A Graph Grammar Based Approach to Automated Multi-Objective Analog Circuit Design

Angan Das and Ranga Vemuri
Department of Electrical and Computer Engineering, University of Cincinnati, Cincinnati, OH 45221-0030, USA.
Email: {dasan, ranga}@ececs.uc.edu

***Abstract*— This paper introduces a graph grammar based approach to automated topology synthesis of analog circuits. A grammar is developed to generate circuits through production rules, that are encoded in the form of a derivation tree. The synthesis has been sped up by using dynamically obtained design-suitable building blocks. Our technique has certain advantages when compared to other tree-based approaches like GP based structure generation. Experiments conducted on an opamp and a vco design show that unlike previous works, we are capable of generating both manual-like designs (bookish circuits) as well as novel designs (unfamiliar circuits) for multi-objective analog circuit design benchmarks.**

I. INTRODUCTION

Automated analog design or circuit synthesis is the art of constructing a sized analog circuit schematic from user specifications, through developed CAD tools. It is essentially a two step process — Topology formation followed by Sizing of the topology. But unlike its digital counterpart, the development of analog CAD synthesis tools has always been slow and inadequate. Herein lies the basic motivation for this work.

Topology formation can be achieved through two different ways — topology selection and topology generation. Owing to heavy designer dependency and huge set-up effort, heuristic-based selection approaches [1], [2], [3] gradually gave way to topology generation. Evolutionary algorithms like Genetic Algorithms (GA) [4] and Genetic Programming (GP) [5] were used to actually generate circuits. To minimize their computational burden, design-specific analog building blocks were then put to use [6]. But they had the overhead of starting with a certified block library. This was recently avoided in [7], [8], where we introduced adaptive block library formation. But again, these works involved a fixed mesh-like topology template that somewhat limits the search space.

Considering all of the above drawbacks, we introduce a graph grammar based approach for the generation of transistor-level analog circuit topologies. Graph-grammar based structure synthesis has been successfully applied to domains like sheet metal component design [9], cellular graph design [10] and epicyclic gear train design [11]. To the best of our knowledge, there has been no corresponding work on automated analog design till date.

A grammar can be defined as a set of rules that can generate a construct from a list of terminals. For example, in a natural language like English, a construct is a sentence and the terminals are words. Graph grammars provide a theoretical foundation for graphical languages [10]. It consists of a set of rules, that illustrates the way of constructing a complete graph from a variety of nodes and allowable interconnections. In this work, we construct a topology graph using basic analog primitives like PMOS and NMOS.

This work was supported by National Science Foundation under award numbers CCF-0429717 and CNS-0421092.

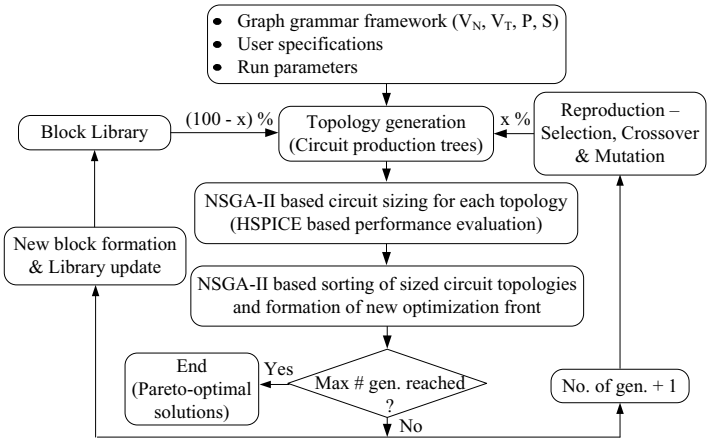


Fig. 1. Synthesis flowchart

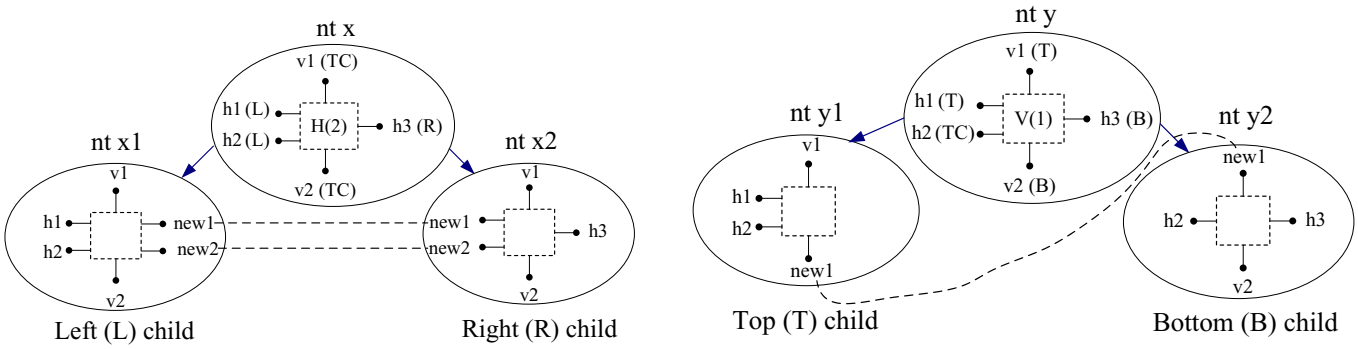
The synthesis process is sped up through usage of bigger and more meaningful building blocks that have been dynamically obtained in a fashion similar to [7], [8]. Finally, it is to be noted that most analog designs are multi-objective optimization problems that often involve conflicting objectives. In this regard, the Non-dominated Sorting Genetic Algorithm (NSGA-II) [12] has been adopted in this framework. Using the above techniques, we have evolved both manual-like, understandable designs as produced in [6], [7] (which we term as *bookish circuits*), as well as novel designs as found in [4], [5], for two analog design benchmarks.

The rest of the paper is organized as follows. Section II introduces the synthesis methodology. In section III, we describe the graph grammar framework. Section IV deals with the reproduction mechanisms, while section V summarizes the adaptive building block library and its formation. Section VI describes the circuit sizer. Experimental results are reported in Section VII and section VIII finally concludes the work.

II. SYNTHESIS METHODOLOGY

The synthesis flow is shown in Fig. 1. Alike any evolutionary algorithm, we maintain a collection of solutions or chromosomes in the form of a generation. Parent generations reproduce to produce offspring solutions. The procedure continues for the allowable number of generations and finally a

978-3-9810801-5-5/DATE09 © 2009 EDAA



Structural disassociation notations for parent node (ntx):
H(2) – Horizontal division with 2 new interconnections (viz. new1 and new2)
Result: new1 & new2 are hor-out for left child (ntx1) and hor-in for right child (ntx2)

Terminal disassociation notations for parent node (ntx):
L – Assigned to left child, R – Assigned to right child,
TC – Assigned to both (T-Connection).

(a)

Structural disassociation notations for parent node (nty):
V(1) – Vertical division with 1 new interconnection (viz. new1)
Result: new1 is ver-out for top child (nty1) and ver-in for bottom child (nty2)

Terminal disassociation notations for parent node (nty):
T – Assigned to top child (must for ver-in of nty), B – Assigned to bottom child (must for ver-out of nty), TC – Assigned to both (T-Connection).

(b)

Fig. 2. Grammar production rules: Structure and terminal disassociation rules (example shown) — (a) Horizontal disassociation and (b) Vertical disassociation

Pareto-optimal set comprising of the best solutions is obtained.

The design components or blocks are stored in a library. The library initially comprises of MOS primitives only. A circuit topology is synthesized using these library blocks through the graph grammar technique. Each topology formed is then sized, whereby device dimensions, and voltage and current bias values are assigned. The multi-objective sizer is based on NSGA-II. NSGA-II produces a set of Pareto-optimal solutions, where no two solutions, belonging to the same optimal front, are inferior compared to each other [12]. Thereby, a wide range of sized topologies is obtained. This whole set of sized circuits is then sorted with NSGA-II again. A new solution front is produced and the process continues.

After each generation, the block library is updated whereby new blocks are added and existing blocks are ranked. This is done as per the techniques described in [7]. Also, from the second generation onwards, offsprings are produced in two ways — one set through the crossover and mutation reproduction mechanisms of parent circuits, while the rest are produced afresh from the better ranked library blocks.

III. GRAPH GRAMMAR: CIRCUIT FORMATION

A graph grammar (G) is symbolized by the 4-tuple:

$$G = (V_N, V_T, P, S)$$

where V_N, V_T, and P denote finite sets of non-terminal symbols (*nt-symbols*) or variables, terminal symbols (*t-symbols*), and production rules respectively. $S(\in V_N)$ stands for the start symbol. The process starts with S. Production rules repeatedly transform an nt-symbol into other nt-symbols or t-symbols. Finally an end graph (solution) is produced that contains t-symbols only. The set of all end graphs ($x : x \in V_T^*$) that can be derived from S by the grammar G is termed as the language (L) of G. L can be expressed as:

$$L = \{x : x \in {V_T}^* \wedge S \underset{G}{\rightarrow} x\}$$

In graph grammar, the solution generation process is represented as a tree structure, popularly known as the derivation tree or production tree. The tree starts with the root node (S), and gradually branches off through production rules. Intermediate nodes represent nt-symbols, while leaf nodes stand for t-symbols. In this work, analog circuit topologies are treated as graphs. They comprise of basic elements like PMOS and NMOS, connected based on certain structural constraints. Fig. 5(a) shows an example topology producing tree (derivation tree), and Fig. 5(b) the corresponding circuit topology (test expt.-1, Sec. VII-A). The following sections detail the individual aspects of the grammar.

A. Start Symbol (S)

The start symbol (S) symbolizes the top-level blackbox for the design. The terminals of the blackbox are designated based on the direction of voltage or current signal propagation [13]. In this work, four terminal sets have been used viz. horizontal-in (hor-in), horizontal-out (hor-out), vertical-in (ver-in) and vertical-out (ver-out). With the above consideration, node $nt0$ in Fig. 5(a) shows the start symbol of a 2-input (V_{IN+} and V_{IN-}), 1-output (V_{OUT}) design. V_{IN+} and V_{IN-} belong to hor-in, V_{OUT} belongs to hor-out, Vdd power rail belongs to ver-in, and Gnd belongs to ver-out.

B. Non-terminal symbols or nt-symbols ($\mathbf{V_N}$)

The nt-symbols represent the substructures that are obtained through the step-by-step decomposition of the main top-level structure, with the decomposition being governed by production rules. These are basically the subcircuits within the main circuit, that are also viewed as blackboxes with a similar set of terminals. Thereby, all tree nodes barring the leaf nodes are composed of nt-symbols ($S \in V_N$). As an example representation of an nt-symbol tree node, in Fig. 2(a), node ntx contains two hor-in terminals ($h1$ and $h2$), one hor-out terminal ($h3$), one ver-in ($v1$) and one ver-out ($v2$) terminal.

C. Terminal symbols or t-symbols ($\mathbf{V_T}$)

The t-symbols form the leaf nodes of the derivation tree, and thereby give the SPICE netlist. These symbols comprise

TABLE I

SALIENT DIFFERENCES (ADVANTAGES) OF OUR TECHNIQUE COMPARED TO GENETIC PROGRAMMING (GP)-BASED STRUCTURE SYNTHESIS

GP-BASED STRUCTURE SYNTHESIS	OUR TECHNIQUE
Generates circuit structures through component creating and connection modifying functions that need to be supplied by a circuit designer. Also, these functions depend on the design class desired.	Breaks the top-level circuit blacbox into hierarchical substructures through simple, generic, and design-independent production rules. The leaf nodes of the circuit formation tree are then mapped to available components.
Automatic generation of subcircuits is not present.	Design-suitable subcircuits, that aid the synthesis run, are automatically generated.
Requires an efficient parser to generate the circuit netlist. The netlist is required for performance evaluation.	Leaf nodes contain the element description as well as the node numbers. They directly produce the circuit netlist. Hence no parser is necessary.
At most times, the size of the evolved design may be large, with several design components.	The circuit size can be restricted, both during the initial generation of solutions, as well as that in subsequent generations evolved through crossover.
Generally requires millions of designs to arrive at the final design.	Thousands of designs are sufficient to generate the desired solution.

of all the actual circuit elements like PMOS, NMOS, resistors, capacitors, etc. required for the design. Fig. 3 shows some of the t-symbols containing NMOS only. An exact similar set of PMOS t-symbols are used as that of an NMOS.

Terminal renaming: To avoid pathological structures, two terminals of the same symbol but of different types, are not interconnected during evolution. Hence, to evolve all kinds of topologies, certain t-symbols have some terminals signified separately though they actually imply the same circuit node. For e.g., in Fig. 5(a), terminals $n1$ and $n4$ of t-symbol $t1$ are both the same drain node of transistor M1. In such cases, these nodes are later renamed (the same) during netlist generation.

D. Production rules (P)

Production rules form the backbone of any grammar. Here, each nt-symbol either produces two new nt-symbols (children nodes) or one t-symbol. nt-symbols at a lower tree depth have higher probability for the former and vice-versa. Hence, production rules are divided into two classes —

(1) nt-symbol → nt-symbol: When an nt-symbol disassociates into two new nt-symbols, the underlying rules are again composed of two sets:

(a) Structure disassociation rules: The disassociation of the parent structure may occur horizontally with the left (L) and right (R) child physically placed adjacent to each other, or vertically with the top (T) child placed above the bottom (B) child. Moreover, the two children produced may be inter-connected through one or more new terminals. Terminal-set wise, they will be 'in' for one and 'out' for the other. Figs. 2(a) and (b) demonstrate horizontal and vertical disassociation respectively. As in Fig. 2(b), 'new1' is the new vertical connection.

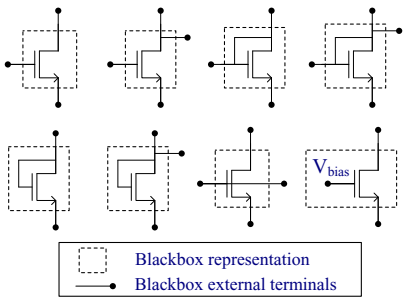


Fig. 3. Terminal symbols: Some examples of initial NMOS only blocks

(b) Terminal disassociation rules: All the directional terminals of the parent structure need to be distributed among its children substructures. In case of a horizontal break-up, the horizontal terminals are assigned either to the left (L) or to the right (R) child. Vertical terminals can be assigned either to L, or to R, or to both of them (TC). But for a lone vertical terminal, TC is the only choice. For e.g., in Fig. 2(a), TC is applied to both the terminals $v1$ and $v2$.

In case of vertical disassociation, all ver-in terminals of the parent are assigned to top (T) child, and all ver-out terminals to bottom (B) child. Horizontal terminals may be assigned to either or both of them. For e.g., in Fig. 2(b), terminal $h1$ is assigned to T, $h3$ to B, and $h2$ to both of them (TC).

(2) nt-symbol → t-symbol: When an nt-symbol is replaced by a t-symbol, there are no disassociation rules. Instead, there is an actual terminal-terminal mapping between those of the nt-symbol and the t-symbol. This helps in generating the SPICE netlist. As shown in Fig. 5(a), terminals $Vdd, n2$ and $n5$ of nt-symbol $nt01112$ correspond to the source, gate and drain nodes of PMOS in t-symbol $t2$. $t2$ represents the transistor M2 in the final circuit of Fig. 5(b).

E. Differences with Genetic Programming (GP)

Although our technique has some apparent similarity with other tree-based techniques like Genetic programming (GP), it certainly has some advantages. Table I enumerates the advantages of our approach when compared to GP-based structure synthesis [5].

IV. REPRODUCTION MECHANISMS

Starting with the second generation, circuit topology generation trees are built in two ways — a certain number of them from the updated block library, and the remaining ones through the crossover and mutation of parent trees.

A. Trees from updated block library (better blocks)

In this reproduction phase, the derivation tree is constructed as per the same production rules. But the rule for replacement of an nt-symbol with a t-symbol is applied with a higher probability. It is so since after the first generation, the t-symbols are not limited to simple primitives only, but also comprise of bigger subcircuits in the form of subtrees (Section V-A). These better ranked subtrees are preferentially chosen over the poorer ones during circuit formation.

B. Crossover

We implement the single point crossover method. Two parents are selected by the crowded comparison operator method [12]. Then, we decide on the number of design elements to be contained from each parent. Accordingly nodes are chosen in both the parent trees, that match each other in terminal properties, and satisfy the size constraints. Finally, these node-rooted subtrees (subcircuits) are swapped between them to produce two offsprings. This kind of crossover provides two advantages. First, it prevents the production of pathological or structurally incorrect circuits. Secondly, the size of the circuit produced is kept within bounds as decided by the user.

C. Mutation

Certain randomly chosen children circuits are mutated in each generation to avoid premature convergence. Mutation is done through subtree replacement. A randomly selected portion of the circuit tree is replaced with a library subtree. Here too, alike crossover, we ensure the structural integrity of the generated circuits and keep a tag on the circuit size.

V. BLOCK LIBRARY: ADAPTIVE FORMATION

Initially, the block library contains simple MOS elements only. But it gradually includes bigger and functionally more useful blocks. The dynamic procedure that continually updates and extends the library is similar to that used in [7].

A. Subtrees from circuit trees: New t-symbols (blocks)

A circuit topology tree gives rise to future subtrees or building blocks. Subtrees are extracted bottom-up out of the main circuit tree as shown in Fig. 4. Physically, they constitute all the subcircuits, though of varying sizes, present in the circuit. These subtrees are then treated as new t-symbols since they may replace any nt-symbol during future circuit tree formation, when the terminal properties of that nt-symbol matches that of the root node of this new t-symbol.

The library is arranged hierarchically, based of number of terminal kinds and design elements contained within a block. Hierarchy helps to fasten the process of — (a) Block selection depending on terminal requirements and (b) Graph isomorphism test to avoid redundancy during new block inclusion.

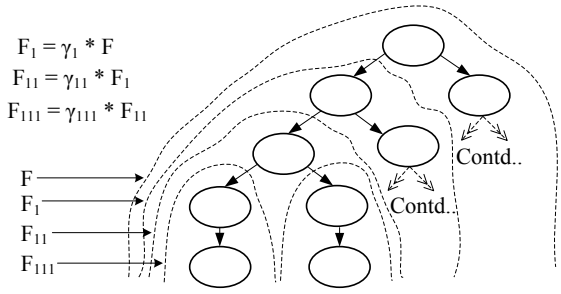
B. Library block parameters

The library blocks possess the same three parameters as outlined in [7]. They are:

1) *No. of occurrences (N_{block}):* Number of times the block has been used for circuit formation.
2) *Average gross Fitness ($Gross\_F_{block}$):* The merit of the block or the level of appropriateness with which the block should be used for the design considered.
3) *Net fitness (Net_F_{block}):* A weighted summation of $\text{Gross\_F}_{\text{block}}$ and N_{block}.

C. Library update

The implication of circuit performance and the heuristic behind its usage in block parameter update have been detailed in [7]. Similar procedures are implemented here to include new blocks (subtrees) and to rank existing blocks after each generation. However, two aspects specific to this work are:



F (child_ckt_portion) = γ * F (parent_ckt_portion)

Notations:
$F_j \rightarrow$ Performance fitness of the circuit portion 'j'
$\gamma \rightarrow$ Proportion of performance share from parent sub-tree
Quantitatively, γ = Fraction of size of child (i.e. no. of leaf-nodes / no. of design components) with respect to parent

Fig. 4. Formation of building blocks (new t-symbols) from circuit trees

*1) **$Gross\_F_{block}$ update*:** For each performance measure, the new gross fitness of an existing block comes from its old gross fitness and its contribution towards the total circuit performance — $\hat{F_{\text{subtree}}}$. For $\hat{F_{\text{subtree}}}$ estimation, we assume that the performance of any tree/subtree may be proportionately distributed into its two immediate children subtrees as shown in Fig. 4. The proportionality factor (γ) is dictated by the relative size of the subtree (number of leaf nodes or in other words - number of design elements) w.r.t. its parent tree/subtree. Greater the relative size, greater the contribution. Following [7], the assumption is claimed to be valid in this evolutionary CAD framework, where we are unaware of the different underlying design heuristics and equations.

*2) **Multi-objective ranking*:** At the end of each generation, the blocks are sorted and ranked based on their net fitness for each performance, as per NSGA-II.

VI. CIRCUIT SIZER

The quality of an analog circuit topology may only be judged after the topology has been sized appropriately [14]. The sizer assigns dimensions and values to all the circuit devices like PMOS, NMOS, voltage source, resistors, and so on. Now, sizing thousands of unknown circuits in a multi-objective design scenario is a non-trivial problem. In this regard, we have implemented the NSGA-II algorithm [12] for the sizer. Within NSGA-II, we adopt the crowded comparison operator for selection, and the SBX method for crossover [12].

The sizer inputs comprise of the range and granularity of design variables like transistor dimensions and voltage source values. Transistors sharing the same gate node are assumed to be matched and their widths are integral multiples of each other. Also, HSPCIE is used for performance evaluation despite some computational effort. It owes to the fact that HSPICE provides accuracy, requires no set-up effort (unlike symbolic methods), and helps to integrate the tool into any existing industry framework.

VII. EXPERIMENTS AND RESULTS

The two experimental design benchmarks chosen for synthesis are that of an operational amplifier and a voltage controlled

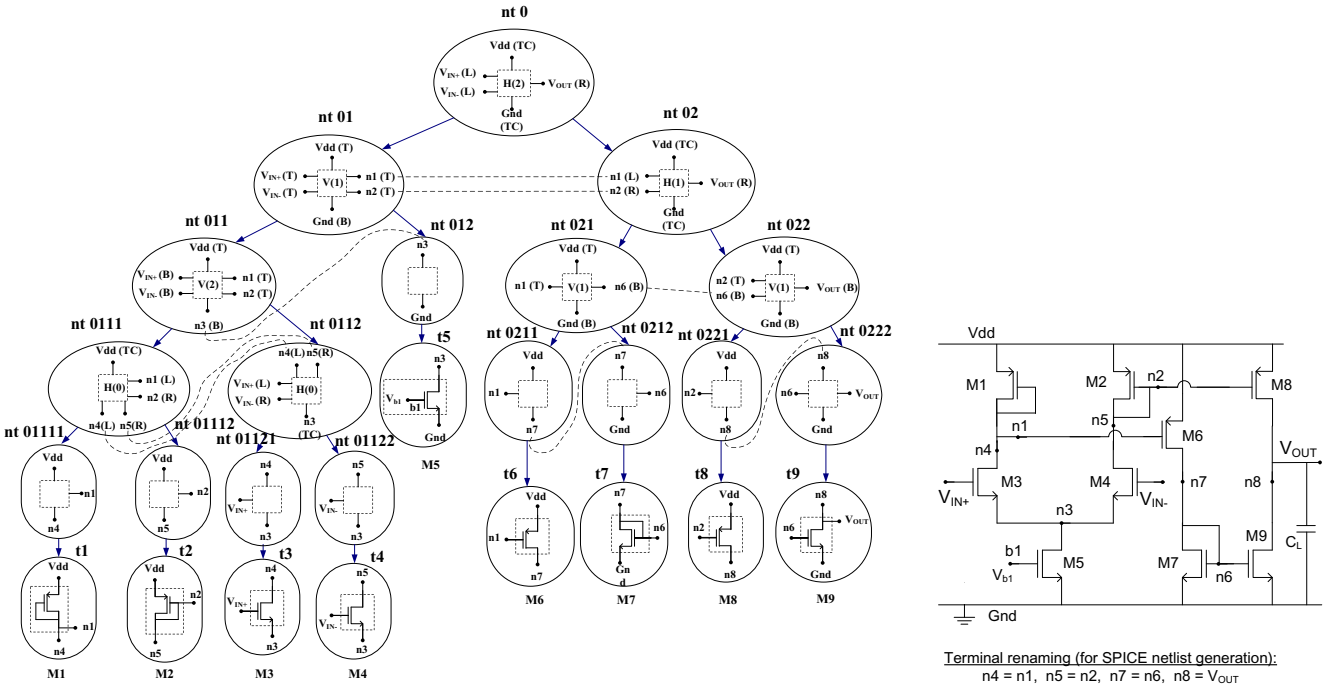


Fig. 5. A (manual-like) solution for the 2-input 1-output opamp design — (a) Topology derivation tree (b) Corresponding circuit schematic

oscillator (vco) design. Table II lists their specifications. The various run parameters for the synthesis tool are set after tuning the algorithm over several runs. Also, in each generation, 30% of the circuits are formed using library blocks and the rest 70% through crossover and mutation mechanisms.

The circuit sizer uses 48 chromosomes and runs for 50 generations for each circuit topology. The process technology file used is 0.18μ. For the sizer, the range of variables are (in the format — min:granularity:max) W_tran $\rightarrow$ 0.36:0.18:80μ; V_bias $\rightarrow$ 0:0.1:5.0V, the symbols carrying their usual meanings. All transistor lengths are fixed at 0.36μ for ease in sizing.

A. Operational Amplifier Synthesis

The opamp is probably the mostly widely used analog circuit, either by itself or as a part of a bigger analog module [13]. Such modules comprise of comparators, filters, pre-amplification stages, signal converters and so on [13]. In this work, a 2-input 1-output opamp design is synthesized that has a load capacitance (C_L) of 10pF.

100 generations with 100 chromosomes per generation is used for the synthesis run. Table II provides the results. As shown in Table II, at the end of the synthesis run, 15 designs are produced that occupy the non-dominated Pareto optimal front. Again, out of these 15 designs, many designs have the same topology, but obviously with varying device sizes.

Unlike previous works [5], [6], our tool produces both manual-like designs or bookish circuits, as well novel topologies. Fig. 5 shows an example of the former. The circuit contains a current mirror, diode-connected loads, and a differential pair. On the other hand, Figs. 6(a)-(b) are examples of novel designs. These help to prove the robustness of the tool. It not only generates bookish solutions, but also unknown yet compliant designs that can greatly aid the human designer.

Similarly, the block library obtained after the synthesis run comprises of both known opamp-friendly blocks like diff-pairs and cascode connections, as well as a host of unconventionally connected MOS structures. Fig. 6(c) shows some of the highly ranked library blocks obtained at the end of the run.

Now, a second synthesis run is conducted, starting with the library obtained after the above first run. 12 well-compliant optimal design solutions are obtained only within 50 generations. This shows that reusing the library for the same class of design speeds up the synthesis process.

To cite a comparison with previous works, the GP-based technique in [5] used 640,000 solutions per generation and generated a single 5dB amplifier circuit after 45 generations. This clearly proves the faster convergence of our techniques compared to previous GP-based synthesis work.

B. Voltage Controlled Oscillator (VCO) Synthesis

A vco is chosen as our next design benchmark. It is an oscillator whose oscillation frequency can be varied through a control voltage (V_c). The frequency range about the center frequency over which the vco produces a linearly increasing frequency with an increase in V_c is termed as the tuning range of the vco. The vco finds wide applications in function generators, phase locked loops and frequency synthesizers [13].

TABLE II
DESIGN SPECIFICATIONS AND RESULTS

SPECIFICATIONS / OBJECTIVES		
Design-I: Operational Amplifier — Specifications: DC Gain $\geq$ 40 dB, 3dB freq. $\geq$ 50 MHz, UGF (Bandwidth) $\geq$ 5 GHz, Phase margin: $45 - 60^\circ$		
Design-II: Voltage controlled oscillator — Specifications: Center oscillation frequency $\geq$ 2 GHz, Tuning range $\geq$ 0.1 GHz		
RESULTS	**Design-I**	**Design-II**
Number of Pareto-optimal designs	15	6

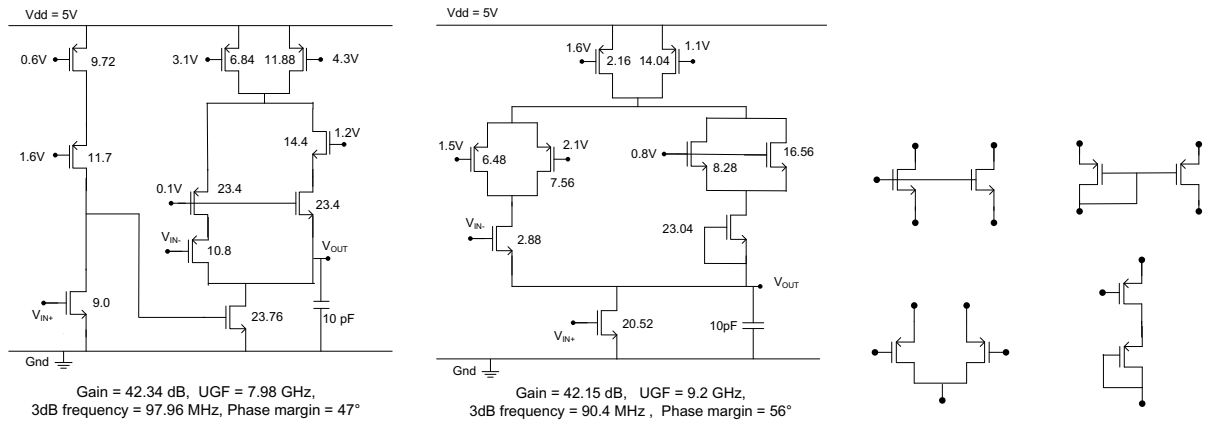


Fig. 6. Operational amplifier design → (a) and (b) Some of the novel designs produced (All transistor widths are in μ, L_{tran} = 0.36μ), (c) Some of the better ranked library blocks obtained

For the vco synthesis, 100 chromosomes are used and the synthesis was run for 50 generations. Fourier analysis is used for performance evaluation, whereby the harmonic closest to the dc component value gives the oscillation frequency. The tool produces six Pareto-optimal compliant designs. This is better than [6], which synthesized a single similar design taking 40 generations, each consisting of 600 chromosomes.

Fig. 7 shows a generated design out of the six optimal solutions obtained. The design produces an oscillation frequency of 4.17 GHz with V_c = 3.5 V. The corresponding output waveform is shown in Fig. 8. Through variation of V_c within an appropriate 2V range, the tuning range obtained is 0.12 GHz. With regard to the block library, it comprises of differential pairs and diode-connected loads among others.

VIII. CONCLUSION

We have introduced a novel graph grammar framework for automated design of analog circuits. Multi-objective guidance has been enabled through the use of NSGA-II. Dynamic generation and subsequent use of design suitable subcircuits speeds up the synthesis run. Our approach has certain advantages when compared to other tree-based generation methods like GP. An opamp and a vco have been synthesized using these techniques. For industrial relevance, our approach involves a generic, context-free, and graph based technique that utilizes minimum design knowledge. Hence it may be applied to structure synthesis across other domains as well.

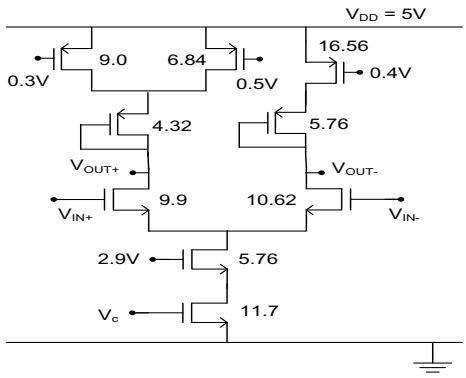


Fig. 7. A vco circuit generated (All transistor widths in μ, L_{tran} = 0.36μ)

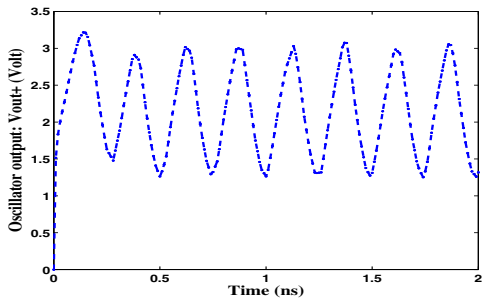


Fig. 8. VCO output waveform: Frequency = 4.17 GHz, V_c = 3.5V

REFERENCES

[1] W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS Opamp synthesis by means of a genetic algorithm," in *Proc. of DAC*, Jun. 1995.

[2] G. Alpaydin, S. Balkir, and G. Dundar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," *IEEE Trans. on Evolutionary Comp.*, vol. 7(3), pp. 240–252, June 2003.

[3] P. Veselinovic, D. Leenaerts, W. van Bokhoven, F. Leyn, F. Proesmans, G. Gielen, and W. Sansen, "A flexible topology selection program as part of an analog synthesis system," in *Proc. of IEEE European Design and Test Conference (ED & TC)*, Mar. 1995, pp. 119–123.

[4] J. D. Lohn and S. P. Colombano, "A circuit representation technique for automated circuit design," *IEEE Trans. on Evol. Comp.*, 1999.

[5] J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Trans. on Evolutionary Comp.*, vol. 1(2), 1997.

[6] T. R. Dastidar, P. P. Chakrabarti, and P. Ray, "A synthesis system for analog circuits based on evolutionary search and topological reuse," *IEEE Transactions on Evolutionary Computation*, Apr. 2005.

[7] A. Das and R. Vemuri, "Topology synthesis of analog circuits based on adaptively generated building blocks," in *Proc. of ACM/IEEE Design Automation Conference (DAC)*, Jun. 2008.

[8] ——, "An automated passive analog circuit synthesis framework using genetic algorithms," in *Proc. of IEEE International Symposium on Circuits & Systems (ISCAS)*, May 2008.

[9] M. I. Campbell, "A generic scheme for graph topology optimization," in *World Congress on Structural and Multidisciplinary Optimization*, 2005.

[10] M. H. Luerssen and D. M. W. Powers, "Graph design by graph grammar evolution," in *IEEE Congress on Evolutionary Computation*, Sep. 2007.

[11] L. Schmidt, H. Shetty, and S. C. Chase, "A graph grammar approach for structure synthesis of mechanisms," *Journal of Mechanical Design*, vol. 122, pp. 371–376, Dec. 2000.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[13] B. Razavi, *Design of Analog CMOS Integrated Cir.* McGraw-Hill.

[14] R. A. Rutenbar, G. G. E. Gielen, and B. A. Antao, *Computer-Aided Design of Analog Intg. Cir. and Sys.* Wiley-IEEE Press, May 2002.