Trustworthy Genetic Programming-Based Synthesis of Analog Circuit Topologies Using Hierarchical Domain-Specific Building Blocks

Trent McConaghy, *Member, IEEE,* Pieter Palmers, *Member, IEEE,* Michiel Steyaert, *Fellow, IEEE,* and Georges G. E. Gielen, *Fellow, IEEE*

***Abstract*—This paper presents MOJITO, a system that performs structural synthesis of analog circuits, returning designs that are trustworthy by construction. The search space is defined by a set of expert-specified, trusted, hierarchically-organized analog building blocks, which are organized as a parameterized context-free grammar. The search algorithm is a multiobjective evolutionary algorithm that uses an age-layered population structure to balance exploration versus exploitation. It is validated with experiments to search across >100 000 different one-stage and two-stage opamp topologies, returning human-competitive results. The runtime is orders of magnitude faster than open-ended systems, and unlike the other evolutionary algorithm approaches, the resulting circuits are trustworthy by construction. The approach generalizes to other problem domains which have accumulated structural domain knowledge, such as robotic structures, car assemblies, and modeling biological systems.**

***Index Terms*—Analog, design automation, evolutionary algorithm (EA), integrated circuit (IC), multiobjective optimization.**

I. Introduction

THE AUTOMATED design of structures such as robotic structures, car assemblies, and circuit topologies has attracted much attention in the evolutionary computation literature, in part because evolutionary algorithms (EAs) like genetic programming (GP) [1] handle the non-vector search spaces of structural design problems more naturally than classical optimization algorithms.

We herein briefly focus on challenges in analog circuit topology (structure) design; we will see that these challenges span other domains as well. In analog topology design, the aim is to generate a graph-like structure of devices (resistors, capacitors, transistors), which are connected by wires, to realize a target electrical behavior. Each device can have sizes (resistance, capacitance, length and width).

The designer has many challenges. First, he or she must design quickly; typically within a time budget. Second, the sized topology should have minimal risk of failure upon manufacturing, because it costs millions of dollars to fabricate and test a design on modern semiconductor processes [2]. A failed design means that another round of design, manufacturing and test is needed, affecting time-to-market and profitability. Third, the designer must either meet target specifications for power, performance, or area (if specifications exist); or more typically, he or she must choose a design based on what the specification tradeoffs. This is hard because different topologies have different advantages, those advantages change depending on the specific manufacturing process, and there are thousands of possible topologies.

To reduce risk of failure, designers typically use building blocks and topologies that are trustworthy by construction. We now explain. A trustworthy analog block (including connections) is one that has been successfully manufactured in silicon and tested. Simulation is not enough because the simulator may not properly model effects[1] that turn out to be critical. A topology is a hierarchical composition of blocks and connections; it is trustworthy by construction if its blocks and connections are trustworthy. The analog design field has accumulated a large number of trusted blocks over the years [3], [4], from differential pairs to bias generators. Combining these blocks in various ways allows a large number of possible trustworthy topologies. Physical realization in the chosen process technology is the ultimate arbiter whether the circuit works, but since manufacturing is so costly, designs must typically pass the "trustworthy" gate before being considered. If the designer trusts the topology, variation is managed, and physical layout is done competently, then the designer can expect the manufactured design to behave similar to simulation.

Designers only try novel designs when the trusted design does not meet the area, power, and performance needs. A novel

Manuscript received December 4, 2008; revised August 24, 2009 and March 15, 2010; accepted November 1, 2010. Date of publication January 24, 2011; date of current version July 29, 2011. This work was supported by Solido Design Automation, Inc., IWT/Medea+ Uppermost, and FWO Flanders. This paper was recommended by Associate Editor G. Greenwood.

T. McConaghy was with Katholieke Universiteit Leuven, B-3000 Leuven, Belgium. He is now with Solido Design Automation, Inc., Saskatoon, SK S7N 3R3, Canada (e-mail: trent_mcconaghy@yahoo.com).

P. Palmers was with ESAT-MICAS, Katholieke Universiteit Leuven, B-3000 Leuven, Belgium. He is now with Mephisto Design Automation, Leuven 3001, Belgium (e-mail: pieter.palmers@gmail.com).

M. Steyaert and G. G. E. Gielen are with Katholieke Universiteit Leuven, B-3000 Leuven, Belgium (e-mail: michiel.steyaert@esat.kuleuven.be; georges.gielen@esat.kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2010.2093581

[1]Such effects include temperature variations, process variations, load variations, unwanted parasitic capacitances and resistances, electromigration effects, packaging effects, leakage effects, and proximity effects. The simulator itself could fail due to nonconvergence or poor device models.

analog topology has one or more novel building blocks. Being novel, these blocks have not been verified in silicon. Therefore, such blocks and topologies pose a higher risk of failure. A novel analog building block is not trusted until it has been fabricated and tested. To minimize risk, the designer typically starts from the best-performing trustworthy design, and adds just enough novelty to meet the performance goals.

To meet the challenges of time constraints, manufacturing risk, and quality design, the designer needs a way to quickly evaluate a broad set of trusted topology options. Our aim is a computer-aided design (CAD) tool to assist the designer in doing this, which fits in industrial design flows for modern semiconductor processes.

Analog design has similar characteristics to other problems in engineering and scientific discovery: the aim is structural design/discovery, and the problem domain has an accumulation of trusted building blocks which can be exploited in the design/discovery process. Some problems, like design of engines or bridges, have further similarities: costly to thoroughly test, higher risk of failure with novel structures, and a multiobjective nature.

This paper presents MOJITO, an EA-based approach that targets such problems. To reliably return trusted designs, MOJITO uses trusted building blocks, hierarchically connected in trusted ways. This is in contrast to previous EA analog synthesis approaches which do not restrict search to trusted designs (see Section II), and therefore risk failure after manufacturing. MOJITO performs multiobjective search, returning a Pareto-optimal set of topologies with associated device sizes. MOJITO stands for multi-objective and topology sizing.
Novel contributions of this paper are as follows.

1) A framework to define a structural synthesis search space that is trustworthy by construction, by reusing the domain knowledge of trustworthy building blocks for the field. Specifically, it uses trusted analog blocks as terminal and nonterminal symbols in a parameterized, context-free grammar. The search space is all the possible derivations and parameterizations of the grammar using the root symbol, terminal symbols and nonterminal symbol. A candidate individual is a sentence within the grammar, representing a candidate topology and device sizes.
2) A novel EA-based search algorithm. It combines a multiobjective EA (MOEA) with one that avoids premature convergence by grouping individuals by age (ALPS) [5]. Some search operators traverse the grammatical space with a tree-based view (like [6]); other operators have a vector-based view (like [7]). It avoids undesirable stealth mutations [8] by examining the phenotype for gene expression before fitness evaluation. A novel multiobjective selection mechanism, TAPAS, preserves topology diversity when searching $\gg$2 objectives.

The rest of this paper is organized as follows. Section II reviews past work in analog synthesis, and Section III introduces MOJITO. Sections IV and V describe the MOJITO search space and search algorithm. Section VI presents experimental results to validate the approach. Section VII concludes.

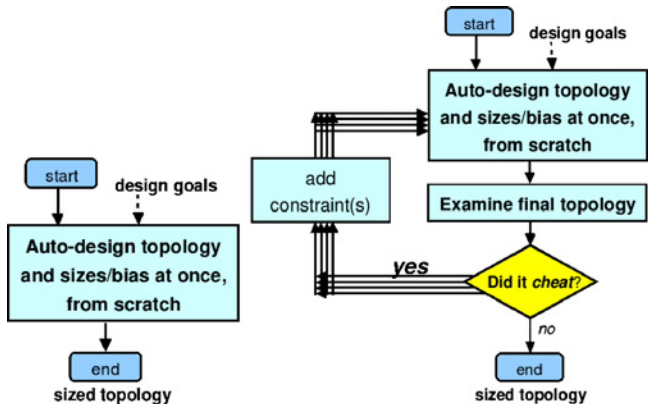


Fig. 1. Left: ideal designer flow when using open-ended GP-based topology synthesis. Right: actual flow when using open-ended GP, which includes iterations of adding constraints. The multiple feedback lines illustrate that there can be many iterations. “Did it cheat” is affirmative if an expert analog designer can identify an issue via inspection or further tests.

II. Previous Analog Synthesis Approaches

This section reviews past approaches to analog structural synthesis, from both EA and analog CAD literature.

GP [1] has a natural ability to handle search spaces with tree-like and graph-like structures (topologies). Accordingly, many approaches [9]–[24] use variants of GP, for the target design flow of Fig. 1 (left). It is simple, taking just the design goals as inputs, and producing a sized topology. GP is given a set of devices (transistors, resistors, and so on) that it can connect in arbitrary ways without rules—all the building blocks are “invented” (or reinvented) from scratch. This is what gives it the open-ended nature. No topology information is input.

The early approaches such as [10], [15], and [18] had few constraints. Any combination of devices and connections was allowed, with fitness measured via a SPICE circuit simulator. While synthesized circuits were optimal according to the simulator, they did not look like human-designed circuits, and sometimes had obvious flaws such as dangling resistors. Such circuits can be found in early papers like [9], up to recent papers like [24]. In contrast to novel manual designs, these computer-generated novel designs were not accompanied by an explanation for the behavior of the design.

Researchers took note of the unfamiliar-looking circuits, and proceeded to iteratively add constraints to block out unwanted behaviors. Fig. 1 (right) illustrates the revised flow. This includes [16], [21], and [22], which added tighter constraints using domain knowledge to improve efficiency and get more palatable-appearing circuits. The key problem is that novel building blocks could still be generated, which are not trustworthy because they have not been verified in silicon.

To overcome the trust issue, another EA approach inspired by [25] replaced the SPICE simulator with reconfigurable analog circuits (e.g., [26]). However, since such circuits have different silicon implementations than application-specific circuits, they have different issues; a functioning evolved reconfigurable block does imply trust in an application-specific block. Also, because reconfigurable analog circuits have significantly worse performance than application-specific analog

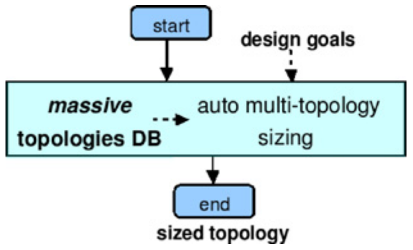


Fig. 2. Proposed flow for trustworthy-by-construction structural synthesis, using MOJITO. A small set of hierarchically organized, pre-specified, and field-specific building blocks combine to create a large set of possible topologies (structures). MOJITO uses GP to search across possible building block combinations (topologies) in a hierarchy-aware fashion.

circuits, they are rarely used in industrial system-on-chip design flows [2]. Our aim is to support designers in these flows.

The CAD literature [27] has approached the analog synthesis problem in different ways. Some approaches [28]–[33] used pre-defined rule-based reasoning or abstract models having transforms to well-known circuit structures. Due to using well-known circuit structures, the designs are trusted. However, the tools required up-front setup efforts of weeks to months, which must be repeated for each circuit type and each new process node, both of which are highly unpalatable for industrial usage. DARWIN [34] and MINLP [35] give trustworthy circuits by predefining a space of designer-known circuit topologies within a fixed-length vector, where variables enable/disable/choose components or subblocks.[2] Unfortunately the approaches rely on a flat definition of the search space specific to the circuit type; they do not show a clear path to generalize or are restricted to <100 topologies.

III. MOJITO Flow

Fig. 2 shows the proposed MOJITO flow. It has the same simple flow as the ideal open-ended GP approaches, but it returns topologies that are trustworthy by construction. This is possible by searching across a database (DB) of pre-defined trusted topologies. This set is sufficiently large that the designer does not have to intervene in a typical design problem. MOJITO's other inputs relate to measuring objectives and constraints, via testbenches which specify the circuit analysis and measures for specific performances, and simulator model files describing how devices like transistors behave for a particular semiconductor process node.

The challenges are how to specify a large topologies DB, and how to search it; Sections IV and V elaborate on this.

IV. MOJITO Search Space

This section describes the framework for a synthesis search space having a large number of possible topologies, which are trustworthy by construction. It also describes an exemplary set of operational amplifier topologies.

[2]There is little conceptual difference between DARWIN and MINLP; the differences lie in search space implementations and choice of search algorithm.

A. Search Space Framework

The search space is the possible derivations of a parameterized, context-free grammar [6]. That is, there is a set of nonterminal and terminal symbols. Each nonterminal has a set of possible derivation rules, where each derivation is an expansion into a list of symbols. The symbols are parameterized in the same fashion that computer-language functions are parameterized: they input parameters which can be propagated to lower-level symbols. One nonterminal is the root symbol.

Trusted analog circuit building blocks are the grammar's symbols. Nonterminal symbols (compound and flexible blocks, defined shortly) can expand according to one or more derivation rules (the possible sub-blocks). The specific building blocks are taken from analog design textbooks [3], [4]. This grammar defines a space of possible topologies and associated device sizings; a sentence in the grammar is a single sized topology.

Equivalently, the possible topologies DB is composed of hierarchically organized blocks. Some sub-blocks do not expand (terminal symbols), and other sub-blocks do expand (nonterminal symbols). The highest-level block (root symbol) expresses any grammatically valid composition of blocks and therefore covers the space of all possible topologies (sentences). We take the block-based perspective in the following description, because it is easier to relate to analog circuit building-block domain knowledge.

Each block has an interface which includes parameters, and external ports (nodes). When a block expands into a sub-block, each sub-block's parameters are a function of its parent block's parameters. A specific block is instantiated when its parameters and external ports have been specified. Therefore, a specific sized topology is instantiated when the parameters and external ports for the highest-level block have been specified.

The DB has three types of blocks: an atomic block (terminal symbol), a compound block (nonterminal with 1 expansion option), and a flexible block (noterminal with >1 option).

1) *Atomic block.* Have no embedded blocks. Being leaf nodes in the building block hierarchy, it is only these blocks that are expressed in the final sized topology (netlist). Fig. 3 gives examples.
2) *Compound block.* These have one or more sub-blocks embedded. Sub-blocks can have internal connections among themselves and to the block's external ports. Fig. 4 gives examples.
3) *Flexible block.* These have the special topological parameter *chosen_part_index*, which, during netlisting, is used to select one of several candidate embedded blocks and respective wirings. Example: a current mirror which may be simple or cascode (*chosen_part_index* = 0 or 1). Fig. 5 gives an example.

Larger blocks are hierarchically composed from smaller blocks, all the way to the top-level block. A hierarchically composed set of blocks that includes flexible blocks allows for >1 possible topologies. Therefore, the top-level block (e.g., operational amplifier) defines the DB of possible topologies and sizings (e.g., possible operational amplifiers). The search space for a given block is the combination of possible values

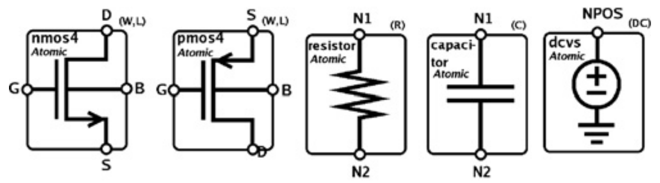


Fig. 3. Example atomic blocks: nmos4 transistor, pmos4 transistor, resistor, capacitor. Example of ports and input parameters: nmos4 has four external ports: *G*, *D*, *S*, and *B*; it has two input parameters, *W* and *L*. Note how DC-controlled voltage source (dcvs) has only one external port; the other port ties directly to ground.

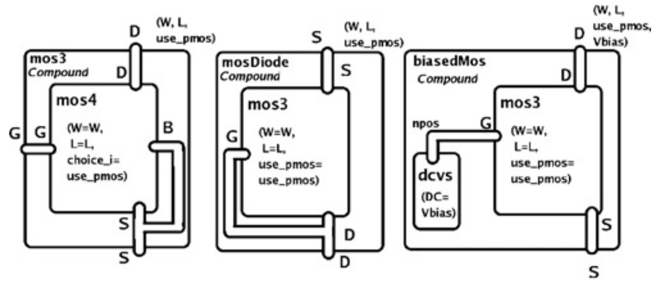


Fig. 4. Example compound blocks. mos3 is a wrapper for mos4, so that the mos4's "B" node is not seen at higher levels. mosDiode ties together two internal ports to only present two external ports. biasedMos uses a 1-port dc-controlled voltage source (dcvs) block to set its gate bias internally.

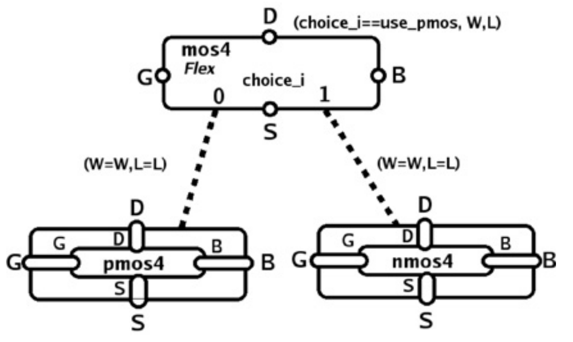


Fig. 5. Example flexible block: mos4 turns the choice of NMOS versus PMOS into a parameter "*chosen_part_index*." Note how parameters get assigned from mos4 to either of its sub-blocks. In this case, both sub-blocks use the mos4's W and L parameters as their own W and L values.

that each of the block's interface parameters and nodes can take. Therefore, the overall search space is the combination of possible values for the top-level block's parameters and nodes. The blocks can be specified in one of many forms: as a text-based grammar, an analog hardware description language, a circuit schematic editor, or a programming language. We use Python [36], where each block is a different method.

B. Search Space Locality

A key aim in designing the search space's building blocks is to make it as easy to search as possible. Good locality means that small changes to the genotype lead to expected small changes in performance (objective function). Good locality is important for an effective search algorithm [8].

In analog circuits, there is a complication to achieving locality. The genotype is expressed into a sized topology, which is then expressed as electrical behavior (via simulation), which is then measured for performance (fitness). Locality

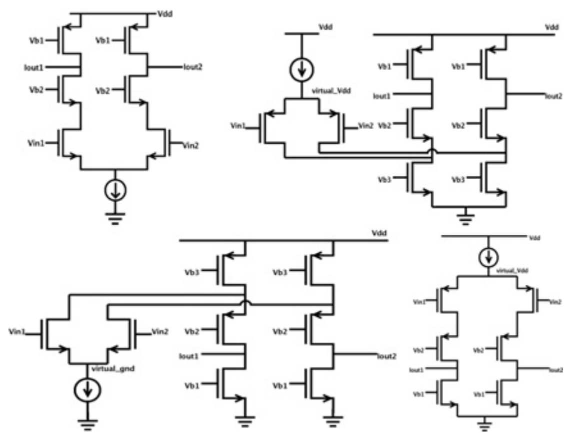


Fig. 6. Four circuits with the same qualitative electrical behavior, but dramatically different topologies. The circuits on the left have PMOS inputs, and on the right have NMOS inputs ($input\_is\_PMOS$ = true/false). The load's rail is *vdd* in the top row, and *gnd* in the bottom row ($loadrail\_is\_vdd$ = true/false).

is easy to achieve if there is a small change in each step of expression. However, in analog circuits, very different building blocks can express the same qualitative electrical behavior at their interface. That is, the same design intent can be captured with different implementations. For example, Fig. 6 shows four circuits. These circuits have the same qualitative electrical behavior: differential amplification from input signal to output signal. However, as the figure shows, they have very different structures. There are many such examples in analog circuit design [3], [4].

Past GP synthesis approaches (Section II) do not handle this, because they do not capture design intent of different building blocks. To illustrate, a GP tree from a past approach does not have a means to convert among the four topologies of Fig. 6 via small genotype changes. Instead, major genotype changes would be needed for this small change to fitness.

Our approach preserves locality in the mapping from genotype to electrical behavior, ignoring the degree of changes to the sized topology. We allow large changes to the topology, even when there are small changes to genotype and electrical behavior. We use this approach at the overall topology level, and within sub-blocks of the topology.

We implement our approach via carefully-designed building blocks. Specifically, we use a flexible block as the qualitative target electrical behavior, and its different possible sub-blocks as the structures implementing the target behavior. For example, a flexible block for the electrical behavior of "differential amplification" can expand into one of the four blocks in Fig. 6. A small change to the genotype will not change the qualitative behavior—they all do differential amplification—but that genotype change could choose a different structure to implement the differential amplification.

C. Example of an "Individual"

The search space is a set of circuit building blocks, and is equivalently all derivations of a parameterized grammar. Therefore, a point in the search space is a circuit, and equivalently is a sentence. In EA terms, it is an individual.

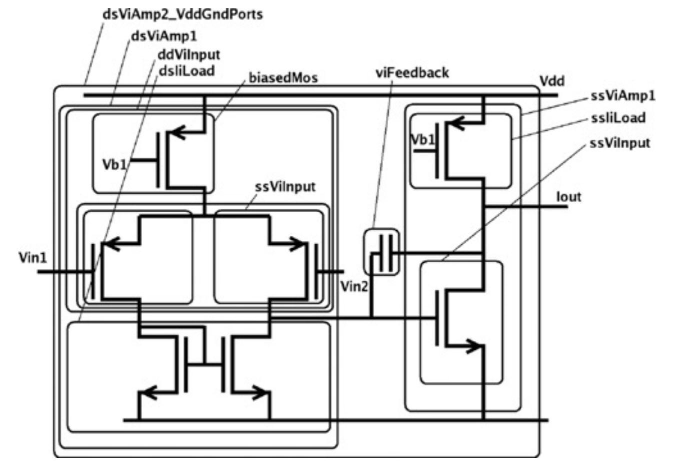


Fig. 7. Example individual ("PMOS-input Miller OTA") shown in schematic form, annotated with MOJITO building blocks.

TABLE I
EXAMPLE INDIVIDUAL: VALUES FOR TOPOLOGY CHOICE VARIABLES

Variable Name	Value
chosen_part_index	1
stage1_loadrail_is_vdd	0
stage1_input_is_pmos	1
stage1_degen_choice	0
stage1_inputcascode_is_wire	1
stage1_inputcascode_recurse	0
stage1_load_chosen_part_index	1
stage2_loadcascode_recurse	0
stage2_load_part_index	0
stage2_inputcascode_is_wire	1
stage2_loadrail_is_vdd	1
stage2_input_is_pmos	0
stage2_degen_choice	0
stage2_inputcascode_recurse	0

This section illustrates the different ways one can view an individual's structure.

We begin with the schematic. It is a graphical depiction how the devices are connected in the topology, with domain-specific symbols for each device type. It concretely defines the circuit, yet is intuitive to analog designers. Fig. 7 is an example individual shown in schematic form. From an analog designer perspective, this schematic is a "PMOS-input Miller OTA" with two stages, the first stage having differential PMOS inputs and simple current mirror load, second stage having single-ended NMOS input and output with single-device PMOS load, and Miller-style feedback with one capacitor.

The schematic is annotated to illustrate its hierarchical composition of the circuit blocks. A choice has been made for each flexible block. The root node is dsViAmp2_VddGndPorts, (as indicated in the top left corner of the schematic). Its sub-blocks are dsViAmp1 (first stage), ssViAmp1 (second stage), and viFeedback (Miller feedback capacitor), which subdivide further until Atomic blocks (leaf nodes) like nmos4, pmos4, and capacitor.

The individual can be represented as a tree, or as a vector of parameter values. The vector's parameters are those needed

TABLE II
EXAMPLE INDIVIDUAL: VALUES FOR DEVICE-VALUE VARIABLES

Variable Name	Value
feedback_C	1.1883e-11 F
stage1_Ibias	0.016297 A
stage1_Ibias2	0.015746 A
stage1_Vds_internal	1.4584-V
⋮	⋮ (30 more)

to instantiate the root block, "ds amp vdd/gnd ports." Some parameters are for topology choices (*chosen_part_index*), and others are for setting specific device values (Is and Vs which translate to Ws and Ls). Tables I and II give the example individual's topology choice values and device-setting values, respectively.

Each parameter in Table I relates to one of the flexible block choices. The parameter *chosen_part_index* decides between one and two stages. A value of 1 indicates that two stages were chosen; this is confirmed by the two-stage schematic in Fig. 7. *stage1_loadrail_is_vdd* = 0 means that stage 1's loadrail is not set to *vdd*, but to *gnd* instead, as already observed. And so on. Note that some variables may be ignored, depending on values of other variables, e.g., if *chosen_part_index*=0 to choose a one-stage topology, then all variables related to the second stage have no effect (i.e., neutrality, see Section V-B).

Table II gives an example of device-setting values. These are parameters which do not affect the topology. Parameters are Is and Vs, not Ws and Ls because an operating-point driven formulation is used [37], where Is and Vs get translated into Ws and Ls at the level of NMOS4 and PMOS4 netlisting.

A final view of an individual is the SPICE netlist. It is a text-based listing, where each line gives a device's connections, type (e.g., resistor), and parameters (e.g., resistance). This is the form used as input to the SPICE circuit simulator, to estimate the individual's performance values.

D. Size of Search Space

These building blocks were used for the design of an opamp topologies DB, integrated into MOJITO. It allows for: one-and two-stage amplifiers, PMOS versus NMOS loads, PMOS versus NMOS inputs, stacked versus folded cascode versus non-cascode inputs, cascode versus non-cascode versus resistor loads, level shifting, different current mirrors, single-ended and differential inputs, and single-ended outputs. Thirty building blocks hierarchically combine to allow 3528 different topologies, counted via simple bottom-up combinatorial counting rules.[3] The space was further extended by supporting symmetrical operational transconductance amplifiers (OTAs), as Fig. 8 (left) shows, by adding current mirror folding blocks. Support was also added for cascoding

[3]These are the topology-counting rules: the count for an atomic block is one; for a flexible block, it is the sum of the counts of each choice block; for a compound block, it is the product of the counts of each of its sub-blocks. But there are subtleties. Subtlety: for a given choice of flexible block, other choice parameters at that level may not matter. Example: if a one-stage amplifier is chosen, do not count choices related to second stage. Subtlety: one higher-level choice might govern >1 lower-level choices, so do not overcount. Example: a two-transistor current mirror should have two choices (NMOS versus PMOS), not four (NMOS versus PMOS x 2).

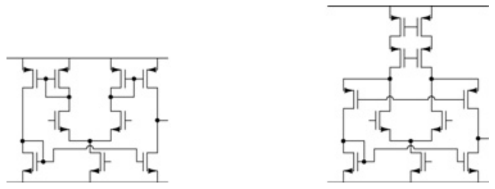


Fig. 8. Left: symmetrical OTA. Right: folded cascoded folder OTA.

TABLE III

SIZE OF OPAMP TOPOLOGY SPACES

Technique	# Topologies	Trustworthy?
GP without reuse, e.g., [18]	$\gg$ billions	NO
DARWIN [34]	24	YES
MINLP [35]	64	YES
GP with reuse: MOJITO	3528 and 101 904	YES

of folding transistors, as Fig. 8 (right) shows. These changes increase the count to 101 904 possible topologies. Table III compares the topology count for different synthesis approaches. It shows that the MOJITO space has ≈1500x more trusted topologies compared to previous work [34], [35]. Topologies can have up to 15 devices: 11 devices for the first stage (e.g., Fig. 8, right), and four devices for the second stage. It is possible to increase the number of components in a topology by adding more complex building blocks, though of course the larger search space will lower convergence rate.

We can compare computational effort for a naive approach that optimizes each topology, one at a time, then merges the results. Assuming that an optimization takes 2000 individuals on average, then 100 000 * 2000 = 200 million individuals are needed. In comparison, we show later that MOJITO does well with 100 000 individuals (2000x fewer).

A large set of options can qualitatively change the designer's perception of the process: rather than doing "selection" from a few dozen topologies, the tool is "synthesizing" the optimal combination of building blocks from a large set of possibilities. The available topologies are broad enough to meet performance aims for many real-world problems. Since the topologies DB only needs to be defined once for a given problem type (e.g., opamp), the designer does not need to view it as an input to the tool, not even if the process node changes.

V. MOJITO Search Algorithm

MOJITO search is a MOEA that uses an age-layered population structure (ALPS) [5] to balance exploration versus exploitation. Some operators traverse the grammatical space in a tree fashion (like [6]), and some operators traverse it in a vector fashion (like [7]) .

The algorithm's aim is formulated as a constrained multi-objective optimization problem

$$\begin{array}{rlll} \text{minimize} & f_i(\phi) & i = 1 ... N_f \\ \text{s.t.} & g_j(\phi) \le 0 & j = 1 ... N_g \\ & h_k(\phi) = 0 & k = 1 ... N_h \\ & \phi \in \Phi \end{array} \qquad (1)$$

where Φ is the "general" space of possible topologies and sizings. The algorithm traverses Φ to return a Pareto-optimal set $Z = \{\phi_1^*, \phi_2^*, \cdots, \phi_{N_{ND}}^*\}$ on N_f objectives, N_g inequality constraints, and N_h equality constraints. Each individual ϕ_l^* is Pareto-optimal. Without loss of generality, we can minimize all objectives and use ≤ 0 inequalities. By definition, a design ϕ is feasible if it meets all constraints: $\{g_j(\phi) \le 0\} \forall j$, $\{h_k(\phi) = 0\} \forall k$, $\phi \in \Phi$. By definition, all the designs in Z are nondominated, i.e., no design ϕ in Z dominates any other design in Z. A feasible design ϕ_a dominates another feasible design ϕ_b if $\{f_i(\phi_a) \le f_i(\phi_b)\} \forall i$, and $\{f_i(\phi_a) < f_i(\phi_b)\} \exists i$. We follow the dominance rules of NSGA-II [38] for handling constraints: a feasible design always dominates an infeasible design, and if two designs are infeasible, then the one with smallest constraint violation is considered dominant.

The next subsection describes the high-level search algorithm, and subsequent sections describe details.

A. High-Level Search Algorithm

We use an EA as the base of our search algorithm because EAs can readily search in a grammar-defined space [6], [7], perform constrained multiobjective optimization (e.g., [38]), and offer flexibility in overall algorithm design [39], [40].

A key issue with most EAs is premature convergence, where the algorithm finds a locally-optimal design and makes no further improvements. This is an issue in synthesis because some sub-blocks may get little chance to size properly before being filtered out via selection. We need to ensure an adequate supply of building blocks [41]. Possible tactics include very large populations like [11], larger mutation rate, larger mutation size, restarting like [42], preselection [43], crowding [44], and deterministic crowding [45]. Many MOEAs have some diversity preservation by including the Pareto set in selection. For example, NSGA-II [38] and SPEA2 [46] have limited-size Pareto archives that are directly included in the selection population; archive size is limited by deterministic crowding and clustering, respectively. The approach [47] has an unlimited-size Pareto archive, from which individuals are periodically selected for reuse in the main population. All these approaches, MOEA and otherwise, can prematurely converge because their genetic building block supply is from just the initial generation; if that material does not include the optimal material, then the run could fail.

Periodic injection of randomly-drawn individuals might help because it allows new genetic material to enter. However, random individuals compete poorly against existing individuals which have already had many generations to improve. To preserve random individuals until they are competitive, the technique of hierarchical fair competition [48] segregates individuals into fitness layers, and restricts competition to within layers. Unfortunately, near-stagnation can occur at some fitness levels because the best individuals per level have no competition.

To overcome this issue, the ALPS [5] segregates by genetic age instead of fitness. Each age layer P_k holds N_L individuals. By example, P_1 might allow individuals with age 0–19, P_2 allows age 0–39, and so on; the top level P_K allows age 0–∞. If an individual gets too old for a fitness layer, it gets removed from that layer. Genetic age is how many generations an individual's oldest genetic material has been around: the age

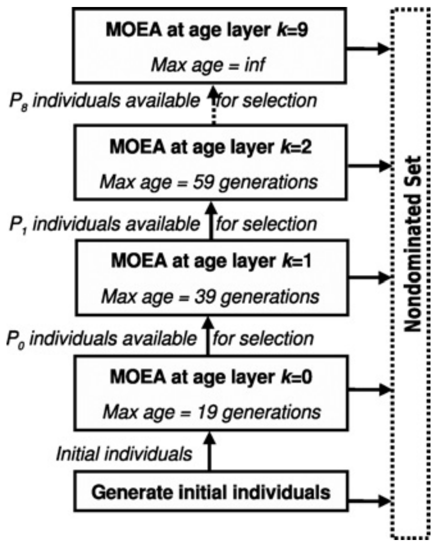


Fig. 9. Structure of MojitoSynthesis(), which uses a multiobjective age-layered population structure (ALPS).

TABLE IV
PROCEDURE MOJITOSYNTHESIS()

```
Inputs: $\Phi$, $N_a$, $K$, $N_L$
Outputs: $Z$
1.  $N_{gen} = 0$; $Z = \emptyset$; $P = \emptyset$
2.  while stop($N_{gen}$, ... ) $\neq$ True:
3.      if ($N_{gen} \% N_a$) = 0:
4.          if $|P| < K$:
5.              $P_{|P|+1} = \emptyset$
6.          $P_{0,i}$ = InitialCircuit($\Phi$), $i = 1...N_L$
7.      for $k$ = 1 to $|P|$:
8.          ($P_k$, $Z$) = OneMOEAGeneration($P_k$, $P_{k-1}$, $Z$)
9.      $N_{gen} = N_{gen} + 1$
10. return $Z$
```

of an initial individual is 0; the age of a child is the maximum of its parents' ages; age is incremented by 1 each generation. Selection at an age layer k uses the individuals at that layer k and layer $k-1$ as candidates, such that younger high-fitness individuals can propagate to higher layers. Every N_a ("age gap") generations, a new age layer may be added, and initial individuals enter layer $k = 0$ as random individuals.

MOJITO search uses ALPS. Fig. 9 and Table IV show MOJITO search structure and behavior, respectively.

We adapted ALPS to handle multiple objectives, via a MOEA at each age layer k (line 9 in Table IV). Whereas a canonical MOEA would select at just layer k, ALPS-based MOEA selection also considers the individuals from layer $k-1$. An external archive holding the Pareto-optimal set Z is maintained.[4] Stopping conditions (line 2) can include a maximum number of individuals $N_{ind,max}$ or a maximum number of generations $N_{g,max}$.

Table V shows the algorithm for the MOEA at each age layer R, for one generation. Note how individuals from the lower layer $k-1$ are imported for selection.

[4]The Pareto archive has no size limit. While the archive's information does not feed back to the search process, the MOEA search algorithm may maintain its own subset, e.g., as NSGA-II does.

TABLE V
PROCEDURE ONEMOEAGENERATION()

```
Inputs: $P_k$, $P_{k-1}$, $Z$
Outputs: $P'_k$, $Z'$
1. $P_{sel}$ = SelectParents($P_k \cup P_{k-1}$)
2. $P_{ch}$ = ApplyOperators($P_{sel}$)
3. $P_{ch}$ = Evaluate($P_{ch}$)
4. $P'_k = P_{sel} \cup P_{ch}$
5. $Z'$ = NondominatedFilter($Z \cup P_{ch}$)
6. return ($P'_k$, $Z'$)
```

The next three sections elaborate on the search operators, multiobjective selection, and generation of initial individuals.

B. Search Operators

1) *Trees Versus Vectors, and Neutrality:* Recall from Section IV that the search space is all derivations of a parameterized grammar. To traverse this space, we could use a grammar-based GP approach which operates on individuals as trees [6], or a grammar-based GP approach which operates on individuals as vectors [7]. A vector-oriented approach is the simplest, though it has issues by ignoring the hierarchy. First, change some of the individual's variables may not change the resulting sized topology at all, because those variables are in sub-blocks that are turned off. From the perspective of a search algorithm, this means that there is neutrality [49], which makes performance more unpredictable [8]. For EAs, another issue is that an n-point or uniform crossover operator could readily disrupt the values of the building blocks in the hierarchy, e.g., the sizes of some sub-blocks' transistors change while others stay the same, thereby hurting the resulting topology's likelihood of having decent behavior. From an EA perspective this means that the "building block mixing" is poor [41].

To overcome these issues, the search algorithm uses both tree-style and vector-style operators, and directly accounts for neutrality.

2) *Mutation Operator:* The mutation operator varies one or more parameters. Continuous-valued parameters follow Cauchy mutation [50] which allows for both tuning and exploration. Integer-valued *chosen_part_index* parameters follow a discrete uniform distribution. Other integer and discrete parameters follow discretized Cauchy mutations. The number of parameters to vary is drawn from the discrete density function of Table VI, which biases toward fewer variables, but sometimes allows significantly more variables to change.

3) *Crossover Operator:* Crossover works as follows: given two parent individuals, randomly choose a sub-block S in parent A, identify all the parameters associated with sub-block S, and swap those parameters between parent A and parent B. It is possible that S shares parameters with other sub-blocks that have the same higher-level block as S; and in that case the new parameters for S will affect those other sub-blocks.

There are cases where operator change a genotypes without affecting the phenotype; i.e., neutrality or stealth mutations operations can exist. Some researchers have found this to be beneficial to efficiency, while others have found it detrimental. To avoid unpredictable changes to search efficiency, MOJITO

TABLE VI
NUMBER OF VARIABLES TO MUTATE IS CHOSEN ACCORDING TO THE FOLLOWING DISTRIBUTION

# Variables to Mutate	Probability	Relative Bias
1	0.769	100
2	0.115	15
3	0.038	5
4	0.023	3
5	0.015	2
6	0.015	2
7	0.015	2
8	0.007	1
Sum	1.0	130

Probability of a specific # variables to mutate = (relative bias)/(sum of relative biases).

avoids stealth operations, as recommended by [8]. Specifically, in a mutation/crossover, the change is retained only if the phenotype (sized topology) changes. Mutation/crossover attempts are repeated until a phenotype change occurs. This does not typically impact synthesis runtime in practice, because the bottleneck is in fitness evaluation.

C. Handling Multiple Objectives

In a MOEA, the key challenge is how to select N_L parents P_{sel} from the candidate parents $P_k \cup P_{k-1}$ in a fashion that reconciles the multiple objectives. MOJITO uses two approaches: NSGA-II [38] which handles 2–3 objectives well, and TAPAS [51] for $\gg$2 objectives.

1) *Multiobjective Selection with 2–3 Objectives: NSGA-II:* NSGA-II [38] is a good starting point because it is relatively simple, reliable, well-studied, and can readily incorporate constraints. It performs selection in three steps. Its first step sorts the candidate parents into nondomination layers F_i, $i = 1...N_{ND}$, where F_1 is the nondominated set, F_2 is what would be nondominated if F_1 was removed, F_3 is what would be nondominated if $F_1 \cup F_2$ was removed, and so on, F contains all the candidates with no duplicates $F_1 \cup F_2 \cup \cdots F_{ND} = P_k \cup P_{k-1}$. NSGA-II's second step fills P_{sel}. It first adds all individuals from F_1, if they all fit, i.e., if $|P_{sel}| + |F_1| \leq N_L$. If there is space left, it then adds all individuals from F_2 if they all fit. If there is space left, it then adds all individuals from F_3 if they all fit. And so on. For the third NSGA-II step, once the P_{sel}-filling step reaches an F_i where not all of F_i's individuals can fit, then a subset of F_i's individuals needs to be chosen. NSGA-II chooses this subset as the individuals with the highest distance from other F_i individuals in the performance space, i.e., deterministic crowding [45].

Many analog design problems have $\gg$2–3 objectives [3]. Unfortunately, NSGA-II does poorly in this scenario [52]. The reason is that with many objectives, all of the population can be nondominated, i.e., there is just one nondomination layer $F_1 = P_k \cup P_{k-1}$. So, the whole population is selected using crowding, which biases toward the corners of the performance space; and not the center points which are close to all designs. That is, NSGA-II focuses on designs that are excellent on 1–2 objectives, yet poor at the rest.

TABLE VII
PROCEDURE ONEMOEA/DGEN()

```
Inputs: $P_k$, $P_{k-1}$, $Z$, $\boldsymbol{W}$
Outputs: $P'_k$, $Z'$
1.  $P'_k = P_k$
2.  for $i$ in $\{1, 2, \ldots, N_L\}$:
3.      $B = \{$best ind. according to $\boldsymbol{w_i}\} \cup$
              $\{$best ind. acc. to $\boldsymbol{w_j}\} \forall j, j = N(\boldsymbol{w_i})$
4.      $P_{sel} = \{\sim unif(B), \sim unif(B)\}$
5.      $\phi_{child}$ = ApplyOperators($P_{sel}$)
6.      $\phi_{child}$ = Evaluate($\phi_{child}$)
7.      $Z'$ = NondominatedFilter($Z \cup \phi_{ch}$)
8.      for $j$ in $N(\boldsymbol{w_i})$:
9.          if $\phi_{child}$ is better than $P'_{k,j}$ acc. to $\boldsymbol{w_j}$:
10.             replace ind. $P'_{k,j}$ with $\phi_{child}$
11. return ($P'_k$, $Z'$)
```

2) *Multiobjective Selection with $\gg$2 Objectives:* This section discusses approaches for $\gg$2 objectives, leading up to TAPAS. The paper [52] benchmarked several approaches, and found that adaptive ranking on Pareto front (ARF) performed best. Like in crowding, ARF prefers some individuals in the Pareto front over others. However, whereas crowding biases to corners of the performance space, ARF biases to individuals that do relatively well on each objective, via the "Adaptive Rank" criterion $AR(\phi) = \sum_{i=1}^{N_f} rank(f_i, \phi, Z)$, where $rank(f_i, \phi, Z)$ is the rank of individual ϕ with reference to the Pareto-optimal set Z, for objective f_i. For a given objective, the best individual has a rank value of 1, the second-best has rank 2, and so on, ARF is implemented like NSGA-II in the first two selection steps, but it differs in the third step: if the Pareto front will fill up all of P_{sel}, then the AR criterion is used; otherwise the usual NSGA-II "crowding" criterion is used. That is, use AR only if $|P_{sel}| + |F_1| \geq N_L$. Therefore, ARF is only used to distinguish among Pareto-optimal individuals, not among the remaining (Pareto-dominated) individuals. While ARF handles $\gg$2–3 objectives better than NSGA-II, its deficiency is the opposite: it gives poor spread across objective space.

MOEA with decomposition (MOEA/D) [53] has characteristics favorable to $\gg$2 objectives. Its idea is to run N_L single-objective local optimizations simultaneously, where each local optimization i minimizes a weighted sum across objective costs $\boldsymbol{w_i}^T \boldsymbol{f}(\phi)$. Each local optimization points to a different direction $\boldsymbol{w_i}$, and directions are well-spread $\boldsymbol{W} = \{\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_{N_L}}\}$. Selection and crossover for a given direction $\boldsymbol{w_i}$ considers the individuals of neighboring directions $j \in N(\boldsymbol{w_i})$. The MOEA/D version of *OneMOEAGen()* is in Table VII and Fig. 10.

In preliminary experiments, we found that MOJITO with MOEA/D could efficiently generate a smooth Pareto Front without performance gaps, but it contained far fewer nondominated individuals than NSGA-II. In a single-topology space this may not be a problem, since the retained individuals are the "best" for one (or more) weights. However, this is undesirable when searching across $\gg$1 topologies, because MOEA/D's local-optimization perspective biases toward easy-to-optimize topologies.

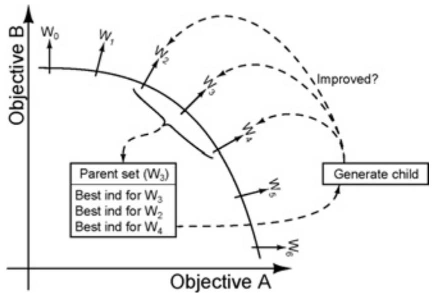
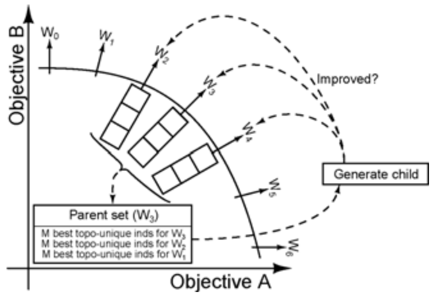


Fig. 10. MOEA/D multiobjective selection in action.



Objective B
Objective A
W0
W1
W2
W3
W4
W5
W6
Improved?
Generate child
Parent set (W3)
M best topo-unique inds for W3
M best topo-unique inds for W2
M best topo-unique inds for W4

Fig. 11. TAPAS multiobjective selection in action.

To address this, TAPAS [51] builds on MOEA/D, by explicitly maintaining a set of unique topologies at each region of the Pareto front. TAPAS stands for topology aware pareto search. TAPAS is like the MOEA/D algorithm of Table VII, except for line 3's mechanism to compute B: for a weight $\boldsymbol{w_i}$ or $\boldsymbol{w_j}$, instead of choosing one best individual according to the weight, the M best unique topologies are chosen. Fig. 11 illustrates (for $M = 3$).

In our experiments, we use NSGA-II for smaller problems (having 2–3 objectives), and TAPAS for the larger problems.

D. Generation of Initial Individuals

This section describes how initial individuals are generated: a simple method for the smaller problems (having 2–3 objectives), and an improved method for the larger problems.

For smaller problems, Table VIII shows the algorithm to generate initial individuals. Recall that an individual ϕ can be specified by a vector $\boldsymbol{d}$ of N_d variables. A random ϕ is created by drawing random vectors: for each variable i (line 3), either draw a value from a continuous uniform random distribution $U([min, max])$ in lines 4–5, or from a discrete set of values with equal bias $U(\{val1, val2, \dots\})$ in lines 6–7.

On larger problems, preliminary experiments showed that the simple algorithm was inadequate. One issue was uneven sampling of topology types; for example, it generates single-stage amplifiers as frequently as two-stage amplifiers, despite the fact that there are many more possible two-stage topologies. This is because there is equal bias to choosing each variable value, such as the choice between one and two stages. To fix this, we instead give equal bias to each possible topology by performing uniform sampling of sentences in a grammar as in [54]. This is implemented by performing a one-time computation of the number of possible topologies for each building block (using the rules of Section IV), and using those counts c are used as the bias on corresponding Flexible block *chosen_part_index* values on the top-level block. Table IX illustrates. The key change is the introduction of lines 4–5, where each choice variable i's value is chosen according to a discrete density function having a probability $p_{i,j}$ for each possible value $d_{i,j}$, $p_{i,j} = c_{i,j} / \sum_{j=1}^{j_{max}} c_{i,j}$, $\sum_{j=1}^{j_{max}} p_{i,j} = 1$, $c_{i,j}$ is the number of sub-topologies if the jth value is used for variable i.

With subsequent preliminary experiments, we found a second issue: most randomly-generated higher-complexity circuits (e.g., folding topologies, 2-stage amplifiers) do not survive more than a few generations. While ALPS generated more topologies in later random injection phases, those would die out too. Upon investigation, we found that randomly-generated complex amplifiers' performances were much worse than simple ones due to poor initial sizing, and that they did not improve as quickly. This is because the more complex amplifiers have more sizing and biasing variables to set reasonably in order to reach a minimal performance bar.

There is plenty of literature (EA and otherwise) reconciling multiobjective optimization and constraint handling (e.g., [38], [55]). However, for this problem domain, there is opportunity to use domain knowledge to: 1) improve survival rate of complex topologies, and 2) optimize each new topology in an inexpensive fashion. Our approach is to defer competition among topologies until each topology is at least nearly feasible. It is acceptable to allow competition once feasible, because each topology will occupy its own niche in the performance space and will therefore be maintained within the multiobjective framework. This is implemented by

TABLE VIII
PROCEDURE INITIALCIRCUIT() (FIRST IMPLEMENTATION)

```
Inputs: $\Phi$
Outputs: $\phi \in \Phi$
1. $\boldsymbol{d}$ = topLevelVariables($\Phi$)
2. $\phi = \emptyset$
3. for $i$ = 1 to $N_d$:
4.     if $d_i$ is continuous:
5.         $\phi_i \sim U([d_{i,min}, d_{i,max}])$
6.     else: #$d_i$ is discrete:
7.         $\phi_i \sim U(\{d_{i,1}, d_{i,2}, \dots, d_{i,max}\})$
8. return $\phi$
```

TABLE IX
PROCEDURE RANDOMDRAWCIRCUIT()

```
Inputs: $\Phi$
Outputs: $\phi \in \Phi$
1.  $\boldsymbol{d}$ = topLevelVariables($\Phi$)
2.  $\phi = \emptyset$
3.  for $i$ = 1 to $N_d$:
4.      if $d_i$ is a choice parameter:
5.          $\phi_i \sim ddf(\{p_{i,1}$ for $d_{i,1}$, $p_{i,2}$ for $d_{i,2}, \dots\})$
6.      else if $d_i$ is continuous:
7.          $\phi_i \sim U([d_{i,min}, d_{i,max}])$
8.      else: #$d_i$ is discrete:
9.          $\phi_i \sim U(\{d_{i,1}, d_{i,2}, \dots, d_{i,max}\})$
10. return $\phi$
```

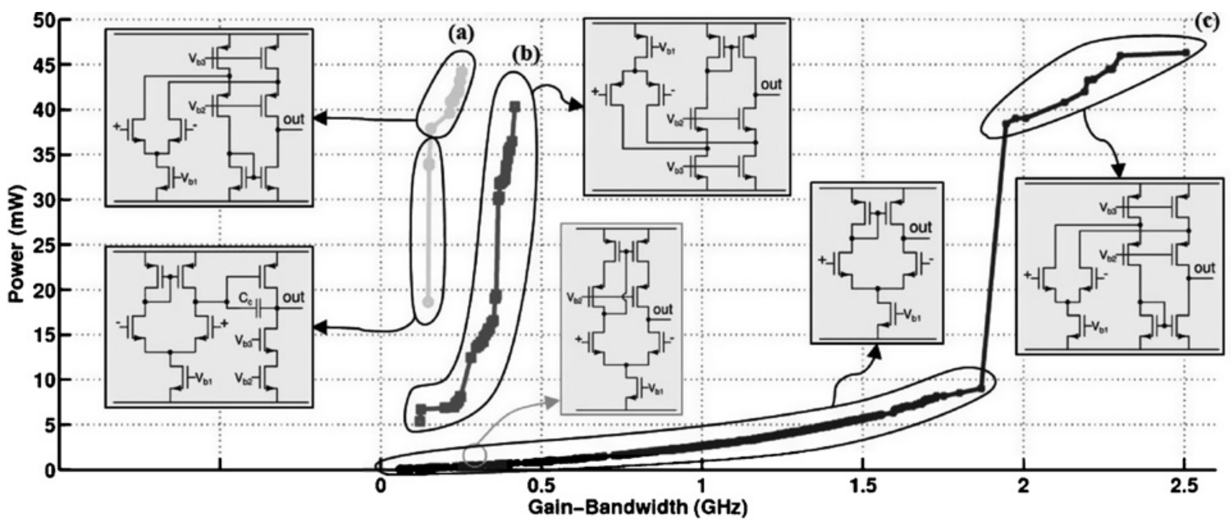


Fig. 12. Results for the experiment of Section VI-B. This plot shows results from three synthesis runs. Set (a) shows a Pareto front for $V_{cmm,in}$ = 1.5-V, set (b) is for $V_{cmm,in}$ = 0.3-V, and set (c) is for $V_{cmm,in}$ = 0.9-V. Aim is to minimize power and maximize gain-bandwidth. Each point is an optimally sized topology; each topology has many different sets of sizings. The expected topologies were found.

TABLE X
PROCEDURE INITIALCIRCUIT() (IMPROVED IMPLEMENTATION)

```
Inputs: $\Phi$
Outputs: $\phi \in \Phi$
1. $\phi$ = randomDrawCircuit($\Phi$)
2. while meetsFuncDOCs($\phi$) $\neq$ True:
3.     $\phi'$ = mutateSizings($\phi$)
4.     if funcDOCsCost($\phi'$) < funcDOCsCost($\phi$):
5.         $\phi = \phi'$
6. while meetsSimDOCs($\phi$) $\neq$ True:
7.     $\phi'$ = mutateSizings($\phi$)
8.     if simDOCsCost($\phi'$) < simDOCsCost($\phi$):
9.         $\phi = \phi'$
10. while meetsPerfConstraints($\phi$) $\neq$ True:
11.     $\phi'$ = mutateSizings($\phi$)
12.     if perfCost($\phi'$) < perfCost($\phi$):
13.         $\phi = \phi'$
14. Return $\phi$
```

performing constraint satisfaction for each topology, using computationally inexpensive constraints. The topology must pass the most inexpensive constraints to proceed to the next level of constraints. Table X illustrates.

The first gate (lines 2–5) has function device operating constraints (DOCs) [56], which measure if current and voltage conditions are being met (e.g., "is the transistor in the saturation operating region?"). Because currents and voltages are part of the individual's design point (see Section IV), then these DOCs can be directly computed without needing simulation. The second gate (lines 6–9) uses DOCs computed from dc simulation. While slower than function DOCs, dc simulation is less expensive than other analyses like transient. The third gate (lines 10–13) uses performance constraints across all the circuit analyses (ac, dc, transient, and so on). It is the most thorough, but the most expensive. Note that *mutateSizings()* is like the mutation operator of Section V-B, except only non-topology parameters get changed.

These improvements enable the reliable generation of complex topologies that are competitive against simpler topologies.

VI. Experimental Results

This section describes the experimental setup, and results for questions regarding MOJITO performance and behavior.

A. Experimental Setup

Experiments of Sections VI-B and VI-C have 2–3 objectives with NSGA-II for MOEA selection, and use the topology DB with 3528 possible topologies. The experiment of Section VI-D has six objectives with TAPAS for MOEA selection, and uses the extended topology DB containing 101 904 topologies. The top-level block had N_d = 50 parameters which include both topology selection variables and sizing variables. In measuring performance, ac/dc analyses took ≈1–4 s to simulate, and transient analyses took ≈10–30 s. MOJITO's search space and search algorithm were implemented about 25 000 lines[5] of Python code [36]. Table XI gives further parameters.

B. Experiment: Hit Target Topologies?

Here, the aim of this experiment is to validate MOJITO's ability to find targeted topologies. The objectives were to maximize GBW, and to minimize power. Three runs were done; the only difference among them is the common-mode voltage $V_{cmm,in}$ at the input. We know that for V_{dd} = 1.8-V and $V_{cmm,in}$ = 1.5-V, topologies must have an NMOS input pair. For $V_{cmm,in}$ = 0.3-V, topologies must have PMOS inputs. At $V_{cmm,in}$ = 0.9-V, there is no restriction between NMOS and PMOS inputs. Each run took ≈overnight on ten single-core 2.0 GHz Linux machines, covering ≈100 000 individuals.

Fig. 12 illustrates the outcome of the experiments. It contains the combined results of the three MOJITO runs. Result

[5]The code breakdown is: 2500 for the building blocks, 500 for constraints and objectives, 3000 for the EA, 2000 to call the simulator and retrieve results, 4000 for data structures, 2500 for parallel processing, and 10 000 for unit tests.

TABLE XI
EXPERIMENTAL SETUP PARAMETERS

Technology	0.18 μm CMOS
Testbench parameters	Load capacitance = 1 pF, supply voltage = 1.8-V, output DC voltage = 0.9-V
Simulator	HSPICE [57]
Constraints	Phase margin $PM > 65$ deg, DC gain > 30 dB, $GBW > 1$ GHz, power < 100 mW, dynamic range > 0.1-V, $SR > $ 1e6-V/s, dozens of device operating constraints
Objectives	See specific experiment
EA settings	Num. age layers $K = 10$, num. individuals per age layer $N_L = 100$, age gap $N_a = 20$, max. num. individuals $N_{ind,max} = 100\,000$. TAPAS: 5 neighbors per weight, $M = 20$ topologies per weight, 20 inds. in age layer initialization

(a) has $V_{cmm,in}$ = 1.5-V, and has indeed only topologies with NMOS inputs. MOJITO chose to use 1-stage and 2-stage amplifiers, depending on the power-GBW tradeoff. Result (b) has $V_{cmm,in}$ = 0.3-V, and MOJITO only returns amplifiers with PMOS input pairs. For result (c), a $V_{cmm,in}$ = 0.9-V has been specified. Though both amplifiers with NMOS and PMOS input pairs might have arisen, NMOS inputs were preferred.

The curve shows the switch in topology around GBW $\approx$ 1.9 GHz, moving from a folded-cascode input (larger GBW) to a simple current-mirror amp (smaller GBW). Note that there would be more amplifiers with folded-cascode input at GBW $<$ 1.9 GHz, but they are not plotted here because they are dominated by the current-mirror amp which needs lower power for the same GBW. Equivalently, the tradeoff indicates that the current-mirror amp has a maximum achievable GBW of $\approx$ 1.9 GHz, beyond which a higher-power amp is needed. To get deeper insight, the designer can use his expertise and experience for a deeper analysis. For example, the analog-design explanation for the step in power is: going from a non-folded to a folded input means that a second current branch is needed, which has its own current/power needs. Interestingly, the search retained a stacked current-mirror load for about 250 MHz GBW.

This experiment validated that MOJITO did find the topologies that we had expected *a priori*.

C. Experiment: Diversity?

The aim of this experiment is to verify that MOJITO could get interesting groups of topologies in a tradeoff of three objectives. The motivation is, whereas a single-objective structural synthesis can only return one topology, more objectives means more possible topologies, because different topologies naturally lie in different regions of the performance space. In this experiment, one run was done. There objectives are: maximize GBW, maximize gain, and minimize area.

Fig. 13 shows the results. We find that MOJITO found diverse structures, as expected. MOJITO found that a folded-cascode opamp gives high GBW but with high area; 2-stage amps give high gain but at the cost of high area; the low-voltage current mirror load is a 1-stage with high gain; and there are many other 1-stage opamp topologies which give a broad performance tradeoff. These are all results that a circuit designer would expect.

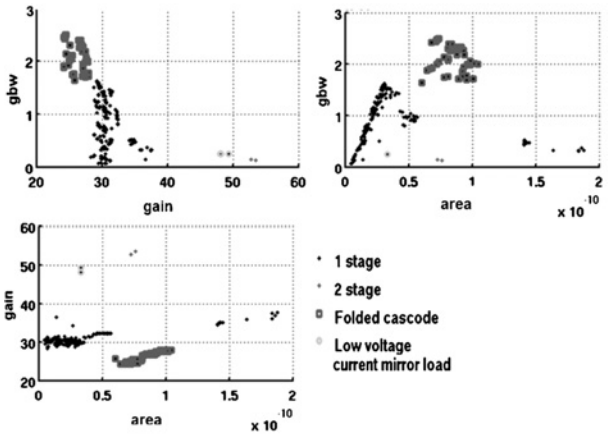


Fig. 13. Results for the experiment of Section VI-C. This illustrates the three-objective Pareto front (maximize GBW, maximize gain, minimize area), two objectives as a time. Individuals are grouped according to some of their structural characteristics (e.g., 1-stage versus 2-stage).

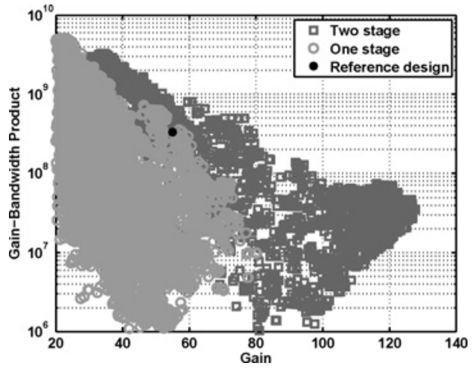


Fig. 14. Results for the experiment of Section VI-D. The plot shows a 2-D cross-section of the 6-D Pareto front.

D. Experiment: Human-Competitive Results?

This section investigates whether MOJITO results are competitive with human-selected and human-sized topologies. The problem has six objectives: maximize DC gain, maximize GBW, minimize power, minimize area, maximize dynamic range (DR), and maximize slew rate (SR). Constraints were device operating constraints, DC gain > 20 dB, GBW > 10 MHz, power < 100 mW, $10^{-14} \leq$ area $\leq 10^{-4}$m^2, PM > 65deg, DR > 0.1-V, SR > 10^6-V/s. The search space contained 101 904 topologies, plus sizes and biases for each. Five 2x 4-core Xeon machines were run for 7 days.

Fig. 14 shows a cross section of the resulting Pareto front in the gain-GBW plane. It has 17 438 designs, comprising 152 unique topologies having 1 or 2 stages. Many of these designs include stages like those in Fig. 8. For comparison, we got a reference design from an expert designer. We see that MOJITO designs compete with the reference design along these axes. MOJITO designs were competitive on all axes, as Table XII shows. Indeed, MOJITO found 59 designs that were better than the manual reference design, distributed over 12 unique topologies. For further verification, we manually inspected the topologies (three of the authors are analog designers); the

TABLE XII
COMPARISON OF MOJITO TO EXPERT REFERENCE DESIGN (EXPERIMENT OF SECTION VI-D)

Performance	Aim	Manual	MOJITO	MOJITO
Topology	–	Symmetrical	Telescopic	Miller
Gain (dB)	Max.	55	53.06	56.24
GBW (MHz)	Max.	330	474	413
DR (V)	Max.	1.2	1.01	1.53
SR (V/μs)	Max.	380	389	554
Power (mW)	Min.	2.26	1.02	7.40
Area (μm^2)	Min.	–	218	3974
PM (deg)	≥ 65	65	67	65.18
DOCs	*met*?	YES	YES	YES

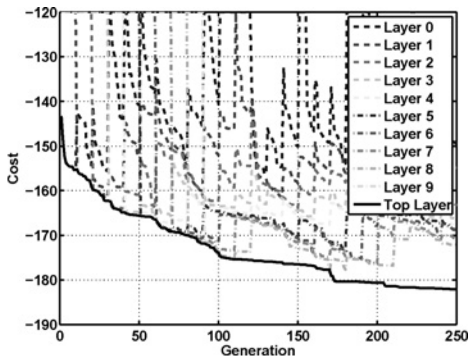


Fig. 15. Results for the experiment of Section VI-D. The plot shows cost versus generation, for each ALPS age layer.

topologies were indeed trustworthy[6] enough to consider for physical silicon implementation.

Fig. 15 shows the convergence over time. Cost is merely the sum of each TAPAS $\boldsymbol{w_i}$'s cost.[7] This figure shows that MOJITO has little risk of premature convergence as it searches for the Pareto Optimal Set, because the lowest age level continually introduces new genetic material, and intermediate age layers allow the new material to compete fairly.

E. Comparison to Open-Ended GP Synthesis

Problems of comparative complexity took open-ended GP 100 million or more individuals [18], and those results were not trustworthy by construction. It was estimated that to get to get a reasonable degree of robustness would take 150 years on a 1000 node 1 GHz cluster [59]. That is, it would have taken [(150 years * 365 days/year * 24 h/day) * 1000 CPUs * 1 GHz]/[(150 h) * 1 CPU * 2 GHz] = 4.4 million times more computational effort than MOJITO to get comparable results. Even if a speedup of 1000x for open-ended GP was achieved, it would still be 4000x slower than MOJITO. By ignoring the domain knowledge that has accumulated over the decades, open-ended GP creates a very difficult problem that could be avoided. Since many disciplines accumulate structural domain knowledge as they mature, codifying this knowledge improves search runtime and result quality.

[6]See definition in Section I.

[7]A convenient single y-variable to track convergence with is a bonus of using MOEA/D or TAPAS, compared to NSGA-II or ARF.

F. MOJITO Applications and Use Cases

To apply MOJITO to other circuit problems, one needs to modify the topologies DB and objectives/constraints. The DB can be modified by building up different blocks, or using a different block as the root node. In this fashion, MOJITO was used for designing current mirrors robust to electromagnetic compatibility effects [60], complex computational circuits [61], and flash analog-to-digital converters [61].

MOJITO has several industrially-applicable use cases. Once it is run for a particular process technology (e.g., 45 nm TSMC), its Pareto-optimal set can be stored as a database, ready to be queried by different designers for different specifications. This is particularly useful for "jellybean IP"—off-the-shelf designs where a simple, non-aggressive solution is quickly needed. The designs can be used in the context of larger designs; for example, MOJITO-designed cell-level opamps can be used in system-level analog-to-digital converters, which themselves can be in chip-level designs such as Wi-Fi or GSM having $>$100 000 transistors. MOJITO itself can be used to explore system-level or chip-level designs, by using Pareto-optimal designs from lower levels in its blocks DB, and sufficiently fast simulation (e.g., with FastSPICE [58] or behavioral modeling [62]). To migrate to a different process, one merely switches simulator model files. In another use case, MOJITO can help designers to creatively design novel blocks: the designer enters the novel block into the blocks DB, runs MOJITO, then determines the region of Pareto front for which the block has provided benefit, and what surrounding circuitry is needed.[8]

The approach generalizes beyond circuits into domains having an accumulation of structural domain knowledge, such as engineering problems (e.g., automotive design, robotics), or scientific modeling problems (e.g., reverse engineering of biological systems). The domain must have trusted building blocks and trusted compositional design principles. To apply, one specifies the building blocks DB, genotype-to-phenotype mapping, phenotype evaluation, and objectives/constraints.

In using a MOJITO styled approach for circuits or other domains, challenges arise. It can take significant effort to capture a field's key structural domain knowledge—days, weeks, or months, depending on the domain and the size of the set. There can be dozens of parameters in the larger blocks, which takes some care to manage. Synthesis runtime for problems that take seconds or minutes to evaluate an individual can still be significant (though of course parallel processing helps). On faster-evaluating problems, it may take repeated applications of the search operators to take a non-neutral step, which may become a bottleneck. But for many domains, these disadvantages are outweighed by the benefits of trustworthy-by-construction synthesis.

VII. CONCLUSION

This paper presented MOJITO, a novel approach for EA-based synthesis of trustworthy structures. MOJITO inputs

[8]A small modification to MOJITO enables it to automatically create and test novel building blocks in the context of otherwise-trustworthy topologies; [63] elaborates on this.

domain knowledge in the form of field-specific, hierarchically-composed building blocks. These blocks have been developed over the years by domain experts. The building blocks DB is organized as a parameterized grammar. MOJITO's EA searches through combinations of these possible blocks to return a Pareto-optimal set of trustworthy structures. It has both tree-oriented and vector-oriented means of searching the derivations of the grammar, and avoids stealth mutations by testing for gene expression before fitness evaluation. MOJITO is demonstrated in the problem domain of analog circuit topology synthesis. It searches across $>100\,000$ different one-stage and two-stage opamp topologies, and returns thousands of trustworthy Pareto-optimal sized topologies. MOJITO can be applied to other problem domains which have built up structural domain knowledge, such as automotive design and scientific modeling.

References

[1] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[2] International Technology Roadmap for Semiconductors. (2008, Apr.) [Online]. Available: http://public.itrs.net

[3] B. Razavi, *Design of Analog CMOS Integrated Circuits*. New York: McGraw-Hill, 2000.

[4] W. M. C. Sansen, *Analog Design Essentials*. Berlin, Germany: Springer, 2006.

[5] G. S. Hornby, "ALPS: The age-layered population structure for reducing the problem of premature convergence," in *Proc. Genet. Evol. Comput. Conf.*, 2006, pp. 815–822.

[6] P. A. Whigham, "Grammatically-based genetic programming," in *Proc. Workshop Genet. Progr.*, 1995, pp. 33–41.

[7] M. O'Neill and C. Ryan, *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Norwell, MA: Kluwer, 2003.

[8] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, 2nd ed. Berlin, Germany: Springer-Verlag, 2006.

[9] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog integrated circuits by means of genetic programming," *IEEE Trans. Evol. Comput.* vol. 1, no. 2, pp. 109–128, Jul. 1997.

[10] J. D. Lohn and S. P. Colombano, "Automated analog circuit synthesis using a linear representation," in *Proc. 2nd Int. Conf. Evol. Syst. Biol. Hardw.*, 1991, pp. 125–133.

[11] J. R. Koza, D. Andre, F. H. Bennett, III, and M. Keane, *Genetic Programming 3: Darwinian Invention and Problem Solving*. San Mateo, CA: Morgan Kaufman, 1999.

[12] J. B. Grimbleby, "Automatic analogue circuit synthesis using genetic algorithms," *IEEE Proc. Circuits Syst. Devices*, vol. 147, no. 6, pp. 319–323, Dec. 2000.

[13] R. Zebulum, M. Vellasco, and M. Pacheco, "Variable length representation in evolutionary electronics," *Evol. Comput.*, vol. 8, no. 1, pp. 93–120, 2000.

[14] C. Goh and Y. Li, "GA automated design and synthesis of analog circuits with practical constraints," in *Proc. Congr. Evol. Comput.*, vol. 1. 2001, pp. 170–177.

[15] H. Shibata, S. Mori, and N. Fujii, "Automated design of analog circuits using cell-based structure," in *Proc. Nasa/DoD Conf. Evol. Hardw.*, 2002, pp. 85–92.

[16] T. Sripramong and C. Toumazou, "The invention of CMOS amplifiers using genetic programming and current-flow analysis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 11, pp. 1237–1252, Nov. 2002.

[17] R. Zebulum, M. Pacheco, and M. Vellasco, *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. Boca Raton, FL: CRC Press, 2002.

[18] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Norwell, MA: Kluwer, 2003.

[19] S. Ando, M. Ishizuka, and H. Iba, "Evolving analog circuits by variable length chromosomes," in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Eds. New York: Springer, 2003, pp. 643–662.

[20] J. Hu and E. Goodman, "Robust and efficient genetic algorithms with hierarchical niching and sustainable evolutionary computation model," in *Proc. Genet. Evol. Comput. Conf.*, 2004, pp. 1220–1232.

[21] T. R. Dastidar, P. P. Chakrabarti, and P. Ray, "A synthesis system for analog circuits based on evolutionary search and topological reuse," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 211–224, Apr. 2005.

[22] C. Mattiussi and D. Floreano, "Analog genetic encoding for the evolution of circuits and networks," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 596–607, Oct. 2007.

[23] A. Das and R. Vemuri, "Topology synthesis of analog circuits based on adaptively generated building blocks," in *Proc. Design Autom. Conf.*, Jun. 2008, pp. 44–49.

[24] Y. Sapargaliyev and T. G. Kalganova, "Unconstrained evolution of analogue computational 'QR' circuit with oscillating length representation," in *Proc. Int. Conf. Evol. Syst.*, LNCS 5216. 2008, pp. 1–10.

[25] A. Thompson, "Evolving electronic robot controllers that exploit hardware resources," in *Proc. 3rd Eur. Conf. Artif. Life*, 1995, pp. 640–656.

[26] A. Stoica, D. Keymeulen, R. Zebulum, A. Thakoor, T. Daud, G. Klimeck, Y. Jin, R. Tawel, and V. Duong, "Evolution of analog circuits on field programmable transistor arrays," in *Proc. NASA/DoD Conf. Evol. Hardw.*, 2000, pp. 99–108.

[27] R. A. Rutenbar, G. G. E. Gielen, and B. A. A. Antao, Eds., *Computer Aided Design of Analog Integrated Circuits and Systems*. Piscataway, NJ: IEEE Press, 2002, pp. 3–30.

[28] F. M. El-Turky and R. A. Nordin, "BLADES: An expert system for analog circuit design," in *Proc. Int. Conf. Circuits Syst.*, 1986, pp. 552–555.

[29] R. Harjani, R. A. Rutenbar, and L. R. Carley, "OASYS: A framework for analog circuit synthesis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 8, no. 12, pp. 1247–1266, Dec. 1992.

[30] H. Y. Koh, C. H. Séquin, and P. R. Gray, "OPASYN: A compiler for CMOS operational amplifiers," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 9, no. 2, pp. 113–125, Feb. 1990.

[31] B. A. A. Antao and A. J. Brodersen, "ARCHGEN: Automated synthesis of analog systems," *IEEE Trans. Very Large Scale Integr. Circuits*, vol. 3, no. 2, pp. 231–244, Jun. 1995.

[32] A. Doboli and R. Vemuri, "Exploration-based high-level synthesis of linear analog systems operating at low/medium frequencies," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 11, pp. 1556–1568, Nov. 2003.

[33] E. Martens and G. G. E. Gielen, "Top-down heterogeneous synthesis of analog and mixed-signal systems," in *Proc. Des. Autom. Test Eur.*, 2006, pp. 275–280.

[34] W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS opamp synthesis by means of a genetic algorithm," in *Proc. Des. Autom. Conf.*, 1995, pp. 433–438.

[35] P. C. Maulik, L. R. Carley, and R. A. Rutenbar, "Integer programming based topology selection of cell level analog circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 4, pp. 401–412, Apr. 1995.

[36] Python. (2008, Apr. 23). *Python Programming Language* [Online]. Available: http://www.python.org

[37] F. Leyn, G. G. E. Gielen, and W. M. C. Sansen, "An efficient dc root solving algorithm with guaranteed convergence for analog integrated CMOS circuits," in *Proc. Int. Conf. Comput.-Aided Design*, 1998, pp. 304–307.

[38] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[39] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer, 1998.

[40] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[41] D. E. Goldberg, *The Design of Innovation: Lessons From and For Competent Genetic Algorithms*. Berlin, Germany: Springer, 2002.

[42] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 1769–1776.

[43] D. J. Cavicchio, "Adaptive search using simulated evolution," Ph.D. dissertation, Dept. Commun. Comput. Sci., Univ. Michigan, Ann Arbor, 1970.

[44] K. A. DeJong, "Analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. Michigan, Ann Arbor, 1975.

[45] S. W. Mahfoud, "Crowding and preselection revisited," in *Parallel Problem Solving from Nature 2*, R. Manner and B.

Manderick, Eds., Amsterdam, The Netherlands: Elsevier, 1992, pp. 27–36.

[46] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evol. Methods Design, Optim. Control Appl. Ind. Prob.*, 2002, pp. 95–100.

[47] G. Smits and M. Kotanchek, "Pareto-front exploitation in symbolic regression," in *Genetic Programming Theory and Practice II*, U.-M. O'Reilly, T. Yu, R. Riolo, and B. Worzel, Eds. Berlin, Germany: Springer, 2004, pp. 283–289.

[48] J. Hu, E. K. Goodman, K. Seo, Z. Fan, and R. Rosenberg, "The hierarchical fair competition framework for sustainable evolutionary algorithms," *Evol. Comput.*, vol. 13, no. 2, pp. 241–277, 2005.

[49] P. Schuster, W. Fontana, P. F. Stadler, and I. Hofacker, "From sequences to shapes and back: A case study in RNA secondary structures," *Proc. Roy. Soc. (London) B*, vol. 255, pp. 279–284, Mar. 1994.

[50] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[51] P. Palmers, T. McConaghy, M. Steyaert, and G. G. E. Gielen, "Massively multi-topology sizing of analog integrated circuits," in *Proc. Des. Autom. Test Eur. Conf.*, 2009, pp. 706–711.

[52] D. Corne and J. Knowles, "Techniques for highly multiobjective optimization: Some nondominated points are better than others," in *Proc. Genet. Evol. Comput. Conf.*, 2007, pp. 773–780.

[53] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[54] H. Iba, "Random tree generation for genetic programming," in *Proc. 4th PPSN Int. Conf. Evol. Comput.*, LNCS 1141. 1996, pp. 144–153.

[55] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Berlin, Germany: Springer, 2004.

[56] H. E. Graeb, S. Zizala, J. Eckmueller, and K. Antreich, "The sizing rules method for analog integrated circuit design," in *Proc. Int. Conf. Comput.-Aided Des.*, 2001, pp. 343–349.

[57] Synopsys, Inc. (2009, Aug. 21). *Product Page: HSPICE* [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Pages/default.aspx

[58] Synopsys, Inc. (2009, Aug. 21). *Product Page: CustomSim* [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/Pages/CustomSim-ds.aspx

[59] T. McConaghy and G. G. E. Gielen, "Genetic programming in industrial analog CAD: Applications and challenges," in *Genetic Programming Theory and Practice III*, T. Yu, R. L. Riolo, and B. Worzel, Eds. Berlin, Germany: Springer, 2005, ch. 19, pp. 291–306.

[60] J. Loeckx, T. Deman, T. McConaghy, and G. G. E. Gielen, "A novel EMI-immune current mirror topology obtained by genetic evolution," in *Proc. Conf. Electro Magnetic Compatibility*, 2009.

[61] P. Gao, T. McConaghy, and G. G. E. Gielen, "ISCLEs: Importance sampled circuit learning ensembles for trustworthy analog circuit topology synthesis," in *Proc. Int. Conf. Evol. Syst.*, Sep. 2008, pp. 11–21.

[62] K. Kundert and O. Zinke, *The Designer's Guide to Verilog-AMS*. Norwell, MA: Kluwer, 2004.

[63] T. McConaghy, P. Palmers, G. G. E. Gielen, and M. Steyaert, "Genetic programming with reuse of known designs," in *Genetic Programming Theory and Practice V*, R. L. Riolo, T. Soule, and B. Worzel, Eds. Berlin, Germany: Springer, 2007, pp. 161–186.

Trent McConaghy (S'95–M'99) received the B.E. and B.S. degrees in computer science (both with great distinction) from the University of Saskatchewan, Saskatoon, SK, Canada, in 1999, and the Ph.D. degree in electrical engineering from Katholieke Universiteit Leuven, Leuven, Belgium, in 2008.

He is currently a Co-Founder and the Chief Scientific Officer with Solido Design Automation, Inc., Saskatoon. He was a Co-Founder and the Chief Scientist with Analog Design Automation, Inc., Ottawa, ON, Canada, which Synopsys, Inc., Mountain View, CA, acquired in 2004. Prior to that, he did research for the Canadian Department of National Defense, Ottawa. He has more than 30 peer-reviewed technical papers and 20 patents granted/pending. He is the author of "variation-aware structural synthesis of analog circuits: a computational intelligence approach," and a Co-Editor of *Genetic Programming Theory and Practice VII* and *Genetic Programming Theory and Practice VIII*. His current research interests include statistical machine learning and evolutionary computation, with transistor-level computer-aided design applications of variation-aware design, knowledge extraction, symbolic modeling, and analog topology design.

Dr. McConaghy's thesis won the International EDAA "Outstanding Dissertation Award."

Pieter Palmers (S'04–M'09) was born in Leuven, Belgium, in 1980. He received the Masters degree in electronic engineering from the Katholieke Universiteit Leuven, Leuven, Belgium, in 2003, and has recently received the Ph.D. degree from ESAT-MICAS, Katholieke Universiteit Leuven.

He is currently with Mephisto Design Automation, Leuven. His current research interests include the field of high speed data converter design and analog design automation.

Michiel Steyaert (SM'92–F'04) received the Masters degree in electrical-mechanical engineering and the Ph.D. degree in electronics from the Katholieke Universiteit Leuven, Leuven, Belgium, in 1983 and 1987, respectively.

From 1983 to 1986, he received the IWNOL Fellowship from the Belgian National Foundation for Industrial Research, Brussels, Belgium, to work as a Research Assistant with the ESAT Laboratory, Katholieke Universiteit Leuven. In 1987, he was responsible for several industrial projects in the field of analog micro power circuits with ESAT as an IWONL Project Researcher. In 1988, he was a Visiting Assistant Professor with the University of California, Los Angeles. In 1989 he was appointed by the National Fund of Scientific Research, Belgium, as a Research Associate, in 1992 as a Senior Research Associate, and in 1996 as a Research Director with ESAT. From 1989 to 1996, he was a Part-Time Associate Professor. He is currently a Full Professor with Katholieke Universiteit Leuven and the Chair of the Department of Electrical Engineering. He authored or co-authored over 400 papers and over 15 books. His current research interests include high-performance and high-frequency analog integrated circuits for telecommunication systems and analog signal processing.

Dr. Steyaert received the 1990 and 2001 European Solid-State Circuits Conference Best Paper Award. He received the 1991 and 2000 NFWO Alcatel-Bell-Telephone Award for Work in Telecommunications ICs. He received the 1995 and 1997 IEEE-ISSCC Evening Session Award, and the 1999 IEEE Circuit and Systems Society Guillemin-Cauer Award. He was recognized as one of the top ten authors in the 50 year history of ISSCC.

Georges G. E. Gielen (S'87–M'92–SM'99–F'02) received the M.S. and Ph.D. degrees in electrical engineering from the Katholieke Universiteit Leuven, Leuven, Belgium, in 1986 and 1990, respectively.

He is currently a Full Professor with Katholieke Universiteit Leuven. He has authored or co-authored five books and more than 300 peer-reviewed papers. His current research interests include the design of analog and mixed-signal integrated circuits, especially analog and mixed-signal computer-aided design (CAD) tools and design automation (modeling, simulation and symbolic analysis, analog synthesis, analog layout generation, analog and mixed-signal testing). He is a Coordinator or Partner of several industrial research projects in this area, including several European projects (EU, MEDEA, ESA).

Dr. Gielen is regularly a member of program committees of international conferences (DAC, ICCAD, ISCAS, DATE, CICC, and so on), and served as the General Chair of DATE in 2006 and of the International Conference on Computer-Aided Design in 2007. He serves on editorial boards of international journals like IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, *Springer International Journal on Analog Integrated Circuits and Signal Processing*, and *Elsevier Integration*. He received the 1995 Best Paper in the *Wiley International Journal on Circuit Theory and Applications*, and was the 1997 Laureate of the Belgian Royal Academy on Sciences, Literature and Arts in the discipline of engineering. He served as an elected member of the Board of Governors of the IEEE Circuits and Systems (CAS) Society. He was the President of the IEEE CAS Society in 2005. He received the IEEE Computer Society Outstanding Contribution Award and the IEEE Circuits and Systems Society Meritorious Service Award in 2007.