Automated Extraction of Expert Knowledge in Analog Topology Selection and Sizing

Trent McConaghy[1,2], Pieter Palmers[1], Georges Gielen[1], Michiel Steyaert[1]
1 ESAT-MICAS, K. U. Leuven, Kasteelpark Arenberg 10, Leuven, Belgium
2 Solido Design Automation Inc., Saskatoon, Canada
{Trent.McConaghy, Pieter.Palmers, Georges.Gielen, Michiel.Steyaert}@esat.kuleuven.be

ABSTRACT

This paper presents a methodology for analog designers to maintain their insights into the relationship among performance specifications, topology choice, and sizing variables, despite those insights being constantly challenged by changing process nodes and new specs. The methodology is to take a data-mining perspective on a Pareto Optimal Set of sized analog circuit topologies, then doing: extraction of a specs-to-topology decision tree; global nonlinear sensitivity analysis on topology and sizing variables; and determining analytical expressions of performance tradeoffs. These approaches are all complementary as they answer different designer questions. Once the knowledge is extracted, it can be readily distributed to help other designers, without needing further synthesis. Results are shown for operational amplifier design on a database containing thousands of Pareto Optimal designs across five objectives.

1. INTRODUCTION

Analog designers use their experience and intuition to choose circuit topologies and to design new topologies. Unfortunately, the topology used may not be optimal, with possible adverse affects on the related product's performance, power, area, yield, and profitability. The suboptimal design may be because the design is on an unfamiliar process node, the designer is time-constrained, or simply because the designer just doesn't have deep experience (it is well recognized that analog design takes decades to master [1]). That said, it still means that a suboptimal topology may be used. Hence, it is desirable to provide support for the designer in topology selection and design, and ideally to catalyze the learning process. Prior CAD research has focused on automated topology selection & design (with nice successes [2]), but has had little emphasis on giving insight back to the user. In fact, by deferring control to automated tools, a designer's learning might slow. Even worse, the designer could end up poorly-equipped when problems arise.

This paper asks: *is there a means to help analog designers maintain and build expert topology-performance knowledge*? The starting point is a recent innovation, which traverses thousands of circuit topologies to automatically generate a database of Pareto-optimal sized circuits [3]. Contributions are:

1. A data-mining perspective on the database to extract the following expert knowledge: (a) a specs-to-topology decision tree, (b) global nonlinear sensitivities on topology and sizing variables, and (c) analytical performance-tradeoff models.
2. A suggested flow in which even reluctant users can conveniently use the extracted knowledge (Figure 1). The database generation and knowledge extraction only needs to be done once per process node, e.g. by a single designer or a modeling group. The knowledge can be stored in a document (e.g. pdf, html), and simply made available to other designers.

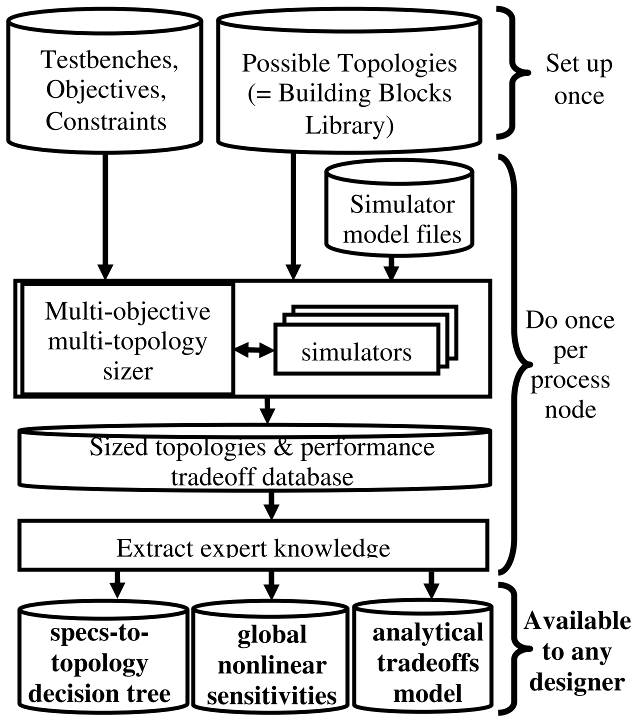
Testbenches, Objectives, Constraints
Possible Topologies (= Building Blocks Library)
Set up once
Simulator model files
Multi-objective multi-topology sizer
simulators
Do once per process node
Sized topologies & performance tradeoff database
Extract expert knowledge
specs-to-topology decision tree
global nonlinear sensitivities
analytical tradeoffs model
Available to any designer

Figure 1: Target flow. The extracted knowledge is readily available to all designers, without requiring them to invoke automated sizing.

This paper's knowledge extraction procedures will be explained using a reference database, generated as described in section 2. Section 3 describes how a specs-to-decision tree is extracted from the database. Section 3.1 describes extraction of global nonlinear sensitivities, and Section 4 extraction of analytical tradeoffs model. Section 5 concludes.

2. GENERATION OF DATABASE

This section describes the setup to generate the sized-topologies database. Table 1 lists the search space and goals. The technology was 0.18μm CMOS with 1.8 V supply voltage. The output DC voltage was 0.9 V, and load capacitance 1pF. HSPICE^{TM} was the simulator.

Table 1: Problem Description

Search Space	50 topology & sizing & biasing 50 variables, comprising 3528 possible opamp topologies (folded, cascode, source degen, 1 & 2 stage, …)
Objectives	5 objectives: Maximize GBW, minimize power, maximize DC gain, maximize dynamic range, maximize slew rate
Constraints	Device operating constraints, DC gain > 20 dB,

	GBW > 1e6, phase margin > 65°, pole margin, dynamic range > 0.1, slew rate > 1e6

We use MOJITO [3] search, which is an evolutionary algorithm with an age-layered population structure [4] to prevent premature convergence, and NSGA-II at each layer to handle constrained multi-objective optimization [5]. Settings were: 100 individuals per age layer; 10 age layers; maximum age per layer: 19, 39, …, 159, 179, ∞. While MOJITO is tuned for cell-level circuits like op amps and bias generators, reference [6] generates system-level multi-objective multi-topology databases, e.g. for ADCs. The knowledge-extraction approaches of this paper apply to any technique that can generate such a database, cell or system level.

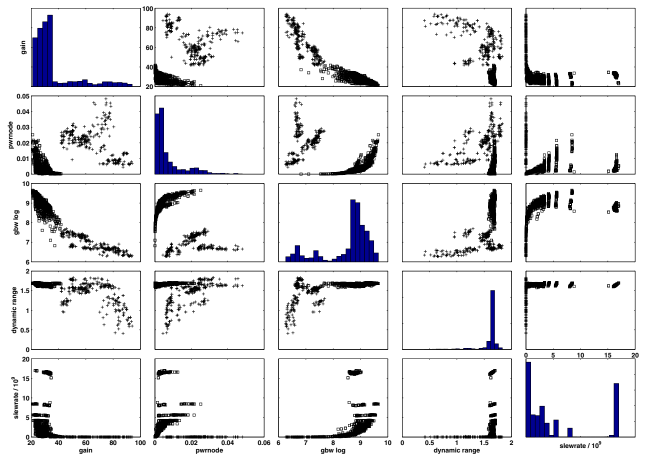

Figure 2: Grid illustrating the Pareto Front (tradeoff of performances). The diagonal entries show histograms of performance; the rest show two-dimensional projections from the five objectives of (top to bottom, left to right): gain, power, GBW, dynamic range, and slew rate. The "□" are 1-stage amplifiers, and "+" are two-stage amplifiers.

The MOJITO run took approximately 12 hours on a Linux cluster having 30 cores of 2.5 GHz each. 180 generations were covered, traversing 3528 possible topologies and their associated sizings. It returned a database of 1576 Pareto-Optimal sized topologies. Figure 2 shows a grid of 2d scatterplots and histograms for the five performance objectives. From the histograms, we can get a quick picture of the distribution and bounds of performances. From the scatterplots, we begin to understand the limits of combinations of performances and take note of trends. The one-stage topologies only occupy a different region of performance space and follow a markedly performance trend than two-stage. The two-stage topologies have several sub-clusters of performances, hinting at further decomposition of topology types.

3. EXTRACTION OF SPECS-TO-TOPOLOGY DECISION TREE

This section describes the automatic extraction of decision (CART) trees [7] that map from performance values to topology choice. Decision trees have a double use: they can directly suggest a choice based on inputs, yet also expose the series of steps underlying the decision. CART trees are in widespread use, such as medicine: "In medical decision making (classification, diagnosing, etc.) there are many situations where decision must be made effectively and reliably. … Decision trees are a reliable and effective decision making technique that provide high classification accuracy with a simple representation of gathered knowledge." Decision trees have not gone unnoticed in analog CAD either, as they have been proposed as the centerpiece of topology-choosing "expert systems", e.g. [9]. Unfortunately, these trees had to be manually constructed which took weeks to months of effort, and were based on rules of thumb that became obsolete as soon as the process node changed. In contrast, this paper constructs the specs-to-topology decision tree *automatically* from data. This is only possible now, because a prerequisite to get the data was a competent multi-topology multi-objective sizer that could output a diverse set of topologies.

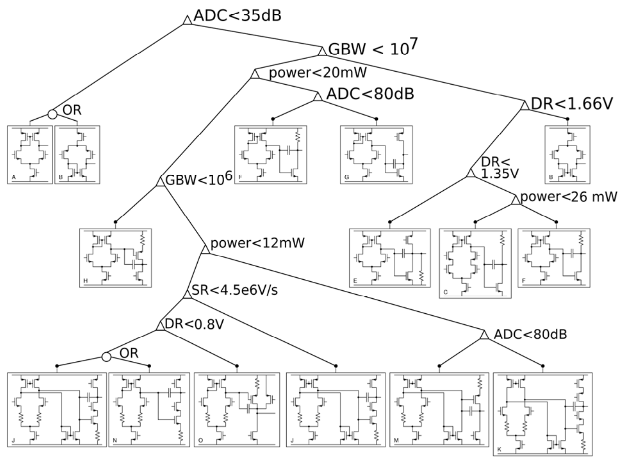

Figure 3: A decision tree for going from specifications to topology. Unlike past expert-system approaches, this was automatically generated.

As input to the tree construction, each topology was assigned a class; the decision variables are the objectives used for generating the dataset; the dataset itself consisted of the non-dominated individuals. Figure 3 shows the tree that was automatically generated. It provides insight into what topologies are appropriate for performance ranges, and actually even gives a suggested topology from a set of input specs. We see that low-frequency gain (A_{DC}) is the first variable selected on, and following through the tree, we see that all specifications play a role for selecting some topologies: gain-bandwidth (GBW), power, slew rate (SR), and dynamic range (DR). When specifications require low gain, the tree suggests single-stage topologies, and two stages when higher gain is required. In cases where very large gain is required with a limited power budget, a two-stage amplifier with large degrees of cascoding (K) is suggested. If power is less of an issue, one can also use a non-cascoded two-stage amplifier (G). Since only Pareto-optimal individuals are used to generate the tree, the choice for the more power-efficient variant implies lower performance for one or more other metrics (in this case e.g. dynamic range). Also reassuring is that while there were thousands of *possible* topologies, just 15 were returned. This is in line with many analog designers' expectation that just a couple dozen opamp topologies serve most purposes. The challenge, of course, is which topologies those are, and for what specs they are appropriate.

It is important to remember that the tree is a classifier at its core, which can help avoid reading too much into it. To aid understanding, we outline its construction. The algorithm starts with just a root node holding all data points. From among all possible combinations of *{split_variable, split_value}*, it chooses the one from that splits off the most data points. That split creates a left and right child, each getting a subset of the data according to the chosen variable and value. The algorithm recurses, splitting each leaf node until there is just one sample at each leaf node or another stopping criterion is hit. There are CART extensions to capture sensitivities to exact split values, but this is at a cost of additional complexity in the reported tree. Another extension is for the user to give preference to choosing certain split variables first, which may result in interesting alternative trees.

An additional benefit of tree extraction is based on there being more than 2-3 objectives, which means the raw data is difficult to visualize; the tree gives alternate perspective among 5 objectives, highlighting which topologies cover which performance regions.

3.1 GLOBAL NONLINEAR SENSITIVITY ANALYSIS

The aim here is to address questions such as: “how much does each topology choice matter? Should I be changing the topology or device sizes? *Which* block or variables should I change?” There may even be more specific questions, such as “how much does cascoding affect gain?” Our approach to handle such questions is to perform global nonlinear sensitivity analysis. We need to be global -- across the range of variables -- because we have thousands of training points, and one cannot do small perturbations on integer-valued design variables such as topology-choice variables. We cannot assume linear because not being local means a Taylor approximation does not apply; topology-choice variables are categorical; and small ad-hoc tests showed that linear models fit poorly.

The sensitivity extraction flow we follow for each performance metric y is:

1. Given: a set of $\{\mathbf{X}, \mathbf{y}\} = \{\boldsymbol{x}_k, y_k\}$, k=1..N Pareto-optimal points where $\boldsymbol{x}_k$ is a d-dimensional topology/sizing input point j and y_k is a corresponding performance value
2. Build a regression model m that maps $\mathbf{X}$ to $\mathbf{y}$
3. From m, compute nonparametric sensitivities $\mathbf{e} = \{e_i\}$, i=1..d
4. Return **e**

Steps 2 and 3 have specific challenges. Step 2, regressor construction, needs to handle numerical *and* categorical input variables, which prevents usage of polynomials, splines / piecewise polynomials, support vector machines, kriging, and neural networks. CAFFEINE [10] works on categorical variables, but it would run very slowly on 50 input variables and 1500 training samples. A CART tree is not appropriate because the model needs to do regression, not classification. However, a relatively recent technology achieves the effect of regression on CART trees by boosting them: stochastic gradient boosting (SGB) [11]. SGB also has acceptable scaling and prediction properties, so we employ it here.

Step 3 needs to compute sensitivities from the model, yet be global, nonlinear, and ideally, nonparametric. The proposed solution defines global nonlinear sensitivity (*impact*) for a variable x_i as the relative error that a scrambled input variable x_i will give in predicting, compared to other variables $\{x_j\}$, j=1..d, $j \neq i$ when they are scrambled. Table 2 gives the algorithm that uses this concept to extract impacts (inspired by chapter 10 of [11]). *NS* is number of scrambles; nmse is normalized mean-squared error.

Table 2: Procedure RegressorImpacts()

```
Input: X, y, m
Output: e
For $i$ = 1 to $d$:
    $e_i = 0$
    Repeat NS times:
        $\mathbf{X}_{scr}$ = X except randomly permute row $i$
        $\mathbf{y}_{scr}$ = $m$.simulate($\mathbf{X}_{scr}$)
        $e_i = e_i$ + nmse($\mathbf{y}$, $\mathbf{y}_{scr}$)
$S = \sum_{i=1}^{d} e_i$
$e_i = \frac{e_i}{S}$, $i$=1..$d$
```

With this flow, we extracted sensitivities for each performance. SGB and CART were coded in about 500 lines of python. SGB parameters were: maximum number of trees = 500, learning rate α=0.10, minimum tree depth = 2, maximum tree depth = 7. *NS* = 500. Build time for an SGB model was about 15 s on a 2.0 GHz Linux machine; impact extraction from the model took about 25 s.

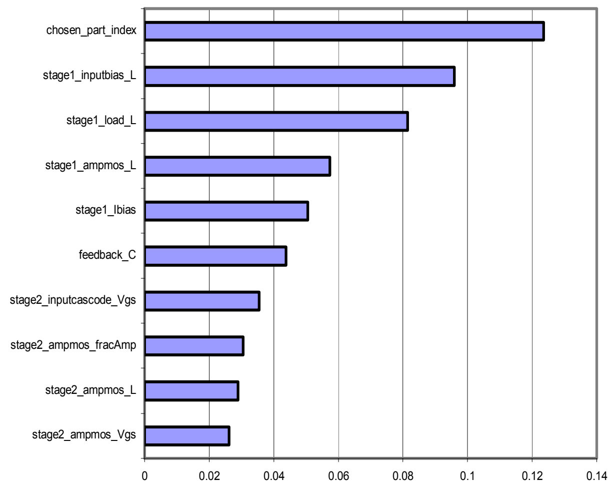

Figure 4: Relative impact of topology, sizing, and biasing variables on GBW, for 10 most important variables

Figure 4 illustrates results for GBW’s 10 most-impacting variables. We see that the most important variable is *chosen_part_index*, which selects one vs. two stages. The variables that are commonly associated with the GBW of opamps -- bias current of the first stage and size of compensation capacitance -- also show up. Interestingly, the figure also indicates a large influence of the length of the transistors in the first stage (input, folding and load). This can be readily explained: these lengths directly influence impedance on the internal nodes, and hence the location of the non-dominant pole. The phase margin requirement (>65°) translates into the requirement that this non-dominant pole frequency is sufficiently higher than the GBW (approx 2x) [13]. It is also interesting to see that for GBW, only one topology parameter made it into the top 10 variables; sizing parameters comprise the other 9. This means that once one vs. two stages is chosen, changing the right sizing variables will make the biggest difference to GBW. Of course, the most sensitive variables can be different for different performance metrics, and the designer must consider all metrics.

4. EXTRACTION OF ANALYTICAL PERFORMANCE TRADEOFFS

Designers often they manually manipulate equations that relate performance tradeoffs [13][14]. Equations facilitate understanding because a direct relationship is expressed and the model is manipulatable to change the output variable. The problem is that hand-derived analytical expressions are based on 1st or 2nd order approximations and may have little relation to the process technology, therefore possibly having error of 20% or 50% or more. The aim here is to extract analytical equations that capture performance tradeoffs, which are naturally in line with the process.

To generate the models, we start with the tradeoffs database, set one performance as the output variable, and the remaining performances as input variables; then we apply CAFFEINE [10]. CAFFEINE automatically generates whitebox models to capture the mapping; it is not constrained to a particular functional template such as polynomials. CAFFEINE actually generates a set of models which trade off complexity for error. CAFFEINE settings were the same as [10], though due to code improvements the runtime was about 30 minutes.

Table 3 shows results for GBW. We expected gain to be strongly related to be GBW, and it turns out that just a linear relation between the two will get < 9% training error. But for a better fit, more complex nonlinear relations are needed leading up to an inverse relationship of GBW with gain or √gain. Slew rate is also needed for a reasonable model. Interestingly, dynamic range and power are not needed to get within 3.5% training error. Cross-examination with the scatterplots (Figure 2) confirms that the strongest tradeoffs are among gain, GBW, and slew rate.

Table 3: Whitebox models Capturing Performance Tradeoff

Train error	Log(GBW) Expression
8.7 %	10.28 - 0.049 * gain
7.8 %	12.57 - 0.69 * √gain
7.3 %	5.65 + 86.5 /gain + 2.92e-11 * slewrate
6.8 %	5.72 + 80.2 / gain + 4.75e-06 * √slewrate
5.7 %	7.30 + 47.76 / gain - 3430 / √slewrate
4.1 %	4.48 + 24.9 / √gain - 8.60e6 / (gain^2 * √slewrate)
3.5 %	16.9 / (1 + 0.15 * √gain + 1.44e-22 * slewrate^2 + 2.56e6 / (gain^2 * √slewrate))

5. CONCLUSION

This paper presented a methodology to help designers maintain their expert insights on the topology-sizing-specs relationship, which is a challenge due to changing process nodes and more. The approach is to take a data-mining perspective on a Pareto Optimal Set of sized analog circuit topologies: extract a specs-to-topology decision tree (via CART); do global nonlinear sensitivity analysis on topology and sizing variables (via SGB and a variable-scrambling heuristic); and generate analytical whitebox models to capture tradeoffs among performances (via CAFFEINE). These approaches are all complementary as they answer different designer questions. Once extracted, the knowledge for a circuit type on a process node can readily be distributed to other designers, without need for more synthesis. Results are shown for operational amplifier design on a database containing thousands of Pareto Optimal designs across five objectives. As a final note, we must emphasize once again that these techniques are meant to *augment* designer experience, not replace it. The designer is key.

6. REFERENCES

[1] J. Williams, Editor. Analog Design: Art, Science, and Personalities. Newnes Press, 1991

[2] R.A. Rutenbar, G.E. Gielen, and B.A. Antao, eds., Computer-Aided Design of Analog Integrated Circuits and Systems, IEEE Press, Piscataway, 2002

[3] T. McConaghy et al., "Simultaneous Multi-Topology Multi-Objective Sizing Across Thousands of Analog Circuit Topologies", *Proc. DAC*, 2007

[4] G.S. Hornby, "ALPS: The Age-Layered Population Structure for Reducing the Problem of Premature Convergence," *Proc. GECCO*, 2006, pp. 815-822

[5] K. Deb et al., "A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II," *IEEE Trans. Ev. Comp.* 6(2), 2002

[6] E. Martens, G. Gielen, "Top-down heterogeneous synthesis of analog and mixed-signal systems"*, Proc. DATE*, 2006

[7] L. Breiman et al, Classification and Regression Trees, Chapman & Hall, New York, 1984

[8] V. Podgorelec et al., "Decision trees: an overview and their use in medicine"*, Journal of Medical Systems* 26(5), Oct. 2002, pp. 445-63

[9] H.Y. Koh et al., "OPASYN: A Compiler for CMOS Operational Amplifiers," *IEEE Trans. CAD* vol. 9, Feb 1990

[10] T. McConaghy, T. Eeckelaert, G. Gielen, "CAFFEINE: Template-Free Symbolic Model Generation of Analog Circuits via Canonical Form Functions and Genetic Programming", *Proc. DATE*, 2005

[11] J. Friedman, "Stochastic Gradient Boosting," *J. Comp. Statistics & Data Analysis* 38(4), 2002, pp. 367-378

[12] T. Hastie, R. Tibshirani, J. H. Friedman. The Elements of Statistical Learning. Springer, 2001.

[13] W.M.C. Sansen, Analog Design Essentials. Springer, 2006.

[14] B. Razavi, Design of Analog CMOS Integrated Circuits. McGraw-Hill, 2000