A Modular Framework for the Evolution of Circuits on Configurable Transistor Array Architectures.

Martin Trefzer, Jörg Langeheine, Karlheinz Meier, Johannes Schemmel
Ruprecht-Karls-University of Heidelberg
Kirchhoff Institute for Physics
Im Neuenheimer Feld 227, 69120 Heidelberg, Germany
martin.trefzer@kip.uni-heidelberg.de, +49 (0)6221 54-9838
http://www.kip.uni-heidelberg.de/vision/projects/eh/

Abstract

This paper gives an overview over the progress that has been made by the Heidelberg FPTA group within the field of analog evolvable hardware. Achievements are the design of a CMOS configurable transistor array (FPTA), the development of evolutionary algorithms (EAs) for analog circuit synthesis and the implementation of a modular framework, which makes it possible to use various substrates and simulation models for evolution experiments. The improvement of the EA is shown by comparing the performance of three implementations in evolving comparators. Additionally, results, obtained from the FPTA for the evolution of oscillators from scratch, are presented as an example for the successful application of the multi-objective Turtle GA. Finally, it is shown that a simplified software model of the Heidelberg FPTA is suitable to assess the real hardware, indicated by the fact that both substrates perform equally well in finding good solutions for comparators. This work aims at creating a customizable, modular framework that facilitates research on the performance and evolvability of possible FPTA topologies in the future.

1. Introduction

There is a great need for analog circuits carrying out complex tasks in a great variety of applications ranging from simple switches to complex DACs, ADCs or OPs. The classical way of designing suitable circuits for specific needs is yet to create a schematic and an according layout, which are conform with an available target technology. Once the functionality of the designed circuit is verified with an analog circuit simulator, a corresponding prototype can be fabricated. Current design flows provide a very high degree of reliability on the design process and the outcome, which is a great advantage. Despite this, in the digital regime, reconfigurable logic chips, namely field programmable gate arrays (FPGAs), are gaining more and more of importance with their increasing speed, complexity and flexibility.

Contrary to that, field programmable analog arrays (FPAAs) are to date not very elaborated and only a few analytic solutions for analog design automation are available [2, 6]. There are three main reasons that make the design of powerful FPAAs very difficult: First, the lack of an analytical language in which the behavior of any analog circuit can be expressed. Second, it is a very challenging task to design a reconfigurable analog substrate which provides both, the flexibility to host a great variety of circuits and sufficient control over the influence of parasitic effects. Third, even if there was yet a substrate fulfilling the latter demands, it is still challenging to develop algorithms, which are able to configure or map existing topologies to such a substrate. As a consequence of this, it is necessary to develop topologies and algorithms at the same time.

Promising approaches on the algorithmic side are made by using evolutionary algorithms (EAs) for the—in some cases—simultaneous synthesis of topology and component sizing of analog circuits [1, 7, 10–12]. If the focus is set on the automatic synthesis of complex topologies, developmental strategies like genetic programming (GP) [8, 13, 16] or heuristic interconnection of building blocks [9] are successfully applied. In addition to that, multi-objective EAs [3, 4] are suitable for taking the numerous variables into account, that need to be optimized for complex problems like analog circuit design.

In this paper, the improvements that have been made by developing more powerful evolutionary algorithms are pointed out by using a comparator as an example analog circuit. It is shown that solutions with a fitness comparable to a manually defined circuit can be synthesized on the

FPTA. Additionally, oscillators of different frequencies can be evolved from scratch on the chip employing the *MO-Turtle GA* presented in [15]. Further, a modular evolution system is introduced which makes it possible to use different substrates and simulators for evolution experiments. This provides the possibility of designing software models of various FPAA architectures, which can be evaluated with the presented framework before fabricating a chip. Tackling the evolution of comparators, the system is proven to work by creating a software model of the FPTA, referred to as the SimFPTA, and by comparing the results to those obtained from the chip. The presented customizable modular framework is intended to facilitate research on the performance and evolvability of future FPAA topologies in hardware.

2. Evolution System

The entire hardware evolution setup consists of the FPTA chip, that hosts the programmable transistor array, a standard PC that runs the evolutionary algorithm (EA) and organizes the voltage test patterns, and a custom made PCI interface card which represents a flexible FPGA based conroller and interface for the chip. Those components provide a flexible realtime measuring system for analog hardware evolution experiments on the FPTA. Since the whole setup is build in a modular manner, the hardware can be easily replaced with any analog circuit simulator. Thus, it is possible to operate various substrates or simulation models of substrates within the presented evolution system. In these experiments, a second substrate is represented by a simplified SPICE model of the current FPTA, referred to as the SimFPTA throughout the remainder of this paper.

2.1. Modular Evolution Environment

The EA library is implemented in a very flexible way using the C++ programming language and is based on the `GALib` of Matthew Wall [17], although numerous additions and changes have been made. Polymorphism is used for all classes representing the evolutionary algorithm, the genome and the experimental setup which facilitates the implementation of new structures that inherit EA functionality from the base classes, i.e. it is easy to add custom genomes and immediately use them within the existing framework. All methods, which are operating on the population and its individuals—namely initialize, mutate, cross, analyze, evaluate and select—, are implemented in static methods, hence, they are available at any time and from anywhere within the application by simply accessing them through pointers to those methods kept in the respective classes. This mechanism allows for arbitrary combinations of operations or even for changing them at runtime, if desired. UML diagrams of the relevant classes are graphed in Fig. 1.

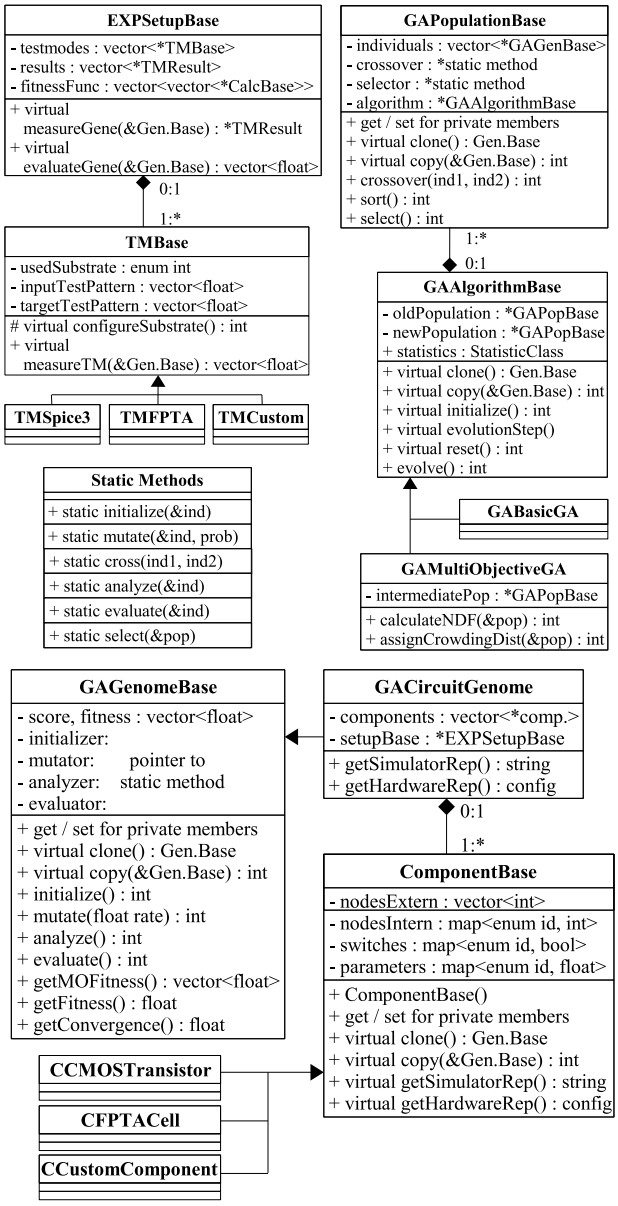

Figure 1. UML diagrams of all relevant classes of the implemented customizable, modular evolution framework.

2.1.1 GAAlgorithmBase: The Evolution Framework

The `GAAlgorithmBase` class is the core of the evolutionary algorithm: It manages the populations and genomes and carries out the evolutionary loop, hence, all presented algorithms are derived from this base class. Only two methods have to be implemented in order to customize a new algorithm, namely the *initialize()* and the *evolutionStep()* method, whereas any other functionality is inherited from

the base class.

2.1.2 GACircuitGenome and ComponentBase

The `GACircuitGenome` is derived from the more general class `GABaseGenome` and is extended with datastructures that contain the circuit components and with a pointer to the experimental setup, described in 2.1.3. Additionally, two new methods (*getSimulatorRep(), getHardwareRep()*) are added and need to be implemented depending on the targeted substrate or simulator. The vector containing pointers to the '`ComponentBase`' objects—that can be specialized to any desired circuit component—represents the actual circuit. In this paper, a representation of the programmable transistor cells of the FPTA chip is used for the experiments, thus, transistor circuits can be evolved either on the FPTA or the SimFPTA.

The `ComponentBase` class contains a list of external and internal nodes, respectively. Switches can be configured in order to connect the actual component to either of those nodes or for directly interconnecting any of the available nodes. Finally, component parameters can be defined for the inner components, i.e. W/L in case of transistors. The external nodes offer the possibility to interconnect multiple components in order to build large circuits.

2.1.3 EXPSetupBase and TMBase: Using different Substrates or Target Technologies

The `EXPSetupBase` class contains a vector of pointers to specialized `TMBase` objects—each representing one testmode of the whole experiment—consisting of the input and the target voltage pattern as well as any information about how to use the respective evolution platform. An important feature is the separation of the evaluation process in two independent steps: measuring (`TMBase`) and fitness calculation (`EXPSetupBase`), which makes it possible to use multiple substrates in parallel and to parallelize measuring.

2.2. Implementations of the Evolutionary Algorithm

In all cases, the genetic operators, namely mutation and crossover, are implemented with regard to cell-based (FPTA) architectures. Therefore, the operators consist of two modules working on cell level and working on array level, respectively. As a consequence of this, the inner structure of the components (cells) is transparent to the used evolutionary algorithm.

2.2.1 The Basic GA

The *Basic GA* is based on the simple genetic algorithm introduced in [5]. It is straight forward implemented for the configurable transistor cells:

Mutation. Connections within and between components are randomly enabled or disabled and the parameters of the components—W,L in case of transistors—are varied due to a probability given by the *mutation rate*.

Crossover. The crossover operator works on cell level and exchanges equally sized rectangular blocks of cells of two selected crossover partners. Size and position of the blocks are randomly chosen. The execution of the crossover operator is controlled by a probability given by the *crossover rate*.

2.2.2 The Turtle GA

Contrary to the *Basic GA*, the *Turtle GA*, reported in [14], produces only circuits that contain no floating nodes or terminals. Consequently, circuits that are synthesized using the *Turtle GA* work in simulation as well as on hardware substrates, thus, can be transferred to various technologies. The genetic operators are defined as follows:

Random Wires (Mutation). The mutation operator randomly selects a node of an arbitrary cell as starting point for the algorithm and randomly decides whether to create or to erase circuit components. Subsequently, the selected node is connected to (or disconnected from) a random, directly connectable target node or transistor terminal, which is provided by the respective substrate. The mutation operator is carried out recursively by selecting the occurring target nodes as new starting points for the next iteration, until the resulting circuit contains no dangling nodes and no floating transistor terminals. The width and length of all active transistors is mutated due to a configurable probability.

Implanting a Foreign Block of Cells (Crossover). The *implanting* crossover operator is carried out in two stages. The first stage exchanges randomly sized and positioned rectangular blocks of transistor cells between two randomly selected individuals. The size of both blocks has to be the same for each individual, whereas the positions of the blocks may differ. Since the crossover operation in general breaks the layout of both previously intact circuits, the second stage fixes the occurring floating nodes by applying the random wires mutation operator to each of them. Consequently, the resulting circuits contain no floating nodes.

2.2.3 The Multi-Objective Turtle GA

A multi-objective (MO) evolutionary algorithm allows individuals with a bad over-all performance to survive as long as they are superior in at least one objective, i.e. they are partially better than all other individuals within the population. The subset of those partially better individuals is called the non-dominated front (NDF). Hence, an MO approach is more suitable for assessing candidate solutions than a single objective approach, since different—sometimes even

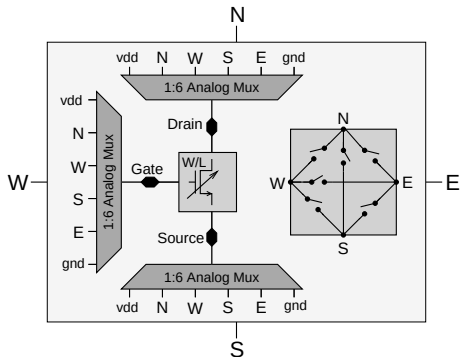

Figure 2. The block diagram of an FPTA and SimFPTA MOS transistor cell.

contradicting—properties of analog circuits have to be separately optimized at the same time. Additionally, a great diversity within the population is inherently provided and it is possible to harvest a whole set of solutions that features both, individuals with a good over-all performance and individuals that are superior in different design corners. Multi-objective optimization has been first proposed in [5] and the presented implementation is based on [4]. In case the *Turtle GA* is run in multi-objective mode, as presented in [15], two additional evaluations have to be carried out for each evolutionary step:

Non-Dominated Sorting. All individuals are classified by calculating their level of non-domination, based on their objective values p_i. An individual p is said to dominate q, denoted by $p \preceq q$, if and only if p is partially less than q (Eq. 1).

$$\forall i \in (1,\ldots,n),\, p_\mathrm{i} \leq q_\mathrm{i} \quad \wedge \quad \exists i \in (1,\ldots,n) : p_\mathrm{i} < q_\mathrm{i} \quad (1)$$

$$\mathrm{NDF} := \{p \in P \mid \nexists p' \in P : p' \preceq p\} \quad (2)$$

All p satisfying Eq. 1, 2 provide the first non-dom. front NDF_1. The succeeding NDFs are found by removing the individuals of NDF_k from the population $P' = P \setminus \mathrm{NDF}_\mathrm{k}$ and by recalculating Eq. 1, 2 for the new population P' until $\mathrm{NDF}_{\mathrm{k}+1}$ is empty.

Crowding Distance. The crowding distance c_dist is a measure for the distribution of the solutions within the fitness landscape. All objective values are considered for calculating the quantity c_dist, which represents an average distance to the nearest neighbors of p and is assigned to each individual of the population. Therefore c_dist is used to steer the evolution towards a uniform distribution of the individuals over the NDF.

2.3. Evolvable Substrates

In this paper, two substrates are employed for circuit evolution: First, the current FPTA [10] and second, a simplified simulation model of the chip, referred to as the SimFPTA. Configurations for both substrates are represented by suitable data structures in software, as can be seen from Fig. 1. In principle, the component classes of the software library can be used to model any FPTA (or circuit) architecture and use it for evolution experiments. Thus, possible new topologies can be tested in simulation and, if a promising architecture is found, a new chip can be manufactured in order to exploit the speed of real hardware, which is significantly faster than simulation.

2.3.1 The FPTA

The FPTA consists of 16x16 configurable CMOS transistor cells (Fig. 2). Hence, a total of 256 programmable transistor cells are available, half of them designed as NMOS and PMOS, respectively. The W/L ratio of those transistor cells can be configured in wide ranges ($W = 1,2,4,8\,\mu$m, $L = 0.6,1,2,4,8\,\mu$m) and each cell is connected to the four nearest neighbors (N,S,W,E). All four external connections can either be directly connected or linked to one of the three transistor terminals (gate, source, drain). Consequently, a great variety of CMOS transistor circuits can be realized on the configurable transistor array and are represented by a corresponding configuration bit string. The cut-off frequency of the circuits on the FPTA is about 4 MHz. A detailed description of the FPTA is given in [10].

2.3.2 FPTA Simulation Model: The SimFPTA

The configurable cells of the SimFPTA feature identical configuration options as the original FPTA, described in the previous subsection, although there are important differences between hardware and SPICE simulation. Most important are the implications of the FPTA's architecture where each programmable transistor is in fact represented by an array of 4x5 plain transistors with different sizes that can be turned on and off, respectively. As a consequence of this, the effective W/L ratio of the resulting transistor is changed. Contrary to that, in the simulation model, a plain transistor with the according W/L is inserted into the circuit. Further, the connectivity is realized by using transmission gates on the FPTA whereas lossless lines could be used in case of the SimFPTA. In this case, the transmission gates have been replaced by resistors with the mean on-resistance of the transmission gates, which is about $R = 330\,\Omega$ [10]. Hereby, the trade-off is to create a model in SPICE which behaves comparable to the FPTA but is less complex, in order to save simulation time.

3. Experimental Setup

The experimental setup is independent of the used substrate. In all experiments, the mutation rate is adapted to the performance of the best individual and takes on values in the range of $p_{\text{mut}} = 0.02..0.1$, whereas the crossover rate is constantly set to $p_{\text{cross}} = 0.2$. A total of 30 evolution runs is carried out for both, the evolution of comparators and oscillators. In all cases, the task is to minimize the fitness.

3.1. Setup for the Comparators

Two test-modes (TMs) are used for the evolution of comparators. The input voltage pattern of the first test-mode consists of a set of seven curves, each a voltage sweep of $V_{\text{in}} = 0..5\,\text{V}$ in 100 samples at one of the switching points $V_{\text{set}} = 1, 1.5, 2, \ldots, 4\,\text{V}$. Hence, the output target voltage has to be $V_{\text{out}} = 0\,\text{V}$ if $V_{\text{in}} < V_{\text{set}}$ and $V_{\text{out}} = 5\,\text{V}$ if $V_{\text{in}} > V_{set}$. Owing to the symmetry of comparators, the circuit's inputs are exchanged for the second test-mode and consequently the output target voltage has to be $V_{\text{out}} = 0\,\text{V}$ if $V_{\text{in}} > V_{\text{set}}$ and $V_{\text{out}} = 5\,\text{V}$ if $V_{\text{in}} < V_{set}$. The GA is allowed to use an area of 8x8 programmable transistor cells and the population size is 50 individuals that are processed for 5000 generations on the FPTA and 2000 generations on the SimFPTA.

3.2. Fitness Calculation for the Comparators

Considering the presence of noise and fluctuations of the analog output, a discrete fitness function is used for the evolution of comparators. Since a precision of $\pm 10\,\text{mV}$ can be assumed for the hardware measurements, an offset of $20\,\text{mV}$ is taken into account in equation 3. Although not necessary for the output obtained from simulation (SimFPTA), the same fitness function (Eqns. 3,4) is used in order to make the results comparable. Fitness is separately calculated for each test-mode and simply summed up in case of single-objective evaluation.

In addition to the two test-modes, minimization of used resources is included in the fitness by adding extra penalties of $1\times$ no. of used routes $+ 2\times$ no. of used transistors. Since, in the phase of exploration, minimizing the resources would be counterproductive, the penalty for maximum ressource consumption is added until the fitness value drops below 500, which already stands for a reasonably good comparator. Thus, a total of 3 independent fitness values is used for the evolution of comparators.

$$\Delta V_i = |V_{\text{target}_i} - V_{\text{measured}_i}| - 20\,\text{mV} \quad (3)$$

$$\text{fitness} = \sum_{i=1}^{\#sam} \sum_{j=1}^{50} \begin{cases} j \cdot \frac{11}{1275} & \Delta V_i > 0.1 \cdot (j-1) \\ 0 & \Delta V_i \leq 0.1 \cdot (j-1) \end{cases} \quad (4)$$

3.3. Experimental Setup for the Oscillators

A truly multi-objective task is the evolution of an oscillator from scratch. One testmode, containing 480 samples with a sampling frequency of 10 MHz, is used to achieve this, hence, the range of the possible output frequencies is $20kHz$ kHz..4.8 MHz. Crucial for this setup is the fact, that there is no input and only one output present in the evolving circuit and further, the output is not bound to fixed constraints, in terms of a target voltage pattern. Merely the output voltage behavior instead of a fixed output voltage pattern is assessed by the fitness function. An area of 10×10 transistor cells is provided to the evolving circuit and the population size is 100 individuals which are evolved for 5000 generations.

3.4. Fitness Calculation for the Oscillators

The fitness measure does not constrain the frequency, phase and signal shape of the circuit's output to fixed values, since it is supposed that this would implicitly exclude usefull pathways for evolution towards oscillating circuits. Consequently, a total of 6 more open and phenomenological fitness criteria are used for the experiments and are listed in Tab. 1

objective	description
1. DC offset	$\lvert\overline{V}_{\text{out}} - 2.5\,\text{V}\rvert$
2. dev. from $\overline{V}_{\text{out}}$	maximize RMS error
3. amplitude span	maximize $V_{\text{out,max}} - V_{\text{out,min}}$
4. no. of zero crossings	#upwards + #downwards
5. inflection points	#not alternating +/-
6. std. dev of periods	std. dev of occurring periods
7. used ressources	sum of used transistors

Table 1. An list of all objectives that are used for the multi-objective evolution of oscillators from scratch. Zero crossings and amplitudes are always measured relative to the mean output voltage. Occurring periods are calculated from those zero crossings.

3.5. Selection Scheme

Tournament selection with a tournament size of 3 is used for all presented experiments. In case of the single-objective evolutions, the aggregated fitness value of all evaluations is considered, whereas in the multiobjective experiments the individual's level of non-domination is considered for selection. In case the scores of both competitors

are equal, the convergence of their fitness is taken into account, which is calculated from their previous fitness values using conv. $= \sum_{\text{objectives}} \frac{\text{fitness}(\text{gen}_i)}{\text{fitness}(\text{gen}_{i-5})}$.

4. Results

Three issues are targeted with the presented experiments: First, the development of an EA which produces circuits that are independent from the substrate on which they are evolved and which more likely delivers solutions that are reduced to relevant components. Second, the achievements of the multi-objective approach can be nicely seen from the results for the oscillators, where it is possible to harvest oscillating circuits of various frequencies from one single evolutionary run. Third, the performance of the FPTA is compared with the performance of the SimFPTA. In all experiments, except for the oscillators, the task is to evolve a comparator.

4.1. Improving Results on the FPTA by Developing the Evolutionary Algorithm

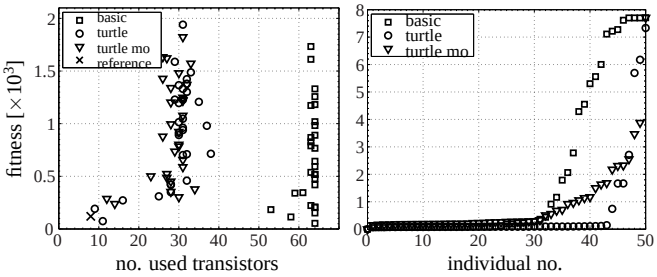

Figure 3. Left: Comparison of the fitness over the no. of used transistors for the three GAs: *Basic GA*, *Turtle GA*, *MO-Turtle GA*. The position of a manually made reference design within the fitness landscape is marked with a cross. Right: Distributions of the fitness values of example results off all three GAs.

The task is to evolve a comparator with three different evolutionary algorithms: The *Basic GA*, The *Turtle GA* and the *MO-Turtle GA*. As can be seen from Fig. 3, the results are spread over equal ranges of fitness, independent of the used EA. Opposite to that, the *Turtle GA* and the *MO-Turtle GA* are extensively resource preserving compared with the *Basic GA* and are performing equally well compared with each other. It is assumed that the *MO-Turtle GA* will outperform the *Turtle GA*, with increasing number of objectives. Examples for the latter are given in Sec. 4.2 and in [15]. As can be seen from Fig. 3, the MO approach achieves the most

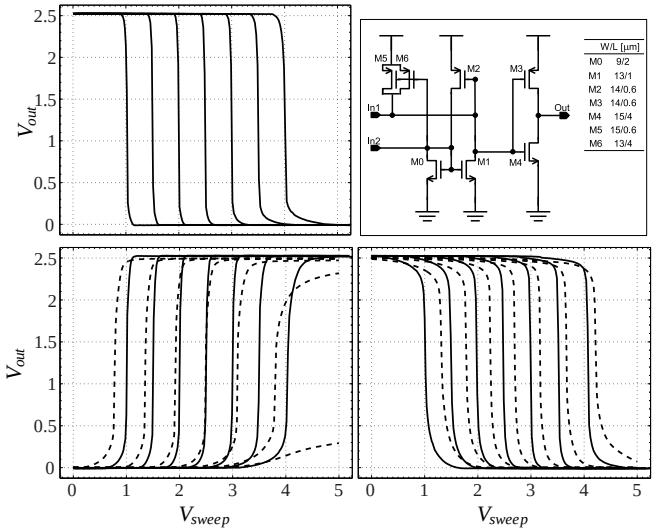

Figure 4. Comparison of voltage characteristics of the best solutions (straight line) with the simulation results (dashed line). Depicted are example measuring for the Basic GA (upper left), the Turtle GA (lower right) and the MO-Turtle GA (lower left). Example for an extracted schematic (upper right).

uniform distribution of fitness, hence, the greatest diversity within the resulting populations.

Example voltage characteristics of solutions with a good (best) performance are shown in Fig. 4. In the case of the *Turtle GA*s, the circuits are extracted into netlists and the output obtained from the FPTA is compared with the results from simulation. For the simulation, the resistance of the transmission gates is approximated with a mean value of 330 Ω and this is the reason for the observed offsets of V_{set}. In cases where the parasitic effects are negligible, netlists with plain transistors are extracted from the resulting individual, thus, schematics for further analysis of the solutions can be drawn. An example schematic of the solution with the lowest resource consumption is shown in Fig. 4, although in this case, the resistors have to be considered for successful simulation.

4.2. A Truly Multi-Objective Result: Oscillators from Scratch

A truly multi-objective result is obtained from the evolution of oscillators from scratch, which has not been achieved yet with any single-objective approach on the FPTA. Additionally, the resulting populations feature numerous oscillator circuits with different frequencies, ranging from 20 kHz to 4.8 MHz. The non-dominated front of a successful evolu-

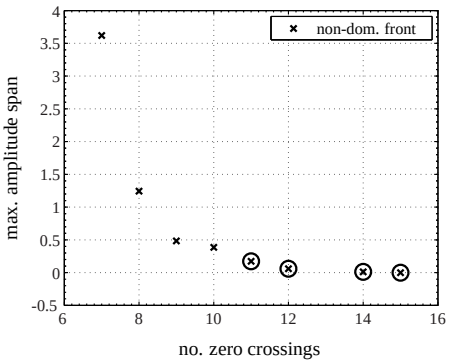

Figure 5. The NDF of a successful evolutionary run is recalculated considering 2 objectives out of 7, namely the *no. of zero crossings* and the *max. amplitude span*

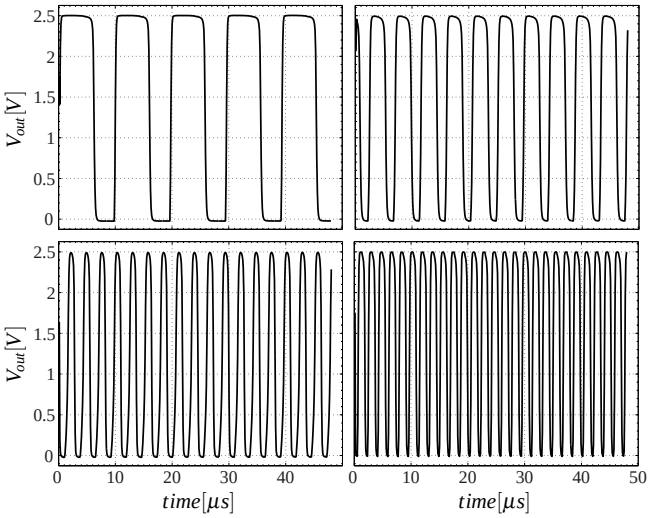

Figure 6. The voltage output of 4 example circuits, obtained from one single evolutionary run and featuring 4 different frequencies (100 kHz**,** 200 kHz**,** 350 kHz **and** 450 kHz**), are depicted above.**

tionary run, which is recalculated considering 2 objectives out of 7, namely the *no. of zero crossings* and the *max. amplitude span*, is graphed in Fig. 5. The output voltage characteristics of the individuals which are marked with a circle in Fig. 5, are graphed in Fig. 6. Each of the 4 resulting circuits oscillates with a different frequency, namely 100 kHz, 200 kHz, 350 kHz and 450 kHz.

4.3. Comparison of Results from the FPTA and the SimFPTA

The evolvability of the FPTA and the SimFPTA is compared by tackling the synthesis of comparators. As can be nicely seen from Fig. 7, the evolutionary runs end up in equal ranges of fitness values and resource consumption for both substrates. Consequently, the SimFPTA is a suitable model for evaluating the evolvability of the real hardware, although the behavior is different in the beginning of evolution, due to the fact that individuals which do not work at all in simulation, nevertheless produce an output on the chip. The latter effect becomes less important, if, as presented here, a GA like the *Turtle GA* is used for the evolution experiments, since a valid circuit is crucial for carrying out a successful simulation. Output voltage characteristics for both test-modes of an individual with a good performance is graphed in Fig. 8.

5. Conclusion and Remarks for Future Experiments

The presented experiments show the progress that is made towards a comprehensive modular evolution framework, which significantly facilitates the development, implementation and immediate application of any module of an evolutionary algorithm. Further, it is possible to support and use different hardware substrates as well as different simulation models for evaluating various custom architectures. In this case, the results obtained from the FPTA and the SimFPTA—a simplified model of the real hardware—show equal performance for the task of synthesizing comparators. Hereby, it is observed that, generally, all circuits that are evolved on the SimFPTA perform similar on the FPTA, however, since a simplified simulation model cannot cope with any parasitic effect of the chip, the inverse is not necessarily true. Regardless of the supported view—avoiding or extensively exploiting parasitic effects—the aim should be to understand or even control the influence of those effects. The *MO-Turtle GA* succeeded in evolving oscillators, comparators and amplifiers [15], therefore, a multi-objective strategy is suggested for the synthesis of complex analog circuits. If the aim is to synthesize circuits that are—mostly in the case of real hardware—not bound to one single substrate, the variation operators have to be designed to produce transferable circuits, as it is realized in the *Turtle GA*. Since evolution on real hardware is significantly faster than in simulation, it is on the one hand an advantage to use a real chip in order to quickly evaluate the performance of the used algorithm. On the other hand, the architecture of the chip cannot be changed unless a new version is designed and fabricated. Thus, a future step for this work is the improvement of the FPTA architecture, based on the experience with the current chip. Once a

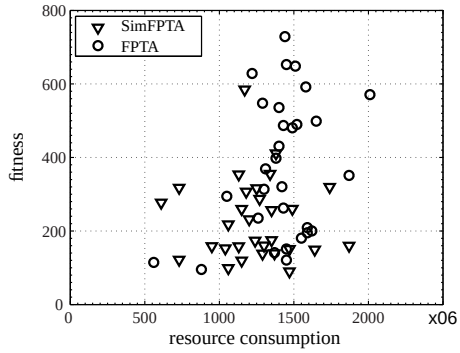

Figure 7. The fitness over resource consumption for 30 evolutionary runs on the FPTA and the SimFPTA, respectively, is graphed above.

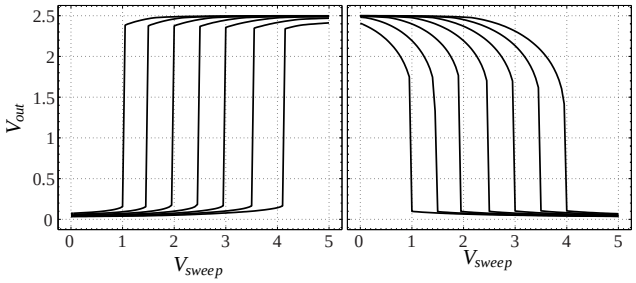

Figure 8. Output voltage characteristics of a good solution obtained from the SimFPTA.

software model of an architecture with a good performance is found, it will be possible to realize a more powerful hardware implementation.

References

[1] V. Aggarwal. Evolving Sinusoidal Oscillators Using Genetic Algorithms. In *5th NASA / DoD Workshop on Evolvable Hardware (EH 2003)*, pages 67–76, Chicago, IL, USA, 9-11 July 2003. IEEE Computer Society.

[2] T. T. Arpad Buermen, Janez Puhan. Robust Design and Optimization of Operating Amplifiers. pages 745–750, December 2003.

[3] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.

[4] K. Deb and T. Goel. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 67–81. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[6] M. Hershenson, S. Boyd, and T. H. Lee. Optimal design of a CMOS op-amp via geometric programming. In *IEEE Transactions on Computer-Aided Design*, pages 1–21, 2001.

[7] L. Huelsbergen, E. A. Rietman, and R. Slous. Evolving oscillators in silico. volume 3, pages 197–204, 1999.

[8] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann Publishers, 1999.

[9] W. Kruiskamp and D. Leenaerts. DARWIN: CMOS opamp Synthesis by means of a Genetic Algorithm. In *DAC '95: Proceedings of the 32nd ACM/IEEE Conference on Design Automation*, pages 433–438, New York, NY, USA, 1995. ACM Press.

[10] J. Langeheine. *Intrinsic Hardware Evolution on the Transistor Level*. PhD thesis, Rupertus Carola University of Heidelberg, Seminarstrasse 2, 69120 Heidelberg, July 2005.

[11] L. Sekanina and R. S. Zebulum. Intrinsic Evolution of Controllable Oscillators in FPTA-2. In J. M. Moreno, J. Madrenas, and J. Cosp, editors, *Evolvable Systems: From Biology to Hardware, Sixth International Conference, ICES 2005*, number 3637 in LNCS, pages 98–107, Sitges, Spain, September 2005. Springer-Verlag.

[12] H. Shibata. *Computer-Aided Design of Analog Circuits Based on Genetic Algorithm*. PhD thesis, Tokyo Institute of Technology, 2001.

[13] T. Sripramong and C. Toumazou. The Invention of CMOS Amplifiers Using Genetic Programming and Current-Flow Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(11):1237–1252, November 2002.

[14] M. Trefzer, J. Langeheine, J. Schemmel, and K. Meier. New Genetic Operators to Facilitate Understanding of Evolved Transistor Circuits. In R. S. Zebulum, D. Gwaltney, G. Hornby, D. Keymeulen, J. Lohn, and A. Stoica, editors, *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 217–224. IEEE Computer Society Press, 2004.

[15] M. Trefzer, J. Langeheine, J. Schemmel, and K. Meier. Operational Amplifiers: An Example for Multi-Objective Optimization on an Analog Evolvable Hardware Platform. In J. M. Moreno, J. Madrenas, and J. Cosp, editors, *Evolvable Systems: From Biology to Hardware, Sixth International Conference, ICES 2005*, number 3637 in LNCS, pages 86–97, Sitges, Spain, September 2005. Springer-Verlag.

[16] P. F. Vieira, L. B. Botelho, and A. Mesquita. Evolutionary Synthesis of Analog Circuits Using Only MOS Transistors. In Zebulum, Ricardo S., Gwaltney, David, Hornby, Gregory, Keymeulen, Didier Lohn, Jason, and Stoica, Adrian, editor, *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 38–45, Los Alamitos, 2004. IEEE Computer Society Press.

[17] M. Wall. *C++ Genetic Algorithm Library, GALib 2.4.6*. Massachusetts Institute of Technology, MIT, 1999.