strive to build *System-on-Chip* (SoC) designs. To manage complexity and time-to-market, SoC designs require a high level of reuse, and cell-based techniques lend themselves

M. Steyaert et al. (eds.), Analog Circuit Design, 95–114.

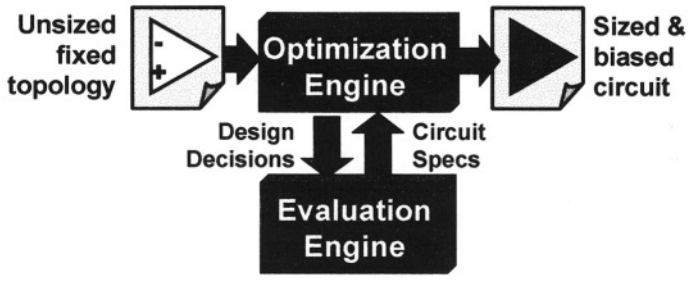

Fig. 1. Abstract model of analog synthesis tools.

well to a variety of strategies for capturing and reusing digital intellectual property (IP). But these digital strategies are inapplicable to analog designs, which rely for basic functionality on tight control of low-level device and circuit properties that vary from technology to technology. The analog portions of these systems are still designed by hand today. They are even routinely ported by hand as a given IC migrates from one fabrication process to another.

A significant amount of research has been devoted to *cell-level analog synthesis,* which we define as the task of sizing and biasing a device-level circuit with 10 to 100 devices. However, as we have noted previously [1], early approaches failed to make the transition from research to practice. This was due primarily to the prohibitive effort needed to reconcile the simplified circuit models needed for synthesis with the "industrial-strength" models needed for validation in a production environment. In digital design, the bit-level, gate-level and block-level abstractions used in synthesis are faithful to the corresponding models used for simulation-based validation. This has *not* been true for analog synthesis.

Fig. 1 illustrates the basic architecture of most analog synthesis tools. An *optimization engine* visits candidate circuit designs and adjusts their parameters in an attempt to satisfy designer-specified performance goals. An *evaluation engine* quantifies the quality of each circuit candidate for the optimizer. Early research focused on trade-offs between the *optimizer* (which wants to visit many circuit candidates) and the *evaluator* (which must itself trade accuracy for speed to allow sufficiently vigorous search). Much of this work was really an attempt

to evade a harsh truth—that analog circuits are difficult and time-consuming to evaluate properly. Even a small cell requires a mix of ac, dc and transient analyses to correctly validate.

In modern design environments, there is enormous investment in simulators, device models, process characterization, and cell sign-off validation methodologies. Indeed, even the sequence of circuit analyses, models, and simulation test-harnesses is treated as valuable IP. Given these facts, it is perhaps no surprise that analog synthesis strategies that rely on exotic, nonstandard, or fast-but-incomplete evaluation engines have fared poorly in real design environments. To trust a synthesis result, one must first trust the methods used to quantify the circuit's performance *during* synthesis. Most prior work failed here.

Given the complexity of, investment in, and reliance on simulator-centric validation approaches for analog cells, we have argued that for a synthesis strategy to have practical impact, it *must* use a simulator-based evaluation engine that is *identical* to that used to validate ordinary manual designs [2]-[5]. Several synthesis and optimization tools that adhere to this dictum have recently emerged, in partial response to these acknowledged problems. Efforts here include academic tools [6]-[8], proprietary (internal) industrial tools [9], and commercially available synthesis systems [10]-[12].

Simulation-based synthesis poses many significant technical challenges. For example, commercial circuit simulators are not designed to be invoked 10,000 times in the inner loop of a numerical optimizer. And the CPU time to visit and simulate this many solution candidates seems, at first glance, intractable even for basic circuits, let alone system-level designs that might require hours just to simulate *once.*

In this paper we review a series of simulation-based synthesis tools developed at Carnegie Mellon University over the last five years that successfully address all these issues [2]-[5]. Our tools rely on four key ideas:

1. We recast the synthesis task as an unconstrained *numerical optimization* problem.

2. We *encapsulate* commercial simulators so that their implementation idiosyncrasies are hidden from our optimization engine.

3. We introduce *distributed global optimization algorithms* that are robust in finding workable circuits, yet require no partially sized starting solution.

4. We exploit *network-level workstation parallelism* to render the overall computation times tractable across a pool of machines.

To demonstrate the practicality of the approach, we review several industrial designs synthesized using these tools.

The remainder of the paper is organized as follows. Section 2 briefly reviews prior approaches. Section 3 gives a brief overview of the basic components of a simulation-based synthesis tool, including issues at both cell and system level. Section 4 reviews several production designs done at Texas Instruments, using prototype versions of the CMU tools. Finally, Section 5 offers concluding remarks.

2. Review of Prior Approaches

2.1 Early Synthesis Approaches for Cells

Referring again to Fig. 1, we can broadly categorize previous work on analog synthesis by how it searches for solutions and how it evaluates each visited circuit candidate. See [13],[14] for more extensive surveys.

The earliest work on synthesis used simple procedural techniques [15], rendering circuits as explicit scripts of equations whose direct evaluation completed a design. Although fast, these techniques proved to be difficult to update, and inaccurate. Numerical search has been used with equation-based evaluators [16]-[18], and even combinatorial search over different circuit topologies [19],[20], but equation-based approaches remain brittle in the face of technology changes. Hierarchical systems [21]-[24] introduced compositional techniques to assemble equation-based subcircuits, but still faced the same update/accuracy difficulties. Qualitative and fuzzy reasoning techniques [25], [26] have been tried, but with limited success. Recent work has emphasized equation-based simplifications that enhance numerical tractability, notably convex modeling with posynomials [27]. However, it remains prohibitively expensive to create these models--indeed, often more expensive than manually designing the circuit. Also, the simplifications required

in these closed-form analytical circuit models always limit their accuracy and completeness.

Symbolic analysis techniques, which have made significant strides of late [28]-[30] offer an automated path to obtaining some of these design equations. However, they remain limited to linear or weakly nonlinear performance specifications. Some "partial" steps toward full simulator-in-the-loop synthesis have also appeared. One option is to simplify the simulator itself, *e.g.,* [1],[13], which targeted fast linear small-signal performance evaluation. Or, one can severely truncate the search process to afford the cost of SPICE simulation for each candidate solution as in [31]. Neither extreme yields a satisfactory and general solution: we either *mis-evaluate* some candidate solutions, or *miss* them altogether by limiting the search process.

2.2 Early Synthesis Approaches for Systems

The literature on cell-level synthesis is rather rich; the literature on system-level synthesis is not. Macromodeling, *e.g.,* [32], plays a central role, since many system-level designs are intractable to simulate flat, at the device level. There is a wider variety of attacks on topology generation, *e.g.,* via templates, pattern matching, scripting, hierarchical performance prediction [33]-[35] since systems have more degrees of freedom than cells. Hierarchical composition in the style of [21] plays a central role, since systems negotiate specifications with subblocks to achieve overall goals. The added degrees of freedom inherent in these more complex designs have limited analog systems synthesized to date to modest size and performance, (*e.g.,* [35]) or to restricted classes of structured systems (*e.g.,* [34] for $\Delta\Sigma$ designs).

2.3 Analog Optimization ≠ Analog Synthesis

Finally, we need to note the distinction between *optimization* and *synthesis* for analog sizing/biasing. Several circuit *optimization* tools have relied on simulator-based methods (*e.g.,* [36]). For circuit optimization we assume some "close" initial circuit solution, and seek to improve it. This can be accomplished with gradient techniques requiring a modest number of circuit evaluations. For example, modern trust-region methods have been used with notable success here [8],[37]. However, in

our model of circuit synthesis, we assume that *no* starting solution is available at all. We know *nothing* about our starting circuit other than acceptable numerical ranges for the design variables, and how to simulate it to evaluate against specifications. This scenario is much more realistic of industrial design tasks, and more difficult as a numerical problem; it requires a more powerful and general solution strategy.

3. Simulation-Based Synthesis Formulation

The problem with early synthesis approaches was their use of circuit evaluation engines *different* from the simulators and simulation strategies that designers actually use to validate their circuits. These engines traded off accuracy and completeness of evaluation for speed. We argue that this is no longer an acceptable trade-off. Our simulation-based synthesis strategies all rely on a few key ideas. Given limitations of space, we review these rather briefly. Additional details appear in [2]-[5].

3.1 Cell-Level Simulation-Based Synthesis

Given un unsized analog circuit topology, our goal is to size/bias the topology to meet an arbitrary set of nonlinear performance goals, as evaluated by an arbitrary detailed circuit simulator. Our architecture for synthesis relies on the following four ideas to accomplish this:

- **Optimization-based formulation:** We use the numerical formulation of OBLX [1], in which the synthesis task is rendered as a single, scalar, weighted-sum cost function whose individual terms measure the divergence of the current solution candidate from the desired performance specification. Individual variables in this optimization represent circuit parameters, '., MOS device sizes, passive component values, bias voltages and currents. The components of the cost function, and their relative weights, are all constructed *automatically*; no "magic numbers" are required from the designer. Our goal is to search this complex, highly nonlinear cost-function for a minimum, which corresponds to a "best" circuit solution.
- **Simulator encapsulation:** Each circuit, which corresponds to a point on this cost-surface, is evaluated via complete, detailed simu-

lation. We use a technique we call *simulator encapsulation* [2] to hide the low-level idiosyncrasies of individual commercial simulators from our optimizer. Encapsulation is physically a layer of "insulating" software that handles the many data formatting issues necessary to communicate with a simulator, and also handles starting each simulator, and restarting it after crashes. Such crashes are common occurrences in synthesis, which frequently visits highly nonphysical circuit candidates. Encapsulation allows us to change simulator details without disturbance to our optimization engine.

- **Distributed global search:** If we seek only to "tweak" an existing design to improve it slightly, we can use deterministic optimization methods that rely on local search to find a superior nearby solution. Such optimization methods often assume that the cost surface is locally smooth, and reasonably well behaved. Unfortunately, in synthesis we can assume no viable starting solution, and the cost surfaces are not only discontinuous, but sometimes even *un-evaluatable*; some combinations of design parameters lead to circuits whose performance will crash a simulator. Hence we focus on derivative-free sample-based global optimization methods. However, to mitigate the CPU time needed to simulate each design candidate, we need search algorithms that scale easily to many processors. Fig. 2 shows the general architecture of our search algorithms. We borrow methods from evolutionary algorithms, and use a population of solutions, many samples of which are independently "evolved" via short bursts of local optimization. We use ideas from annealing, genetic, and direct optimization [2]-[5]. The population of solutions helps avoid local minima in search, and allows robust scaling behavior, even up to 20 or more CPU nodes.
- **Workstation-level parallelism:** Because simulation-based search is expensive in time, we exploit workstation-level parallelism [2]. This means our optimizer simply schedules necessary simulation tasks on any machine in a pool of designated CPUs. A critical attribute of our search algorithms is their ability to parallelize both individual circuit simulations and the numerical search process itself. In other words, when multiple simulations are required to evaluate a *single* circuit candidate, these can be distributed across

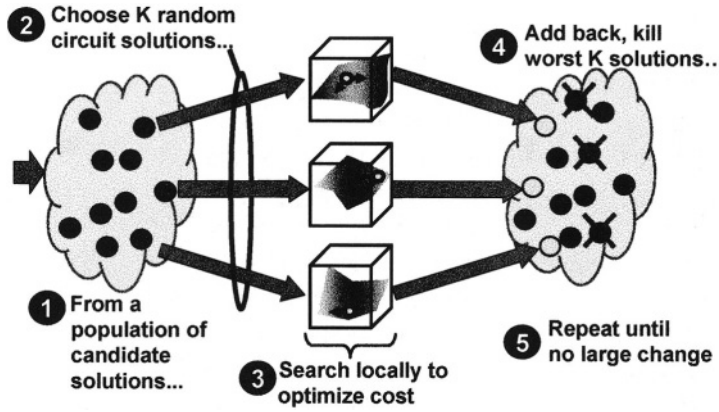

Fig. 2. General architecture for distributed global search

parallel machines. But in addition, multiple circuit candidates are also being simultaneously evolved. Control mechanisms in the search algorithms force convergence to a *set* of solutions of *similar* quality. Distribution of jobs to processors, restarting crashed simulators, balancing the load when some simulations are fast (*e.g.,* a dc evaluation) while others slow (*e.g.,* a full transient THD analysis), are all managed transparently and automatically.

3.2System-Level Simulation-Based Synthesis

In simulation-based synthesis, we simulate *each* design candidate during numerical search. The new problem we face is that system-level blocks are often vastly more expensive to simulate at device-level (if they can simulate *at all*) than cells. This can defeat our preferred simulator-in-the-loop formulation. We have explored three alternative strategies for coping with system-level complexity [4]:

- **Flat synthesis** chooses to ignore the hierarchical system-level structure, flattening it down to a single, potentially large circuit, and treating it just as cell-level synthesis. Where this method works, and the resulting run times are acceptable, it is the easy choice to make. Unfortunately this approach does not always work. Not only are the simulation times for large circuits problematic, but simulator convergence also becomes an issue. The reason is that numeri-

cal search often visits *exotically* parameterized designs--circuits with behaviors that deviate widely from the norms for which commercial simulators are designed.

- **Iterative-sequential synthesis** mimics top-down design practice. At the top level of our design, we replace subsystems with simplified behavioral macromodels, guess appropriate model parameters for these subsystems, and formulate synthesis as the task of choosing the remaining top-level component values to satisfy system-level goals. This is a straightforward simulation-based synthesis task since our "simulations" are just evaluations of subcircuit models. The real problem is the need to move down the design hierarchy one level at a time, and deal with the fact that even given good macromodels, predicting feasible trade-offs among the parameters for a subsystem can be difficult. We resolve this by iterating the design, using human insight to adjust specifications at each level.
- **Concurrent synthesis** is a novel alternative to the above two strategies. The system-level design and its component subsystems evolve *simultaneously.* Unlike a fully flattened design, the system still uses macromodels for its components, and synthesis sets their input parameters. In contrast, the component cells use complete device-level models and detailed circuit simulation. We link the two synthesis processes into a single numerical problem via a transformation of the cost function. We add terms that coerce agreement between the macromodel *parameters* evolving at the top of the design hierarchy, and the actual simulated *behaviors* of the device-level components at the bottom of the hierarchy. For example, at the top we may be evolving the *gain* and *bandwidth* of an amplifier, while at the bottom we are evolving MOS *widths/lengths* to achieve this *gain/bandwidth.* We refer to such specifications as being *dynamically set* since they evolve naturally as a negotiation between the system and its components. The virtue of the concurrent approach is that it reduces iteration steps, and automatically avoids designs in which macromodel parameters and device-level simulated behaviors disagree.

Each of these approaches is useful, though in different design scenarios. We explore two of these in the following section.

4. Simulation-Based Synthesis Results

We have implemented versions of these ideas in several prototype synthesis systems built at CMU, notably the tools named **MAELSTROM** [2], and **ANACONDA** [3]-[5]. We review here a few results obtained from experiments conducted at Texas Instruments (TI) in Dallas, Texas, using two versions of **ANACONDA.**

4.1 Cell-level Synthesis: TI Opamps

We benchmarked an early version of **ANACONDA** on the three opamps shown in Fig. 3, which are all examples of production TI circuits. The power amp was designed in a 1μm CMOS process and the other two opamps were designed in a 0.6μm CMOS process. Table 1 gives the input performance constraints, simulation environment, and run-time statistics for each synthesized circuit. For our experiments, the performance constraints were set by running nominal simulations on original expert hand designs supplied by TI. Each design was synthesized using a pool of 24 Sun Sparc workstations. Because runtime is highly dependent on the size of the circuit and the type of simulation being performed, there was significant variation among the circuits. For example, for the power amp, each THD analysis took a few CPU seconds, and consequently, that circuit ran longer.

Fig. 4 shows the results of 5 repeated synthesis runs for each circuit. This sample size of 5 gives a good picture of the statistical spread in solutions for our stochastic algorithms (i.e., these are 5 runs *in sequence,* not the best 5 out of a large number of runs). Given the large number of performance specifications for each circuit, we summarize these results as a power-versus-area scatter plot for each circuit. In all but one case, the synthesized circuits met *all* of the performance constraints specified in Table 1. The one exception provides excellent justification for why we value simulation-based evaluation: this folded cascode misses its UGF spec (159MHz < 162MHz) by less than 2%, but this is enough to give an almost 15% improvement in area. For designs pushed to tight performance limits, simulation gives us results that are much easier to *trust* than any approximate analytical equation.

For each design, the objectives were to minimize area and static power dissipation. In addition, each of the synthesized designs was ver-

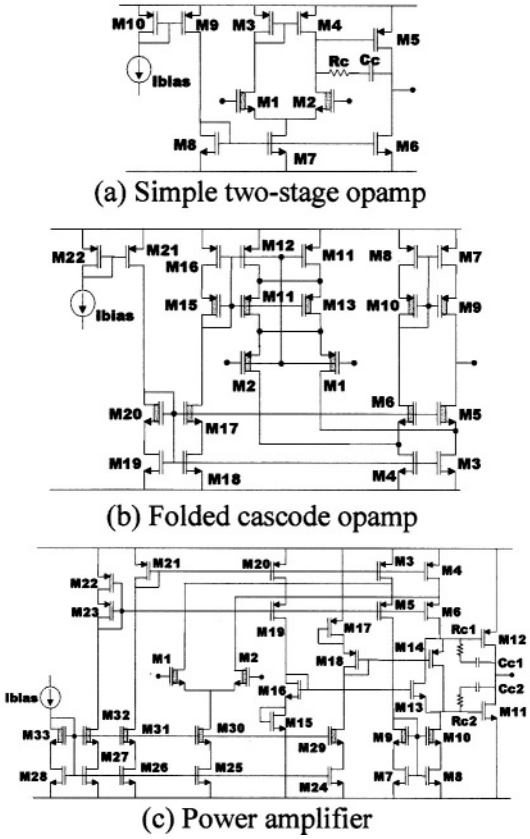

Fig. 3. Industrial test circuits for our synthesis experiments.

ified for robustness by performing Monte Carlo simulations with 3σ process, 10% voltage supply and 0 to 100C° temperature variations. The resulting histograms for each performance characteristic showed that each synthesized design was as robust as the original hand design; Fig. 5 shows one of those histograms.

These results are noteworthy in several respects. First, these were production-quality industrial analog cells with difficult performance specifications. Second, our synthesis approach used as its evaluation engine the *identical* simulation environment used by TI's designers to

Table 1. Detailed specifications, simulation environment, and synthesis runtime results for TI test circuits of Fig. 10.

Environment Spec		*Two-Stage*	*Folded-Cascode*	*Power Amp*
T=25°C				
Vdd	V	2.7	2.7	5
C_L	pF	1.5	1.75	100
R_L	Ω			25
Performance Constraint				
Gain	dB	≥ 68	≥ 71	≥ 92
UGF	MHz	≥ 260	≥ 162	≥ 0.6
PM	deg.	≥ 56	≥ 52	≥ 84
GM	dB	≥ 17		
Noise@1kHz[1]	nV $Hz^{-0.5}$	≤ 145	≤ 70	≤ 52
Noise@10kHz	nV $Hz^{-0.5}$			≤ 40
Noise@10MHz	nV $Hz^{-0.5}$	≤ 3	≤ 4	
CMRR	dB	≥ 76	≥ 108	> 138
PSRR (Vss)	dB	> 85	> 89	> 90
PSRR (Vdd)	dB	> 70	> 72	> 94
Settling Time	ns	≤ 4.6[4]	≤ 16[5]	
THD[2]	%			< 0.06
THD[3]	%			< 0.1
Runtime Info				
Independent Variable Count		15	13	20
Ave. Perturbations per run[6]		6966	5740	10248
Ave. Runtime[7] (hrs)		2.8	1.8	10

1 – All noise specs are input referred
2 – 4.0V p-p 1kHz input
3 – 2.6V p-p 1kHz input, R_L=5Ω
4 - 0.5V input step, 10-Bit accurate final voltage
5 - 0.1V input step, 10-Bit accurate final voltage
6 – Total number of circuits evaluated is approx. 10x perturbation count
7 – Using a pool of 16 300MHz SUN Ultra10 and 4 dual 300MHz Ultra2 workstations running Solaris 2.5.1.

validate their manual designs. As a result, we could deal accurately with difficult design specifications such as noise, settling time, and THD, which require detailed simulations to evaluate complex nonlinear effects. Finally, nearly all these synthesized circuits compared favorably with their manually designed counterparts, both in performance and in robustness across corners.

4.2System-Level Synthesis Results: TI EQF Filter

Digital Subscriber Line (DSL) technologies combine sophisticated analog and digital signal processing to deliver high-speed digital data and conventional analog voice data over existing copper telephone wires. To understand how synthesis could be applied to systems, we

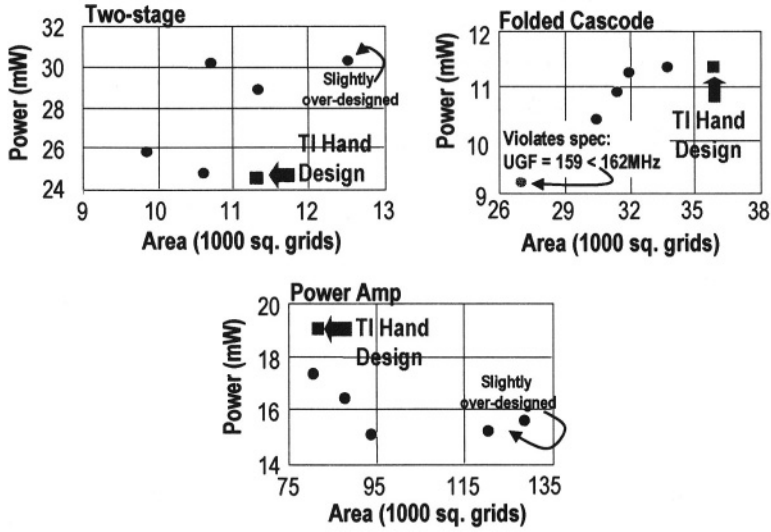

Fig. 4. TI opamp synthesis results for circuits of Fig. 3. Circles show 5 runs from synthesis; squares show TI hand designs.

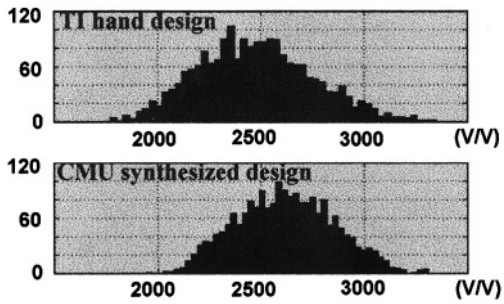

Fig. 5. Monte Carlo histograms for two-stage TI opamp low-frequency gain, across process, voltage, and temperature variations

were asked by TI to target the frontend of the remote modem CODEC receiver (Fig. 6.) from the ADSL design they introduced at the 1999 International Solid State Circuits Symposium [38]. We used a later version of ANACONDA [4] on the *equalizer filter* (EQF) subsystem at the front-end of the analog signal path. Signals in copper phone lines attenuate severely with increasing frequency and line length, and cable

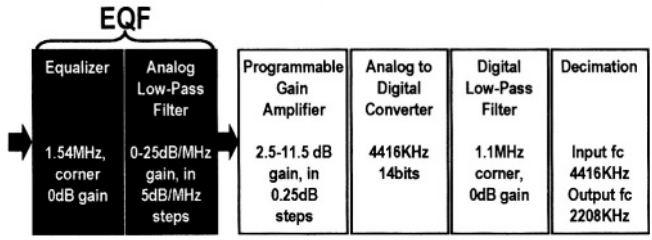

Fig. 6. Architecture for remote modem CODEC receiver. The section targeted (shaded) is the analog equalizer & low-pass filter: EQF.

bundles introduce considerable crosstalk. The equalizer amplifies the attenuated line signal, the filter extracts the digital data from the spectrum without corrupting the nearby analog voice traffic, and the combined EQF must do this under stringent noise and area constraints set by the CODEC.

The EQF itself is shown in Fig. 7 and comprises five identical low noise operational amplifiers (Fig. 8) connected via R's, C's and CMOS switches. The equalizer has six separate modes (Fig 9) to compensate for high frequency line attenuation across the frequency range of interest, 25kHz to 1104kHz. Fig. 9 also shows some of the EQF spectral mask.

We fixed the topology of the EQF and formulated synthesis as the task of designing parameter values to meet performance specifications. This respects the fact that "libraried" analog blocks are most likely to be stored as topologies that can be re-parameterized to handle new specifications, or fabrication processes. Moreover, expert designers routinely choose good topologies to optimize gross system function, and then spend enormous effort iteratively resizing them; the problem is to determine if a proposed sizing can realize the specified performance in the face of many interacting second-order circuit effects. Hence, it is this sizing/biasing we seek to automate.

As a system, EQF has one layer of hierarchy, and five instances of a single component, the low-noise amplifier of Fig. 8. It was designed in a proprietary TI 0.6um CMOS process. At the top level, the EQF has 46 R's, 32 C's and 36 CMOS switches. Pole and zero location constraints from the transfer function set a large number of the R's and C's. The amplifier is itself a complex cell, and has 20 independent variables.

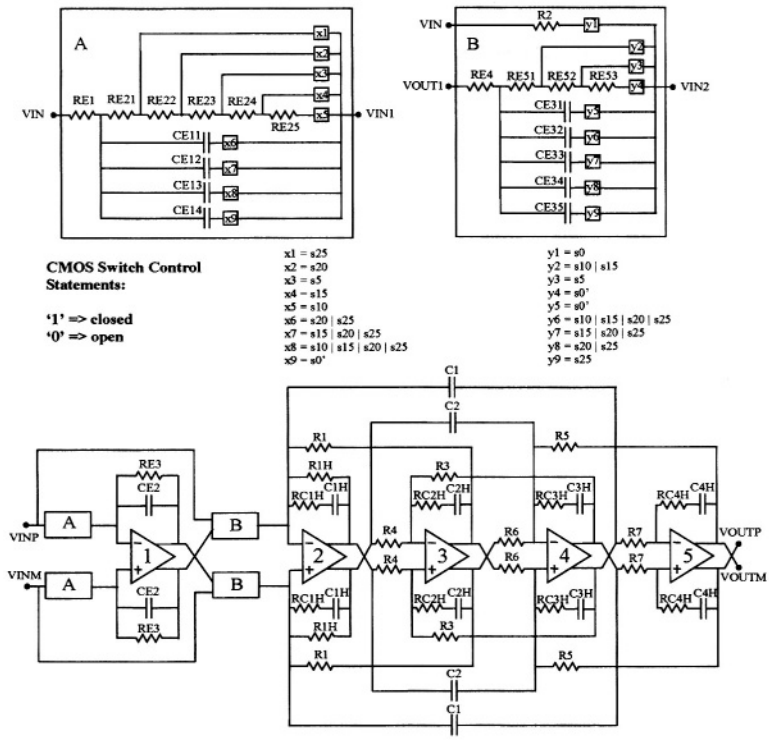

Fig. 7. Schematic for the Equalizer/Filter (EQF)

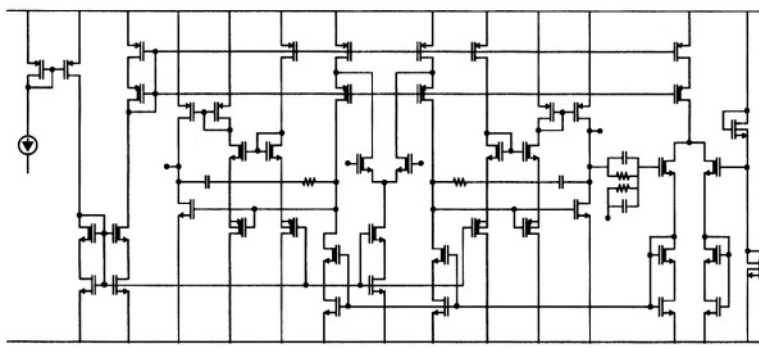

Fig. 8. Schematic for low-noise opamp in the above EQF circuit

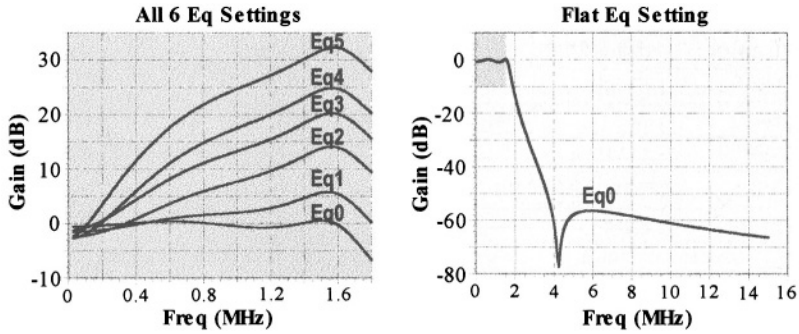

Fig. 9. Frequency response of the EQF. Left: the passband response for each of the six equalizer settings. Right: shows the frequency response with the equalizer set to 0dB gain.

In [4] we synthesized components of the EQF, and the complete EQF, in several different styles. Here, we review just two of these results:

- **Flat component synthesis:** since we have both the system and component specifications from an expert hand design, we re-synthesized individual pieces of the design to see if we could match the manual effort. Fig. 10 shows one such experiment, in which we redesigned the low-noise amplifier five times. All designs but one met all specifications, and that one "miss" had a few devices with V_{eff} low by a few mV. Again, the virtue of simulation-based synthesis is that we see these effects more precisely; note that compromising V_{eff} gives us a significant, but *unmanufacturable,* area savings. Each synthesis run took roughly 10 hours
- **Concurrent system synthesis:** we also resynthesized the entire EQF using the concurrent approach outlined in the previous section. The top-level EQF circuit was modeled using a proprietary TI opamp macromodel, to allow fast, flat simulation. The bottom-level low-noise amplifiers were modeled flat at device level. Concurrent synthesis forced convergence of the performance requirements evolved for the opamps at the top, with the actual evolved opamp behaviors at the bottom. Fig. 11 shows the resulting passband responses with the equalizer gain set to 0dB. The two important observations to make are that the ripple in the passband and the cut-

Nominal Specs @25°C	
Gain	135dB
PM	54°
UGF	110MHz
Noise	6.3nV/Hz$^{1/2}$
THD	0.160%

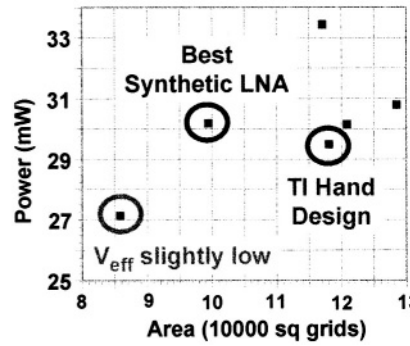

Fig. 10. Synthesis results for low-noise amp in EQF. Five CMU synthesis results and the TI hand design are shown.

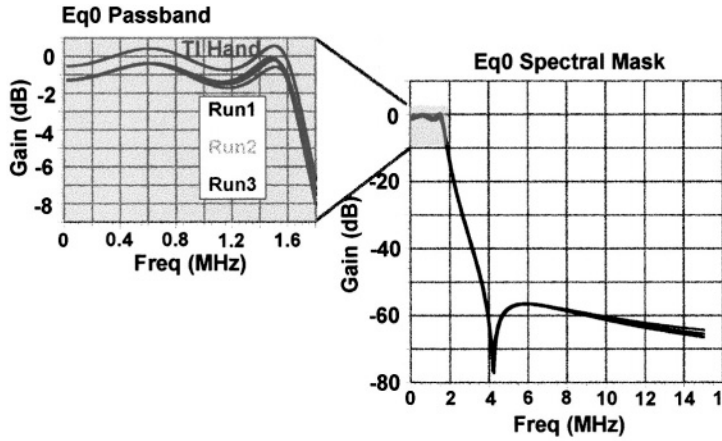

Fig. 11. Results from three CMU synthesis runs for full EQF.

off frequency are nearly identical for the hand design and the three synthesized designs. When setting up this experiment, we decided to allow the gain to vary slightly to provide an *extra* degree of freedom to the optimization engine. The difference in gain between the hand design and the synthesized design can be compensated for using the programmable gain amplifier (see Fig. 6), which is the next stage in the CODEC receiver. Fig. 11 shows the EQF's response over the entire frequency range of interest. We observe

from the figure that the cutoff frequency, stopband attenuation, and overall shape of the response are nearly identical for the hand design and the three synthesized EQF designs. Comparing the other performance goals for the synthetic and expert design: Run1 and Run3 had nearly identical noise, but with 7% and 17% *better* area, respectively; design Run2 has slightly better noise, but at a cost of 14% more area. EQF took a few months to design by hand; our tools were able to complete each EQF run in 12 hours.

To the best of our knowledge, the EQF design is the largest, most complex, most thorough controlled experiment ever undertaken to demonstrate how simulation-based analog synthesis can be applied to a state-of-the-art industrial analog system. We successfully redesigned the EQF block in the TI ADSL frontend in several different ways, and examined the trade-offs involved. Simulation-based synthesis, with a mix of macromodels, transistor-level detailed simulation, and vigorous global numerical search, can yield practical results on large, complex designs.

5. Conclusions

To trust a synthesis result, one must first trust the methods used to qualify each solution candidate *during* synthesis. Analog designers necessarily evolve an intimate relationship with the simulators, models, etc., that they use on a daily basis to evaluate their circuits. To make analog synthesis practical, we have argued that synthesis must use *identical* verification methods internally. We have developed a family of simulation-based synthesis tools at CMU that have been successfully used across many designs. Other similar tools are also being developed elsewhere. Open problems here are how to scale these techniques "up" to ever bigger systems, and how to handle the layers of macromodeling needed to do this.

Acknowledgment: This paper reviews joint work done in collaboration with several colleagues whose contributions were essential to this research: L. Richard Carley, Michael Krasnicki and Rodney Phelps of CMU, and James R. Hellums of TI. The work was funded in part by the Semiconductor Research Corp., by the National Science Foundation under contract 9901164, and by Rockwell and Texas Instruments.

References

[1] E. Ochotta, R.A. Rutenbar, L.R. Carley, "Synthesis of High-Performance Analog Circuits and ASTRX/OBLX," *IEEE Trans. CAD,* vol. 15, no. 3, Mar. 1996.

[2] M. Krasnicki, R. Phelps, R. A. Rutenbar, L.R. Carley, "MAELSTROM: Efficient Simulation-Based Synthesis for Custom Analog Cells," Proc. ACM/ IEEE DAC, June 1999.

[3] R. Phelps, M. Krasnicki, R.A. Rutenbar, L.R. Carley, J.R. Hellums, "ANACONDA: Robust Synthesis of Analog Circuits Via Stochastic Pattern Search," *Proc. IEEE Custom Integrated Circuits Conference,* May 1999.

[4] R Phelps, M. Krasnicki, R.A. Rutenbar, L.R. Carley, J. Hellums, "A case study of synthesis for industrial-scale analog IP: Redesign of the equalizer/filter frontend for an ADSL CODEC," *Proc. ACM/IEEE DAC,* June 2000.

[5] R. Phelps, M. Krasnicki, R. A. Rutenbar, L. R. Carley, J. R. Hellums, "Anaconda: Simulation-based synthesis of analog circuits via stochastic pattern search," *IEEE Trans. CAD,* vol. 19, no. 6, June 2000.

[6] P. J. Vancorenland, G. Van der Plas, M. Steyaert, G. Gielen, and W. Sansen, "A Layout-aware Synthesis Methodology for RF Circuits", *Proc. ACM/IEEE ICCAD,* Nov 2001.

[7] P. Vancorenland, C. De Ranter, M. Steyaert, G. Gielen, "Optimal RF Design Using Smart Evolutionary Algorithms," *Proc. ACM/IEEE DAC,* June 2000.

[8] R. Schwenker, J. Eckmueller, H. Graeb, K. Antriech, "Automating the Sizing of Analog CMOS Circuits by Consideration of Structural Constraints," *Proc DATE99,* March 1999. vol. 19, no. 6, June 2000.

[9] M.J., Krasnicki, R. Phelps, J.R. Hellums, R.A. Rutenbar, L.R. Carley, "ASF: A Practical Simulation-Based Methodology for the Synthesis of Custom Analog Circuits," Proc. ACM.IEEE ICCAD, Nov. 2001.

[10] S Dugalleix, F Lemery, A Shah, "Technology migration of a high-performance CMOS amplifier using an automated front-to-back analog design flow," Proc Design Auto. & Test in Europe (DATE), March 2002.

[11] E. Hennig, R. Sommer, L. Charlack, "An automated approach for sizing complex analog circuits in a simulation-based flow," Proc Design Auto. & Test in Europe (DATE), March 2002.

[12] T. McConaghy, "Intelligent Systems Solutions for Analog Synthesis," *Integrated Communications Design,* Penwell, January, 2002

[13] E. Ochotta, T. Mukherjee, R.A. Rutenbar, L.R. Carley, *Practical Synthesis of High-Performance Analog Circuits,* Kluwer Academic Publishers, 1998.

[14] G.G.E. Gielen and R.A. Rutenbar, "Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits, Proc IEEE, vol. 88, no. 12, Dec. 2000.

[15] M. Degrauwe *et al., "*Towards an analog system design environment," *IEEE JSSC,* vol. sc-24, no. 3, June 1989.

[16] H.Y. Koh, C.H. Sequin, and P.R. Gray, "OPASYN: a compiler for MOS operational amplifiers," *IEEE Trans. CAD,* vol. 9, no. 2, Feb. 1990.

[17] G. Gielen, *et al,* "Analog circuit design optimization based on symbolic simulation and simulated annealing," *IEEE JSSC,* vol. 25, June 1990.

[18] F. Leyn, W. Daems, G. Gielen, W. Sansen, "A Behavioral Signal Path Modeling Methodology for Qualitative Insight in and Efficient Sizing of CMOS Opamps," *Proc. ACM/IEEE ICCAD,* 1997.

[19] P. C. Maulik, L. R. Carley, and R. A. Rutenbar, "Integer Programming Based Topology Selection of Cell Level Analog Circuits," *IEEE Trans. CAD,* vol. 14, no. 4, April 1995.
[20] W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS Opamp Synthesis by Means of a Genetic Algorithm," *Proc. 32nd ACM/IEEE DAC,* 1995.
[21] R. Harjani, R.A. Rutenbar and L.R. Carley, "OASYS: a framework for analog circuit synthesis," *IEEE Trans. CAD,* vol. 8, no. 12, Dec. 1989.
[22] B.J. Sheu, *et al.,* "A Knowledge-Based Approach to Analog IC Design," *IEEE Trans. Circuits and Systems,* CAS-35(2):256-258, 1988.
[23] E. Berkcan, *et al.,* "Analog Compilation Based on Successive Decompositions," *Proc. of the 25th IEEE DAC,* pp. 369-375, 1988.
[24] J. P. Harvey, *et al.,* "STAIC: An Interactive Framework for Synthesizing CMOS and BiCMOS Analog Circuits," *IEEE Trans. CAD,* Nov. 1992.
[25] C. Makris and C. Toumazou, "Analog IC Design Automation Part II--Automated Circuit Correction by Qualitative Reasoning," *IEEE Trans. CAD,* vol. 14, no. 2, Feb. 1995.
[26] A. Torralba, J. Chavez and L. Franquelo, "FASY: A Fuzzy-Logic Based Tool for Analog Synthesis," *IEEE Trans. CAD,* vol. 15, no. 7, July 1.996.
[27] M. Hershenson, S. Boyd, T. Lee, "GPCAD: a Tool for CMOS Op-Amp Synthesis", Proc. ACM/IEEE ICCAD, pp. 296-303,1998
[28] G. Gielen, P. Wambacq, and W. Sansen, "Symbolic ANalysis Methods and Applications for Analog Circuits: A Tutorial Overview, "Proc. IEEE, vol. 82, no. 2, Feb., 1990.
[29] C.J. Shi, X. Tan, "Symbolic Analysis of Large Analog Circuits with Determinant Decision Diagrams," *Proc. ACM/IEEE ICCAD,* 1997.
[30] Q. Yu and C. Sechen, "A Unified Approach to the Approximate Symbolic Analysis of Large Analog Integrated Circuits," *IEEE Trans. Circuits and Sys.,* vol. 43, no. 8, August 1996.
[31] F. Medeiro, F.V. Fernandez, R. Dominguez-Castro and A. Rodriguez-Vasquez, "A Statistical Optimization Based Approach for Automated Sizing of Analog Cells," *Proc. ACM/IEEE ICCAD,* 1994.
[32] Y-C Ju, V.B. Rao and R. Saleh, "Consistency Checking and Optimization of Macromodels", *IEEE Transactions on CAD,* August 1991.
[33] B. Antao and A. Brodersen, "ARCHGEN: Automated Synthesis of Analog Systems", *IEEE Transaction on VLSI Systems,* June 1995.
[34] F. Medeiro, B. Pérez-Verdú, A. Rodríguez-Vázquez, J. Huertas, "A vertically-integrated tool for automated design of SD modulators," *IEEE Journal of Solid-State Circuits,* Vol. 30, No. 7, pp. 762-772, July 1995.
[35] A. Doboli, et al, "Behavioral synthesis of analog systems using two-layered design space exploration," *Proc. ACM/IEEE DAC,* June 1999.
[36] W. Nye, *et al,* "DELIGHT.SPICE: an optimization-based system for the design of integrated circuits," *IEEE Trans. CAD,* vol. 7, April 1988.
[37] A.R. Conn, R.A. Haud, C. Viswesvariah, C.W. Wu, "Circuit optimization via adjoint lagrangians," Proc. ACM/IEEE ICCAD, Nov. 1997.
[38] R. Hester, *et al.,* "CODEC for Echo-Canceling, Full-Rate ADSL Modems," *IEEE Int'l Solid-state Circuits Conference,* pages 242-243. 1999.