Reverse-Engineering Nonlinear Analog Circuits with Evolutionary Computation

Theodore W. Cornforth[1] and Hod Lipson[2]

[1] Department of Biological Statistics and Computational Biology, Cornell University, Ithaca, USA 14853
twc63@cornell.edu
[2] Department of Mechanical and Aerospace Engineering, Cornell University, Ithaca, USA 14853
hl274@cornell.edu

Abstract. The design of analog circuits by hand is a difficult task, and many successful approaches to automating this design process based on evolutionary computation have been proposed. The fitness evaluations necessary to evolve linear analog circuits are relatively straightforward. However, this is not the case for nonlinear analog circuits, especially for the most general class of design tasks: reverse-engineering an arbitrary nonlinear 'black box' circuit. Here, we investigate different approaches to fitness evaluations in this setting. Results show that an incremental algorithm outperforms naive approaches, and that it is possible to evolve robust nonlinear analog circuits with time-domain output behavior that closely matches that of black box circuits for any time-domain input.

1 Introduction

Analog circuit design is a challenging task that typically requires a domain expert with years of experience [14]. As a result, the costs associated with developing circuits for new applications can be very high. Many studies reported in the literature have attempted to automate the circuit design process in an attempt to reduce the human effort required [4,5,7,13]. Approaches involving evolutionary computation have been particularly successful, and these generally allow both the topology and component sizes of a circuit to be optimized. Perhaps the earliest work in this area was by Koza and colleagues, who employed genetic programming and a developmental encoding [8]. Later work showed that simpler methods based on genetic algorithms could also obtain good results [1, 9, 10, 19].

All approaches to the automation of analog circuit design involving evolutionary computation rely on fitness evaluations in which the behavior of a candidate circuit is simulated and compared with a design specification. For linear circuits, such as those comprised only of passive resistors, capacitors, and inductors, these fitness evaluations are relatively straightforward as the behavior of a linear circuit is fully characterized by its frequency and phase responses [16]. In a typical application, the AC frequency response of an evolved circuit can be compared with the desired frequency response using a measure such as the mean-squared

O.H. Ibarra et al. (Eds.): UCNC 2014, LNCS 8553, pp. 105–116, 2014.
DOI: 10.1007/978-3-319-08123-6_9, © Springer International Publishing Switzerland 2014

error [8]. This error then serves as the basis for assigning a fitness value to the evolved circuit (Fig. 1A.). For nonlinear circuits, such as those with transistors, diodes or other nonlinear components, fitness evaluations can be considerably more difficult, as it is generally not possible to fully characterize the behavior of a nonlinear circuit with a single measurement such as AC frequency or phase response.

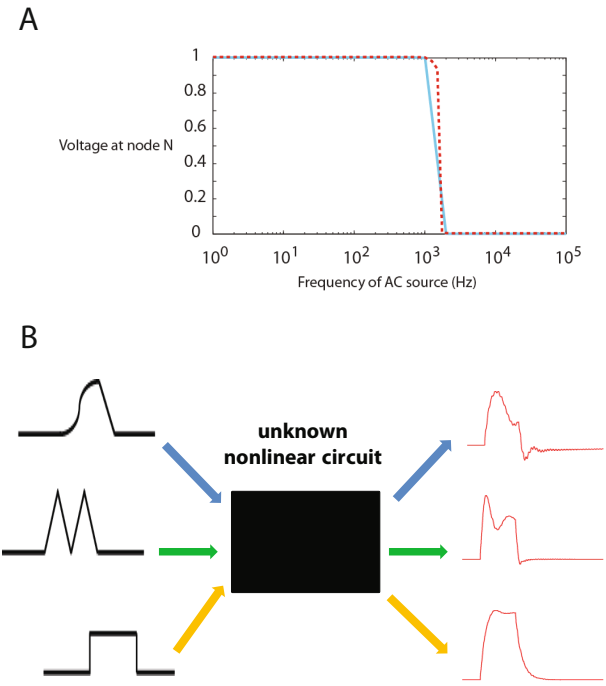


Fig. 1. Fitness evaluations for linear and non-linear circuits. **A.** The fitness of an evolved linear circuit can easily be assessed by comparing its frequency response (red) to a desired frequency response (blue). In this case the desired behavior is that of a low pass filter. **B.** To reverse-engineer an unknown 'black box' nonlinear circuit, time domain inputs (black) are used to probe the black box. The resulting outputs (red) are compared with the outputs of an evolved circuit in response to the same inputs as a means of assigning a fitness value to the evolved circuit.

Despite this complicating factor, some studies have successfully 'forward-engineered' nonlinear analog circuits with evolutionary computation [8, 15]. In these studies, a problem-specific design specification is developed and the task is to evolve a circuit that meets this design specification. Depending on the nature of the target design, Fourier analysis, DC sweeps, time-domain transient analysis, or other analyses are applied to an evolved circuit and used to compare its behavior to the design specification. Although time-domain analyses are the

most computationally expensive, these are often considered necessary to evolve the most robust circuit designs [11, 15].

The forward-engineering tasks considered in previous studies of nonlinear analog circuit evolution are fundamentally different from a reverse-engineering task in which a nonlinear circuit must be evolved without a design specification. Instead, the circuit is evolved to match the behavior of an unknown 'black box' nonlinear circuit (Fig. 1B.). The black box circuit must be probed with different voltage or current inputs in order to obtain information used as the basis for evaluating the fitness of evolved circuits, yet the appropriate type of stimuli to use for these probes would generally not be known in advance. This is especially true if the goal is to evolve a circuit that generalizes well and reproduces the time-domain output behavior of a black box circuit for arbitrary time-domain inputs not used during evolution.

In this study, we investigate different algorithms for selecting the type of probe stimuli to use for the efficient evolutionary design of nonlinear analog circuits in the most general reverse-engineering setting. Many possibilities exist for such an algorithm, including the incremental approaches that have proven very powerful in other applications of evolutionary hardware [18]. Using randomly-generated nonlinear circuits in order to evaluate different approaches in as unbiased a manner as possible, our results show that incrementally presenting probe stimuli is a particularly efficient means of guiding nonlinear circuit evolution. Importantly, the evolved circuits generalize well and match target circuit behavior for arbitrary stimuli.

2 Methods

2.1 Genetic Algorithm

We use a genetic algorithm in which circuits are represented as variable-length linear chromosomes, where each gene contains four elements (Fig. 2). The circuits considered in this work were composed of five component types: resistors, capacitors, inductors, pnp bipolar transistors, and npn bipolar transistors. In the case of resistors, capacitors, and inductors, the elements in a gene specify the type, connection nodes, and parameter value of the component. To mimic the limitations of physical hardware, these components had one of 96 possible parameter values taken from a log10 scale with 12 values per decade. These values ranged from 1.0 – 8.2x10^7 pF, nH, and Ω for capacitors, inductors, and resistors respectively. In the case of bipolar transistors, NG-Spice default models are used and the four-element gene specifies transistor type (pnp or npn) and the connection nodes for emitter, base, and collector.

Each individual in the population of 64 circuits was initialized by concatenating 1-9 randomly chosen genes. The resulting circuit was checked for topological validity. This included checks for dangling nodes, disconnected subcircuits, inductor loops, the lack of capacitor paths to ground, and other common problems that would make the circuit unsimulatable in Spice.

```
...
resistor3    n3     n9     123.45Ω
resistor8    n8     n2     31.1112Ω
inductor4    n4     n5     1.132H
capacitor3   n5     n2     1361.8F
resistor7    n3     n4     45.0Ω
inductor9    n16    n8     0.2529H
inductor3    n3     n12    6.3H
resistor5    n9     n2     19.9054Ω
capacitor1   n15    n6     26.37F
...
```

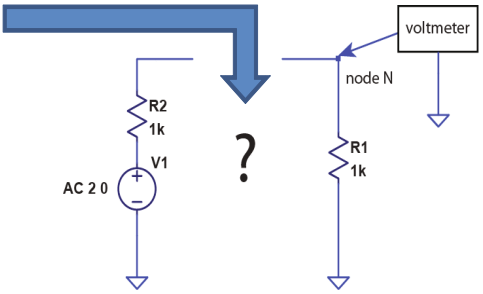


Fig. 2. Circuit representation. Circuits were represented as linear chromosomes, with each gene (such as the one highlighted in red) corresponding to one component. This list specifies the components in a variable portion of an otherwise invariant embryonic circuit. Note that the embryonic circuit contains both a voltage source and an output node at which voltage measurements are made.

As in standard genetic algorithms, crossover and mutation were used to create new members of the population. Parents were selected at random from the population of 64 and used to generate two offspring with a 90% chance of crossover. Instead of one point, two point, uniform or another standard type of crossover for linear chromosomes, we used the topology-aware crossover described in [3]. Regardless of whether the two offspring were generated by crossover, each gene in each offspring was mutated with a probability of 5%. A mutation consisted of the change from one type of element to another, a change of connections from one node to another, or a change in parameter values. We did not find it necessary to use mutations to add or delete genes as crossover in the variable-length chromosomes provided sufficient variation in offspring size. Offspring were created in this manner until 64 offspring were obtained. Both crossover and mutation ensured that topologically valid circuits resulted from the operation.

We used genotypic age to perform survival selection with the Age-fitness Pareto algorithm [17] for some experiments. Survival selection was performed by culling the combined population of parents and offspring using tournament selection with replacement and tournaments of size 2. This process was implemented with a Pareto tournament scheme in which two random members of the combined population were selected. If one of the pair had both lower fitness and higher age than the other, it was thrown out. The survivor was then returned to the pool. This continued until the population size was reduced back to 64. Age was defined as the number of generations in which an individual had been present in the population. Offspring inherited the age of the older parent in the case of crossover. One new, randomly generated individual with an age of 0 and created in the same way as members of the initial population was added to the population each generation.

When evaluating the fitness of a circuit, the linear chromosome was translated into a netlist recognizable by the NG-Spice circuit simulator [12]. This netlist

specified the variable portion of an otherwise invariant embryonic circuit common to all evolved and target circuits (Fig. 2). Time-domain transient analysis was used to measure the voltage of the circuit at the output node in response to a voltage source with an arbitrary waveform as the input. The differences between evolved circuit outputs and target outputs were compared by calculating the sum of squared errors in the time domain. Random voltage source waveforms were specified by 20 parameters: 18 random parameters corresponding to the first nine coefficients of the Fourier series, one parameter controlling the scaling of the waveform with respect to time, and one parameter controlling the phase offset of the waveform.

2.2 Stimulus Selection

As the focus of this study was on comparing different methods of selecting probe stimuli for obtaining information about target circuits and evaluating the fitness of evolved circuits, we employed several methods for doing so. The simplest method ('fixed single') was to probe the target black box circuit with a single randomly generated voltage source waveform. The resulting voltage output was then used as the basis for assigning fitness values to evolved circuits. Each fitness evaluation consisted of applying the same input to the evolved circuit and comparing the resulting output in the time domain with that output originally obtained for the target circuit. Fitness was calculated as the sum of squared errors between target output and the output of the evolved circuit.

A second method for stimulus selection, 'fixed four', was inspired by the fact that voltage ramp and step functions are very common hand-designed waveforms used to probe circuits [11]. Two fixed but different voltage ramps and two fixed but different step functions were each used to probe the target and evolved circuits. Fitness was assigned as the sum of the errors on each of the four targets.

The third stimulus selection method, 'switching single', was the same as fixed single except that a new randomly generated target input/output pair was generated periodically. In other words, the stimulus switched at regular intervals during evolution. This method was used to determine whether overfitting to a fixed single input/output pair was a significant problem.

Finally, we also employed incremental stimulus selection. Inspired by work in digital circuit evolution [18] and other evolvable hardware domains, this method involved periodically generating a new random stimulus, applying it to the black box target circuit, and then adding this new input/output pair to the set of test cases used to evaluate fitness of an evolved circuit. The fitness of an evolved circuit was then defined simply as the sum of its fitnesses for each individual stimulus. The 'switching incremental' algorithm employed this approach without Age-fitness Pareto optimization whereas the 'age-fitness incremental' algorithm was identical except that Age-fitness Pareto optimization was used.

For all five algorithms, the time required to evolve circuits was almost entirely spent on the computationally expensive Spice simulations required for fitness evaluation. As there are significant differences in the number of fitness evaluations required for the five algorithms for a given number of evolutionary generations, all

experiments below account for these differences and use the same total number of fitness evaluations (Spice simulations) for all algorithm comparisons. Given the computational expense of Spice simulations, we performed evolutionary runs for a predetermined number of fitness evaluations instead of following the standard practice of letting the genetic algorithm run until stagnation. However, we found that the number of fitness evaluations used was sufficient to draw preliminary conclusions about the performance of the various algorithms.

2.3 Target Circuits

We initially used a hand-designed RTL inverter circuit as a reverse-engineering target. A RTL inverter is a simple transistor switch that implements logical negation. Not counting the elements of the embryonic circuit, this circuit had three elements as shown in (Fig. 3). Each of 40 evolutionary trials used this same simple circuit as a target to obtain baseline information about the performance of the algorithms.

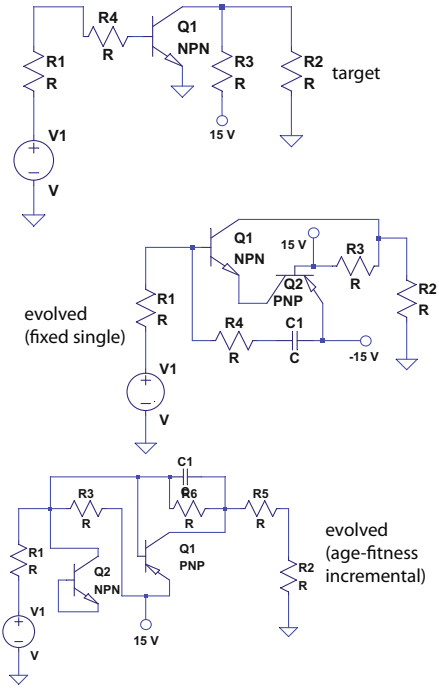


Fig. 3. RTL inverter target and evolved circuits. The target circuit is shown at the top. The other circuits are those evolved to match the behavior of this target circuit using a fixed single stimulus (middle) and the age-fitness incremental algorithm (bottom).

In order to evaluate different stimulus selection methods in as unbiased a manner as possible, we also used randomly-generated nonlinear circuits as the

black box targets. An example of such a random circuit is shown in (Fig. 4). These random circuits were generated by creating random netlists with 3–11 elements and at least one nonlinear element (bipolar transistor). The elements specified by a netlist were inserted into the variable portion of the embryonic circuit and the resulting circuit was simulated. If the circuit was simulatable and for 25 randomly chosen input stimuli the outputs of the circuit were of sufficient variability, the circuit was included in the set of black box targets. Otherwise, it was discarded and a new random circuit was generated. This process continued until 40 random circuits with a variety of sizes were generated.

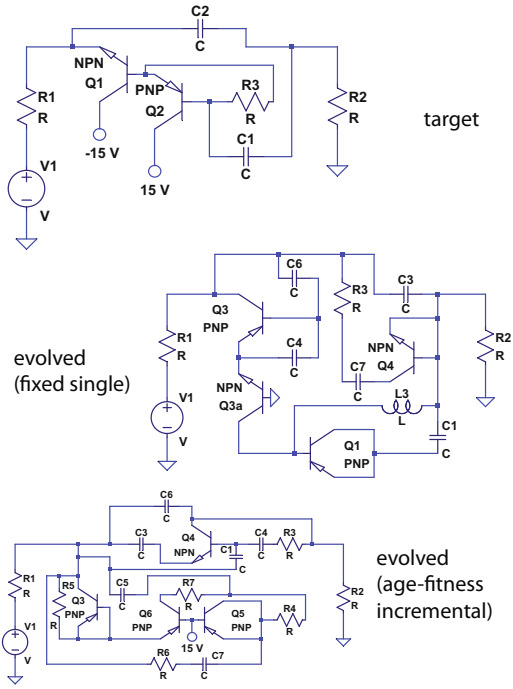


Fig. 4. Random target and evolved circuits. One randomly generated target circuit is shown at the top. The other circuits are those evolved to match the behavior of this target circuit using a fixed single stimulus (middle) and the age-fitness incremental algorithm (bottom).

3 Results

Each of the five stimulus selection algorithms were applied to the target circuits in 40 trials. The results summarizing evolutionary progress across all 40 trials for all five algorithms with the RTL inverter as the target are shown in Fig. 5. Similar results were obtained for random circuits as shown in Fig. 6. In both cases, presenting stimuli with the incremental algorithm was clearly the most

effective method of evolving circuits that generalized well and that matched the performance of the target circuits for several random stimuli not used during evolution. In contrast, little benefit was gained by using different non-incremental approaches given a fixed computational budget.

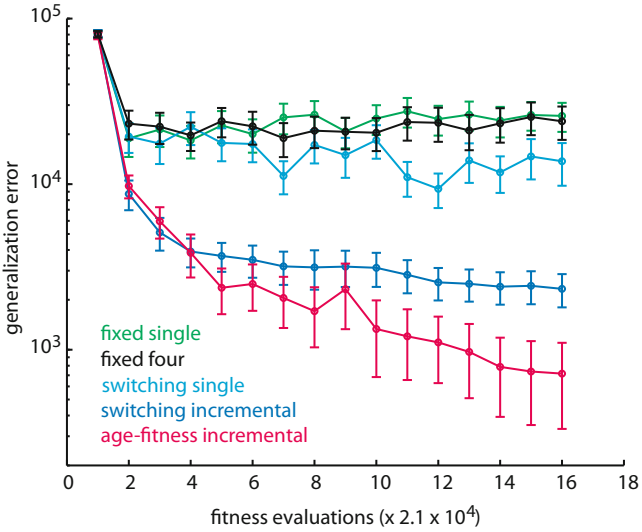


Fig. 5. Generalization error for the five stimulus selection methods with the RTL inverter target. At regular intervals (approximately once every 2.1×10^4 fitness evaluations), each member of the current population of circuits was evaluated on 25 randomly chosen stimuli not seen during training. The circuit with the lowest total error was then recorded. The mean value of this error across each of 40 trials is plotted, with error bars representing plus or minus the standard error of the mean.

Fig. 3 shows two examples of circuits obtained with a fixed single stimulus and with the age-fitness incremental algorithm, as well as the RTL inverter target circuit whose behavior they were evolved to match. Although the evolved circuits do not match the target circuit in size or structure, the circuit evolved using the age-fitness incremental algorithm closely matched the behavior of the target circuit for arbitrary stimuli (Fig. 7). A similar conclusion can be drawn from the results for random circuits as shown in Fig. 4 and Fig. 8.

4 Discussion and Conclusion

In this study, we compared different methods of fitness evaluation for reverse-engineering nonlinear analog circuits with a genetic algorithm. Stimulus presentation with an incremental algorithm proved to be a particularly effective means of performing these fitness evaluations. This method successfully produced circuits that closely matched the behavior of black box target circuits.

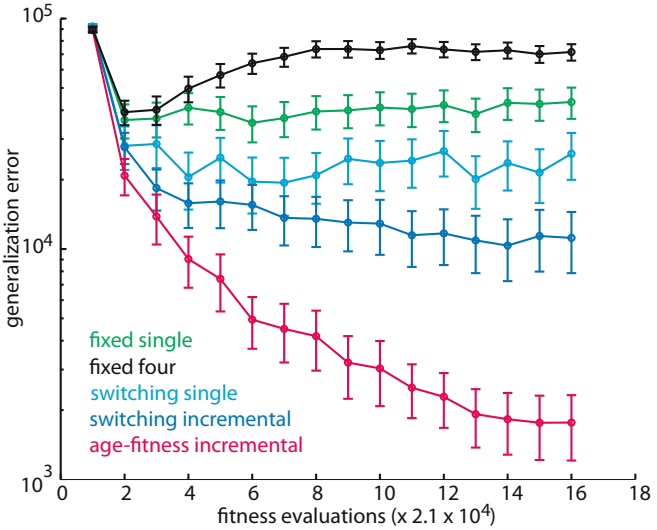


Fig. 6. Generalization error for the five stimulus selection methods with random targets. Axes and bars are the same as in Fig. 5.

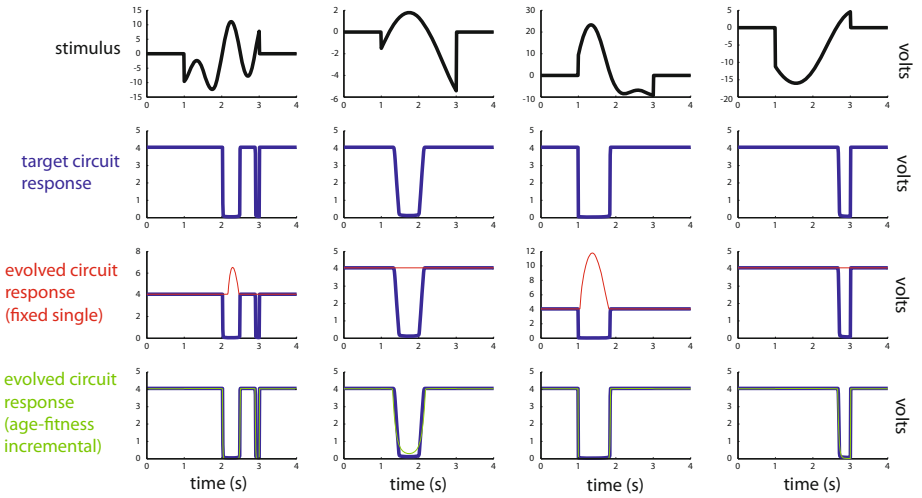


Fig. 7. RTL inverter target and evolved circuit behavior. Four randomly chosen stimuli not used during training (top row) were applied to the target circuit, which generated the responses shown in the second row. The response of the circuit evolved with a fixed single stimulus shown in Fig. 5 to those same four stimuli is shown in red in the third row. These red curves are plotted over the target responses for reference. The response of the circuit evolved with the age-fitness incremental algorithm shown in Fig. 5 to those four stimuli is shown in green in the bottom row plotted over the target responses.

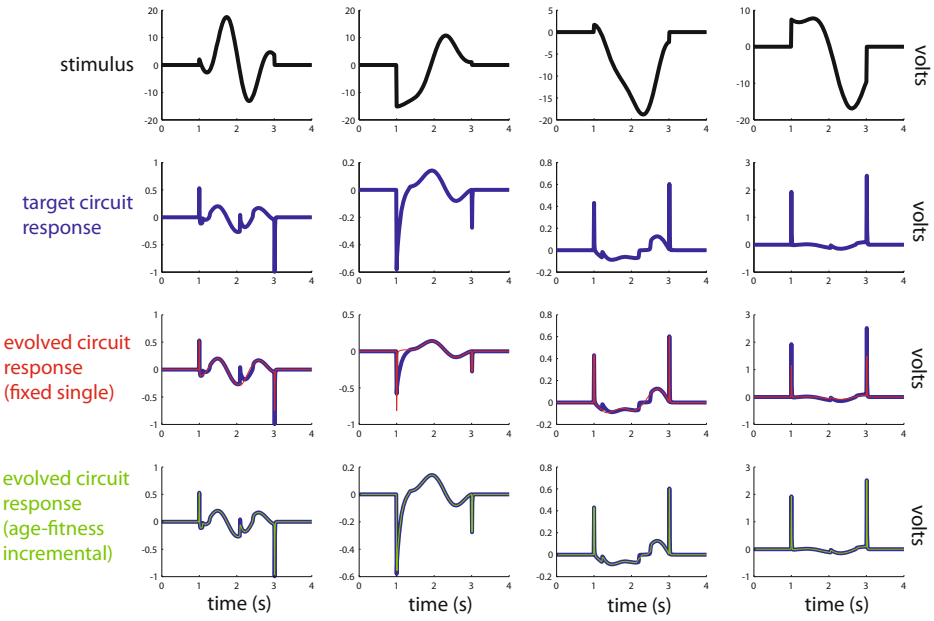


Fig. 8. Random target and evolved circuit behavior. Panels are the same as in Fig. 7 except that results are for circuits shown in Fig. 6.

One weakness of the approaches studied here is that the evolved circuits fail to match their respective target circuits in size and structure, even when they match them in behavior. It could then be argued that the evolved circuits are no more valuable as models of the target circuit than a neural network or other universal function approximator that can also reproduce the input/output behavior of the target circuit. However, we argue that using circuits as the function approximator confers two advantages. First, circuits are almost certainly the most useful function approximators for a reverse-engineered circuit if the function approximator is to be implemented in hardware. Second, we hypothesize that circuits and electrical components, although computationally expensive to simulate, are a natural representation with which to approximate the input/output functions embodied by the target nonlinear circuits and are therefore easier to identify than neural networks or other non-domain specific representations. We are currently testing this hypothesis and preliminary results are supportive.

For future work, we intend to extend the results here in several ways. First, we are studying random target circuits of larger sizes, as reverse engineering targets in real life applications tend to contain considerably more components than 3-11 as in the circuits considered here. Second, we will consider physical circuits implemented in hardware as targets as an important means of confirming the results obtained here with simulations. Third, we intend to investigate the prospect of coevolving stimuli instead of randomly generating them as in the current study. Such an approach has proven effective in other evolutionary

computation problem domains such as symbolic regression [2]. Fourth, formally characterizing the complete input/output behavior of nonlinear dynamical systems such as the analog circuits in this work would be feasible with enough computational effort [6]. Such steps will be important to provide a firm theoretical basis for concluding that the evolved circuits are robust under real-life operating conditions, a conclusion that was only informally demonstrated with the experiments described in the present work. Finally, the approaches here will be applied to additional circuits with specific, known functions such as cube root circuits [8]. Although the results will not be as general as when the targets are random circuits, this will be useful as a baseline comparison of the approaches considered here to the more common evolutionary hardware methods oriented to forward-engineering tasks.

Acknowledgments. Support was provided by the Tri-Institutional Training Program in Computational Biology and Medicine, National Science Foundation grant ECCS 0941561 on Cyber-enabled Discovery and Innovation (CDI), National Institutes of Health NIDA grant RC2 DA028981, and the U.S. Defense Threat Reduction Agency (DTRA) grant HDTRA 1-09-1-0013.

References

1. Ando, S., Ishizuka, M., Iba, H.: Evolving Analog Circuits by Variable Length Chromosomes. In: Ghosh, A., Tsutsui, S. (eds.) Advances in Evolutionary Computing. Springer (2003)
2. Bongard, J., Lipson, H.: Automated Reverse Engineering of Nonlinear Dynamical Systems. PNAS 104, 9943–9948 (2007)
3. Das, A., Vemuri, R.: An Automated Passive Analog Circuit Synthesis Framework Using Genetic Algorithms. In: Proc. IEEE VLSI 2007, pp. 145–152 (2007)
4. El-Turky, F.M., Nordin, R.A.: BLADES: An Expert System for Analog Circuit Design. In: Proc. IEEE ISCAS, pp. 552–555 (1986)
5. Harjani, R., Rutenbar, R.A., Carley, L.R.: OASYS: A Framework for Analog Circuit Synthesis. Trans. Comp.-Aided Des. Integ. Cir. Sys. 8, 1247–1266 (2006)
6. Hedrich, L., Barke, E.: A Formal Approach to Nonlinear Analog Circuit Verification. In: Proc. IEEE ICCAD 1995, pp. 123–127 (1995)
7. Koh, H.Y., Sequin, C.H., Gray, P.R.: OPASYN: A Compiler for CMOS Operational Amplifiers. IEEE Trans. Comput.-Aided Des. 9, 113–125 (1990)
8. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A., Dunlap, F.: Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming. IEEE Trans. Evol. Comp. 1, 109–128 (1997)
9. Kruiskamp, W., Leenaerts, D.: DARWIN: CMOS Opamp Synthesis by Means of a Genetic Algorithm. In: Proc. of the 32nd Design Automation Conference, pp. 433–438 (1995)
10. Lohn, J.D., Colombano, S.P.: Automated Analog Circuit Synthesis Using a Linear Representation. In: Sipper, M., Mange, D., Pérez-Uribe, A. (eds.) ICES 1998. LNCS, vol. 1478, pp. 125–133. Springer, Heidelberg (1998)
11. Mydlowec, W., Koza, J.: Use of Time-domain Simulations in Automatic Synthesis of Computational Circuits Using Genetic Programming. In: Proc. GECCO 2000, pp. 187–197 (2000)

12. NG-Spice, http://sourceforge.net/projects/ngspice
13. Rutenbar, R.A.: Analog Design Automation: Where Are We? Where Are We Going? In: Proc. l5th IEEE CICC, pp.13.1.1–13.1.8 (1993)
14. Rutenbar, R.A., Gielen, G.G.E., Antao, B.A.: Computer-Aided Design of Analog Integrated Circuits and Systems. Wiley-IEEE Press (2002)
15. Sapargaliyev, Y.A., Kalganova, T.G.: Open-Ended Evolution to Discover Analogue Circuits for Beyond Conventional Applications. Genet. Prog. Evol. Mach. 13, 411–443 (2012)
16. Schaumann, R., Van Valeknburg, M.E.: Design of Analog Filters. Oxford University Press, New York (2001)
17. Schmidt, M.D., Lipson, H.: Age-Fitness Pareto Optimization. Genetic Programming Theory and Practice 8, 129–146 (2010)
18. Torresen, J.: A Scalable Approach to Evolvable Hardware. Genet. Prog. Evol. Mach. 3, 259–282 (2002)
19. Zebulum, R.S., Pacheco, M.A., Vellasco, M.: Comparison of Different Evolutionary Methodologies Applied to Electronic Filter Design. In: Proc. IEEE WCCI, pp. 434–439 (1998)