Design of Posynomial Models for Mosfets: Symbolic Regression Using Genetic Algorithms

Varun Aggarwal[1] and Una-May O'Reilly[2]

[1] Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology varun_ag@mit.edu
[2] Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology unamay@csail.mit.edu

Summary. Starting from a broad description of analog circuit design in terms of topology design and sizing, we discuss the difficulties of sizing and describe approaches that are manual or automatic. These approaches make use of black-box optimization techniques such as evolutionary algorithms or convex optimization techniques such as geometric programming. Geometric programming requires posynomial expressions for a circuit's performance measurements. We show how a genetic algorithm can be exploited to evolve a posynomial expression (i.e. model) of transistor (i.e. mosfet) behavior more accurately than statistical techniques in the literature.

1 Introduction

Analog circuit design remains an important part of electronic design even after the advent of digital electronics. This is because some components of an electronic system must be analog. Some examples are voltage reference circuits, converters (analog to digital and vice versa), clock generators, circuits for processing the input signal before it is digitized (e.g. filters and amplifiers) and circuits for processing analog output signals. Because research also shows [12] that analog systems can be designed to consume several orders of magnitude lesser power than digital circuits, interest in analog design remains strong. Complementarily, there is active interest in the improvement and development of methods for computer-aided design (CAD) of analog circuits. Design and verification of analog circuits has not yielded to automation and thus analog design is a bottleneck in achieving short time-to-market and robustness.

An analog circuit is composed of components that include transistors (fets, bipolar junction transistors), resistors, capacitors and inductors.[1] Each of these components has a certain behavior that is expressed in terms of the

[1] Inductors are difficult to fabricate and are generally used in RF circuits.

current(s) flowing through it and the voltage(s) across its nodes. This behavior depends on one or more parameters that are numerically expressed. For example, capacitance is the parameter for a capacitor and width and length parameterize a mosfet. Models to express the behavior of capacitors and mosfets are shown in Figure 1 and Table 1. Here, the mosfet model is inaccurate (a first order approximation).

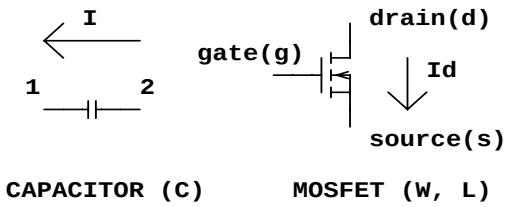


Fig. 1. Basic Analog

Component	Model
Capacitor:	$v_{21} = \frac{i}{sC}$
Mosfet:	$\mathrm{I}_d = k\frac{W}{L}(V_{gs} - V_t)^2(1 + \lambda V_{ds})$

Table 1. Capacitor and (first order) Mosfet models.

These components are connected to each other to form a circuit topology that implements a certain function. A circuit topology may be defined as a certain (wired) connection of the nodes of components with constraints on unspecified parameters to implement a given function. Figure 2 shows a topology of a *differential pair* that implements the differential amplification function given in Equation 4. The topology is a connection of mosfets, resistors and a current source. There are constraints on the parameters of the components as expressed in Equations 2 and 3. The second constraint is popularly termed a *matching requirement* in the analog design community. While there are additional constraints on parameters to keep the mosfet in saturation, we omit these for simplicity. For the given differential pair topology, the parameters of the components must be determined according to its *gain* requirements as per Equation 1. In general, the step of determining the parameters of a topology is called *circuit sizing*.

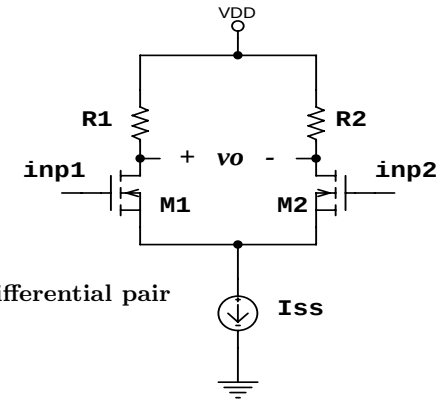


Fig. 2. Differential pair

$$a_v = f(R, W_i, L_i, I_{ss}, V_{DD}) \tag{1}$$

Constraints

$$\frac{W_1}{L1} = \frac{W_2}{L2} \tag{2}$$

$$R1 = R2 = R \tag{3}$$

CircuitFunctionality

$$v_o = -a_v(v_{inp1} - v_{inp2}) \tag{4}$$

Here a_v is gain, v_0 is AC output and v_{inp1}, v_{inp2}: AC input

Every analog circuit has a function and performance measurements.[2] For instance, the function of a differential pair is that its output voltage is proportional to the difference of its input voltages. The proportionality constant, *gain* or a_v, is a performance measurement of the differential pair. Other measurements for a differential amplifier are, for example, lower bound or upper bound on gain, unity gain frequency, phase margin, noise and slew rate. Given a model of the behavior of each component, its parameter values and the topology, the measurements will have specific values. The analog design problem is an inversion of this derivation: given a required function, a set of requirements for (the values of) performance measurements and a class of components (e.g.,

[2] Performance measurements are also called specifications. The term *specification* in the analog domain is also used in the context of a requirement. In this submission, to avoid ambiguity we avoid the use of "specification" unless the context is completely clear.

use only mosfets), a designer must come up with the connection of components (topology) and the component parameters to meet the function and performance measurements requirements.

There are two steps in analog circuit design. First, a topology is designed that satisfies the functional requirements of the circuit. There are always multiple candidate topologies and the decision to choose one among them is informed by the performance measurements requirements. In the context of our example, a difference amplifier can be realized via the differential pair in Figure 2 or via a differential pair followed by a simple amplification state. If high gain is a performance requirement, the latter will be chosen and for low gain the former will be chosen. However, the latter circuit will consume more power given it has more components and this might be a crucial tradeoff in the design.[3] This first step of topology selection involves making expertise informed decisions among choices that have unfinalized outcomes because the parameter values of the components are not decided upon.

The second step is circuit sizing, that of determining the parameter values of all components to meet the required performance measurements. Given the parameter values, the performance of the circuit can be determined, but as in the case above, we need to solve the inverse problem, i.e. to find the components' parameters' values given the performance measurement requirements.

The next two sections give an insight into how the parameters of circuits map to performance measurements and elucidate the methodologies to do sizing.

1.1 Circuit Sizing: Complex behavior models and Interconnection Effects

Circuit sizing is complex even though it is possible to determine circuit performance measurements when given the behavior of components, their parameter values and interconnection,

The resistors and capacitors have simple linear behavior, however this is not the case with the transistor. Simplistically, the transistor can be viewed as an active device with a variable current that is controlled either by current (BJT) or voltage (FET) in a non-linear relationship. The actual behavior of transistor is far more complex with multiple interactions between its three nodes (as observed on fabrication). The model of this relationship depends on the technology used (different substrate, doping concentration, fabrication methodology, minimum feature size, etc. and their combinations imply different technologies) to fabricate the device. The models may differ a lot depending on the fabrication technology.

Beyond single component behavourial complexity, the behavior of the entire circuit is even more complex due to interaction between the complex

[3] The description and tradeoffs have been kept simple throughout this example for clarity. In reality, they are much more complex.

models of the individual devices. One result is very large expressions for performance measurements. These are computationally expensive to solve (in [13], it is shown that the gain expression of a 3 transistor circuit consist of 63 distinct terms in numerator and 908 distinct terms in denominator). Another factor in increased complexity is that many performance measurements do not have closed form expressions. This implies they must be determined iteratively or numerically (e.g., slew rate). This leads to a computationally expensive and non-intuitive mapping between parameter values and performance measurements.

An accurate, yet time consuming means of measuring circuit performance is to use a circuit simulator with component models acquired from the fabrication phase. SPICE[14] is one such circuit simulator which handles all the model and inter-model complexity. It takes as input the circuit expressed as a netlist and outputs its behavior, from which the performance measurements can be derived. SPICE is a world standard for circuit simulation and the final verification tool for the circuit before it goes to silicon or PCB.

1.2 Circuit Sizing: Manual and Automatic Methodologies

There are both manual and automatic methodologies for sizing circuits.
Manual Design: Given the dismal cognition-starved, non-intuitive factors in sizing, it is hard to imagine how analog circuit sizing could be done manually. The reality is contrary to this intuition. Analog circuit sizing has been conventionally done manually and even today, is done manually by highly-paid analog design engineers!

Roughly, the design methodology is the following: The designer uses an approximate quantitative first order model for the transistor behavior (strongly informed by his prior design experience with the fabrication technology). The simplified quantitative expressions are used to model the interaction between the components to come up with simplified expressions for the performance measurements. Using these expressions, the component parameters are worked out. This is an iterative process where at one step the designer may select parameters that manage to satisfy one measurement requirement but which "fall out" on others. At the next step some adjustment is made to (hopefully) bring the measurements closer (or completely) to requirements. Intuition of the designer regarding the higher-order interactions between components and feedback from simulations (which in turn sharpens intuition) informs the readjustments to components to meet specifications. As the designer works more and more on a given topology, intuition about the higher order interaction of components improves and yields expertise in sizing circuits optimally. At each step, the design is simulated on SPICE for verification with respect to requirements.
Automated blackbox Optimization: SPICE technically performs a mapping function between parameter values and performance measurements. It is empirically known that such a direct mapping function would be misbehaved

and multimodal. Also, performance measurements are coupled and in tradeoff. The problem is very high dimension in input variable space; the number of parameters varies from 10s to a few hundreds. For instance, a simple opamp has around 13 parameters that need to be set. Thus, rather than replacing SPICE with a function, it can be exploited as a blackbox. This conceptualization of SPICE lays the foundation for casting the sizing problem as a large scale multi-objective optimization problem for which a blackbox function is available. There has been a lot of work in applying different stochastic blackbox optimization algorithms (also termed non-structural) to sizing such as genetic algorithms [17] and simulated annealing[16]. Genetic algorithms and programming has been applied to the combined topology and sizing design problem [9, 7, 15, 1].

Equation-based approaches have also been used instead of invoking SPICE in the optimization loop[6]. Here, a simplified model for the transistor is used and multiple symbolic equations are derived to express the performance measurements. These equations, though not completely accurate, take much less time than SPICE to provide the performance measurement values. The process of sizing is thus accelerated at the cost of accuracy.

Equation-driven global optimization: When the sizing problem is solved by blackbox optimization techniques that use multiple symbolic expressions which measure the circuit's performance, any exploitation of the *structure* of the performance measurement equations is ignored. This observation reveals the possibility that a structured optimization algorithm (like linear programming, quadratic programming), if it could exploit the structure of the symbolic equations, would also be able to solve sizing. In an exciting development in sizing methodology, [11] showed that in the case of an opamp, circuit performance measurement equations could be accurately yet approximately expressed in *posynomial* form (which we shall define in detail in Section 2) to be solved by geometric programming. The approach used inaccurate transistor models and considered only simple interconnection effects. Geometric programming is a structural optimization technique which can determine the global optimum for objectives and constraints expressed in posynomial form. It uses interior-point methods [2] and solves in a few seconds. In [11, 5, 3], it was shown that geometric programming can be used to size various analog circuits such as PLLs, opamps, OTAs and inductor circuits in a few minutes.

As convenient as the geometric programming technique might initially appear, the devil is in its details. Most significantly, all circuits do not render accurately to posynomial equations that express *real* (i.e. complex) transistor models and multiple interaction effects. When these inaccuracies then are translated into problem objectives and constraints, they lead to the determination of a faulty global optimum, i.e. the optimum of an inappropriate (or wrong) problem. The extent to which simple interaction terms impose inaccuracy on the equations depends on the topology and needs to assessed on a case-by-case basis. There may be a large unacceptable impact for a certain topology, while it could be trivial for another.

This paper investigates finding accurate posynomial expressions that are high fidelity models of a real transistor. With more accurate performance measurement equations incorporated into geometric programming's objectives and constraints, there is improved potential for sizing optimization. We have used genetic algorithms to design posynomial models for mos transistors.

2 Geometric Programming

To be more explicit, geometric programming [2] is a special type of convex optimization which exploits the posynomial or monomial form of objectives and constraints. Let $\overline{x}$ be a vector of n real, positive variables. A function f is called a posynomial function of $\overline{x}$ if it has the following form:

$$f(x_1, \ldots, x_n) = \sum_{k=1}^{t} c_k x_1^{\alpha_{1,k}} x_2^{\alpha_{2,k}} \ldots x_n^{\alpha_{n,k}}, \quad c_j > 0, \quad \alpha_{i,j} \in \Re$$

When $t = 1$, the expression is called a monomial. Geometric programming solves an optimization problem of the following form:

$$\begin{array}{ll} \text{minimize} & f_0(\overline{x}) \\ \text{subject to} & f_i(\overline{x}) \leq i, \; i = 1, \ldots, m, \\ & g_i(\overline{x}) = 1, \; i = 1, \ldots, p, \\ & x_i > 0, \quad i = 1, \ldots, n \end{array}$$

Here f_i and f_0 are posynomials while g_i are monomials. A geometric program can be solved for the global optimum in a few seconds using interior point methods. In the next section, we will show how an opamp sizing can be expressed as a geometric program. Note that posynomials exclude the expression of a negative term but can express a fraction.

2.1 Circuits as a Geometric Program

Circuit performance measurements fall in two categories: small signal and large signal. The process of expressing the circuit for geometric programming involves expressing its small signal and large signal measurements as posynomials.

Small Signal Measurements: The transistor despite being a non-linear device has nearly linear behavior for small changes in voltage and current. This fact is exploited (in general) by all analog circuits to implement different behaviors. The measurements of these behaviors are termed small signal performance measurements. One example is gain, the ratio of the output voltage and the input voltage when the circuit is excited by a *small* input voltage.

The circuit small signal performance is measured by expressions that interpret the interconnection of the topology and embed a *small signal model* of

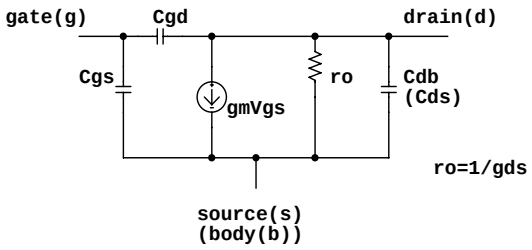
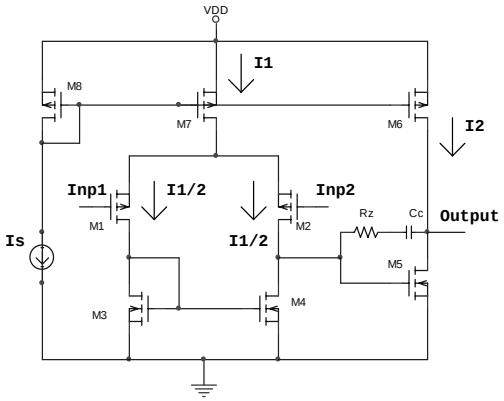


Fig. 3. Small Signal model of mosfet

the mosfet. A small signal model for the mosfet is shown in Figure 3. Its parameters gm, gds (or r_o, Cdb, Cgs and Cgd) are also shown in Figure 3. The symbol g stands for transconductance, r for resistance and C for capacitance. These parameters are themselves each a function of the transistor width (W), length (L) and the current (I_d) flowing through it.

Large signal Specifications: The transistor is essentially a non-linear device. The performance measures related to its non-linear characteristics are called large signal specifications and do not consider the linearity assumption of the small signal model. Slew rate and voltage swing are examples of large signal performance measures for an opamp. The value of current flowing through the transistor is also ascertained by the transistor large signal model, which is required to express the value of small signal parameters (small signal parameters are a function of W, L and I_d).



Current Constraints (Example.)

$$I_1 = \frac{W_7 * L_8}{L_7 * W_8} Is$$

Small Signal Measures

$$G = \frac{gm_2 * gm_5}{(gds_2 + gds_4)(gds_5 + gds_6)}$$

$$w_c = \frac{gm_2}{Cc}$$

Large Signal Measure

$$Veff_1 + Veff_7 < VDD - Vcm(\max) + V_{t1}$$

Fig. 4. opamp for geometric programming example

The process of expressing the circuit as a posynomial for geometric programming is shown for an opamp in Figure 4. To size the given circuit, the width (W_i) and length (L_i) of all transistors, the input current (I_{ss}), the parameter values of the resistor (R) and capacitor (C) have to be determined. The currents flowing through all mosfets are expressed using the large-signal behavior of the mosfet and topological connection information. They are a function of the input currents, W and L of transistors (example included in Figure 4). These are monomial constraints. Small signal performance measures, for instance, (G) and unity-gain frequency (w_c) are expressed in terms of small signal parameters. For maximization of gain or imposing a lower bound constraint on it, the inverse of the gain has to be a posynomial. This can be achieved by expressing gds and gm as posynomial and monomial respectively. Same holds true for w_c. The large signal performance measures may be expressed using the large signal model parameter Vt, $Veff$ and substitution of empirically derived values. The figure shows the constraint on V_{eff} of transistor $M1$ and $M7$ due to a lower bound of the maximum common-mode input voltage ($Vcm(max)$). This can also be expressed as a posynomial constraint.

In this formulation, each value of the small signal parameters (for e.g., gm) and certain large-signal parameters (for example, Vt) of a mosfet must be expressed as a posynomial (or monomials in certain cases) function of the width, length and current flowing through the mosfet. This is what we dub the MOS posynomial modeling problem.

3 The MOS Posynomial Modeling Problem

The goal of MOS posynomial modeling is to express all small signal parameters and some large signal parameters (henceforth called output variables, Y_i) as posynomial function of the transistor width (W), length (L) and the current (I_d) (henceforth called, input variables, $\overline{X}$). This is shown below.

$$Y_i = \sum_{k=1}^{t} a_k W^{\alpha_{k,1}} L^{\alpha_{k,2}} I_d^{\alpha_{k,3}}, \quad a_j > 0, \quad \alpha_{i,j} \in \Re$$

As can be seen, the number of terms, the value of the coefficients of each term and the exponent of each input variable in each term fully specifies the posynomial expression.

The values of the output variables for any value of input variables can be found by SPICE simulation of the transistor. A large set of points (in the order of 10s of thousands) enumerating the value of output variables for values of input variable is available[4]. Our GA must find a posynomial function for each output variable in terms of the input variables, which minimizes the

[4] The large number of sampling points is necessary for comprehensive representation

mean square error between the actual value and calculated value (posynomial function) for each variable over the complete data set.

In earlier studies, log regression has been used to fit monomials [2], however this approach cannot be extended for fitting posynomial models. In other works [4, 10], posynomial models with exponents as integers between -2 and 2 have been suggested. Like a GA, these approaches make no claim to finding a globally optimal solution. In our approach, the exponent can take any real value which gives the model more expressive power.

4 Our Genetic Algorithm for MOS Modeling

We have designed a genetic algorithm (GA) to synthesize a posynomial for each of the output variables of a mosfet model. The genetic algorithm has to be executed for each output variable separately. The demonstrated method is generic for fitting a posynomial to a given set of input and output data.

4.1 Posynomial Representation:Genotype to phenotype mapping

The mapping of genotype of phenotype is shown in the Figure 5. Our phenotype is a posynomial expression. The genotype is a matrix of real numbered values as shown in Figure 5. Each row represents a term of the posynomial. The number of rows is fixed. A choice parameter associated with each row decides whether the row is actually used or not (1:used, 0:don't care). This allows us to have posynomials with varying number of terms in the population. The number of rows is equivalent to the maximum number of possible terms in the posynomial. Each column is associated with one of the 3 input variables. The value in a cell encodes the exponent of the variable (represented by the column) for the term (represented by the row). All cell values are in a specified range $[minVal, maxVal]$. The coefficient of each term is not a part of the genotype.

This genotype might be interpreted to state that the value of the choice parameter helps exploration of posynomials with different number of terms. This will not be sufficient due to high possibility of bloat. Because of how we determine the coefficients (that are not in the genotype), a coefficient may become zero valued. This incorporates automatic *feature selection* (term selection) in the algorithm. The determination of coefficient values and how the number of terms of the posynomial varies will be discussed in terms of fitness evaluation in Section 4.2. This representation expresses the exponents as real values (in a given bound) and our algorithm is not biased to set exponents as absolute zeros. Another option for the model matrix would be to restrict the exponents to integers, with a bias to pushing small exponent terms to zero. Intuitively, while the current representation would evolve expressions with fewer terms (since each term has a high degree of freedom/expressibility), the

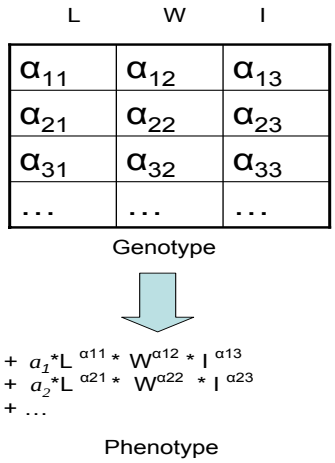


Fig. 5. GA genotype to phenotype mapping

latter would evolve expression with more terms. Since there is no *a priori* information about the model, our choice of the model matrix is arbitrary.[5]

4.2 Fitness Evaluation

The GA evolves the exponents of all variables for each term. To determine the complete posynomial form of the candidate solution, the coefficient of each term must be determined. This is done deterministically, given the specific values of the exponents, with a minimization of mean square error (MSE) objective. We formulate a Quadratic Programming (QP) problem from the MSE objective function (because it is second degree) along with linear constraints that all coefficients are positive. The coefficients found by QP are the global optima for the given exponents. A QP solver within Matlab is used to solve the QP problem. The minimum value of the error (minimum MSE) is a measure of the accuracy of the posynomial.

The complete dataset of SPICE derived MOS behavior is substantially large (about 70000 points) and the formulation and solving of QP becomes computationally expensive for the whole dataset. Therefore, we only use a small randomly sampled fraction of the dataset.[6] Using this smaller fraction requires that the evolved model does not overfit the sampled points. To ensure this, we use 2-fold cross validation on the sampled data set and use the cross validation MSE as the fitness of the individual. At each generation, we fit the

[5] The decision is not completely arbitrary since monomials with non-integer exponents yield low error results.

[6] A more principled approach to do this could be by use of Design of Experiments [8]

coefficients of only the best of generation solution on the entire dataset and calculate its MSE. This error value lets us know, if in any phase of evolution, the algorithm starts overfitting to the sampled data set. The error value is not used to provide any feedback to the evolutionary algorithm.

To summarize, the candidate solution is derived by evolution of its exponents and QP optimization of its coefficients. It is evaluated on a fixed randomly sampled small fraction of the complete data set and the cross-validation error is used as the candidate's fitness.

It is worth noting here, that the problem at hand is different from a typical model building problem, where one needs to deliver a model which generalizes well to unseen data. Here a huge data-set for prediction is available but we are using only a part of it since our algorithm is computationally intractable with the complete data set. There is also a hypothesis that using more data will not help.[7] Our final adjustment of the coefficients for the exponents of the GA solution for optimal performance on the complete dataset is a flexibility that is not available in typical model building problems.

Also noteworthy is an effect arising from using QP to find the coefficients. The QP problem has the constraints that all coefficients should be more than zero. This can be visualized as an optimization problem with feasible space bounded by hyperplanes. Each hyperplane has value of one of the coefficients as zero everywhere on it. The intersection of the hyperplanes have more than one coefficient equal to zero. The solution of the QP problem in many cases lie on one of the constraining hyperplanes or their intersection. This pushes some of the coefficients to exact zero. Thus QP implicitly performs *feature selection* on the evolved terms by setting the coefficients of useless terms to zero. From the perspective of GAs, QP is identifying useful terms within the genotype. It also makes the genotype implicitly variable length though with an upper bound. It is an open question whether this QP-based selection of terms also generates useful building blocks across the population that could be exploited by a GA combination operator.

4.3 Variation Operators

The GA employs a coarse grained uniform crossover operator that exchanges terms. The order of terms (rows in the genotype matrix) has no significance since all terms are additive so each row is chosen from one of the two parents. Since the evolved expression is a sum-of-products, we consider the product terms as building blocks, which combine through the addition operator to form the solution.

We used term-wise uniform crossover. Two individuals are chosen from the selected population and the new individual is created by choosing each row

[7] This is akin to bioinformatics experiments, where in motif predicting algorithms, no improvement in accuracy is observed after the data-set becomes larger than a given number.

from one of the two parents. This operator is used to do *inter building block* recombination.

The mutation operator is used to perturb the real-numbered values in the cells of the matrix. A normal distribution centered at zero with a given variance (λ) is added to the real numbered value. The variance is adaptively decreased according to the genetic algorithm generation to make the algorithm explorative initially and exploitative in later stages. We investigated two strategies:

Strategy 1: Mutation is carried out in two phases. In the first phase, a term (equivalent to a row) is chosen for mutation by a given probability (p_{term}). In the second phase, each cell in the chosen row is mutated by a given probability (p_{cell}). This scheme in phase one chooses a building-block to mutate. The term-wise mutation rate is set to conserve a given number of building blocks in the population depending on the selection pressure. In the second phase the intention is to direct appropriately step-sized, local search within each identified building block. The cell-wise mutation rate is set to encourage incremental search in the product-term space.

Strategy 2: Strategy 2 exploits the fact that the coefficients indicate whether a given term is a useful (i.e. contributing to fitness) component to a particular genotype's solution. We could hypothesize that a useful term in one solution might be useful in another solution and thus is an evolutionary building block. A zero coefficient term would not be a useful building block. Thus, in this strategy, terms with zero coefficients and non-zero coefficients are treated differently. A term with a zero coefficient is mutated by a probability (p_{zero_term}), the operation being reinitialization of the term randomly. A term with a non-zero coefficient is mutated by a different probability ($p_{non\_zero\_term}$) and is mutated cell-wise in the same way as Strategy 1. Here, the coefficients of the terms are considered in tandem with the genotype for the purposes of mutation only. This strategy is exploiting a hypothesis that a term with a non-zero coefficient is a building block. If this is true, the strategy will generate useful explorative variation only in circumstances where it seems advantageous (i.e. when a zero coefficient term has evolved).

5 Experiments

We used the proposed genetic algorithm to evolve posynomial models for 9 mos model output variables in terms of its W, I and L for an NMOS transistor. The specific model output variables are shown in the top row of Table 3. The silicon technology for the mosfet model is TSMC $0.18u$. Approximately 70,000 points were extracted from SPICE simulation which swept the complete operating range of the transistor (in saturation). We sampled 2000 points uniformly from the complete set for fitness evaluation and used 2-fold cross-validation.

The GA is standard. Each genotype of generation 0 is initialized using a uniform random distribution bounded by $[minVal, maxVal]$ for each cell

element. The choice parameter is randomly initialized to 1 or 0 such that the average number of terms per individual in the initial generation is 3. We use a generation based GA with tournament selection. Each tournament produces one new member of the next generation. The genetic algorithm parameters are given in Table 2. The probability of term mutation for both Strategy 1 and Strategy 2 were roughly hand calculated for learning a linkage of 3 terms given the tournament size.

Parameter	Value
Population Size	50
Tournament Size	6
[minVal, maxVal]	[-3, 3]
Initial λ	1
λ rate	Halved every 20 generations
$p_{crossover}$	0.5
Mixing Ratio	0.7
p_{term}	0.45
p_{cell}	0.5
p_{zero_term}	0.7
$p_{non\_zero\_term}$	0.3
Maximum number of terms	5

Table 2. GA Parameters

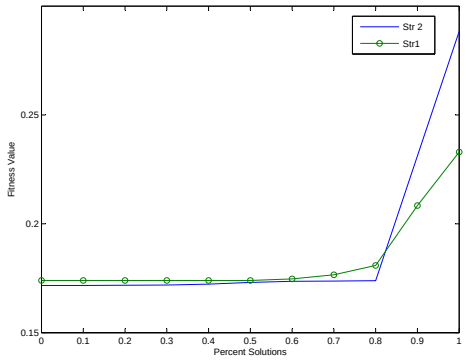


Fig. 6. Quantile plot for best solutions of Strategy 1 and Strategy 2 over 10 runs

5.1 Experiment 1

Here we evolved posynomial models for output variable $Veff$ using Strategy 1 and Strategy 2. The average best-fitness over all runs is misleading due to outlier runs. Thus, we use quantile plots which show the fitness value range for a percent of individuals. Figure 6) shows the quantile plot for best-fitness over 10 runs for both strategies. One can observe that 8 out of 10 solutions for Strategy 2 have lower error value than those of Strategy 1. However, one solution from Strategy 2 does much worse than Strategy 1 and maybe considered an outlier. The fitness of the posynomial expressions which gave the least error for complete data over all runs and generations was $1.030e^{-4}$ for Strategy 1 and $1.0220e^{-4}$ for Strategy 2. These results are just indicative in favour of Strategy 2 and no strong claim can be made on their basis. The primayy aim of this investigation is to find out whether our GA can find a better posynomial than log-trained monomials. This is addressed in Experiment 2.

5.2 Experiment 2

We evolved posynomials for all 9 mos output variables using the genetic algorithm. We ran 2 runs for each output variable with the same settings as above for 1000 generations. In each generation, the coefficients of the best individual and its error were re-determined according to the complete set. The posynomial expressions which gave the least error for complete data over all runs and generations are reported in Table 4. For each parameter the coefficient of each term and the respective exponents are reported.

For comparison, monomials for output variables were created by a three step process. First, log regression [2] was done to find a set of coefficients and exponents. The exponents and coefficients were re-tuned to minimize MSE using a gradient-descent method in the second step. In the third step, the coefficients for the given exponents were optimized globally by a QP formulation. A comparison of error of these monomials and GA evolved posynomial is shown in Table 3.

	gm	gds	$Veff$	Vt	$V_{d(sat)}$	C_{gd}	C_{gs}	C_{db}	r_o
Monomial	$1.36e^{-8}$	$1.57e^{-11}$	$5.08e^{-4}$	$3.46e^{-5}$	$2.87e^{-4}$	6.28^{-34}	$3.94e^{-29}$	$3.48e^{-32}$	$3.35e^{13}$
Monomial	$4.75e^{-9}$	$5.62e^{-13}$	$1.04e^{-4}$	$8.53e^{-6}$	$7.61e^{-5}$	1.99^{-35}	$3.55e^{-29}$	$1.28e^{-34}$	$1.25e^{12}$
Imp (%)	65.1	96.4	79.5	75.3	73.4	96.8	9.7	99.63	96.2

Table 3. Results of Experiment 2. The bottom row (Imp) shows the percentage improvement of the GA evolved posynomial over over the monomial derived by a three step process (see text for details). Improvement measures decreased error.

Coefficient	L exponent	W exponent	I_d exponent
gds			
$1.394403e^{-08}$	-0.4964759	-0.1530169e	0.6236803
$3.822282e^{+04}$	1.926779	-1.993445	3.000000
$3.340184e^{-05}$	-0.6053242	0.5020036	0.5457113
$7.580838e^{-21}$	-3.000000	0.5954294	0.3877911
gm			
$6.142170e^{-02}$	-0.3766175	0.4750202	0.5247892
V_{eff}			
$1.010186e^{+02}$	0.5113741	-0.5138530	0.5071154
$6.589195e^{-01}$	0.5540773	-1.458791	1.621552
V_t			
$1.100209e^{-01}$	-0.09881736	0.003999609	0.000000
$9.979964e^{+05}$	1.476890	-0.06611535	0.000000
$V_{d(sat)}$			
$1.521139e^{-01}$	0.07657982	-0.04372403	0.08441106
$9.201251e^{+06}$	2.151408	-2.249152	2.269076
$2.012515e^{+02}$	0.5960877	-0.5342329	0.5364634
$1.499778e^{-02}$	1.367515	-1.578360	0.5740558
C_{gd}			
$9.281660e^{+01}$	3.000000	-0.8642648	1.858666
$3.756514e^{-16}$	-0.6405048	1.774402	-0.7871581
$1.335686e^{+03}$	2.631523	0.9.397126	0.04175101
$6.854030e^{-10}$	-0.01477968	1.002911	0.00050428
C_{gs}			
$7.175508e^{-03}$	1.080458	0.9303338	0.05072020
$2.179441e^{-03}$	0.9974063	0.9505211	0.04358394
$4.078940e^{-07}$	0.2597072	1.171186	0.05483664
C_{db}			
$4.333407e^{-10}$	0.0092217	0.9867695	0.0018283
$5.303325e^{-10}$	0.0013345	1.014629	-0.0012439
r_o			
$5.928747e^{+09}$	1.560661	-0.4045190	-0.6062304
$3.738613e^{-18}$	-2.572708	1.495797	-2.566089
$1.487947e^{-04}$	-0.3950394	1.965040	-2.956083
$1.077567e^{+11}$	2.380938	1.812263	-2.580723
$3.027457e^{+04}$	0.5501161	-0.3637335	-0.5637681

Table 4. Results of Experiment 2. These are GA evolved posynomial mosfet models.

6 Summary

We have broadly described the process and methods of analog circuit design in terms of topology selection and sizing. A description of sizing as an optimization problem along with brief descriptions of hand methodologies and automatic methodologies such as black-box optimization and geometric programming was given. Geometric programming can solve an optimization problem in seconds provided the problem objectives are expressed as posynomials and the constraints are expressed as monomials. This prompted us to design a GA to evolve posynomial mos models that are embedded within posynomials that express the circuit's small signal measurements. We designed a GA with a fixed length genotype that implicitly has variable encoding of posynomial terms via its accompanying use of QP for coefficient optimization. For this particular mos the GA provides **much better** models than statistically log linear fitted monomial based models. The given approach is a general posynomial model building approach and not specific to only mos parameters. However, care has to be taken if building models for higher dimensional input space because sparseness of expressions will be required. In this case a modified GA or geometric programming might be useful.

7 Future Work

While we have only focused on posynomial models in this submission, a broader goal is to evaluate the value of GA evolved posynomial models in terms of how much they improve the quality of circuit sizing. Comparisons can be made to hand-sized circuits and circuits sized with posynomial models derived by other means (e.g statistically derived mos monomial models and hand written circuit level posynomials). While the GA provides better accuracy, the extent to which this improved accuracy helps with sizing is unquantified as yet. The cost of the accuracy of the GA derived mos models is the computational expense and time to evolve them. High cost is unimportant because the model is very reusable. In other words, the cost is amortized over many uses of the model.

But are posynomials' inherent errors worse than using polynomials and more computationally intensive techniques? There could be potential advantages to using any kind of EA to derive less restricted models and then using another EA (i.e. in place of geometric programming) that exploits the model. This sounds easy but it is not straight forward in the circuit sizing domain because of the complexity of large signal models and the ability to express a models parameterized for a technology.

Acknowledgements

We would like to thank Vladimir Stojanovic, Ranko, Joel Dawson, Trent McConaghey for their assistance with test test and discussions

References

1. J. Botelho, B. Leonardo, P. Vieira, and A. Mesquita. An experiment on nonlinear synthesis using evolutionary techniques based only on CMOS transistors. In J. Lohn, R. Zebulum, J. Steincamp, D. Keymeulen, A. Stoica, and M. I. Ferguson, editors, *2003 NASA/DoD Conference on Evolvable Hardware*, pages 50–58. IEEE Computer Society, 2003.
2. S. Boyd, S.-J. Kim, L. Vaudenberghe, and A. Hassibi. A tutorial on geometric programming. In *Technical report, Electrical Engineering Department, Stanford University*, 2004.
3. D. M. Colleran, C. Portmann, A. Hassibi, C. Crusius, S. S. Mohan, S. Boyd, T. H. Lee, and M. del Mar Hershenson. Optimization of phase-locked loop circuits via geometric programming. In *IEEE Custom Integrated Circuits Conference*, pages 377–380, 2003.
4. W. Daems and G. Gielen. Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions*, 22(5):517–534, 2003.
5. M. del Mar Hershenson, S. S. Mohan, S. P. Boyd, and T. H. Lee. Optimization of inductor circuits via geometric programming. In *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 994–998, New York, NY, USA, 1999. ACM Press.
6. G. Gielen, K. Swings, and W. Sansen. An intelligent design system for analogue integrated circuits. In *EURO-DAC '90: Proceedings of the conference on European design automation*, pages 169–173, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.
7. J. Grimbleby. Automatic analogue circuit synthesis using genetic algorithms. *IEE Proceedings Circuits, Devices and Systems*, 147(6):319–323, Dec. 2000.
8. W. D. Kelton. Design of experiments: experimental design for simulation. In *WSC '00: Proceedings of the 32nd conference on Winter simulation*, pages 32–38, San Diego, CA, USA, 2000. Society for Computer Simulation International.
9. 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 Transactions on Evolutionary Computation*, 1(2):109–128, July 1997.
10. X. Li, P. Gopalakrishnan, Y. Xu, and T. Pileggi. Robust analog/RF circuit design with projection-based posynomial modeling. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 855–862, Washington, DC, USA, 2004. IEEE Computer Society.
11. M. Mar, H. Stephen, P. Boyd, and T. Lee. GPCAD: A tool for CMOS op-amp synthesis, 1998.
12. C. Mead. Neuromorphic electronic systems. 78(10):1629–1636, 1990.

13. S. Pookaiyaudom and S. Jantarang. Automatic circuit simplification for meaningful symbolic analysisusing the genetic algorithm. In *1996 IEEE International Symposium on Circuits and Systems*, volume 1, pages 109–112, 1996.
14. T. Quarles, A. Newton, D. Pederson, and A. Sangiovanni-Vincentelli. SPICE 3, version sf5 user's manual. 1994.
15. H. Shibata, S. Mori, and N. Fujii. Automated design of analog circuits using cell-based structure . In *Evolvable Hardware*, pages 85–92. IEEE Computer Society, 2002.
16. P. Siarry, G. Berthiau, F. Durdin, and J. Haussy. Enhanced simulated annealing for globally minimizing functions of many-continuous variables. *ACM Trans. Math. Softw.*, 23(2):209–228, 1997.
17. B. D. Smedt and G. Gielen. WATSON: Design space boundary exploration and model generation for analog and rf ic design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):213–224, 2003.