2003 AAAI Spring Symposium on Computational Synthesis held March 24-26, 2003 at Stanford University in Palo Alto, California.

Scalability Issues in Evolutionary Synthesis of Electronic Circuits: Lessons Learned and Challenges Ahead

Adrian Stoica
Ricardo S. Zebulum
Didier Keymeulen
M. I. Ferguson
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109
adrian.stoica@jpl.nasa.gov

Xin Guo
Chromatech, Alameda CA 94501

Abstract

This paper describes scalability issues of Evolutionary-driven automatic synthesis of electronic circuits. The article begins by reviewing the concepts of circuit evolution and discussing the limitations of this technique when trying to achieve more complex systems. The paper continues by describing techniques developed by the authors to partially overcome the limitations of evolution, such as the use of domain knowledge, design re-use and the development of hardware accelerators, such as stand-alone board-level evolvable system (SABLES), which was built to speed up the evolutionary design of electronic circuits. We also propose new directions of research that address scalability, such as: 1) evolutionary compilation of descriptions from behavioral Hardware Description languages (HDL) to structural HDL (for both the case of digital and analog/mixed signal) 2) evolutionary synthesis, converting from synthesizable analog HDL to circuits and 3) hardware-software partitioning (co-design) for CPU/FPGA hybrids.

1. Introduction

Evolution-driven automatic synthesis of electronic circuits has been demonstrated for simple circuits, yet its efficiency in obtaining complex electronic circuits operating under real-world conditions is still to be proven. Complex circuits are often associated with large circuits. Even a simply formulated requirement on which a circuit can easily be evaluated (e.g. a 12 bit analog to digital converter) may require a complex circuit solution. Complexity may also be in the requirements (often hard to formally specify, and which can translate in complicated evaluation tests for each individual) even if the resulting solution circuit is quite simple/compact. In certain cases it may be useful to observe the distinction between the two aspects of complexity and treat them differently.

Evolutionary synthesis has been mainly a bottom-up approach. The components used in most demonstrations of evolutionary circuit synthesis were primitive elements, for example device-level (transistor, capacitor, resistor) for analog circuits, or gate level for digital circuits. For any complex system the number of components used may be relatively large, leading to many ways of interconnecting them, and consequently to a very large and irregular search space.

In addition to the challenge of sampling a large search space, there is also the problem that evaluations in simulations may take long, and what is worse, scale badly with increasing complexity of the design. For example, in automated analog circuit design it is known that the SPICE analysis/simulation scales badly with the number of nodes of the circuit. Thus it appears natural to try to reduce the number of evaluations and to speed-up simulations/evaluations.

This paper is organized as follows. Section 2 reviews the concepts of evolution-driven automatic circuit synthesis. Section 3 describes techniques to address the scalability problem, including the use of domain knowledge, building block encapsulation, Mixed Model Search and the development of a hardware accelerator in the form of a stand-alone board-level evolvable system (SABLES). Section 4 proposes new techniques to address this problem, such as using structural descriptions when evolving from higher level specifications. Section 5 concludes the paper.

2. The Evolutionary Search

The evolutionary/genetic search is tightly coupled with a coded representation that associates each circuit to a "genetic code" or chromosome. The simplest representation of a chromosome is a binary string, a succession of 0s and 1s that encode a circuit. The status of the switches (ON or OFF) determines a circuit topology and consequently a

Copyright © 2003, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

specific response. Thus, the topology can be considered as a function of switch states, and can be represented by a binary sequence, such as "1011...", where a '1' is associated to a switch turned ON and a '0' to a switch turned OFF.

First, a population of chromosomes is randomly generated. The chromosomes are converted into circuit models for evaluation in SW (extrinsic evolution) or into control bitstrings downloaded to programmable hardware (intrinsic evolution). Circuit responses are compared against specifications, and individuals are ranked based on how close they come to satisfying them. In preparation for a new iteration, a new population of individuals is generated from the pool of best individuals in the previous generation. This is subject to a probabilistic selection of individuals from a best individuals pool, followed by two operations: random swapping of parts of their chromosomes, the *crossover* operation, and random flipping of chromosome bits, the *mutation* operation. The process is repeated for several generations, resulting in increasingly better individuals. Randomness helps to avoid getting trapped in local optima. Monotonic convergence (in a loose Pareto sense) can be forced by unaltered transference to the next generation of the best individual from the previous generation. There is no theoretical guarantee that the global optimum will be reached in a useful amount of time; however, the evolutionary/genetic search is considered by many to be the best choice for very large, highly unknown search spaces. The search process is usually stopped after a number of generations, or when closeness to the target response has reached a sufficient degree. One or several solutions may be found among the individuals of the last generation.

3. Techniques for Scalability

3.1 Use of Domain Knowledge

One possible way to reduce the search space (possibly eliminating large regions of the search space) is through the use of domain-knowledge. The incorporation of domain-knowledge is sometimes implicit in the choice of representations, restriction or bias of possible configurations. For example, in one experiment it was difficult to use evolved logic gate circuits as building blocks for more complex designs, because the evolved gate was not able to drive similar gates. This was addressed by restricting inputs only to connect to transistor gates and not to drain, source or substrate; as a consequence higher input impedance solutions and better loading characteristics were implicitly obtained, and the building-block became reusable.

In the current state of our research this technique was only applied to enforce that evolved building blocks could be cascaded to build complex circuits. However, it is not yet proven that the enforcement of human design guidelines might aid the Evolutionary Algorithm to synthesize complex circuits only from primitive components, i.e., without cascading simple evolved circuit building blocks.

Evolutionary Algorithms can also rediscover classic human designs, as shown in Koza et al. (1999). They have more chances to infringe on existing patented circuits if the representation is constrained by allowing only usual connections among components. This is usually the case of developmental approaches, in which user guidelines to build the circuit from the chromosome are used.

3.2 Building Block Encapsulation

It appears natural to say that the search space can be reduced if one keeps an acceptable scope/focus, increasing level of abstraction e.g. through the use of increasingly higher level building blocks. The method proposed is based upon encapsulation and design re-use and one example is the evolution of a 4-bit Digital to Analog Converter (DAC). It has been observed that evolution had difficulty in solving this task using low-level components (such as transistors, resistors and capacitors) as building blocks. For instance, the literature reported that a total of 45,000,000 circuits had to be evaluated to achieve the solution for the 3-bit DAC problem using Genetic Programming (Bennett et al. 1999). The objective of our experiment was therefore to overcome this limitation and synthesize a 4-bit DAC *hierarchically*. At first, a 2-bit DAC was evolved from scratch, e.g., using only MOS transistors as components for evolution. This circuit was then employed as a building block for evolution to synthesize a 3-bit DAC, which was subsequently used as building block for the synthesis of a 4-bit DAC. Figure 1 depicts the evolved circuit schematic and Figure 2 shows its response. In this particular experiment, the design re-use was not only limited to previously evolved DAC circuits (building blocks), but also employed other well-known building blocks, such as current mirrors.

The experiment took less than one minute in a SPARC Ultra 2 Sun workstation evaluating about 200,000 individuals. The dramatic reduction in time compared to other experiments is due to the fact that the DC operating point of encapsulated building blocks had already been defined, and their behavior was simulated using a high-level description language. For further details on the experiment refer to Zebulum, Stoica and Keymeulen (2001).

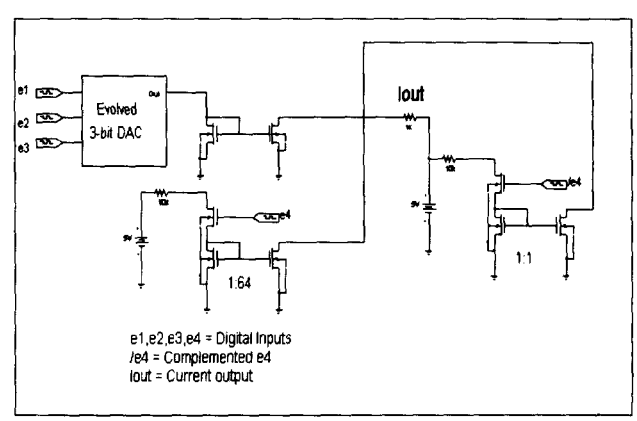

Figure 1: Hierarchically evolved 4-bit DAC.

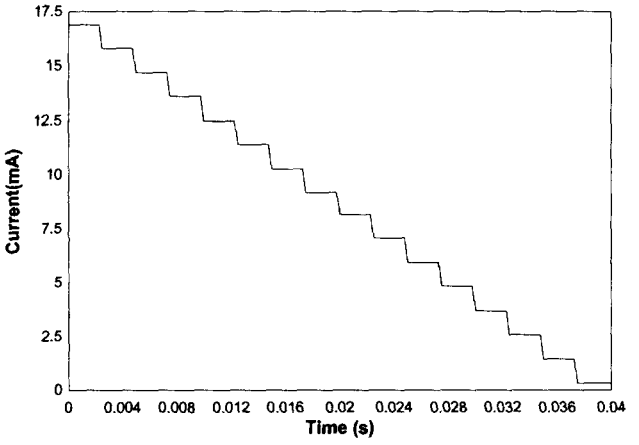

Figure 2 – Response of the of the circuit shown in Figure 1.

Building block encapsulation is a promising path to the evolutionary synthesis of complex circuits. In the context of the experiment described in this section, however, more investigation is needed to refine the representation. Although it seems natural to employ simpler DACs to evolve higher resolution ones, the circuit size gets very large comparing to classic human design. Even though it is straightforward to evolve a *n-bit DAC* using *(n-1)-bit DACs* as building blocks (leading to large circuits), it is still a difficult task for evolution to synthesize a *n-bit DAC* using *(n-2)-bit DACs* or simpler building blocks.

Instead of using evolution as a tool for automatic composition of useful modules, these building blocks can also be selected using domain knowledge. One example is the use of Operational Amplifiers (OpAmps) in the evolution of analog circuits. OpAmps are standard building blocks for analog processing systems, being therefore natural candidates for the evolution of a regular analog design. The use of OpAmps as higher level building blocks enabled the hardware evolution of filters and signal separators, as discussed in section 3.4.

3.3 Mixed Model Search

Increasing the level of abstraction by using simpler models also speeds up simulations. Once a building block has been "encapsulated", one can for example replace back-to-back transistors that form a switch with resistors, leading to circuits that simulate faster. From this point of view developing coherent equivalent models may be an efficient approach. Building coherent equivalent models of various levels of abstraction is made possible using the JPL-introduced concept of mixtrinsic evolution (Stoica, Zebulum and Keymeulen 2000). In mixtrinsic evolution models of various nature, or models of same nature but of different levels of resolution are used part of the search population.

A set of candidate circuits C1 through CN is defined by a set of "chromosomes" that are fed to a high-resolution model to produce N high-resolution models M1 through MN. A simulator simulates the physical behavior of each of the models M1 through MN in response to a predetermined stimulus. The responses of each model are compared to a desired response and a fitness function is produced for each model, namely the fitness functions F1 through FN. A standard search process determines the next search points based on this information.

In the proposed Mixed Model Search each one of the candidate circuit C1 through CN is modeled by both a high-resolution model and by one (or more) low-resolution models. Thus, for the N candidate circuits C1 through CN, there are N pairs of models M1, m1 through MN, mn. There are N high-resolution models M and N low-resolution models m. The pair of high/low resolution models (e.g., M2 and m2) (family model) representing a particular candidate circuit (e.g., C2) produces a pair of fitness functions (e.g., F2 and f2). A combiner combines each pair of fitness functions to produce a combined score for the corresponding candidate circuit. For example, the combiner may compute the average of the two fitness functions as the combined fitness function or score. The combined score for each candidate circuit is provided to a search process that controls the simulator. The average may be a weighted average in which, for example, the fitness function of a higher resolution model is given more (or less) weight than that of a lower resolution model. Alternatively, the average may be unweighted.

A computational savings may be realized by employing only one model for each candidate circuit during any single iteration of the evolution process. With each iteration of the evolution process, a different resolution level model is assigned to each (or at least many) of the candidate circuits. As a result, after a number of iterations, each candidate circuit has been modeled with all levels of resolution. Such assignments may be carried out in a random fashion. For example a model could be considered for each candidate circuit, different candidate circuits being modeled with a model of a different resolution level. The first two candidate circuits C1 and C2 could be modeled with a high-resolution model (M1, M2 respectively) while the third candidate circuit C3 is modeled with a low-resolution model (m3). The simulator produces data that would lead to a fitness function from each model (F1, F2, f3, etc.) which is provided to the search decision mechanism.

3.4 SABLES

As previously stated, one drawback of sampling large spaces for circuit evolution is the simulation time when the circuits are evaluated in SW (extrinsic evolution). An alternative way to speed up evaluation is the use of reconfigurable devices, possibly as emulators, accelerating the evaluation. A stand alone board level Evolvable System (SABLES), developed for autonomous portable experiments integrates a Field Programmable Transistor Array (FPTA-2) and a DSP implementing the Evolutionary Platform (EP). The system is connected to the PC only for the purpose of receiving specifications and communicating back the results of evolution for analysis.

JPL has developed the concept and has experimented with a new generation of reconfigurable devices, called Field Programmable Transistor Arrays (FPTA) (Stoica et al. 2001). The lack of evolution-oriented devices, in particular for analog, has been an important stumbling block for researchers attempting evolution in intrinsic mode (with evaluation directly in hardware). The FPTA has transistor level reconfigurability, supports any arrangement of programming bits without danger of damage to the chip (as is the case with some commercial devices). Three generations of FPTA chips have been built and used in evolutionary experiments. The latest chip, FPTA-2, consists of an 8x8 array of reconfigurable cells. Each cell has a transistor array as well as a set of other programmable resources, including programmable resistors and static capacitors. The FPTA-2 cell consists of 14 transistors connected through 44 switches and it is able to map different building blocks for analog processing, such as two and three stages OpAmps, logarithmic photo-detectors, or Gaussian computational circuits. Figure 3 shows the details of the FPTA cell for the first and for the latest version of the FPTA chip.

The evolutionary algorithm was implemented in a DSP that directly controlled the FPTA, together forming a board-level evolvable system with fast internal communication ensured by a 32-bit bus operating at 7.5MHz. Details of the EP were presented in (Ferguson et al. 2002). Over four orders of magnitude speed-up of evolution was obtained on the FPTA chip compared to SPICE simulations on a Pentium processor (this performance figure was obtained for a circuit with approximately 100 transistors; the speed-up advantage increases with the size of the circuit). The evaluation time depends on the tests performed on the circuit. Many of the evaluation tests performed required less than two milliseconds per individual, which for example on a population of 100 individuals running for 200 generations required only 20 seconds. The bottleneck is now related to the complexity of the circuit and its intrinsic response time. SABLES fits in a box 8" x 8" x 3".

The following experiment illustrates an evolution on SABLES (Stoica et al. 2002). The objective of this experiment is to synthesize a half-wave rectifier circuit. The testing of candidate circuits is made for an excitation input of 2kHz sine wave of amplitude 2V. A computed rectified waveform of this signal is considered as the target. The fitness function rewards those individuals exhibiting behavior closer to target (using a simple sum of differences between the response of a circuit and target) and penalizes those farther from it. After evaluation of 100 individuals, they are sorted according to fitness and a 9% portion (elite percentage) is set aside, the remaining individuals undergoing first crossover (70% rate), either among themselves or with an individual from elite, and then mutation (4% rate). The entire population is then reevaluated. In this experiment only two cells of the FPTA were allocated.

Figure 4 displays snapshots of evolution in progress, illustrating the response of the best individual in the population over a set of generations. The first caption shows the best individual of the initial population, while the subsequent ones show the best after 5, 50 and 82 generations.

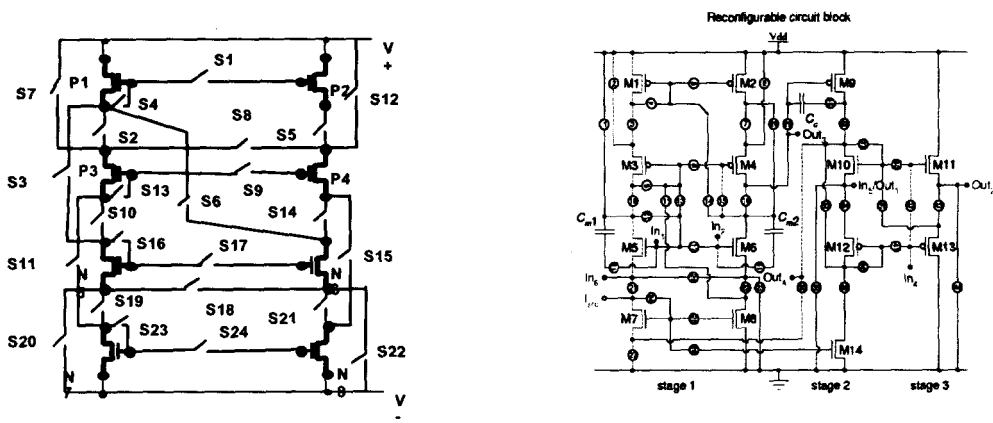

Figure 3: FPTA cell topology as used in two FPTA chips (FPTA-0 in the left and FPTA-2 in the right.

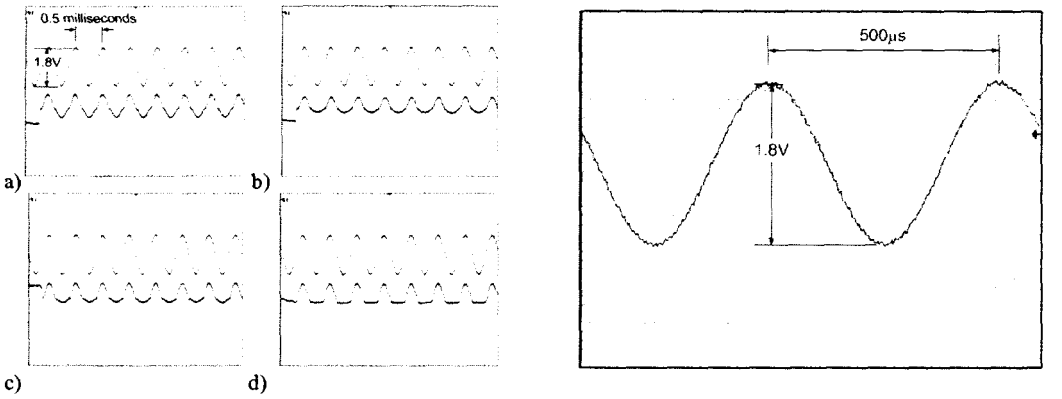

Figure 4. Evolution of a halfwave rectifier showing the response of the best individual of generation a) 1, b) 5, c) 50 and finally the solution at generation d) 82. The final solution is illustrated on the right.

With SABLES enabling rapid evolvable hardware experiments, the focus has shifted from the hardware platform to algorithms. More specifically the focus became overcoming problems related to the formulation of requirements in a way that facilitate evolutions, and the translation of target specifications into the language of evolution, including representations, fitness function and parameters of the algorithm.

Other experiments were performed on SABLES, such as the evolution of oscillators and signal separators (Zebulum et al. 2003). SABLES has been a successful platform for quick (less than 5 minutes) on-line automatic synthesis of simple circuits building blocks, such as rectifiers, filters, oscillators and logic gates. However, we were not able to synthesize complex circuits using this system yet. Although we have a powerful tool for evolution, we still have to find how to use it so as to explore its full capacity. In order to keep the search space manageable, we have restricted most of our experiments to use only a small fraction of the chip, usually two cells out of 64. The search space to be sampled when using all cells in the chip is about 2^{5000} (5,000 configuring bits) or 10^{1500}. Assuming that we can evaluate 10^4 circuits per second and leave an experiment running for one month, we are only able to sample 10^{10} individuals, a negligible fraction of the search space. One approach to circumvent this problem is to constrain the cells to behave as OpAmps as discussed earlier and evolve only the cells' connectivity, thereby reducing the search space. This method was tried with partial success in the evolution of signal separators using 10 cells of the chip (Zebulum et al. 2003).

4. Future Approaches and Challenges

4.1. Evolving from high-level specifications

- From behavioral HDL to circuit through synthesizable HDL.

One way to approach the scalability problem is to first admit that what we address is an open problem for both analog *and* digital (and for system design in general). The perception in the evolvable hardware community exists that the digital automated design/synthesis problem is solved by current techniques and tools. What is missed is that only *structural* VHDL (Verilog) is synthesizable, while *behavioral* VHDL is not. The reason is simple: structural VHDL offers a problem decomposition! Thus the tools only have to deal with implementation of a simpler block, and also the set of library elements offers easy/direct matches. (The boundary between behavioral and structural depends on the vendor supported language/extension and size of IP library, etc). However, evolutionary design is behavior-oriented and poorly-specified in form of response curves instead of using standard language such as HDL. In this context the following two research directions appear promising:
a) *Evolutionary-based compilation of behavioral HDL to structural HDL* (analog or/and digital). Force specifications to be in a standard behavioral language.
b) *Evolutionary-based synthesis of structural AHDL.* Structural AHDL may be the required first step to automatic analog synthesis. The building blocks may be sufficiently small to allow evolution to find optimal solution.
We believe that by decomposing the problem first into a *functional to structural* translation and then a *structural to primitives* one, the chances of evolution to approach complex system are much improved.

-Hardware/software evolutionary co-design for FPGA/CPU hybrids and other Systems-on- a-Chip
This direction is especially timely in the context of the embedded systems industry rapidly moving toward integration of programmable and reconfigurable devices, with a powerful convergence toward hybrid FPGA/CPU architectures. Ultimately these will use flavors of on-chip hybrids such as the new Xilinx Virtex II Pro chip. There are no tools allowing designers to go from system level

specification, e.g., in Matlab/Simulink and convert it to an efficient hardware/software allocation/partitioning.

4.2 Complexity in Functional Descriptions

A remaining largely unaddressed challenge is related to increased complexity in functional descriptions. Even in the case of the simple halfwave rectifier experiment previously described, providing a complete set of specifications is not so obvious. The operational range of the evolved rectifier in the frequency domain is one potential pitfall, since in principle the circuit behavior should be evaluated for the overall frequency domain in which it is expected to work. In that particular case, the circuits were only tested at 2kHz, and it has been verified that the solutions only worked in the decade 500Hz-5kHz.

The need for upfront complete specifications is also reflected in the evolution of a logic gate. An example described in Stoica et al. (2002) evaluated a circuit targeted as NAND gate providing input stimulus (using a SPICE transient analysis) with changes in the microsecond range. The correct behavior for this timescale was quickly achieved by evolution. However, an incorrect behavior was observed when the same circuit was simulated in the timescale of seconds. Conversely, when the circuit is evaluated at a large timescale evolution often led to slow gates. The method applied to correct this situation was a derivative of the mixed model search described previously in this paper. In this case the different models are subject to different analysis. A two-transient analysis for each candidate circuit was performed during evolution, the first on a small timescale and the second on a larger timescale, solving the problem. For each circuit, the combined fitness measure was chosen to be the worse between the two evaluations, so that the genetic algorithm was driven to achieve a correct behavior at both timescales. However, in order to speed up evolution, we can assign the candidate solutions to one model or another during successive generations and thus letting evolution remove solutions that do not behave well on both models.

5. Conclusions

New approaches are needed in order to reach the full potential of evolvable systems. Scalability and completeness of specifications are primordial. In order to sample more efficiently the large search space generated in circuit design problems, techniques such as use of domain knowledge and building blocks encapsulation are currently being used. These techniques still have to be refined to allow the evolution of complex circuits that are competitive with human design. Another important issue is to find the right balance between reducing the search space while keeping the power of Evolutionary Algorithms to find novel solutions.

It is proposed to approach these problems through hardware description languages, formulating the requirements/ specifications in HDL. It is also proposed to use intermediate, structural level representation, and thus employ evolution in two phases: behavioral to structural, and structural to circuits. Evolution proved efficient in the design of circuits satisfying multiple constraints.

Acknowledgements

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the National Aeronautics and Space Administration (NASA).

References

Koza, J. et al. 1999. Genetic Programming III: Darwinian Invention and Problem Solving. San Francisco, CA: Morgan Kaufmann.

Bennett, F. et al. 1999. Evolution by means of genetic programming of analog circuits that perform digital functions. In Banzhaf, Wolfgang, Daida, Jason, Eiben, A. E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E. (editors). 1999. GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida USA. San Francisco, CA: Morgan Kaufmann: 1477 - 1483.

Zebulum, R. S., Stoica, A., and Keymeulen, D. 2001 Experiments on the Evolution of Digital to Analog Converters. Published in the Proceedings of the 2001 IEEE Aerospace conference, March, 2001, Montana. ISBN: 0-7803-6600-X (published in CD).

Stoica, A, Zebulum, R. S. and Keymeulen, 2000. Mixtrinsic Evolution. In T. Fogarty, J. Miller, A. Thompson and P. Thompson, (eds.), In Proc. of the Third International Conference on Evolvable Systems (ICES2000). April, 2000, Edinburgh, UK. New York, USA, Springer Verlag: 208-217.

Stoica, A. et al. 2001. Reconfigurable VLSI Architectures for Evolvable Hardware: from Experimental Field Programmable Transistor Arrays to Evolution-Oriented Chips. *IEEE Transactions on VLSI*, IEEE Press, Volume 9, Number 1, ISSN 1063-8210: 227-232.

M.I. Ferguson et al. 2002. An Evolvable Hardware Platform based on DSP and FPTA. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2002), USA. Memlo Park, CA: AAAI Press: pp145-152.

Stoica, A. et al. 2002. Evolving Circuits in Seconds: Experiments with a Stand-Alone Board Level Evolvable System. In Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware, Alexandria, VA, IEEE Computer: 67-74.

Zebulum, R.S. et al. 2003. Automatic Evolution of Signal Separators using Reconfigurable Hardware. In Proceedings of the 2003 International Conference on Evolvable Systems (ICES) to be held in Norway, Forthcoming.