ASTRX/OBLX: Tools for Rapid Synthesis of High-Performance Analog Circuits

Emil S. Ochotta, Rob A. Rutenbar, and L. Richard Carley
Electrical and Computer Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract—We describe ASTRX/OBLX, a synthesis system that can size high-performance analog circuit topologies to meet user-supplied linear performance specifications without designer-supplied equations. We present synthesis results for a large suite of circuit benchmarks and show that, when compared to prior approaches, ASTRX/OBLX can synthesize high-performance circuits with up to 3 orders of magnitude less initial design effort.

I. Introduction

A surprising number of modern technologies rely on a core of analog circuitry; cellular telephones, magnetic disk drives, and compact disc players are just a few such examples. Tomorrow's neural networks, speech recognition systems, and ultra low-power personal communication devices will require more mixed analog/digital Application Specific Integrated Circuits (ASICs). To maximize profit, ASIC designers must minimize time-to-market. Digital CAD tools facilitate this by providing a rapid path to silicon for large-scale digital designs. Unfortunately most analog circuits, although small in size, are still designed manually by experts using time-consuming techniques that have remained largely unchanged in the past 20 years. With the advent of logic synthesis tools and semi-custom layout techniques to automate much of the digital design process, the analog section may consume 90% of the overall design time, while consuming only 10% of the ASIC's die area.

In [1] we outlined a new approach to analog circuit synthesis, *i.e.*, automating part of the analog design process by translating a specification into a circuit schematic with sized devices, and described a prototype implementation called ASTRX/OBLX. The scope of this work was synthesis for *cell-level* (10 to 50 devices) *linear* circuits: starting from a transistor schematic, we sought both to design a dc bias point and size all devices to meet performance targets such as gain and bandwidth. In this paper, we validate our initial work by synthesizing a suite of benchmark circuits that blankets essentially all the previous analog cell synthesis results, and by explicitly comparing to these prior results using three critical metrics for any analog synthesis tool:

- *Accuracy:* the discrepancy between the synthesis tool's internal performance prediction mechanisms, and those of a detailed circuit simulator that uses realistic device models.
- *Synthesis time:* the CPU time required by the synthesis tool.
- *Preparatory effort:* the designer-time/effort required to render a new circuit design in a form suitable for the tool to complete.

In comparison to prior approaches, ASTRX/OBLX typically requires hours, rather than weeks/months to compose a new circuit problem; tens of lines of constraints, rather than thousands of lines of executable code; and hours of CPU time on a fast workstation, instead of seconds or minutes—the price we pay for ease of formulation.

This paper is structured as follows. Section II reviews prior approaches to analog circuit synthesis. Section III presents the basic formulation underlying ASTRX/OBLX, while Section IV presents a circuit synthesis example to show how this formulation is applied to a real synthesis problem. In Section V we revisit some of the more interesting aspects of ASTRX/OBLX in more detail. Section VI describes results from our suite of circuit benchmarks and compares to prior approaches. Finally, Section VII offers concluding remarks.

II. Review Of Prior Approaches

Previous approaches to analog circuit synthesis [2][3][4][5][6][7][8][9] have failed to make the transition from research to practice. This is due primarily to the prohibitive one-time effort required to derive the complex equations that drive these synthesis tools. Because they rely on a core of equations, we refer to all previous approaches to synthesis as "equation-based". At each step in the synthesis process, performance equations are used to evaluate the evolving circuit design, determine how well it meets its specifications, and provide feedback to a search mechanism to guide the synthesis process. Because of their reliance on equations, these systems are still limited in the crucial areas of accuracy and automation. Let us examine these issues in greater detail.

Accuracy: Equation-based approaches rely heavily on simplifications to circuit equations and device models. The performance of the synthesized circuit often reflects the limitations of the equations used to model it, rather than the inherent limitations of the topology. The need for high performance circuits using the latest technologies invalidates the use of many of these simplifications. For example, in a 3µ MOS process, $I_{DS} = K' W/2L\,(V_{GS} - V_T)^2$ is a workable model of the current-voltage relationship for a device, and equation-based approaches take advantage of the fact that it can be inverted to allow either voltage or current as the independent variable. But this is a grossly inaccurate model for a device with a submicron channel length. The need to support complex device models and circuit design goals that push the limits of these models is fundamentally at odds with equation-based strategies that rely on these simplifications.

Automation: Equation-based tools appear to design circuits quickly. But, the run-time of the tools is misleading because it does not consider the time required to derive the circuit equations. Even for a relatively simple analog circuit, these equations are very complex, require considerable analog design expertise to derive, and must be entered as thousands of lines of program code. For a textbook design this process can take weeks [7], while for an industrial design it can take several designer-years [3], and the process must be performed for each new circuit topology added to the synthesis tool's library. Moreover, adding these equations typically requires a user who is a programmer, an analog designer, and intimate with the internal architecture of the tool. As a result, it is almost always easier for an industrial mixed-signal ASIC designer to design circuits manually rather than dedicate the effort required to teach these tools to do it. There have been several attempts to make it easier to codify these performance equations. For example: OASYS [5] structures circuit equations hierarchically in an attempt to provide reusable circuit building blocks, and ARIADNE [4] provides a symbolic simulator to assist in deriving transfer functions for linear circuits. However, even the most promising of these attempts, symbolic simulation, has yet to overcome sub-

stantial technical obstacles before it can fully automate performance equation derivation for high-performance circuits. Symbolic simulation of a few tens of devices can generate expressions with tens of thousands of terms. Despite ongoing efforts [10] with these methods, we are unaware of any effective pruning strategy that can generate expressions that are accurate over the range of circuit parameters that must be explored during synthesis.

One simple solution is to replace the equations with a direct simulation technique. This is the approach that was first proposed for analog circuit optimization decades ago [11]. A more recent example, DELIGHT.SPICE [12], replaces performance equations with SPICE and employs a gradient-based optimization technique for search. However, DELIGHT.SPICE is an *optimization* tool, not a *synthesis* tool, a distinction that has proved insurmountable in the past decade. The key hurdle that has not been overcome to make this transition from *optimization* to *synthesis* is the inefficiency of simulation and its impact on starting point sensitivity.

Efficiency/Starting Point Sensitivity: Because SPICE-class simulators are slow, the search mechanism must invoke the simulator as *in*frequently as possible. As a result, simulation-based methods use local optimization techniques that require few iterations to converge. These techniques must be primed with a good initial circuit design, otherwise, an optimization may not converge or may converge to a local minima significantly worse than the circuit's best capabilities [12]. In circuit synthesis, a local optimizer is not practical both because the search space contains many local but non-global minima [6][12] and because the user cannot be expected to provide a good initial circuit design.

In the remainder of this paper, we describe an alternative synthesis formulation that combines the strengths of both equation-based and simulation-based approaches.

III. ASTRX/OBLX: BASIC FORMULATION

In this section, we present the full synthesis formulation of ASTRX/OBLX, including core ideas originally presented in [1]. We begin with the specific design goals that guided the evolution of this formulation and the key ideas that form its foundation. We then outline the architectural aspects of their realization in ASTRX/OBLX.

A. Design Goals for ASTRX/OBLX

Our design goals for a new analog circuit synthesis architecture are to address directly the automation, accuracy, and efficiency problems we identified with previous approaches. First, it should require only hours rather than weeks/months of preparatory effort to design a new circuit. Second, our new system should yield accurate performance predictions for high-performance circuits rather than suffering from problems due to device model or performance equation simplifications. And third, the system should find high-quality circuit design solutions without regard to starting point rather than getting trapped in the nearest local minima. Our primary goal here is efficiency. We wish to streamline the path from a circuit idea to a sized circuit schematic. Equation-based synthesis tools use only minutes of CPU time but require the designer to spend months deriving, coding, and testing equations. We believe the following scenario is more appealing: after an afternoon of effort, a circuit designer goes home while the synthesis tool completes the design overnight. Realizing this scenario is the primary goal behind ASTRX/OBLX.

B. Ideas Underlying the ASTRX/OBLX Strategy

To achieve these goals our circuit synthesis strategy relies on five key ideas: synthesis via optimization, asymptotic waveform evaluation, simulated annealing, encapsulated device evaluators, and the relaxed-dc numerical formulation. We describe these ideas below.

Synthesis via Optimization: We perform fully automatic circuit synthesis using a constrained optimization formulation, but solved in an unconstrained fashion. As in [7][8][12], we map the circuit design problem to the constrained optimization problem of (1). Where $\underline{x}$ is the set of independent variables—geometries of semiconductor devices or values of passive circuit components—we wish to change to determine circuit performance; $f(\underline{x})$ is a set of objective functions that codify performance specifications the designer wishes to optimize, *e.g.* power or bandwidth; and $g(\underline{x})$ is a set of constraint functions that codify specifications that must be beyond a specific goal, *e.g.*, gain ≥ 60dB. Scalar weights, w_i, balance competing objectives.

$$\underset{\underline{x}}{\text{minimize}} \sum_{i=1}^{k} w_i \cdot f_i(\underline{x}) \qquad \text{s.t.} \quad \underline{g}(\underline{x}) \le 0 \tag{1}$$

To allow the use of simulated annealing, we perform the standard conversion of this constrained optimization problem to an unconstrained optimization problem with the use of additional scalar weights. As a result, the goal becomes minimization of a scalar cost function, $C(\underline{x})$, defined by (2).

$$C(\underline{x}) = \sum_{i=1}^{k} w_i f_i(\underline{x}) + \sum_{j=1}^{l} w_j g_j(\underline{x}) \tag{2}$$

The key to this formulation is that the minimum of $C(\underline{x})$ corresponds to the circuit design that best matches the given specifications. Thus, the synthesis task becomes two more concrete tasks: evaluating $C(\underline{x})$ and searching for its minimum. However, performing these tasks is not easy. In equation-based synthesis tools, evaluating $C(\underline{x})$ is done using designer-supplied equations. To achieve our automation goals, we must avoid the months/years of preparatory effort it may take to derive these equations. Moreover, in searching for the minimum, we must address the issues of starting point independence and global optimization, since $C(\underline{x})$ may have many local minima.

Asymptotic Waveform Evaluation: To evaluate circuit performance, *i.e.* $C(\underline{x})$, without designer-supplied equations, we rely on an innovation in simulation called Asymptotic Waveform Evaluation (AWE)[13]. AWE is an efficient approach to analysis of arbitrary linear circuits that is several orders of magnitude faster than SPICE. By matching the initial boundary conditions and the first $2q-1$ moments of the actual circuit transient response to a reduced q-pole model, AWE can predict small-signal circuit performance using a reduced complexity model. AWE is a general simulation technique that can be applied to any linear circuit and yields accurate results without manual circuit analysis. Thus, AWE replaces performance equations, but does so at a fraction of the run-time cost of SPICE-like simulation.

Simulated Annealing: We have selected simulated annealing [14] as the optimization engine that will drive our search for the best circuit design in the solution space defined by $C(\underline{x})$. This method provides the potential for global optimization in the face of many local minima. Simulated annealing has a theoretically proven ability to find the global optimum under certain restrictions [15]. Although these restrictions are not enforceable for most industrial applications, the proofs suggest an algorithmic robustness that has been validated in practice [16]. Because annealing incorporates controlled *hill-climbing* it can escape local minima and is starting-point independent. Annealing has other appealing properties including its ability to optimize without derivatives. Further, although annealing typically requires more function evaluations than local optimization techniques, it is now achieving competitive run-times on problems for which tuned heuristic methods exist [17]. Because annealing directly solves *un*-constrained optimization problems, we require the scalar cost function of (2).

Encapsulated Device Evaluators: To model active devices, we rely on a compiled database of industrial models we call *encapsulated device evaluators*. These provide the accuracy of a general purpose simulator while making the synthesis tool independent of low-level device modeling concerns. As with any analysis of a circuit, we use models to linearize non-linear devices, generating a small signal circuit that can be passed to AWE. Using one or two equation approximations instead of the hundreds of equations used in industrial device models is no longer a viable alternative in a practical synthesis sys-

tem. Unlike equation-based performance prediction, where assumptions about device model simplifications permeate the circuit evaluation process, with encapsulated devices all aspects of the device's representation and performance are hidden and obtained only through requests to the evaluator. In this manner, the models are completely independent of the synthesis system and can be as complex as required. For our purposes we rely entirely on device models adopted from detailed circuit simulators such as Berkeley's SPICE 3.

Relaxed-dc formulation: To avoid a CPU intensive dc operating point solution after each perturbation of the circuit design variables, we rely on a novel re-casting of the unconstrained optimization formulation for circuit synthesis [8] we call the *relaxed-dc formulation*. Supporting powerful device models is not easy within a synthesis environment because we cannot arbitrarily invert the terminal relationships of these models and choose which variables are independent and which are dependent. This critical simplification enables equation-based approaches to solve for the dc bias point of the circuit analytically and, as a result, very quickly. In contrast, when the models must be treated numerically, as in circuit simulation, an iterative algorithm such as Newton-Raphson is required. For synthesis, this approach consumes a substantial amount of CPU time that we would prefer not to waste on intermediate circuit designs that are later discarded. Instead we explicitly formulate Kirchhoff's laws, which are solved implicitly during dc biasing, and include them in $g(\underline{x})$, the constraint functions in (2). Just as we must formulate optimization goals such as meeting gain or bandwidth constraints, we now formulate dc-correctness as yet another goal to meet.

C. ASTRX/OBLX System Architecture

In ASTRX/OBLX, we combine these five ideas to create an architecture that provides a fully automated path from an un-sized circuit topology and a set of performance specifications to a completed, synthesized circuit. This path is comprised of two phases:

- **Compilation by ASTRX:** Compilation generates code that implements the cost function, $C(\underline{x})$. To evaluate this cost function, ASTRX will compile in the appropriate links to the encapsulated device evaluators and AWE. Because of our relaxed-dc formulation, ASTRX must also derive the dc-correctness constraints that will enforce Kirchhoff's laws and encode them in the cost function.
- **Solution by OBLX:** This cost function code is then compiled and linked to OBLX, our solution library, which uses simulated annealing to numerically find its minimum, thereby designing the circuit.

IV. SYNTHESIS EXAMPLE

In this section we present a complete synthesis example to make concrete the entire path from problem to solution. In our example, we focus on the input the designer must provide to ASTRX and the steps performed by ASTRX to translate this input into $C(\underline{x})$.

A. Describing The Circuit Under Design

Assume we wish to size and bias the simple differential amplifier topology shown in Fig. 1a to maximize differential gain (A_{dm}) such that unity gain frequency (UGF) is at least 1MHz and the slew rate (SR) is at least 1V/µs. There are two main parts to the input description file, the topology of the circuit under design, and the performance specifications the completed design must achieve. For our example, assume the dimensions of M1 and M2 and the values of I and Vb are unknown. (*i.e.*, M3 and M4 are given.) We begin by listing these independent variables: $\underline{x} = \{W, L, I, Vb\}$. Note that we wish to determine only a single width and length because the circuit is differential, so the two transistors (M1, M2) must be matched. We can describe this to ASTRX with the topological description of the circuit by listing all the elements, their port nodes, and their values. M1 and M2 can be automatically matched by using the same expression for their dimensions. Assuming the variables, $\{W, L, I, Vb\}$, are declared elsewhere in the description, this information (whose format is designed after the familiar SPICE notation) is:

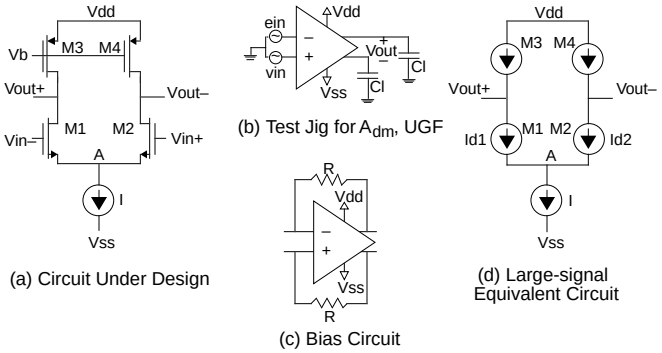

Fig. 1: Design Example

```
.subckt  oa     in+     in-     out+    out-    nvdd    nvss
M2       out+   in-     A       nvss    Ne      w='W'   l='L'
M1       out-   in+     A       nvss    Ne      w='W'   l='L'
M3       out+   nvb     nvdd    nvdd    Pe      w=2u    l=1.2u
M4       out-   nvb     nvdd    nvdd    Pe      w=2u    l=1.2u
vvb      nvb    nvss    'Vb'
ib       A      nvss    'I'
.ends
```

Our next task is to describe the performance specifications. Since ASTRX/OBLX simulates circuit performance with AWE, we must provide a *simulation oriented* definition of each specification. To describe a specification, we need to provide a *test jig*, a set of measurements to make on the jig, and any simple arithmetic that must be performed to calculate the values we are interested in. The test jig is important because it supplies the environment (stimulus, load, supplies, etc.) in which the circuit is to be tested. We can use the test jig shown in Fig. 1b to measure both A_{dm} and UGF. For ASTRX/OBLX, this would look as follows:

```
xamp    in+     in-     out+    out-    nvdd    nvss    oa
vdd     nvdd    0       vddval
vss     nvss    0       vssval
vin     in+     0       0       ac      1
ein     in-     0       0       in+     1
cl1     out+    0       Cl
cl2     out-    0       Cl
.pz     tf      V(out+) vin
.obj    Adm     'dc_gain(tf)'   good = 1000     bad = 10
.spec   ugf     'ugf(tf)'       good = 1Meg      bad = 10k
```

Here, the `.pz` card tells AWE to determine the transfer function `V(out+)/Vin` directly. It is then straightforward to write expressions for A_{dm} and UGF using pre-defined functions available in ASTRX. In the above example, the `good` and `bad` values are used both to specify bounds on specifications and to normalize the contributions of each performance metric to $C(\underline{x})$. For our example, we use a single test jig for A_{dm} and UGF, but in general we can specify several jigs with which to measure circuit performance with AWE.

Until now, we have neglected the slew rate. This is because measuring slew rate would require a transient simulation, which is not straightforward with AWE. However, unlike gain and unity gain frequency, slew rate is described with an easily derived expression. If we assume that we are interested only in the rate at which the output slews downwards, we can write this expression by inspection as $SR = I/(2(Cl + Cd))$, where Cd is the capacitance at the output node due to the transistors. In ASTRX this specification becomes:

```
.spec SR 'I/(2*(Cl+xamp.m1.cd+xamp.m3.cd))' good=1Meg bad=10k
```

ASTRX supports a complete equation-based description of circuit performance, thus ASTRX/OBLX has all the flexibility of previous equation-based techniques but also provides direct support for performance measured with AWE. Given a simple circuit, symbolic equations can be derived for A_{dm} and UGF. However, as experiments with symbolic simulation have shown, these equations can be huge, *e.g.*, 10,000+ terms for a circuit with 10 devices, and the number of terms grows exponentially with circuit size. In contrast, AWE uses a numeric technique and can evaluate A_{dm} and UGF in a few tens of millisec-

onds for circuits of this size. Moreover, AWE's algorithmic complexity is roughly that of an LU factorization, approximately $O(n^{1.4})$ where n is the number of nodes in the circuit. The speed of AWE and the ability to describe linear performance specifications without deriving circuit equations are two of the chief advantages of ASTRX/OBLX over previous synthesis approaches.

Our description of the synthesis problem is almost complete. However, we must also provide one additional circuit, a bias circuit. To understand the need for this circuit, we must discuss the treatment of non-linear devices in more detail. Like a detailed circuit simulator, ASTRX/OBLX must know the node voltages in the circuit to act as input to the device evaluators. An evaluator converts the dimensions and port voltages of the device into a set of linear elements that models the device's behavior at that operating point. After replacing each transistor with its model, we can then use AWE to evaluate the circuit's performance. To obtain these voltages, we separate the small-signal and dc bias concerns for the circuit under design—a technique familiar to analog designers. For our example, we can use the bias circuit of Fig. 1c. The topology of this bias circuit is the last piece of input the designer must supply to ASTRX.

B. Mapping the Synthesis Problem Into An Optimization Problem

In this section, we show how ASTRX maps the problem description into the components of the cost function of (2). We begin with the independent variables, $\underline{x}$. This is quite simple: we use the independent variables specified by the designer. Thus, $\underline{x} = \{W, L, I, Vb\}$. However, we shall see that we must add other variables to $\underline{x}$ as well.

Next we must determine $f(\underline{x})$, the set of objective functions. For our example, the only objective is to maximize A_{dm}. Thus, $f(\underline{x})$ contains only the function that calculates A_{dm}, which we label $f_{Adm}(\underline{x})$. (Following [12], we use the good and bad values given by the user to transform $f_{Adm}(\underline{x})$, such that we *maximize* A_{dm} but *minimize* $C(\underline{x})$.)

To understand the additions we need to make to $\underline{x}$, we trace the information required to calculate $f_{Adm}(\underline{x})$. Recall that we require a bias circuit to provide device port voltages as input to the evaluators. However, we did not discuss how we would solve for the node voltages in that bias circuit. In ASTRX/OBLX, solving for these voltages is not a separate procedure, instead we simply include the node voltages in $\underline{x}$. Thus, ASTRX adds variables for the independent node voltages in the bias circuit of Fig. 1c. Now, $\underline{x} = \{W, L, I, Vb, \text{Vout+}, \text{Vout–}, V_A\}$. This is the first component of the relaxed-dc formulation.

We complete the relaxed-dc formulation by forcing the node voltages to take values such that Kirchhoff's laws are obeyed. To accomplish this we replace the transistors in the circuit we are trying to design (Fig. 1a) with large-signal models returned by the device evaluators, giving the circuit of Fig. 1d. Kirchhoff's current law (KCL) can then be written at each node in the circuit. *e.g.*, at node A: $\text{I} - \text{Id1} - \text{Id2} = 0$. This KCL equation must be met when our optimization is complete. To ensure this, we include the equation in the set of constraint equations, $g(\underline{x})$, in $C(\underline{x})$. Specifically, we can write the KCL equation at node A as the constraint equation:

$$g_{\text{A}}(\underline{x}) = \max(0, |\text{I} - \text{Id1} - \text{Id2}| - \tau_{abs}) \tag{3}$$

Thus, $g_{\text{A}}(\underline{x})$ contributes a penalty to the cost function whenever the KCL error at node A is larger than some numeric tolerance, τ_{abs}[1]. We formulate the other KCL equations in the same fashion, creating $g_{\text{Vout+}}(\underline{x})$ and $g_{\text{Vout–}}(\underline{x})$. Together, these three constraints complete the relaxed-dc formulation.

For our example, the other members of $\underline{g}(\underline{x})$ correspond to the other performance specifications we wish to design for. Thus, we need constraint functions for UGF and SR. Including these final terms, we obtain (4), the final form of the cost function ASTRX generates and OBLX must minimize to complete the circuit design.

$$\begin{aligned} C(\underline{x}) = {} & w_{Adm} f_{Adm}(\underline{x}) + w_{UGF} g_{UGF}(\underline{x}) + w_{SR} g_{SR}(\underline{x}) + \\ & w_{\text{A}}\, g_{\text{A}}(\underline{x}) + w_{\text{Vout+}} g_{\text{Vout+}}(\underline{x}) + w_{\text{Vout–}} g_{\text{Vout–}}(\underline{x}) \end{aligned} \tag{4}$$

V. ASTRX/OBLX Design Issues

In this section, we revisit the ASTRX/OBLX formulation, presenting further details of its design. We then focus on some of the interesting implications the relaxed-dc formulation has on circuit synthesis.

A. Algorithmic Aspects of ASTRX/OBLX

As seen in the example, the tasks performed by ASTRX can be summarized as (a) determine the set of independent variables ($\underline{x}$), (b) generate large-signal equivalent circuits for biasing, (c) write KCL constraints for the large-signal circuits, (d) generate small-signal equivalent circuits for AWE, (e) generate cost terms for each circuit performance metric specified by the user, and (f) write all the code that describes the cost function for this circuit synthesis problem.

A somewhat subtle aspect of compilation is determining the set of independent variables, $\underline{x}$. The user specifies most of these, but ASTRX must find a set of independent node voltages to include in $\underline{x}$ as part of the relaxed-dc formulation. To do so, ASTRX performs a tree-link analysis of the large-signal equivalent circuit, which is built from the input netlist with the help of device templates provided by the encapsulated device evaluators. Whenever a node voltage cannot be trivially determined, its value becomes another variable in $\underline{x}$.

To completely describe the annealing formulation used in OBLX, we must describe its four principal components: the *representation* of the synthesis problem manipulated by the annealer; the *moves* used to transform one circuit configuration into another; the *cost function* that evaluates the quality of each visited circuit configuration; and the *control mechanisms* that direct the overall cooling process.

Representation: The problem representation is conceptually straightforward: the variables in $\underline{x}$ map to aspects of the evolving circuit design, such as device sizes and node voltages. However, we do not represent all the variables in $\underline{x}$ as continuous values. Node voltage values must clearly be continuous to determine an accurate bias point. Device sizes, however, can reasonably be regarded as discrete quantities, since we are limited by how accurately we can etch a device. Moreover, there is considerable advantage to be had from properly discretizing device sizes: the coarser the discretization, the smaller the space of reachable sizes that must be explored. Because small changes in device sizes make proportionally less difference on larger devices, we typically use a logarithmically spaced grid.

Move-Set: Given the present state, $\underline{x}$, the *move-set* is the set of allowable perturbations on it, $\Delta\underline{x}$. The basic problem is the need for an efficient mechanism to generate each perturbation. For discretized variables there is always a smallest allowable move, an *atomic* perturbation. The problems here are what larger moves should be included in the move-set for efficiency and how to decide when to use these larger moves. For continuous variables the situation is more complex. For an n-dimensional real-valued state, $\underline{x} \in R^n$, determining the right *smallest* $\Delta\underline{x}$ is itself a difficult issue. In OBLX, we may need to explore across a voltage range of several volts and then converge to a dc bias point with an accuracy of a few microvolts. We aid this convergence by augmenting our move-set with moves that employ the Newton-Raphson algorithm. Newton-Raphson moves all node voltages simultaneously, using the gradient information in the bias circuit's nodal admittance matrix to move toward dc-correctness. A simulator performs a complete Newton-Raphson before it evaluates circuit performance. However, because we are using a relaxed-dc formulation, we do not require a full Newton-Raphson solve after each move OBLX proposes. Instead, we combine random moves of members of $\underline{x}$ with full and partial Newton-Raphson solves to create a palette of *move classes* from which the annealer must select. We then use an automatic move selection method adapted from Hustin [19] that allows

[1] The large-signal model and this formulation of the KCL constraint is somewhat simplified for clarity. For more detail, see [18].

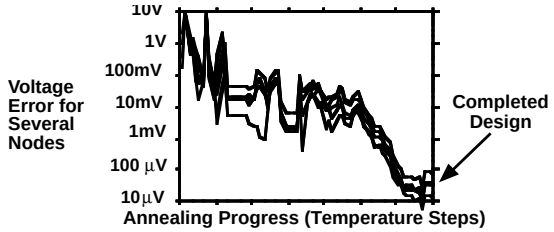

Fig. 2: Discrepancy From KCL Correct Voltages During Optimization

the annealer to determine the most appropriate move class—purely random, gradient-directed, or a combination—at a given point in the optimization process. The use of partial gradient-based moves within an annealing formulation is not new, *e.g.*, the theoretically globally-convergent continuous annealing strategy in [20]. In practice, this technique allows OBLX to converge to a dc operating point at least as reliably as a detailed circuit simulator.

Cost-Function: The heart of the annealer is the circuit specific cost function, $C(\underline{x})$, generated by ASTRX, which maps each visited circuit configuration $\underline{x}$ to a scalar cost. The cost function was described by example in Section IV. In general, it has the form:

$$C(\underline{x}) = \underbrace{C^{obj}}_{\text{objective}} + \underbrace{C^{perf} + C^{dev} + C^{dc}}_{\text{penalty terms}} \quad (5)$$

Where each term in (5) represents a group of related terms. There are two distinct kinds of terms: the *objective* terms implement $f(\underline{x})$ in (2) and must be minimized, while the *penalty* terms implement $g(\underline{x})$ and must be driven to zero. Here, C^{obj} and C^{perf} implement the user supplied figures of merit; C^{dev}, forces devices to be in particular regions of operation; and C^{dc}, implements the relaxed-dc formulation.

Control Mechanisms: To control the annealing process, we have implemented the general purpose cooling schedule of Lam [17] as modified by Swartz [21]. Our freezing criteria, which determines when the annealing has completed, has been developed specifically for our analog synthesis application. The design is complete when both the discrete variables have stopped changing and the changes in the continuous variables are within a specified relative tolerance.

Another aspect of the annealing control process is the issue of numeric constants. Numeric algorithms often require large numbers of constants that tune the algorithm such that it reliably produces high-quality solutions. If these algorithms are poorly designed, the constants will need to be adjusted for each new problem solved. Substantial effort has been spent designing ASTRX/OBLX such that it is truly an automation tool, *i.e.*, the user is not required to provide problem-specific constants. One key aspect of this process is the use of adaptive algorithms to replace the majority of the numeric constants that would otherwise be needed within OBLX. For example, the scalar weights in $C(\underline{x})$ have been replaced with an adaptive weight algorithm. These techniques ensure that an analog circuit designer can use ASTRX/OBLX without understanding its internal architecture. See [18] for a complete discussion of these techniques.

B. Implications Of The Relaxed-DC Formulation

As a result of the relaxed-dc formulation, early in the optimization process the sum of the currents entering a given node has a significant, non-zero value. An important issue to address is what it means to evaluate the performance of a circuit that is not dc-correct. One way to view this circuit is to imagine an additional current source at each node. This current source sinks the current required to ensure dc-correctness. Then, the goal of the Kirchhoff's law constraints added to $g(\underline{x})$ is to reduce the value of these currents sources to zero. When evaluating circuit performance, the fact that these current sources will not be in our final design means that our predicted performance will differ slightly from the final performance. This error factor allows us to visit many more possible circuit configurations within a given period of time, albeit evaluating each a little less accurately. As the optimization proceeds the current sunk by these sources goes to zero and the performance prediction becomes completely accurate. This evolution is shown in Fig. 2. By the end of the annealing process, the circuit will be dc-correct within tolerances not unlike those used in circuit simulation.

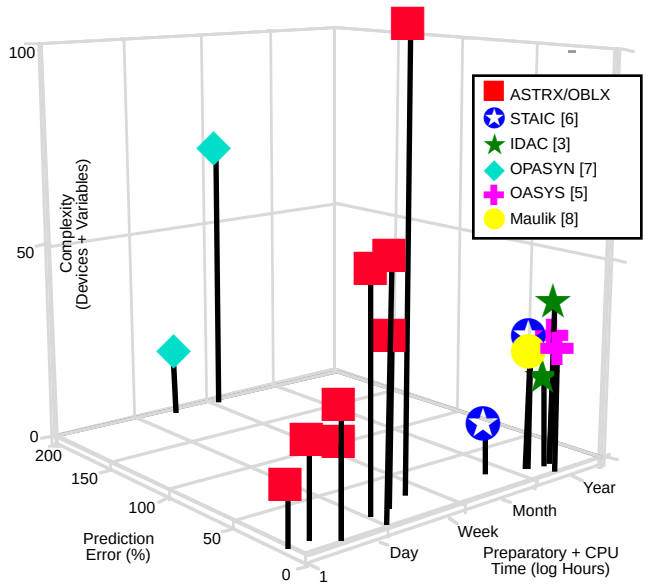

Fig. 3: Complexity, Error and First Time Design Effort. ASTRX/OBLX Compared with Prior Approaches

This analogy points out a seeming inconsistency in our overall synthesis formulation. We clearly do not intend that each annealing move visits a dc-correct circuit, *i.e.*, where the current sources ensuring dc-correctness are zero-valued. Nevertheless, we evaluate each circuit using detailed device models and highly accurate AWE techniques. Clearly this accuracy is not fully exploited early in the optimization when these error currents are substantial. Simpler models could be used in these early stages, but, practically, the evolution from simple to accurate models would be difficult because different models often predict significantly different performance, so changing models might substantially alter the value of $C(\underline{x})$. Moreover, this accuracy is not entirely wasted: the annealer still learns much from these early circuits. For example, if we need to achieve more gain in our circuit, we probably need to increase the g_m on some critical device. We do not need a precise dc bias point to know that we must either increase that device's width, its current, or both. We can successfully make coarse decisions such as this even in the presence of the extra current sources introduced by the relaxed-dc formulation.

VI. Results: Circuit Benchmarks

The primary goal of ASTRX/OBLX is to reduce the time it takes to size and bias a new circuit topology to meet performance specifications. Fig. 3 summarizes a representative selection of previous analog circuit synthesis results. Here, each symbol represents synthesis results for a single circuit topology. The length of the symbol's "tail" represents the complexity of the circuit, which we quantify as the sum of the number of devices in the circuit and the number of the variables the user asks the synthesis tool to determine. Note that this graph includes two circuits published in [22], including the most complex result of which we are aware. The other axes represent metrics for accuracy and automation. The prediction error axis measures accuracy by plotting the worst case discrepancy between the synthesis tool's circuit performance predication and the predictions of a circuit simulator. The time axis measures automation by plotting the sum of the preparatory time spent by the designer and CPU time spent by the tool to synthesize a circuit for the first time. (In [6] where preparatory time was not published, we equated 1000 lines of circuit-specific custom

code to a month.) Fig. 3 reveals three distinct classes of synthesis results. The first class is on the right and contains the majority of previously published work. Here, the synthesis tool predicts performance with reasonable accuracy, but only because a designer has spent months to years of preparatory time deriving the circuit performance equations. The limited range and performance of these prior results is perhaps the best indicator that the first-time effort to design a circuit has always been a substantial barrier to obtaining a broader range of results. The second group of results, those on the left, trade reduced preparatory effort for substantially reduced circuit performance prediction accuracy. Finally, the center group of results is for ASTRX/OBLX. In contrast to the other two groups, generating each new design with ASTRX/OBLX typically involved an afternoon of preparation followed by 5-10 annealing runs performed overnight, yet produced designs that matched simulation with at least as much accuracy as the best prior approaches.

Results of ASTRX's analyses of our complete suite of benchmark circuits are listed in Table 1. Note that just 5 of these topologies (Simple OTA, OTA, Two-Stage, Folded-Cascode, and Comparator) cover essentially all previously published synthesis results. (We make the reasonable assumption that a circuit topology can represent topologies that vary in only minor detail.) Because detailed synthesis results for circuit Comparator appear in [22], we confine our discussion to the remaining circuits. For our complete suite, Table 1 gives the number of lines required for each synthesis problem description and the results of ASTRX's analysis of the problem. Recall that ASTRX compiles the problem description, generating the cost function as C code that will be compiled and linked with OBLX. The number of lines is reported as two separate values: (1) the lines required for the netlists of the circuit under design and the test jigs (about the number of lines that would be required to simulate the circuit with SPICE), and (2) the lines for the independent variables and performance specifications (lines specific to ASTRX/OBLX). In all, the amount of time and effort required to create an ASTRX/OBLX input description is quite modest. Table 1 also includes the number of independent node voltage variables added by ASTRX to codify the relaxed-dc formulation. Because the device models we employ contain internal nodes, these added variables typically outnumber the user-specified variables. Finally, Table 1 shows the size of the linearized small-signal test jig circuit(s) generated by ASTRX to be evaluated by AWE for each new circuit configuration and the size of the bias circuits generated by ASTRX using the large-signal models for the non-linear devices.

The schematics for our benchmark circuits are shown in Fig. 4 and the basic synthesis results for this suite are in Table 2. CPU times given are on an IBM RS/6000-550. It is impossible to compare directly circuit performance of these ASTRX/OBLX synthesized circuits with that of circuits synthesized with other tools because of the unavailability of device model parameters to describe the processes for which they were designed. As a result, we compare the *accuracy* of ASTRX/OBLX to that of previous approaches. Because of the simplifications made during circuit analysis, results from equation-based synthesis tools differ from simulation by as much as 200% (see Fig. 3). In contrast, for the small-signal specifications where AWE predicts performance, ASTRX/OBLX results match simulation almost exactly. By simulating linearized versions of these circuits we have determined that these minor differences are due to differences between the models

TABLE 1. RESULT OF ASTRX'S ANALYSES

		Circuit						
		Simple OTA	**OTA**	**Two-Stage**	**Folded Cascode**	**Comparator**	**BiCMOS Two-Stage**	**Novel Folded Cascode**
Input (lines)	Netlist/Models	30	34	43	65	131	39	68
	Synth. Specific	28	33	40	56	68	33	51
x	User-Supplied	7	11	19	28	19	12	27
	Node Voltages	14	24	26	70	57	26	84
$C(x)$	Terms	56	85	88	212	169	86	246
	Lines of C	1443	1809	1894	3408	3088	1723	3960
Circuits	Circuit Type[a]: nodes, elements	B: 20, 31 A: 20, 67	B: 28, 49 A: 29, 114	B: 34, 54 A: 33, 118	B: 75, 138 A: 75, 324	B: 65, 126 A: 63, 265 A: 64, 266 A: 29, 115	B: 33, 54 A: 32, 105	B: 90, 167 A: 90, 395

a. Type 'A' is a linearized, small-signal AWE circuit. Type 'B' is a bias circuit.

TABLE 2. BASIC SYNTHESIS RESULTS, BSIM AND GUMMEL-POON MODELS, 1.2μ PROCESS

	Specification: OBLX / Simulation				
Attribute	**Simple OTA**	**OTA**	**Two-Stage**	**Folded Cascode**	**BiCMOS Two-Stage**
Cload (pF)	1	1	1	1.25	1
Vdd	5	5	5	5	5
dc gain (dB)	↑[a]: 36.6 / 36.6	↑: 40.4 / 40.2	≥60: 66.4 / 66.4	≥70: 70.1 / 70.1	↑: 99.1 / 99.1
gain bandwidth (MHz)	≥50: 50.1 / 50.6	≥25: 25.0 / 25.4	≥10: 10.6 / 10.6	↑: 72.4 / 72.1	≥50: 73.7 / 75.1
phase margin (°)	≥60: 71.4 / 74.8	≥45: 57.9 / 57.8	≥45: 87.3 / 86.5	≥60: 80.0 / 80.0	≥45: 45.2 / 49.6
PSRR (Vss)	≥20: 21.9 / 21.9	≥40: 42.1 / 42.0	≥20: 31.0 / 30.9	≥105: 107 / 107	≥60: 78.9 / 79.0
PSRR (Vdd)	≥20: 36.8 / 36.8	≥40: 52.8 / 52.8	≥40: 45.8 / 45.8	≥105: 125 / 125	≥40: 52.2 / 52.2
output swing (V)	≥2.3: 3.7 / 3.6	≥2.5: 4.0 / 4.0	≥2: 2.7 / 2.8	≥±1.0: ±1.5 / ±1.5	≥2: 3.3 / 4.0
slew rate (V/μs)	≥10: 130 / 131	≥10: 51.6 /48.2	≥2: 3.8 / 4.0	≥50: 67 / 57	≥10: 10 / 9.5
active area ($10^3\mu^2$)	↓: 2.8	↓: 0.9	↓: 2.1	↓: 46	↓: 11.9
static power (mW)	≤1: 0.72 / 0.72	≤1: 0.33 / 0.34	≤1: 0.16 / 0.16	≤15: 10 / 10	≤20: 1.3 / 1.5
time/ckt. eval (ms)	36	37	38	116	38
CPU time (min. / run)	6	9	16	120	12

a. ↑ means maximize, while ↓ means minimize.

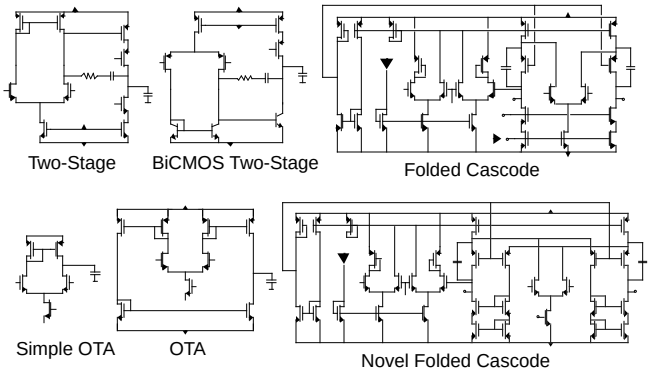

Fig. 4: Circuit Schematics

used during simulation with HSPICE [23] and our models adopted from Berkeley SPICE.

Presently ASTRX/OBLX employs 3 different encapsulated device models: the level 3 MOS model from SPICE, the BSIM MOS model [24] and the Gummel-Poon model for BJT devices. To demonstrate the importance of supporting different models, we synthesized the same circuit (Simple OTA) with three different model/process combinations: BSIM/2µ, BSIM/1.2µ, and MOS3/1.2µ. ASTRX/OBLX was given (and achieved) the same specifications for each but was told to minimize active area. As expected, the BSIM/2µ design required the largest area ($580\mu^2$). But, surprisingly, the two designs for the *same* 1.2µ process also differed substantially in area: $300\mu^2$ for BSIM and $140\mu^2$ for MOS3. Clearly the choice of device model greatly affects circuit performance prediction accuracy.

The final two circuits in our benchmark suite show the ability of ASTRX/OBLX to design high-performance and novel circuit topologies. The first such circuit, a BiCMOS two-stage amplifier, shows the ability of ASTRX/OBLX to handle a mix of MOS and bipolar devices. Synthesis and simulation results appear as the last column of Table 2.

Our final design, a novel folded cascode fully differential opamp, is a new high-performance design recently published in [25] and as such is a significant test for any synthesis tool because the performance equations cannot be looked up in a textbook. Moreover, the performance of the circuit is difficult to express analytically, and as many as 6 poles and zeros may non-trivially affect the frequency response near the unity gain point. Table 3 is a comparison of a redesign of this circuit using ASTRX/OBLX with the highly optimized manual design for the same 2µ process. Surprisingly, ASTRX/OBLX finds a design with higher nominal bandwidth at the cost of less area. Although we are pleased with the ability of OBLX to find this corner of the design space, this does not mean that ASTRX/OBLX out-performed the manual designer. In fact, the manual designer was willing to trade *nominal* performance for better estimated yield and performance over varying operating conditions. Adding this ability to ASTRX/OBLX is one of our highest priorities for future effort.

TABLE 3. COMPARISON WITH MANUAL DESIGN FOR CIRCUIT NOVEL FOLDED CASCODE

Attribute	Manual Design	Automatic Re-Synthesis Spec:	OBLX / Sim
Cload (pF)	1		1
Vdd (V)	5		5
dc gain (dB)	71.2	≥ 71.2:	82 / 82
gain bandwidth (MHz)	47.8	↑[a]:	89 / 89
phase margin (°)	77.4	≥ 60:	91 / 91
PSRR (Vss)	92.6	≥ 93:	112 / 112
PSRR (Vdd)	72.3	≥ 73:	77 / 77
output swing (V)	± 1.4	± 1.4:	± 1.4 / ± 1.3
slew rate (V/µs)	76.8	≥ 76:	92 / 87
active area ($10^3\mu^2$)	68.7	↓:	56
static power (mW)	9.0	≤ 25.0:	12 / 12
time/ckt. eval (ms)			83
CPU (min. / run)			116

a. ↑ means maximize, while ↓ means minimize.

VII. CONCLUSIONS

We have presented ASTRX/OBLX, tools that accurately size high-performance analog circuits to meet user-supplied specifications but do not require prohibitive preparatory effort for each new circuit topology. For a suite of benchmark analog circuits that covers nearly all previously published synthesis results, we have validated our formulation by showing that ASTRX/OBLX requires several orders of magnitude less preparatory effort yet can predicts results more accurately.

Acknowledgment: This work was funded by the Semiconductor Research Corporation and the National Science Foundation.

REFERENCES

[1] E.S. Ochotta, R.A. Rutenbar, and L.R. Carley, "Equation-Free Synthesis of High-Performance Linear Analog Circuits," *Proc. 1992 Brown/MIT Conf.*, The MIT Press.
[2] E. Berkcan, *et al.*, "Analog Compilation Based on Successive Decompositions," *Proc. of the 25th IEEE DAC*, pp. 369-375, 1988.
[3] M. Degrauwe *et al*., "Towards an analog system design environment," *IEEE JSSC*, vol. sc-24, no. 3, June 1989.
[4] G. Gielen, *et al.*, "Analog circuit design optimization based on symbolic simulation and simulated annealing," *IEEE JSSC*, vol. 25, June 1990.
[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] J. P. Harvey, *et al.*, "STAIC: An Interactive Framework for Synthesizing CMOS and BiCMOS Analog Circuits," *IEEE Trans. CAD*, Nov. 1992.
[7] 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.
[8] P. C. Maulik, L. R. Carley, and R. A. Rutenbar, "A Mixed-Integer Nonlinear Programming Approach to Analog Circuit Synthesis," *Proc. DAC*, pp. 693-703, June 1992.
[9] B.J. Sheu, *et al.*, "A Knowledge-Based Approach to Analog IC Design," *IEEE Trans. Circuits and Systems*, CAS-35(2):256-258, 1988.
[10] F. V. Fernandez, *et al.*, "Accurate Simplification of Large Symbolic Formulae," *Proc. IEEE ICCAD*, pp. 318-321, Nov. 1992.
[11] R.A. Rohrer, "Fully Automatic Network Design by Digital Computer, Preliminary Considerations," Proc. IEEE vol. 55, Nov. 1967.
[12] W. Nye, *et al.*, "DELIGHT.SPICE: an optimization-based system for the design of integrated circuits," *IEEE Trans. CAD*, vol. 7, April 1988.
[13] R.A. Rohrer, *et al.*, "AWE Inspired," *Proc. IEEE CICC*, May 1993.
[14] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, 13 May 183.
[15] F. Romeo and A. Sangiovanni-Vincintelli, "A Theoretical Framework for Simulated Annealing", *Algorithmica (1991), 6: 302-345.*
[16] J. M. Cohn, *et al.*, "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing," *IEEE JSSC*, vol. 26, no. 3, March, 1991.
[17] J. Lam and J.M. Delosme, "Performance of a New Annealing Schedule," *Proc. 25th ACM/IEEE DAC*, pp. 306-311, 1988.
[18] E.S. Ochotta, "Synthesis of High-Performance Analog Cells in ASTRX/OBLX," Ph.D thesis, Carnegie Mellon University, 1994.
[19] S. Hustin and A. Sangiovanni-Vincentelli, "TIM, a new standard cell placement program based on the simulated annealing algorithm", presented at IEEE Physical Design Workshop on Placement and Floorplanning, Hilton Head, SC, April 1987.
[20] S.B. Gelfand and S.K. Mitter, "Simulated Annealing Type Algorithms for Multivariate Optimization," *Algorithmica (1991), 6: 419-436.*
[21] W. Swartz and C. Sechen, "New Algorithms for the Placement and Routing of Macrocells," *Proc. IEEE ICCAD*, pp. 336-339, Nov. 1990.
[22] E.S. Ochotta, L.R. Carley, and R.A. Rutenbar, "Analog Circuit Synthesis for Large, Realistic Cells: Designing a Pipelined A/D Converter with ASTRX/OBLX," *Proc. CICC*, May 1994.
[23] Metasoft Corp. HSPICE manual, 1990.
[24] B. Sheu, *et al*., "BSIM: Berkeley short-channel IGFET model for MOS transistors," IEEE JSSC, vol. sc-22, no. 4, Aug. 1987.
[25] K. Nakamura and L.R. Carley, "A current-based positive-feedback technique for efficient cascode bootstrapping," *Proc. VLSI Circuits Symposium*, June 1991.