A Case Study of Synthesis for Industrial-Scale Analog IP: Redesign of the Equalizer/Filter Frontend for an ADSL CODEC

Rodney Phelps, Michael J. Krasnicki, Rob A. Rutenbar, L. Richard Carley, James R. Hellums*

Department of Electrical and Computer Engineering, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213 USA

*Mixed Signal Products, Texas Instruments Incorporated
Dallas, Texas 75243 USA

Abstract: *A persistent criticism of analog synthesis techniques is that they cannot cope with the complexity of realistic industrial designs, especially system-level designs. We show how recent advances in simulation-based synthesis can be augmented, via appropriate macromodeling, to attack complex analog blocks. To support this claim, we resynthesize from scratch, in several different styles, a complex equalizer/filter block from the frontend of a commercial ADSL CODEC, and verify by full simulation that it matches its original design specifications. As a result, we argue that synthesis has significant potential in both custom and analog IP reuse scenarios.*

I. INTRODUCTION

Modern fabrication technologies support the integration of many formerly discrete functions onto a single die. To manage complexity and time-to-market, these *system-on-chip* (SoC) designs require a high degree of reuse, a situation that has generated growing interest in techniques for creating silicon *intellectual property* (IP), and an evolving marketplace for IP. However, the vast majority of efforts in IP creation and packaging address only digital logic. This is an alarming situation because a large fraction of new ICs require an interface to the external, continuous-valued analog world.

IP and reuse strategies for the digital portion of these mixed-signal designs can successfully exploit cell-based tools for synthesis, mapping, and physical design. IP offered as a *soft* netlist can be retargeted via logic synthesis; IP offered as a *hard* layout can by integrated via physical synthesis. Unfortunately, with respect to current logic-centric design flows, analog blocks still fit poorly. Although a small fraction of the overall design size (*e.g.*, 10,000 to 20,000 analog transistors is typical), the analog portion of these systems remains a real design challenge. Worse, these analog blocks are still designed by hand, usually one transistor and one rectangle at a time.

The widening gap between reuse-centric digital design and manual analog design has been addressed in two very different ways. One option is *hard analog IP* in the form of layouts for common system-level analog blocks, *e.g.*, data conversion, network physical layers, phase lock loops, *etc.* Limited (but growing) selections of such analog IP blocks are now available from larger foundries, and from a new generation of third-party IP providers who target the most widely used foundries. (See [1] for a recent survey.) This is an appealing model for analog reuse--but a limited one. The problem is that the analog space cannot be fully covered by any finite library of blocks. There are hundreds of circuit-level topologies for cells (10-50 devices) and systems (10-100 cells) in use today, and each performance parameter is *continuous*. To take a crude example: a cell with 10 continuous parameters has roughly 1000 library variants even if we limit each parameter to only "low" and "high," and ignore completely that device-level circuit/layout decisions must change as we move from process to process. Routine functions can, of course, be created and rendered as part of a useful analog IP portfolio. But the analog side of mixed-signal SoCs often comprises more

than just "routine" blocks connecting to "routine" interfaces. Analog functions are often *aggressively* custom, to deal with next-generation interfaces, application-specific sensors or transducers, or nontraditional trade-offs between analog and digital signal processing. Indeed, on many such designs, this custom analog is a large component of the value added by a single-chip implementation.

The alternative approach is *analog synthesis*, which seeks to transform some abstract description into a working circuit. There is an extensive literature on cell-level analog synthesis, e.g., [2]-[13], but considerably fewer attempts at analog system synthesis, e.g., [14]-[17]. None of these techniques is in widespread use today. We believe that these fail in practice because of the difficulties in reconciling the simplified models required for synthesis with the industrial-strength simulation environments required for validation. Many techniques trade-off accuracy in the heuristic evaluation of circuits in favor of speed, to support a more vigorous numerical search. To date, strategies that rely on circuit evaluators unrelated to (often inconsistent with) current simulation-based practice have not been regarded as trustworthy by working designers, who rely on simulation to correctly assess critical second-order circuit effects. There is enormous investment in modeling, characterization, test harnesses, and simulation-based sign-off for validation in current analog design flows; synthesis strategies that cannot leverage this investment face significant obstacles.

In [18],[19] we introduced new techniques for efficient *simulation-based synthesis* for custom analog cells. By combining novel global numerical search algorithms, workstation-level parallelism, and software encapsulation methods that insulate search from the idiosyncrasies of circuit simulators, we were able to synthesize a range of custom cells competitive with industrial designs, using full "SPICE-in-the-loop" search. Our goal in this paper is to demonstrate that some *system-level* analog designs are also within the reach of these techniques. The central problem is how to retarget simulation-based synthesis to systems where the cost of a full circuit-level simulation of a design candidate is so large as to render optimization intractable. The key idea is a hierarchical decomposition in which cell-level macromodels are used to search for an optimal system-level design, while *concurrently* a full transistor-level design evolves for each cell, matching it to the abstracted cell behaviors evolving at system level.

To counter the persistent criticism that analog design--especially design beyond basic cells--is intrinsically "too hard" to make synthesis practical, we present a case study of synthesis for one significant block, an equalizer/filter (EQF), in the frontend of a state-of-the-art CODEC for a full-rate ADSL modem introduced by Texas Instruments in [20] in 1999. This design is "aggressively custom" in the sense argued above: it uses neither routine system nor cell-level circuit topologies, and meets difficult analog performance goals. Our study takes the form of a legacy IP reuse scenario: we have full access to a working design and the usual documentation archived with such designs, but *no access to the designer* (who no longer works in this group). We develop hierarchical simulation-based synthesis techniques that are successfully able to *redesign* this entire block's sizing/biasing to meet its original commercial specifications.

The remainder of the paper is organized as follows. Sec. II briefly surveys prior synthesis work. Sec. III introduces the EQF benchmark circuit. Sec. IV develops our hierarchical synthesis formulation. Sec. V presents experimental results. Sec. VI offers concluding remarks.

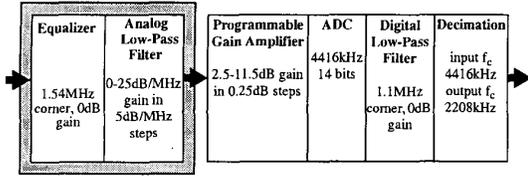| Equalizer	Analog Low-Pass Filter	Programmable Gain Amplifier	ADC	Digital Low-Pass Filter	Decimation
1.54MHz corner, 0dB gain	0-25dB/MHz gain in 5dB/MHz steps	2.5-11.5dB gain in 0.25dB steps	4416kHz 14 bits	1.1MHz corner, 0dB gain	input f_c 4416kHz output f_c 2208kHz

Fig. 1 Architecture for Remote Modem CODEC Receiver. The section targeted (shaded) is the analog equalizer & low-pass filter.

II. PRIOR WORK

There is an extensive literature on cell-level synthesis, covering both generation of circuit topologies [2],[3] and sizing and biasing. The earliest approaches used procedural scripting [4], which proved fragile in the face of circuit and especially process changes. Hierarchical attacks [5-7] allowed composing of reusable subcircuits, but remained limited by their script-based underpinnings. Equation-based approaches substituted numerical search for simple scripting, and were able to attack a wider set of designs [8-10]; however, they still suffered from accuracy problems and limitations imposed by the need for closed-form models. Symbolic analysis techniques [11] can automatically extract some, but not yet all of these required equations. Attacks based on custom, lightweight simulators coupled with numerical search proved yet more capable [12],[13], but still lacked the ability to evaluate some circuit performance specifications.

The literature on system-level synthesis for analog is considerably thinner. Macromodeling, e.g., [14], 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 [15-17], since systems have more degrees of freedom than cells. Hierarchical composition in the style of [5] plays a central role, since systems negotiate specifications with sub-blocks to achieve overall goals. The added degrees of freedom inherent in these more complex designs have limited analog systems synthesized to date to very modest size and performance.

None of these techniques is widely used. In [18],[19] we argued that the necessary remedy is efficient *simulation-based analog synthesis*. These techniques are the starting point for our work.

III. OVERVIEW OF EQF CIRCUIT BENCHMARK

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. *Asymmetric* variants like ADSL offer full-duplex communication, but bias bandwidth usage toward data downloads (*e.g.*, video on demand provided at several Mbps), rather then uploads (*e.g.*, email, web clicks, etc., supported at a few hundred Kbps). ADSL connections require a pair of modems, one at each end of the copper line. We focus on the *remote modem* at the user's end. The *CODEC* is the interface between the modem's DSP core and the line itself; its architecture (from[20]) appears in Fig. 1. We focus on a complex subsystem at the front-end of the analog signal path, the *equalizer filter* (EQF). Copper transmission presents significant design challenges: signals attenuate strongly with increasing frequency and line length, and typical cable bundles introduce considerable crosstalk. The equalizer amplifies the attenuated line signal which is subsequently extracted by the filter. The combined EQF must do this under stringent noise and area constraints set by the overall CODEC.

The EQF itself is shown in Fig. 2 and comprises five identical low noise operational amplifiers (LNAs) connected via R's, C's and CMOS switches. The equalizer consists of opamp1 and shares opamp2 with a fourth-order elliptical low pass continuous-time filter. Typically, the equalizer would require two opamps, but a novel circuit architecture merges equalizer and filter, eliminating one opamp.
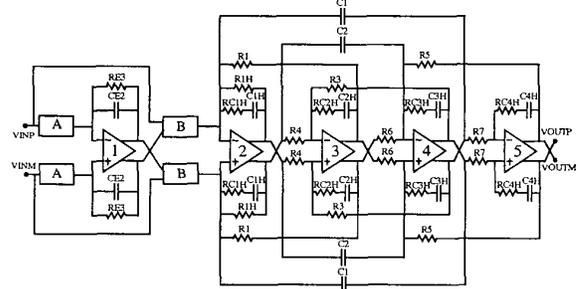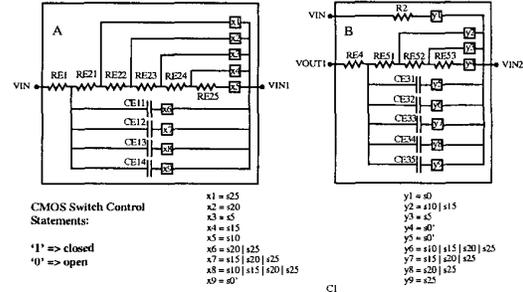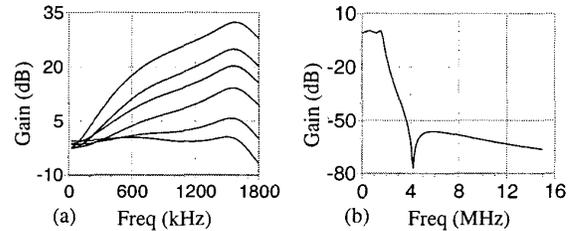

Fig. 2 Schematic for the Equalizer/Filter

Fig. 3 Frequency response of the EQF. (a) shows the passband response for each of the six equalizer settings. (b) shows the frequency response with the equalizer set to 0dB gain.

The equalizer has six separate modes (Fig. 3a) to compensate for high frequency line attenuation. As shown in Fig. 2 the control signals on the switches program the gain to one of {0, 5, 10, 15, 20, 25}dB across the frequency range of interest, 25kHz to 1104kHz. The low pass filter itself is a standard design, with the exception of resistors RC1H-RC4H in the feedback path. Ideally, these would not be required, but because the opamp has finite bandwidth and the passband is relatively wide, these resistors compensate for peaking near the cutoff frequency. In other words, during initial design, it was decided that these small resistors were a better choice than to increase the performance of the amplifier. Fig. 3 shows the spectral mask for the EQF.

IV. SYNTHESIS FORMULATION

A. Synthesis Styles for System-Level Designs

Following [13],[18],[19], we fix the topology of the EQF and formulate synthesis as the task of designing parameter values to meet performance specifications. This does simplify the problem, but also 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 and biasing we seek to automate.

We formulate synthesis as cost-based numerical search: a minimum of an appropriately constructed cost function corresponds to a "best" circuit. We use the functional form developed in [13],[18],[22]. Creating an appropriate cost function is mostly mechanical, and we can automate much of the process; solving for a useful minimum is not.

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 suggest three alternative strategies for coping with system-level complexity:

- **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. 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 numerical 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 analytical models or very simple circuits. 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 must avoid the situation where the system design "works"--but only if its components comply with unachievable performance goals.

- **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. 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.

The flat and iterated-sequential synthesis styles can be accommodated with no problems in our existing simulation-based synthesis framework. They differ only in the nature of the simulation, and the number of separate synthesis tasks to be undertaken. The concurrent style, however, requires a modification of our numerical formulation.

We map the circuit design problem to a constrained optimization problem, where x is the set of independent design variables; $\{f(x)\}$ is a set of objective functions that codify performance specifications to optimize, *e.g.*, noise; and $\{g(x)\}$ is a set of constraint functions that codify hard specifications to meet, *e.g.*, *gain* > 60dB. We perform the standard conversion to an unconstrained optimization problem, substituting suitable normalizing and penalty functions for each term [13]. Scalar weights balance competing objectives. As a result, the goal becomes minimization of a scalar cost function, $C(x)$, defined by (1).

$$(\underline{x}) = \sum_{i=1}^{k} w_{1,i}\hat{f_i}(\underline{x}) + \sum_{j=1}^{l} w_{2,j}\hat{g_j}(\underline{x}) \tag{1}$$

Suppose now that we have n subsystems in our system, and our independent variables x include both model parameters for the macromodel of each subsystem, and actual device-level parameters for the detailed design of each subsystem. We force the model parameters and simulated performance of each subsystem to *converge* by adding a set of penalty functions, $\{p(x)\}$, to the cost, as shown in (2). (3) shows the penalty function for a single parameter. The *measured* value is the value obtained via SPICE simulation of the actual subsystem; *modelparam* is the current macromodel input, set as an independent variable.

$$C(\underline{x}) = \sum_{i=1}^{k} w_{1,i}\hat{f_i}(\underline{x}) + \sum_{j=1}^{l} w_{2,j}\hat{g_j}(\underline{x}) + \sum_{m=1}^{n} w_{3,m}\hat{p_m}(\underline{x}) \tag{2}$$

$$\hat{p_m}(\underline{x}) = |modelparam - measured| \tag{3}$$

This formulation allows us to mix higher-level macromodels and lower-level detailed models, yet treat the overall synthesis task as a single, simulation-based numerical search. The added penalty functions coerce the consistency we need between different models.

B. Global Numerical Search Algorithm

This cost function creates a difficult, highly nonlinear, discontinuous optimization problem. We attack this by combining the population-based search ideas from ANACONDA [19] with some of the annealing ideas from MAELSTROM [18]. The architecture is shown in Fig. 4.

1. **Population of partial circuit solutions:** we maintain a large population of partial solutions. The population itself helps combat the problem of cost surfaces with local minima. Each element is one sample of that cost surface. We maintain a suitably diverse set of samples, and preferentially update the population so that lower-cost samples survive and higher-cost samples are culled.

2. **Limited population update:** from a population of P partial solutions, we select k candidates, apply a short annealing improvement process to each candidate, then replace these in the population. Note that as a result, individual elements may improve or degrade on cost after annealing. From this updated population of $P+k$ solutions, we remove the k solutions of highest (worst) cost.

3. **Evolution by problem-shared annealing:** the update process for a selected candidate is a *shared* annealing. We maintain k parallel annealers (k=3 in Fig. 4), and each selected candidate undergoes a small number of annealing perturbations before being returned to the overall population. Each annealer sees locally a single numerical optimization problem; in reality, it sees a sequence of snapshots of independent optimizations, each sampling different regions of the same underlying cost surface. The annealers each run a global, fixed-length cooling schedule [13]. The open question here is what is the cost C that is the current *state* of annealing?
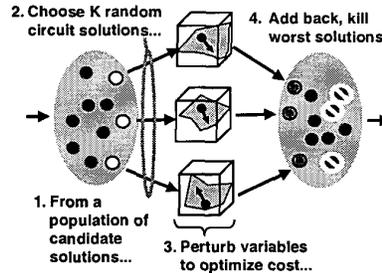

Fig. 4 Overview of population-based global numerical optimization.

Suppose after move Δ_i we have annealing cost C_i, and we then swap in a new circuit candidate to perturb, which has its own current annealing cost D_i. We maintain a form of thermal equilibrium by requiring each annealer to *choose* which of these is the cost of the annealing state *before* the next perturbation Δ_{i+1}. To do this, the annealer makes a Metropolis-weighted probabilistic choice between C_i and D_i; either can be the next new state. This is essentially the notion of parallel quasi-equilibrium from [21]. See [22],[23] for additional details about this annealing process.

The population model supports significant practical parallelism: we routinely run on 20-30 workstations. In concert, the population model plus the shared-annealing update are effective as a global optimizer for simulation-based synthesis. These ideas have been implemented in an updated version of ANACONDA.

V. EXPERIMENTAL SYNTHESIS RESULTS

We describe experiments in the three synthesis styles discussed in Sec. IV. Experiments were done at TI using TISpice as the simulation engine, on a compute farm of 20 to 30 Sun UltraSparcs.

As a system, EQF has one layer of hierarchy, and five instances of a single component, the LNA circuit [20] of Fig. 5. It was designed in one of TI's 0.6μm CMOS processes. 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 LNA is itself a complex cell, and has 20 independent variables. As we shall see, the number of optimizable degrees of freedom depends on the synthesis style we choose.

A. Flat Synthesis for EQF

As expected, attempts here were unsuccessful. A good design for EQF can be simulated at device level, but not inside an optimization loop that seeks to visit 10,000 to 100,000 candidate solutions. In addition, we experienced simulator convergence difficulties for the "exotic" parameterizations visited early in synthesis.

B. Iterated Sequential Synthesis for EQF

We undertook several synthesis experiments in this style, including both exploratory synthesis (top-level only, no device details), and fully detailed synthesis.

In the initial stages of design it is convenient to be able to evaluate a system-level choice before each analog cell is fully designed. This allows design dead-ends, especially poor topology choices, to be quickly recognized. To evaluate the EQF's system-level topology we performed a high-level exploration that determined the relationship between noise and area. Because EQF's top-level R's are small but noisy and its C's are large but noiseless, noise versus area is a fundamental trade-off for the EQF system. To accomplish this, we replaced each LNA with a simple two-pole opamp model, and used a symbolic package (MATLAB) to derive a symbolic expression for the output noise via nodal analysis on EQF's adjoint network. At this stage in the design, we assumed that each opamp was noiseless, and ignored the
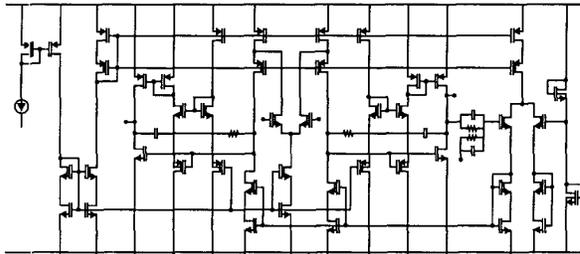

Fig. 5 The low noise operational amplifier used in the EQF.
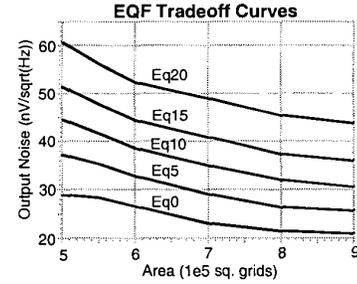

Fig. 6 Area vs. noise trade-off curves for EQF design exploration.

CMOS switch parasitics. This allowed us to estimate the absolute minimum noise for a given area.

After eliminating the dependent variables (*e.g.*, set by the transfer function) we were left with twelve independent variables that could be set by our optimization engine. We used the derived symbolic expressions as our "simulation" engine and instructed the optimization engine to minimize noise for a specified area. A double-sided penalty function was added to the cost function (2) to ensure that the specified area was obtained for each run. Using this approach we were able to generate the area versus noise trade-off curves shown in Fig. 6. These curves give us a good idea if the EQF system topology makes sense given the noise specifications for the design. Is the required area reasonable? Is it possible to meet the noise specification with the current topology? As it turns out, these results are reasonable but optimistic, as we shall see when we later compare this exploration to detailed synthesis results.

In our second approach we synthesized the EQF using a two step process. First, the top-level devices and components were sized, then the LNA was synthesized separately. Because we synthesized the top-level first, the simpler 2-pole model of opamps 1-5 in Fig. 2 was replaced with the more detailed circuit-level macromodel in Fig. 7. One drawback here is that if we wish only to synthesize the top-level components alone, the LNA macromodel parameters need to be set first. Some of the parameters, such as DC gain and the pole locations, can be calculated reasonably well during high-level design exploration. However, other parameters such as noise and input capacitance can only be approximated initially, then further refined once the LNA is actually synthesized. This is the reason this style of synthesis is ultimately iterative.

As with the exploration experiment, after pruning dependent variables, we were left with twelve independent designables for the top-level. Because we already had a finished hand design for EQF (*i.e.*, our "legacy IP"), we simply set the values for the LNA macromodel to the actual performance parameters measured from the hand design.

To test our approach we first synthesized the top-level several times to the same performance specifications as specified for the hand EQF design. Our synthesized results were *all* comparable to the hand design. More interestingly, the synthesized EQF's component values were very
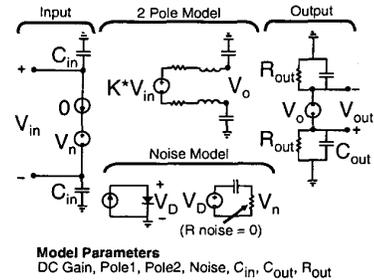

Model Parameters
DC Gain, Pole1, Pole2, Noise, C_{in}, C_{out}, R_{out}
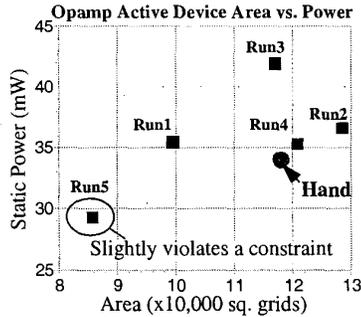
Fig. 7 LNA macromodel.

Fig. 8 Area vs. power scatter for 5 *consecutive* LNA syntheses. All met specs; Run5 had a few devices with an effective voltage smaller than desired.

Table 1. Nominal performance specs for the LNA. R1-R5 correspond to nominal characteristics for the scatter plot points of Fig. 8.

Vdd = 3.3V

	Hand	R1	R2	R3	R4	R5
DC Gain (dB)	135	135	135	135	135	135
UGF (MHz)	110	121	126	132	118	118
PM (deg)	54	54	54	54	54	54
Noise (nV/Hz$^{-0.5}$)[1]	6.33	6.22	6.07	5.88	6.32	6.28
THD (%)[2]	0.16	0.13	0.12	0.14	0.15	0.15

1 Max input referred in range 25kHz-1104kHz

2 1MHz 2Vp-p input voltage, 5kΩ load

close to being the *same* component values as the hand design. We regard these as good results because the hand design had been aggressively optimized by an experienced designer for several weeks.

Next we synthesized the LNA five *consecutive* times. The results are summarized as the area versus power scatter plot shown in Fig. 8 as well as the list of each opamp's nominal performance characteristics shown in Table 1. All five designs met the *nominal* performance constraints as defined by the hand design. Notice that four of the five synthesized results are within 20% of the hand design in terms of area and power. The circled result in Fig. 8 had a few devices with an effective voltage smaller than desired. This affected robustness across manufacturing variations; we verified each design across 3σ process, 0-100°C temperature and 10% supply voltage variations.

Finally, the entire EQF system was verified by replacing the symbolic macromodel with a synthesized opamp and simulating the complete EQF. This process was done for *each* of the four good opamp designs. *All* four met the EQF system level performance specifications.

A population size of 50 circuits, with 5 annealing streams, was used to synthesize the top-level. For each run approximately 25,000 design points were evaluated in roughly 2 hours. The LNA was synthesized using a population size of 100, with 10 annealers. Each run evaluated approximately 100,000 designs, resulting in a runtime of 10 hours.

C. Concurrent Synthesis for EQF

In our third experiment, we synthesized the top-level and LNA concurrently. The advantage of this approach is that the LNA's performance parameters are derived *dynamically* over the course of each synthesis run. An example of dynamically setting a performance parameter is shown in Fig. 9. The *difference* between the LNA's measured input referred noise and the macromodel's noise parameter is effectively zero when the synthesis run completes: the top-level and component synthesis tasks have *negotiated* the right specification here.

The number of independent variables for the entire system was 39: 12 top-level variables, 7 designable macromodel parameters and 20 LNA designables. Three *consecutive* synthesis runs were performed
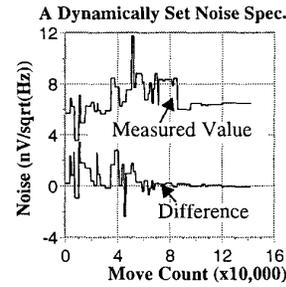

Fig. 9 Actual *measured value* from TISpice for input referred noise of the LNA over the lifetime of a synthesis run. The *difference* between the *measured value* and the evolving macromodel parameter value is also shown.

Table 2. - Derived performance characteristics for each of the three synthesis run.

Vdd = 3.3V

	Hand	Run1	Run2	Run3
DC Gain (dB)	135	131	132	130
UGF (MHz)	110	74	100	126
PM (deg)	54	64	53	56
Noise (nV/Hz$^{-0.5}$)[1]	6.33	8.35	7.16	6.44
THD (%)[2]	0.16	0.16	0.11	0.12

1 Max input referred in range 25kHz-1104kHz

2 1MHz 2Vp-p input voltage, 5kΩ load

and, as before, several steps were taken to verify the results. First, each LNA was verified separately. Table 2 shows each LNA's nominal performance characteristics which were dynamically derived over the course of the synthesis runs. The LNAs were verified across manufacturing variations, and all had histograms with reasonable spreads when compared to the hand design. After verifying the LNAs, each was inserted into its respective top-level design, and the entire EQ was verified by simulation.

Fig. 10 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 cutoff 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. 1), 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.
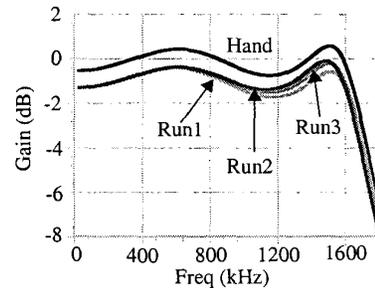

Fig. 10 Frequency response in the passband for the hand design and each of the three synthesis runs.
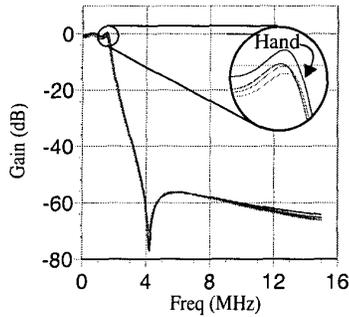
5

Fig. 11 Frequency response: hand EQF design and 3 three synthesis runs.

Table 3. - Noise and area results for three consecutive runs.

Nominal Noise Specs @25°C (nV/sqrt(Hz))

Max Output Noise 25kHz-1104kHz

	Hand	Run1	Run2	Run3
Eq0	36	35	33	33
Eq5	40	38	37	38
Eq10	44	41	40	43
Eq15	49	47	45	48
Eq20	55	54	52	54
Eq25	108	108	104	108

Area (1000 sq. grids)

	Hand	Run1	Run2	Run3
LNA	117	93.5	146	96.5
Top*	680	709	708	560
Total	1265	1177	1438	1043

(*) Area consumed by system level components.

Table 3 shows the final results for the EQF's area and output noise. Notice that the areas for the synthesized designs are close to the hand design: designs Run1 and Run3 have 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. It is also worth noting that the values for noise and area are worse than those predicted in our exploration experiment. For example, from Fig. 6, the predicted output noise value for a top-level area of 700,000 sq. grids was approximately $23nV/\sqrt{Hz}$. Table 3 shows that the final output noise actually obtained was $33nV/\sqrt{Hz}$ for Run2. This difference comes as no surprise: the symbolic macromodel ignored several effects, and was thus overly optimistic. Nevertheless, this does provide significant insight into the overall envelop of performance achievable by EQF in this technology. Any full synthesis flow will likely begin with high-level exploration, followed by detailed synthesis to set final component values.

The population size used was 100, with 10 annealers. Fig. 12 shows how the cost of the population evolved over the course of each
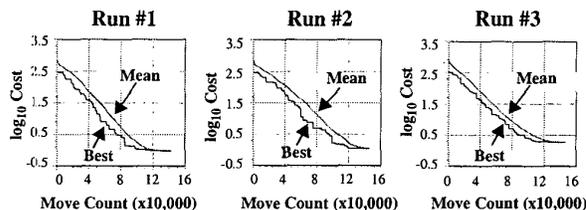

Fig. 12 Log cost versus the number of design points evaluated. The best circuit's cost in the population and the mean cost of the circuit population are shown over the lifetime of each synthesis run.

synthesis run. The log of the circuit population's mean and best cost are shown as a function of the move count. The move count is the total number of design points evaluated, about 150,000 total in this case, resulting in a total runtime of 12 hours.

VI. CONCLUSIONS

To the best of our knowledge, this is the largest, most complex, most thorough controlled experiment ever undertaken to demonstrate that recent developments in analog synthesis have direct application to state-of-the-art industrial analog systems. We have successfully redesigned the EQF block in the ADSL frontend in several different ways, and examined the various 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 this important problem. We believe these ideas have significant potential in both fully custom analog design, and IP creation/reuse scenarios.

REFERENCES

[1] Steve Ohr, "Analog IP Slow to Start Trading", *EETimes*, Issue 1053, March 22 1999 (Also:.http://www.eet.com)
[2] W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS Opamp Synthesis by Means of a Genetic Algorithm," *Proc. 32nd ACM/IEEE DAC*, 1995.
[3] 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.
[4] M. Degrauwe *et al.*, "Towards an analog system design environment," *IEEE JSSC*, vol. sc-24, no. 3, June 1989.
[5] 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.
[6] B.J. Sheu, *et al.*, "A Knowledge-Based Approach to Analog IC Design," *IEEE Trans. Circuits and Systems*, CAS-35(2):256-258, 1988.
[7] J. P. Harvey, *et al.*, "STAIC: An Interactive Framework for Synthesizing CMOS and BiCMOS Analog Circuits," *IEEE Trans. CAD*, Nov. 1992.
[8] 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.
[9] G. Gielen, *et al.*, "Analog circuit design optimization based on symbolic simulation and simulated annealing," *IEEE JSSC*, vol. 25, June 1990.
[10] M. Hershenson, S. Boyd, T. Lee, "GPCAD: a Tool for CMOS Op-Amp Synthesis", *Proc. ACM/IEEE ICCAD*, pp. 296-303, 1998
[11] G. Gielen, P. Wambacq, W. Sansen, "Symbolic Analysis Methods and Applications for Analog Circuits: A Tutorial Overview," *Proc. IEEE*, vol. 82, no. 2, Feb., 1994.
[12] 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.
[13] E. Ochotta, R.A. Rutenbar, L.R. Carley, "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX," *IEEE Trans. CAD*, vol. 15, no. 3, March 1996.
[14] Y-C Ju, V.B. Rao and R. Saleh, "Consistency Checking and Optimization of Macromodels", *IEEE Transactions on CAD*, August 1991.
[15] B. Antao and A. Brodersen, "ARCHGEN: Automated Synthesis of Analog Systems", *IEEE Transaction on VLSI Systems*, June 1995.
[16] 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.
[17] A. Doboli, et al, "Behavioral synthesis of analog systems using two-layered design space exploration," *Proc. ACM/IEEE DAC*, June 1999.
[18] M. Krasnicki, R. Phelps, R.A. Rutenbar, L.R. Carley, "MAELSTROM: Efficient Simulation-Based Synthesis for Analog Cells," *Proc. ACM/IEEE Design Automation Conference*, June 1999.
[19] 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.
[20] R. Hester, *et al.*, "CODEC for Echo-Canceling, Full-Rate ADSL Modems," *IEEE Int'l Solid-State Circuits Conference*, pages 242-243. 1999.
[21] P.J.M van Laarhoven and E.H.L. Aarts, *Simulated Annealing: Theory and Applications*, D. Reidel Pub. Co./Kluwer, Dordrecht, Holland, 1987.
[22] M. Krasnicki, *Generalized Analog Circuit Synthesis*, M.S. Thesis, Dept. of ECE, Carnegie Mellon University, Dec. 1997.
[23] R. Phelps, *Population-Based Synthesis for Analog Cells and Systems*, Ph.D. Thesis, Dept. of ECE, Carnegie Mellon University, June 2000 (expected).