Chapter 19

GENETIC PROGRAMMING IN INDUSTRIAL ANALOG CAD: APPLICATIONS AND CHALLENGES

Trent McConaghy[1] and Georges Gielen[1]
[1]*Katholieke Universiteit Leuven, Leuven, Belgium*

Abstract This paper investigates the application of genetic programming to problems in industrial analog computer-aided design (CAD). One CAD subdomain, analog structural synthesis, is an often-cited success within the genetic programming (GP) literature, yet industrial use remains elusive. We examine why this is, by drawing upon our own experiences in bringing analog CAD tools into industrial use. In sum, GP-synthesized designs need to be more robust in very specific ways. When robustness is considered, a GP methodology of today on a reasonable circuit problem would take 150 years on a 1,000-node 1-GHz cluster. Moore's Law cannot help either, because the problem itself is 'Anti-Mooreware' – it becomes more difficult as Moore's Law progresses. However, we believe the problem is still approachable with GP; it will just take a significant amount of 'algorithm engineering.' We go on to describe the recent application of GP to two other analog CAD subdomains: symbolic modeling and behavioral modeling. In contrast to structural synthesis, they are easier from a GP perspective, but are already at a level such that they can be exploited in industry. Not only is GP the only approach that gives interpretable SPICE-accurate nonlinear models, it turns out to outperform nine other popular blackbox approaches in a set of six circuit modeling problems.

Keywords: analog, CAD, synthesis, industrial, genetic programming, robust, yield

1. Introduction

One of the flagship problems in Genetic Programming is that of analog structural synthesis, where the aim is to automatically determine the circuit

components, interconnections, and suggested component dimensions to meet a set of circuit design goals. This is an industrially relevant problem and a challenge to automated design techniques.

In this domain, GP has evolved several patent-quality circuits (Koza et al., 2003), which is a remarkable success by almost any measure. It is an especially notable accomplishment from an artificial intelligence perspective because "patent-worthiness" is a good measure of success for testing techniques in automated "creative" design.

Given such impressive results, a GP researcher might have expected GP to be barnstorming the field of analog design. However, this is not the case; GP is actually not in use *at all* for topology design in industry. In fact, industrial analog engineers and CAD developers would be very surprised to hear that analog synthesis is considered a success within the field of GP. In effect, the bar of "GP success," even success on industrially relevant problems, is different than the bar of "usefulness to industry." How can GP make the transition? In this paper, we draw upon our experiences in industrial analog CAD, with the aim to identify what would make GP useful to that field.

This chapter is organized as follows. We first describe analog CAD's context, then how GP-based synthesis would fit in. We highlight industrial robustness issues and tactics, which we use to reframe the problem of GP-based synthesis. Then, we show two other analog CAD applications where GP is making inroads: symbolic modeling and behavioral modeling.

2. The Problem Domain: Analog CAD

Context. Electronic Design Automation (EDA) is the field devoted to building computer-aided design (CAD) tools for electrical engineers. Because of the massive size of the semiconductor industry and the constant changes in design constraints due to Moore's Law, EDA is an active industry, with billions in revenue every year. Analog CAD (Gielen and Rutenbar, 2002) is a subfield devoted to tools for analog circuit designers.

Design "Implementation". When researchers in GP read about GP for analog synthesis, they're used to reading about "front-end design," in which the problem input is circuit specifications (*e.g.* get power consumption < 10mW), and the target output is a "netlist," which describes the synthesized circuit in terms of components, interconnections, and component dimensions.

That's actually just one step in a much broader flow. Somehow, that netlist has to get into the real world, *i.e.* as part of a discrete circuit, or as a "chip" (VLSI circuit). The industrial value is in chips. The back-end flow is as follows: Once the netlist is determined, it is converted into a "layout," which is essentially a set of overlapping polygons, where specific shapes represent specific types of components and interconnects. The layout is integrated into an overall system

layout, which is sent to a billion-dollar fabrication facility. The system layout is used for creation of process masks, which are a sort of physical filter on whether to dope / etch / *etc.* different parts of a silicon wafer. Process mask generation can cost hundreds of thousands of dollars or more. Using the masks, many chips at once are fabricated on a wafer. The chips are sliced apart from each other, then packaged, and finally tested.

If a problem is detected after a step, then the process backtracks to the previous step. The most expensive step is creation of the process masks, so this is where it is most important to avoid backtracking. In a worst case, which still often happens in practice, a fabricated chip does not work at all, and to make it work one needs to go back to front-end design. This is known as a "respin." Obviously, respins are to be avoided because of mask costs, but even more importantly, loss of profitability in time-to-market.

A new analog topology significantly raises the chance of a respin due to lack of experience with that topology; this makes adoption of an analog structural synthesis tool a risky proposition (and costly to try). But, ultimately, GP would need to demonstrate working chips.

3. GP Application: Analog Structural Synthesis, Part I

Designer Perspective

Since the late 1980's, analog designers have been presented with impressive-sounding claims about "analog synthesis." Researchers have labeled "analog synthesis" to mean many things, including global parameter optimization, automated conversion from netlist to layout, and automated topology design (the version that GP targets). For a survey, see (Gielen and Rutenbar, 2002).

Our focus here is automated topology design. Most analog designers would acknowledge that if such a technology actually worked, it would drastically change the field. Their counterparts in digital design have already experienced such a revolution: the mid 1980's introduction of digital circuit logic synthesis.

Unlike digital synthesis, few claims of analog synthesis have held true. The analog synthesis techniques were typically too unscalable or brittle to be useful in industry. Of the dozens of various types of analog synthesis technologies reported over the last twenty years, just a few have found their way into industrial use, and that was only recently (Synopsys, 2005; Cadence, 2005b; Cadence, 2005a). None of these do automated topology design. Thus, when designers hear about a new structural synthesis technology, from GP or elsewhere, they immediately question them, and to a much stronger degree than automation-friendly digital designers.

How do the claims of GP look, from a designer's perspective?

For starters, they're not shocked, even when they see the patent results. With every other structural synthesis technology reported until now, something

was missing, something that limited its widespread industrial use. Despite their limited understanding of GP, designers have no real reason to treat GP specially. They simply believe that something's missing for GP too.

They're right. When an analog designer digs more deeply into the GP methodology for automated topology design, he/she finds problems. Some are obvious (to an electrical engineer), and some are subtle. But, whereas prior analog structural synthesis approaches had showstopping problems of brittleness and scalability, we believe that GP has no such problems. Instead, GP faces "engineering-style" challenges in problem setup, and especially in improving GP's speed.

Current Industrial Practice

It is fruitful to look at what flow and automation tools that industry uses which are closest to the analog structural synthesis problem.

Figure 19-1 illustrates the overall flow of front-end design for cell-level circuits.

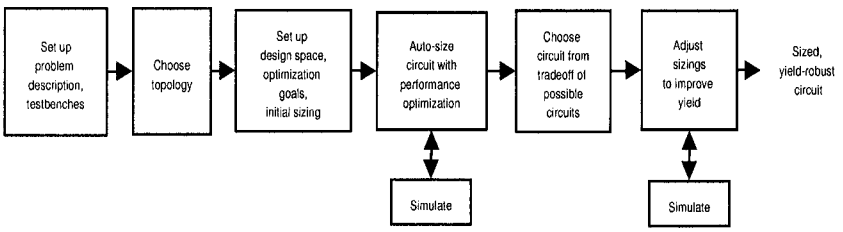

Figure 19-1. State of the Art Industrial Front End Analog Design Flow

The automation happening at the front end is in local / global optimization tools (Synopsys, 2005; Cadence, 2005b), which take in a fixed topology, and automatically determine the component values in order to best meet the design specifications. This step is often referred to as circuit sizing or circuit optimization, rather than synthesis. The topology has been manually designed beforehand. Yield improvement is typically manual, though there is a shift to automation there too.

These tools need to make chips that meet certain performance measures once they've been manufactured. Thus, the tools need a means for estimating performance and taking robustness into account.

Performance Estimation and Robustness

In analog synthesis, robustness is strongly related to performance estimation. A performance estimator takes in a candidate design (*i.e.* a topology and component values in our case), and estimates the performances of the circuit.

To achieve a robust design, one has to estimate performance as accurately as possible.

The ideal performance estimator would predict with 100% accuracy how a design performs after layout, manufacturing, and testing without actually fabricating it. It would run quickly enough to be invoked thousands or millions of times throughout optimization, to allow automated exploration of designs. SPICE is the most accurate and general estimator, but there are also faster, less general, less accurate ones.

Layout issues. "Layout parasitics" are effects that were not accounted for prior to layout. An example layout parasitic is when the material between two wires acts like a circuit component (*e.g.* a capacitor) which is supposed to be an open circuit.

Environmental conditions. The manufactured chip will need to work at the desired performance level, even as temperatures change, power supply changes, and load changes. These are conditions of the circuit's operating environment.

Manufacturing variations. When manufacturing a VLSI circuit, random variations get introduced into the implementation of the designs as an inherent effect of the fabrication process. The automated tool must model this and handle it.

The simplest model is so-called "Fast/Slow corners," which in effect try to capture the 3-sigma extremes in each type of transistor's operating speed due to manufacturing variations. This approach is popular for its simplicity and availability. However, corners do not model the problem well because they do not bracket the variations in analog design goals (they are really only suitable for digital design).

Some approaches build empirically-based statistical models to estimate a probability density function, such as (Power et al., 1994). These models almost always make assumptions that render them inaccurate, for example, assuming that certain random variables are independent when they are not, or ignoring local statistical variations as in (Alpaydin et al., 2003).

One approach (Drennan and McAndrew, 2003) uses a more physical basis for randomness modeling and is quite accurate, though an implication is that for every transistor, 8 random variables are introduced; thus, a medium sized circuit could have hundreds of random variables.

Analog Structural Synthesis Problem

The problem of analog structural synthesis is the same as the sizing problem, except the design space is broadened drastically, to include choice of the topology (devices and connections among devices, in addition to device sizes).

Synthesis cannot make assumptions about the topology; this has big implications, which we will discuss later.

Current Industrial Practice: Details

We are now ready to ask how the industrial tools account for robustness.

For environmental variations, they use a set of user-defined "corners," with each corner specifying a temperature, power supply, *etc.* SPICE is used to estimate performance for each corner, and the worst-case value is taken.

For layout, they can ignore it for a first-pass design. Then, after layout has been done, if layout parasitics degrade the performance too much, the most important parasitics can be inserted into the design and a local optimization performed.

For manufacturing variations, they (Synopsys, 2005; Cadence, 2005b) use model corners, which as mentioned, is less accurate. There are many other approaches in the literature (Phelps et al., 2000; Schenkel et al., 2001; Smedt and Gielen, 2003), but each is forced to trade off accuracy for feasible runtime, or pessimistic design. GP tactics such as (Teller and Andre, 1997; Hu and Goodman, 2004b) are too expensive for refining designs.

4. Analog Design for Robustness (on a Fixed Topology)

This section highlights how a fixed topology implicitly brings robustness, or conversely, what other robustness issues must be considered when evolving a topology.

Robustness in Manual Topology Design

By definition, optimization approaches operate on manually designed topologies. For VLSI circuits, and perhaps as a surprise to GPers, *manually-designed topologies are almost always designed with robustness in mind.*

We now examine what analog designers do to make topologies more robust. We will refer to a well-known circuit shown in Figure 19-2.

Topologies Are Designed For Process Variations. The effect of "local" or "mismatch" variations within a chip ("mismatch") has always been smaller than "global" variations which are between chips and between runs (1-2% vs. 10-20%).

The main tactic to deal with global variations is to design structures in which performance is a function of *ratios of sizings*, rather than absolute values. For example, in common-source gain stages, a load resistor would have variation of 10-20%. So, designers use a PMOS load instead, matched up to an NMOS gain transistor, and gain is dependent on the ratios (*e.g.* in Figure 19-2, M5a is a resistive load for M3a).

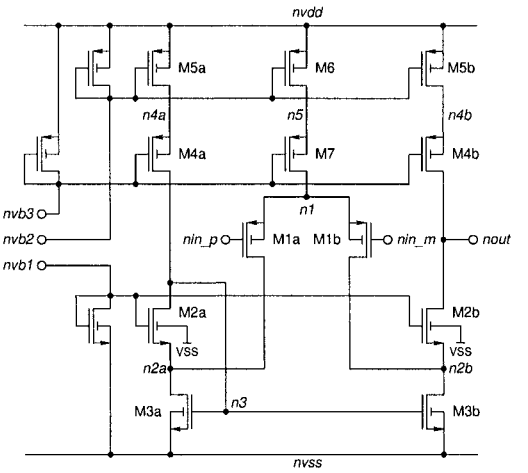

Figure 19-2. "High-speed operational transconductance amplifier (OTA)" analog circuit

Differential design is another tactic to move away from "absolute" values. Here, "mirrors of structures" are created, and the circuit operates on a difference between two voltages, rather than one voltage and ground. The Figure 19-2 OTA is symmetrical about a vertical axis centered on M5 and M7; the output is a function of the difference between the positive and negative inputs, nin_p and nin_n.

A precise current is expensive to generate; it's a much better idea to generate one or a few reference currents and copy them throughout the circuit with "current mirrors." The OTA does this: the three transistors on the left are the "biasing" circuitry to generate currents, which are then copied throughout the circuit. Sometimes a single current can be shared, rather than trying to match two separate currents. The OTA's differential pair (M1a and M1b) does this: instead of having different "tail" currents, they share the same current which goes through M6 and M7.

Negative feedback is a well-known general engineering technique for compromising some performance in the interest of precision. Analog circuits often do this too, such as for improving common-mode rejection ratio of a differential amplifier or for reducing variation of an amplifier's gain (Razavi, 2000).

Trust and Re-Use. The topology is trusted because it has been created and characterized by expert analog designer(s), and has been fabricated and tested in many process generations. Topology re-use is widespread because past success means more confidence that the topology will work. A new topology is typically a derivative of an existing topology, because similarity maintains trust.

SPICE can lie. SPICE can lie due to problems in its device models, convergence, and perhaps inadequate models of parasitics. SPICE transistor models seem to be in a continually inadequate state, with known deficiencies (*e.g.* non-smooth transitions from one operating region to another). Part of the difficulty is that the models have to work for several processes, typically require hundreds of parameters that should be easy to extract, and strive to have as good a physical basis as possible. Because of this, designers consciously avoid transistor operating regions where the models are known to be inadequate.

Whitebox Constraints. Topologies have whitebox constraints based on the strategy underlying the topology's design. Every transistor in a circuit has been designed with the assumption that it will be operating in a specific operating region; there is a good chance that the assumptions break down outside those constraints.

Clear Path To Layout. The designer knows that, for manually-designed topologies, there is a clear path to layout; to a large extent, the designer has already anticipated the parasitics. Layout designers also have tactics to improve robustness, such as: folding transistors, guard rings, and careful routing to avoid cross-coupling between sensitive wires (Hastings, 2000; Lampaert et al., 1999). Analog *layout* synthesis is another analog CAD subproblem (Rutenbar and Cohn, 2000); it is difficult to model and solve well, as illustrated by continued research activity. When layout parasitics are more pronounced, such as in RF design, there are ways to tighten the coupling between sizing and layout design (DeSmedt and Gielen, 2003; Zhang et al., 2004; Bhattacharya et al., 2004).

To properly account for layout effects in synthesis, one possibility is to unite the front-end design space (topology and circuit sizes) with the back- end space (layout), and approach the whole problem at once, as in Section 5.2 of (Koza et al., 2003). Unfortunately, runtime was 1.5 orders of magnitude slower, and that work drastically simplified the layout synthesis problem – it didn't even extract the parasitics from the layout before simulating the netlist.

Synthesis Exaggerates "Cheating" of Search Algorithms. We say a "cheat" occurs when design has good measured performances, but which upon inspection is useless (*e.g.* not physically realizable). An example is too many long, narrow transistors; the solution is to add more constraints on width/length ratios. Each added constraint takes time to detect, correct, and re-run. There is more opportunity for structural synthesis to cheat compared to optimization, because synthesis design space is drastically larger, and SPICE can cheat more readily. Evolvable hardware research is filled with examples of odd designs; however, in non-reprogrammable analog VLSI, one cannot embrace odd designs because of the high cost of fabrication.

5. GP Application: Analog Structural Synthesis, Part II

An Updated Model of the Analog Synthesis Problem

Most earlier GP structural synthesis work such as (Koza et al., 1999; Lohn and Colombano, 1998; Zebulum et al., 2002; Sripramong and C.Toumazou, 2002; Koza et al., 2003) did not have a very thorough model of the problem compared to analog CAD optimization, but is has been getting better recently. In (Koza et al., 2004a), corners have been added to account for environmental and (very roughly) manufacturing variations. And, they employ testbenches directly from an industrial CAD vendor (Synopsys, 2005). Though some recent research has not yet acknowledged the need for more robustness (Dastidar et al., 2005).

GP does not have whitebox constraints, because it does not make assumptions about what region each transistor will operate in. GP actually has stronger performance measures in one regard: it also tries to match waveforms of behavior.

Compared to analog CAD optimization work, GP's biggest deficiency in problem modeling is its lack of a good model of manufacturing variations. The closest, robust HFC (Hu and Goodman, 2004a), did have Monte Carlo sampling, but the randomness model is not suitable for VLSI circuits.

Beyond analog CAD optimization, GP-evolved circuits must somehow get the same advantages as a manually-designed topology. Such circuits must get designer trust, including an explanation and formulae for behavior; ultimately, successful fabrication and testing. On the way, there are the hurdles of SPICE (mis)behavior, layout parasitics, search space cheats, and extra challenges from first-order process variations.

New Computational Challenges

Ultimately, the only way to accurately model manufacturing variations is via *simulation* on *good statistical models*. Let us examine the runtime of a typical structural synthesis run that uses brute force Monte Carlo sampling. Except for layout, we will temporarily ignore all the extra challenges wrought by a non-fixed topology.

Let us say: 8 corners (for environmental variations), 10 Monte Carlo samples (for manufacturing variations, 10 is optimistic), and simulation time of 1 minute for a circuit at one corner and one sample on all testbenches on a 1 GHz machine. Parasitic-extracted layouts might mean 10x longer. Larger designs and/or longer-than-transient analyses could easily take 6x, 60x, or even 600x longer to simulate.

It is typical for a GP run to explore 100 million designs for more challenging problems. 1 billion or even 10 billion would not be unreasonable (Koza et al., 2003). But let us have 1,000 1-Ghz machines in parallel.

Then, total run time = 152 years! And it's even longer for tougher problems, where simulation time is 6x-600x longer and number of individuals is 10x-100x more. One might ask if Moore's Law can ease this challenge.

The Impact of Moore's Law

Mooreware vs. Anti-Mooreware. GP is considered an example of "Mooreware" (Koza et al., 1999), where an algorithm becomes more effective with more computational power, and therefore with the march of Moore's Law over time.

However, Moore's Law, when attacking VLSI design problems, is a double-edged sword. Each new technology generation also requires more modeling effort, and therefore more compute time! For example, the need for substrate noise modeling is growing; to model this takes 30 minutes on four modern processors (Soens et al., 2005), *i.e.* 120x more computational effort.

Thus, analog synthesis is an "Anti-Mooreware" problem: it gets more difficult as Moore's Law progresses. So, we cannot rely on the "Mooreware" aspect of GP to eventually be fast enough.

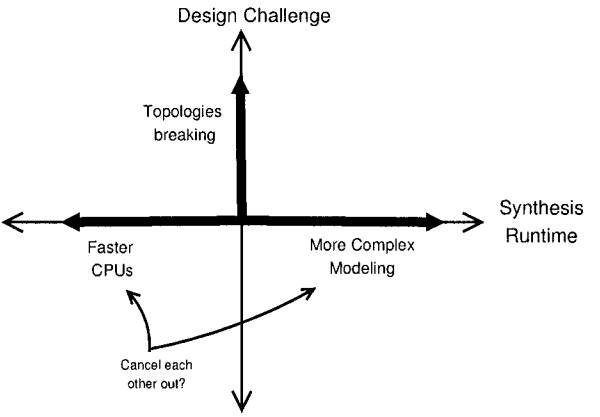

Figure 19-3. Effects of Moore's Law on Analog Structural Synthesis

Moore's Law Breaks Topologies. Topologies are getting constrained in new ways due to Moore' Law. Here is an example. Supply voltages and threshold voltages are steadily decreasing, but threshold voltages cannot scale as quickly because of fundamental physical constants. At some point, "cascode" configurations, which stack two transistors on top of each other, are unusable

Table 19-1. GP-generated symbolic circuit models with < 10% train and test error.

Perf. Char.	Expression
ALF	-10.3 + 7.08e-5 / id1 + 1.87 * ln(-1.95e+9 + 1.00e+10 / (vsg1*vsg3) + 1.42e+9 *(vds2*vsd5) / (vsg1*vgs2*vsg5*id2))
fu	10^(5.68 - 0.03 * vsg1 / vds2 - 55.43 * id1+ 5.63e-6 / id1)
PM	90.5 + 190.6 * id1 / vsg1 + 22.2 * id2 / vds2
voffset	- 2.00e-3
SRp	2.36e+7 + 1.95e+4 * id2 / id1 - 104.69 / id2 + 2.15e+9 * id2 + 4.63e+8 * id1
SRn	- 5.72e+7 - 2.50e+11 * (id1*id2) / vgs2 + 5.53e+6 * vds2 / vgs2 + 109.72 / id1

(*e.g.* M4b and M5b in figure 19-2 are in cascode). The alternatives are less ideal: folded cascodes mean larger power consumption, and extra stages mean slower speed and instability risk. Figure 19-3 summarizes.

The Road Ahead for GP and Structural Synthesis

GP has come a long way along the road of analog structural synthesis and the milestones have been remarkable, but a full industrial-strength version is orders of magnitude away.

Speeding up GP sufficiently may actually be possible because there are so many facets to the problem and the algorithms. It comes down to an "algorithm engineering" problem. There are possible speedups at (1) the general EA level, for example in population management, handling modularity / hierarchy, exploiting advances in theory, reuse of run information, in representation and operators, parallelism; (2) at the robustness level, for example exploiting the transparency in manufacturing variations, environmental variations, and simulation analyses; and (3) at the domain-specific level of cell-level analog circuits, for example to guide design of representation, operators and building blocks, special constraints, faster performance estimators. Koza has elaborated on some possibilities (Koza et al., 2004b).

6. GP Application: Symbolic Modeling

Given the overall goal of finding ways to aid analog engineers in the design process, we can ask ourselves what other problems GP might help in. That's a question that we asked in the last year, and so far we've demonstrated two other industrially-relevant applications. Let's examine each, starting with symbolic modeling.

In all designs that an engineer does, the more he or she understands a circuit, the more he will be able to improve it (in terms of performance and yield), and the more productive he or she will be. This is independent of whether the tools are automated or manual. Equations are a very useful tool for helping designers improve understanding, *e.g.* equations that map design variables (*e.g.* component values) to circuit performances (*e.g.* power consumption). Such equations have traditionally been created by hand, but they are so useful that since the early 90s, there has been considerable research effort to devise algorithms to automate this (Gielen, 2002). This subfield of of analog CAD is called "symbolic analysis" when the equations are directly extracted from the topology, or "symbolic modeling" when the equations come from SPICE simulations. The ideal approach would produce SPICE-accurate, interpretable equations of arbitrary nonlinear circuits. So far, no approach could do all those things at once.

Interestingly (and almost surprisingly), no one had yet used GP in symbolic regression mode on SPICE-generated training data. So, we applied it, with a few modifications to GP to keep the expressions readily interpetable (McConaghy et al., 2005). Table 19-1 gives models for each of six different performance expressions, for the circuit previously examined (Figure 19-2).

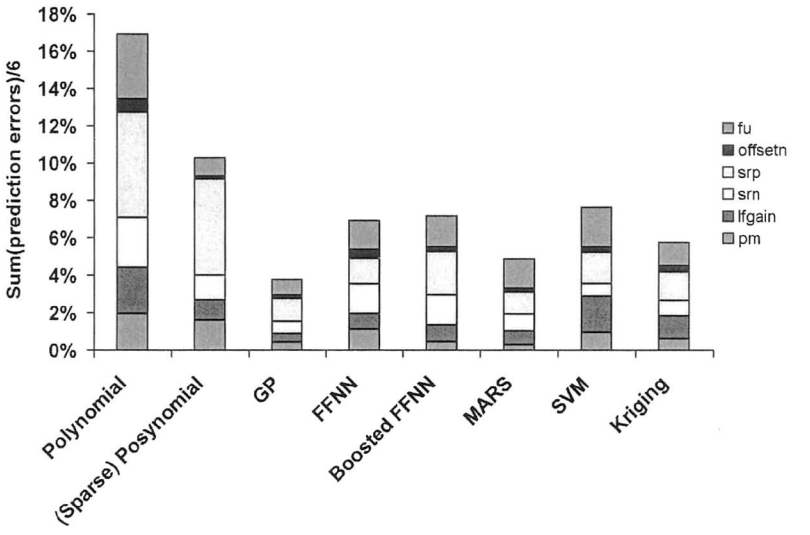

Figure 19-4. Comparison of prediction error for several state-of-the-art modeling approaches.

GP turned out to predict remarkably well. In a separate study on six circuit datasets (McConaghy and Gielen, 2005a), we found that GP could generate nonlinear expressions that outperformed several state-of-the-art approaches, as shown in Figure 19-4.

Table 19-2. GP-generated behavioral models for a latch circuit.

Train error	Expression
15.11%	dx1/dt = nBit dx2/dt = Bit * x1
6.25%	dx1/dt = - 21.3 - 9.28e-03 * bufclk * x1 + 1.0e+04 * nBit * bufclk
3.32%	dx1/dt = 2.21e-02 - 3.72e-02 * x1 - 21.8 * Bit*nBit * bufclk dx2/dt = nBit * bufclk * x1 dx6/dt = x1

7. GP Application: Behavioral Modeling

Another challenge in circuit design is how to manage system-level design. One of its sub-problems is how to simulate a whole system in a feasible time, ideally fast enough to optimize with. A good approach is behavioral models, which approximate the dynamic behavior of each of the system's sub-blocks. Automatically devising behavioral models is very difficult: it's common for a student to spend his whole Ph.D on (manually) designing a good behavioral model for one building block! There's a long history of attempts to automated approaches as well, starting from linear, progressing to weakly nonlinear, and finally recent successes in strongly nonlinear behavioral models. But those approaches are, once again, black box. With behavioral modeling, even more than symbolic analysis, trustworthiness of a model is very important, and black-box models compromise that because there is no guarantee how the model will perform under other input stimuli.

Once again, we saw opportunity. We adapted our GP system to build dynamic models, and tested it on a strongly nonlinear circuit (McConaghy and Gielen, 2005b). It successfully built interpretable behavioral models with good prediction ability. Table 19-2 gives some of the behavioral models generated, at different levels of complexity and accuracy.

8. Conclusions

While GPers have considered analog synthesis a success story for GP, and with good reason from an AI perspective, it still remains for GP to be put into industrial analog design practice.

To understand why, we examined the problem context and the details of how a design is implemented. It comes down to achieving more robust designs, with the main aim of reducing risk of costly manufacturing respins. Furthermore, it needs to be trusted by the designer. To address this, the GP computational effort goes up drastically, and Moore's Law cannot be relied upon to help because the

problem is "Anti-Mooreware." Thus, we have a grand "algorithm engineering" challenge for clever GP researchers.

Structural synthesis is not the only opportunity for GP in analog CAD. We demonstrated GP as applied to two other applications, symbolic modeling and behavioral modeling, where the barrier to entry was far lower, and the industrial payoff much sooner.

GP is not barnstorming the field of analog design... yet. But it is slowly gaining ground in multiple aspects of analog CAD.

9. Acknowledgements

The first author would like to thank John Koza, Matthew Streeter, Sameer Al-Sakran, Lee Jones, and Martin Keane for the invigorating discussions which motivated the writing of this paper.

References

Alpaydin, G., Balkir, S., and Dundar, G. (2003). An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. *IEEE Transactions on Evolutionary Computation*, 7(3):240–252.

Bhattacharya, Sambuddha, Jangkrajarng, Nuttorn, Hartono, Roy, and Shi, Richard (2004). Correct-by-construction layout-centric retargeting of large analog designs. In *Proceedings of the Design Automation Conference*.

Cadence (2005a). Neocell product. *Website of Cadence Design Systems Inc.*

Cadence (2005b). Neocircuit product. *Website of Cadence Design Systems Inc.*

Dastidar, T.R., Chakrabarti, P.P., and Ray, P. (2005). A synthesis system for analog circuits based on evolutionary search and topological reuse. *IEEE Transactions on Evolutionary Computation*, 9(2):211–224.

DeSmedt, B. and Gielen, Georges G.E. (2003). Watson : Design space boundary exploration and model generation for analog and rf ic design. *IEEE Transactions on Computer-Aided Design*, 22(2):213–223.

Drennan, P.C. and McAndrew, C.C. (2003). Understanding mosfet mismatch for analog design. *IEEE Journal of Solid State Circuits*, 38(3):450–456.

Gielen, G.E. (2002). Techniques and applications of symbolic analysis for analog integrated circuits: A tutorial overview. In Rutenbar, R.A., Gielen, G.E., , and Antao, B.A., editors, *Computer Aided Design of Analog Integrated Circuits and Systems*, pages 245–261. IEEE Press, Piscataway, NJ.

Gielen, G.E. and Rutenbar, R.A. (2002). Computer-aided design of analog and mixed-signal integrated circuits. In Rutenbar, R.A., Gielen, G.E., , and Antao, B.A., editors, *Computer Aided Design of Analog Integrated Circuits and Systems*, chapter 1, pages 3–30. IEEE Press, Piscataway, NJ.

Hastings, Alan (2000). *The Art of Analog Layout*. Prentice-Hall.

Hu, J. and Goodman, E. (2004a). Robust and efficient genetic algorithms with hierarchical niching and sustainable evolutionary computation model. In *Proceedings of the Genetic and Evolutionary Computing Conference.*

Hu, Jianjun and Goodman, Erik (2004b). Topological synthesis of robust dynamic systems by sustainable genetic programming. In O'Reilly, Una-May, Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice II*, chapter 9. Kluwer, Ann Arbor.

Koza, John R., Andre, David, Bennett III, Forrest H, and Keane, Martin (1999). *Genetic Programming 3: Darwinian Invention and Problem Solving.* Morgan Kaufman.

Koza, John R., Jones, Lee W., Keane, Martin A., and Streeter, Matthew J. (2004a). Towards industrial strength automated design of analog electrical circuits by means of genetic programming. In O'Reilly, Una-May, Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice II*, chapter 8. Kluwer, Ann Arbor.

Koza, John R., Keane, Martin A., and Streeter, Matthew J. (2004b). Routine high-return human-competitive evolvable hardware. In Zebulum, Ricardo S., Gwaltney, David, Horbny, Gregory, Keymeulen, Didier, Lohn, Jason, and Stoica, Adrian, editors, *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 3–17, Seattle. IEEE Press.

Koza, John R., Keane, Martin A., Streeter, Matthew J., Mydlowec, William, Yu, Jessen, and Lanza, Guido (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* Kluwer Academic Publishers.

Lampaert, Koen, Gielen, Georges G.E., and Sansen, Willy (1999). *Analog Layout Generation for Performance and Manufacturability.* Kluwer Academic Publishers.

Lohn, J.D. and Colombano, S.P. (1998). Automated analog circuit synthesis using a linear representation. In *Proceedings of the Second International Conference on Evolvable Systems: From Biology To Hardware*, pages 125–133. Springer-Verlag.

McConaghy, Trent, Eeckelaert, Tom, and Gielen, Georges G. E. (2005). Caffeine: Template-free symbolic model generation of analog circuits via canonical form functions and genetic programming. In *Proceedings of the Design Automation and Test Europe Conference.*

McConaghy, Trent and Gielen, Georges G. E. (2005a). Analysis of simulation-driven numerical performance modeling techniques for application to analog circuit optimization. In *Proceedings of the International Symposium on Circuits and Systems.*

McConaghy, Trent and Gielen, Georges G. E. (2005b). Ibmg: Interpretable behavioral model generator for nonlinear analog circuits via canonical form functions and genetic programming. In *Proceedings of the International Symposium on Circuits and Systems.*

Phelps, R., Krasnicki, M., Rutenbar, R.A., Carley, R., and Hellums, J.R. (2000). Anaconda: Simulation-based synthesis of analog circuits via stochastic pattern search. *IEEE Transactions on Computer Aided Design.*

Power, J.A., Donellan, B., Mathewson, A., and Lane, W.A. (1994). Relating statistical mosfet model parameters to ic manufacturing process fluctuations enabling realistic worst-case design. *IEEE Transactions on Semiconductor Manufacturing*, 7:306–318.

Razavi, Behzad (2000). *Design of Analog CMOS Integrated Circuits.* McGraw-Hill.

Rutenbar, Rob A. and Cohn, John M. (2000). Layout tools for analog ics and mixed-signal socs: A survey. In *Proceedings of the ACM International Symposium on Physical Design*, pages 76–83.

Schenkel, F., Pronath, M., Zizala, S., Schwencker, R., Graeb, H., and Antreich, K. (2001). Mismatch analysis and direct yield optimization by spec-wise linearization and feasibility-guided search. In *Proceedings of the Design Automation Conference.*

Smedt, B. De and Gielen, Georges G.E. (2003). Holmes: Capturing the yield-optimized design space boundaries of analog and rf integrated circuits. In *Proceedings of the Design Automation and Test Europe Conference*, page 10256.

Soens, C., Wambacq, P., Plas, G. Van Der, and Donnay, S. (2005). Simulation methodology for analysis of substrate noise impact on analog / rf circuits including interconnect resistance. In *Proceedings of the Design Automation and Test Europe Conference.*

Sripramong, T. and C.Toumazou (2002). The invention of cmos amplifiers using genetic programming and current-flow analysis. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems.*

Synopsys (2005). Circuit explorer product. *Website of Synopsys Inc.*

Teller, Astro and Andre, David (1997). Automatically choosing the number of fitness cases: The rational allocation of trials. In Koza, John R., Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max, Iba, Hitoshi, and Riolo, Rick L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 321–328, Stanford University, CA, USA. Morgan Kaufmann.

Zebulum, R., Pacheco, M., and Vellasco, M. (2002). *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms.* CRC Press.

Zhang, Gang, Dengi, E. Aykut, Rohrer, Ronald A., Rutenbar, Rob A., and Carley, L. Richard (2004). A synthesis flow toward fast parasitic closure for radio-frequency integrated circuits. In *Proceedings of the Design Automation Conference*, pages 155–158.