Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digital CMOS circuits

Adrian Stoica, Didier Keymeulen[1], Raoul Tawel, Carlos Salazar-Lazaro, and Wei-te Li
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109
adrian.stoica@jpl.nasa.gov

Abstract

The paper describes the architectural details of a fine-grained Programmable Transistor Array (PTA) architecture and illustrates its use in evolutionary experiments on the synthesis of both analog and digital circuits. A PTA chip was built in CMOS to allow circuits obtained through evolutionary design using a simulated PTA to be immediately deployed and validated in hardware and, moreover, enables a benchmarking and comparison of evolutions carried out via simulations only (extrinsic evolution) with the chip-in-the-loop (intrinsic) evolutions. The evolution of an analog computational circuit and a logical inverter are presented. Synthesis by software evolution found several potential solutions satisfying the a-priory constraints; however, only a fraction of these proved valid when ported to the hardware. The circuits evolved directly in hardware proved stable when ported to different chips. In either case, both software and hardware experiments indicate that evolution can be accelerated when gray-scale (as opposed to binary switches) were used to define circuit connectivity. Overall, only evolution directly in hardware appears to guarantee a valid solution.

1 Introduction

Evolvable Hardware (EHW) is reconfigurable hardware whose configuration is under the control of an evolutionary algorithm. The search for an electronic circuit realization of a desired transfer characteristic can be made in software as in *extrinsic* evolution, or in hardware as in *intrinsic* evolution. In extrinsic evolution the final solution is downloaded to (or become a blueprint for) the hardware. In *intrinsic* evolution the hardware actively participates in the circuit evolutionary process.

In the context of electronic synthesis on reconfigurable devices, the architectural configurations are encoded in "chromosomes" that define the state of the switches connecting elements in the reconfigurable hardware. The main steps in evolutionary synthesis of electronic circuits are illustrated in Figure 1. First, a population of chromosomes is randomly generated to represent a pool of circuit architectures. The chromosomes are converted into circuit models (for extrinsic EHW) or control bitstrings downloaded to programmable hardware (intrinsic EHW). Circuit responses are compared against specifications of a target response and individuals are ranked based on how close they come to satisfying it. Preparation for a new iteration loop involves generation of a new population of individuals from the pool of the best individuals in the previous generation. Here, some individuals are taken as they were and some are modified by genetic operators, such as chromosome crossover and mutation. The process is repeated for a number of generations, resulting in increasingly better individuals. The process is usually ended after a given number of generations, or when the closeness to the target response has been reached. In practice, one or several solutions may be found among the individuals of the last generation.

A variety of circuits have been synthesized through evolutionary means. For example, Koza used Genetic Programming (GP) to grow an "embryonic" circuit to one that satisfies desired requirements [1]. This approach was used for evolving a variety of circuits, including filters and computational circuits. An alternative encoding technique for analog circuit synthesis, which has the advantage of reduced computational load was used in [2] for automated filter design. On-chip evolution was demonstrated by Thompson [3] using an FPGA as the programmable device, and a Genetic Algorithm (GA) as the evolutionary mechanism. More details on current work in evolvable hardware are found in [4], [5], [6], and [7].

[1] Also member of the Electrotechnical Laboratory, Tsukuba, Japan.

Intrinsic evolution can speed-up the search for a solution-circuit by several orders of magnitude compared to evolution in software. This is especially true if one simulates large, complex analog circuits (and if the circuit response is rapid, i.e. not for circuits with time constants of seconds). Moreover, since the software simulation relies on limited accuracy models of physical hardware a solution evolved in software may behave differently when downloaded in programmable hardware; such mismatches are avoided when evolution takes place directly in hardware. Hardware evolution also scales well with both size of the circuits and model accuracy. The more complex the circuits, and more accurate their models, the longer the time it takes for their evaluation in software; on the other hand the time is approximately the same in hardware evaluations.

Evolutions of analog circuits reported in [1] and [2] were performed in simulations, without concern for a physical implementation, but rather as a proof-of-concept to show that evolution can lead to designs that compete, or even exceed in performance those of humans. No analog programmable devices exist that would support the implementation of the resulting design (but, in principle, one can test their validity in circuits built from discrete components, or in an ASIC), and thus intrinsic evolution was not possible. The problem of intrinsic evolution on programmable analog devices is very interesting. One reason is that the potential of analog processing is much greater than what is exploited today. Analog circuitry has advantages in cost, size and power consumption (compared to digital), and can directly process signals that are continual in time and amplitude, albeit at a finite resolution. Even a single transistor has many functions that can be exploited in computation functions such as generation of square, square-root, exponential and logarithmic functions. Other functions include voltage-controlled current sources, analog multiplication of voltages, and short term and long term analog storage [8]. The basic combinations of transistors offer a rich repertoire of linear and nonlinear operators available for local and collective analog processing. Using evolution, the benefits of analog processing can be exploited, while its disadvantages are reduced/eliminated.

Interest in analog computing dwindled with the emergence of robust and precise digital computing techniques. Analog computers were not easily programmable, were prone to temperature induced drifts, and required precise matching of components to perform quasi-accurate computations. If the evolutionary mechanism proves sufficiently powerful for evolving complex analog circuits, then its combination with reconfigurable analog devices will be able to capture the benefits of analog in new applications. A support of research in this area is also the current belief that the evolutionary search works better with analog than with digital circuits. The possible explanation lies in the fact that analog behaviors are relatively smoother search spaces. Thus, new perspectives are possible: evolutionary searches offer automatic programming; sufficiently precise equivalent components could be obtained if the programmable analog components offer controllability of their operating points; drifts that can be compensated for by adjusting operating points or, if the drifts are too strong, by a new search for a different optimal circuit configuration

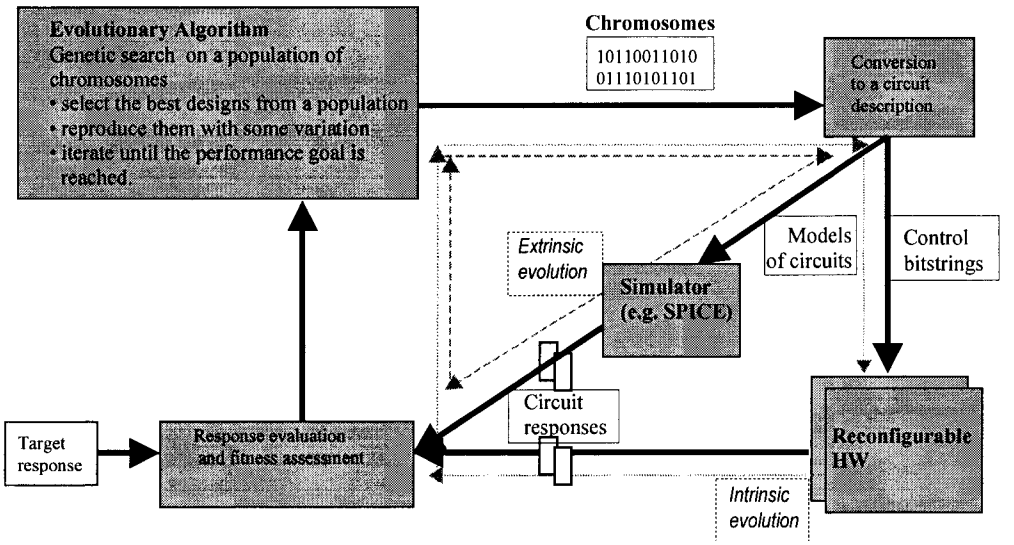

Figure 1. Evolutionary synthesis of electronic circuits

and operating point. Analog computation on simple low-power circuits can boost the emerging applications areas of "smart matter" and distributed high bandwidth adaptive sensing.

A shift in the design approach, from reconfigurable devices to evolution-oriented devices (evolvable devices) would facilitate hardware evolvability. This paper presents the Programmable Transistor Array (PTA) as a platform for experiments in evolutionary synthesis of both analog and digital CMOS electronic circuits, along with experiments that are expected to lead to design guidelines for a truly evolvable chip. Several experiments in evolutionary synthesis of CMOS circuits performed in simulations as well as on a test chip, led to the observations reported in the following. This paper is organized as follows: Section 2 presents the PTA rationale, and the PTA concept. Section 3 presents the experimental setup, including details of the software evolutionary design tool, the PTA chip, and a hardware tesbed. Section 4 describes a software experiment in which a CMOS circuit with an imposed current-voltage characteristic was synthesized by evolution, and discusses the effects the impedance of the switches has on evolution. Section 5 presents intrinsic evolution (on PTA chips) of the same analog computational circuits discussed in Section 4, as well as of a logical inverter. Section 6 compares the software and hardware experiments and presents some lessons learned from the experiments. Section 7 comments on related work. Lastly, Section 8 presents our conclusions.

2 Programmable Transistor Array

2.1 Rationale for analog reconfigurable devices

Current efforts in the evolution of hardware have been limited to simple circuits. For experiments with digital circuits, this limitation may be caused by a lack of power of evolutionary techniques in such search spaces. For analog circuits the limitation appears to come from a lack of appropriate reconfigurable analog devices to support the search. This precludes searches directly in hardware and requires evolving on hardware models. Such models require evaluation with circuit simulators such as SPICE; the simulators need to solve differential equations and, for anything beyond simple circuits, they require too much time for practical searches of millions of circuit solutions. A hardware implementation offers a big advantage in evaluation time for a circuit; the time for evaluation is determined by the goal function. For example, considering an A/D converter operating at a 100 kHz sampling rate the electronic response of the A/D converter is available within 10 microseconds, compared to (an over-optimistic) 1 second on a fast computer running SPICE; this advantage increases with the complexity of the circuits. In this case the 10^5 speedup would allow evaluations of populations of millions of individuals in seconds instead of days. There is also another characteristic that makes electronics an attractive domain to apply evolution; the higher the frequency at which a circuit needs to function, the shorter is its evaluation time, making the design of very high frequency circuits an excellent candidate for evolutionary design.

2.2 Rationale for fine-grained granularity

Most reconfigurable devices are digital, and while several levels of granularity are in use, the most common ones are configurable at the gate-level. In the analog programmable devices the reconfigurable active elements are Operational Amplifiers, such as in Field Programmable Analog Arrays (FPAA) with only very coarse granularity and few programmable components, allowing specified functionality with good precision, having a limited range of possible EHW experiments.

The optimal choice of elementary block type and granularity is task dependent. At least for experimental work in evolvable hardware, it appears a good choice to build reconfigurable hardware based on elements of the lowest level of granularity. Virtual higher-level building blocks can be considered by imposing programming constraints. An example of this would entail forcing groups of elementary cells to act as a whole (e.g. certain parts of their configuration bitstrings with the interconnections for the N transistors implementing a NAND would be frozen). Ideally, the "virtual blocks" for evolution should be automatically defined/clustered during evolution (an equivalent of the Automatically Defined Functions predicted and observed in software evolution).

2.3 PTA concept

The idea of a programmable transistor array was introduced first in [11]. The PTA is a concept design for hardware reconfigurable at transistor level. As both analog and digital CMOS circuits ultimately rely on functions implemented with transistors, the PTA appears as a versatile platform for the synthesis of both analog and digital (and mixed-signal) circuits. Further, it is considered a more suitable platform for synthesis of analog circuitry than existing FPGAs or FPAAs, extending the work on evolving simulated circuits to evolving analog circuits directly on the chip. The PTA module is an array of transistors interconnected by programmable switches. The status of the switches (ON or OFF) determines a circuit topology and consequently a 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 by convention one can assign 1 to a switch turned ON and 0 to a switch turned OFF . The PTA is planned to expand much like an FPGA with versatile functional cells. Figure 2 illustrates an example of a PTA module consisting of 8 transistors and 24 programmable switches. In this example the transistors P1-P4 are PMOS and N5-N8 are NMOS, and the switch-based connections are in sufficient number to allow a majority of meaningful topologies for the given transistor arrangement, and yet less than the total number of possible connections. Programming the switches ON and OFF defines a circuit for which the effects of non-zero, finite impedance of the switches can be neglected in the first approximation. An example of a circuit drawn with this simplification is given in Figure 3.

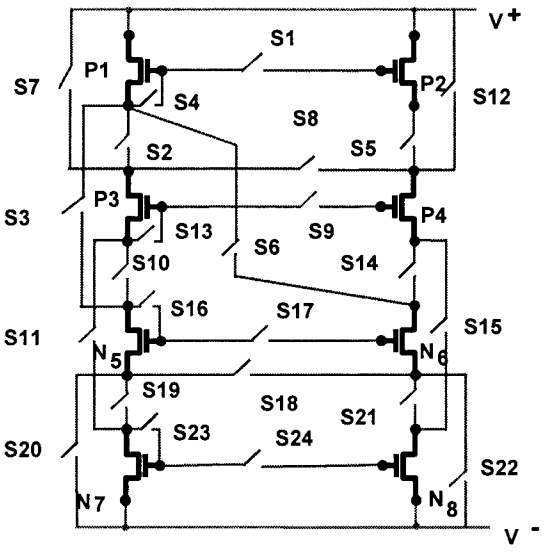

Figure 2. Module of the Programmable Transistor Array

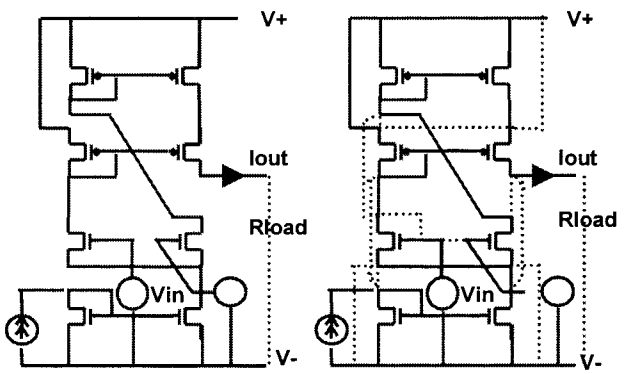

Figure 3. Schematic of a simple circuit implemented on a PTA module (with leakage through the finite resistance of OFF switches as dotted lines on the right figure).

The left drawing illustrates the ideal circuit, the right drawing shows with dotted lines the finite resistance of open switches. A power supply, input signals and a biasing current source have been added.

3 Testbed for evolutionary experiments

3.1 A software tool for evolutionary design

An evolutionary design tool was developed to facilitate experiments in simulated evolution. The tool illustrated in Figure 4 can be used for synthesis and optimization of new devices, circuits, or architectures for reconfigurable hardware. These operations get performed before any hardware gets fabricated. The tool proved very useful in testing architectures of reconfigurable HW and demonstrating evolution on them before the fabrication of a dedicated reconfigurable chip. In its current implementation the tool uses the public domain Parallel Genetic Algorithm package, PGAPack, and a public domain version of SPICE 3F5 as circuit simulator. An interface code links the GA with the simulator where potential designs are evaluated, while a GUI allows easy problem formulation and visualization of results. At each generation the GA produces a new population of binary chromosomes, which get converted into voltages in netlists that describe candidate circuit designs. Netlists are further simulated by SPICE. More details about the tool are given in [10].

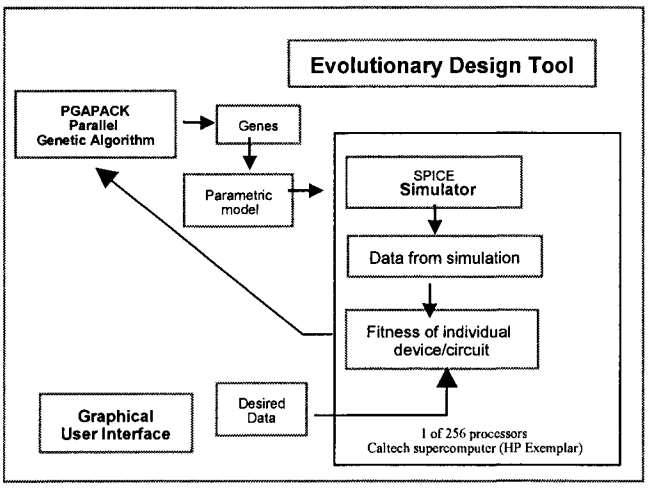

Figure 4. A software tool for evolutionary design tool

3.2 PTA chip and hardware testbed

Successful evolution on the simulated PTA encouraged the development of a test chip implementing the PTA architecture. Experiments with the chip could offer an estimate on how reliable is the evolution on SW models. More importantly, evolution of the circuit directly on the chip becomes possible, and at an expected accelerated pace of over two orders of magnitude compared to the simulation (estimated ~5 seconds compared to ~20 minutes on the supercomputer for the experiment described). As in the experimental simulations, the size of the transistors was fixed. The programmable switches were implemented with transistors, acting as simple T-gate switches. The considerations for this choice were:

- The switch has to pass analog signals
- The resistance of the switch needed to be variable between low (~tens/hundreds of ohms) and high (in excess of tens and hundreds of MOhms).
- Intermediate resistance values were necessary but linearity (R=R(Vgate-control)) was not important.

The analog gradual switches act in circuit evolution very much like resistive weights in an artificial neural network.

Each chip implements one PTA module. In this implementation only four layers of pairs of transistors (two PMOS and two NMOS) were chosen for simplicity. The PTA architecture allows the implementation of bigger circuits by cascading PTA modules. To offer sufficient flexibility the module has all transistor terminals connected via switches to expansion terminals (except those connected to power and ground). Further issues related to chip expandability are treated in [11]. The chip was fabricated as a Tiny Chip through MOSIS, using 0.5-micron CMOS technology. The test board with four chips mounted on it is illustrated in Figure 5.

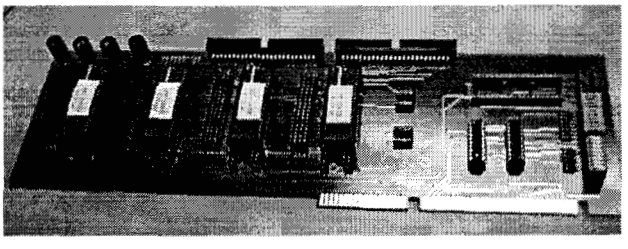

Figure 5. A test board with four PTA chips

The hardware testbed was built around National Instruments data acquisition hardware and software (LabView). A graphical tablet is attached to the system, allowing the user to introduce the characteristic of the desired circuit in a graphical way.

4 Evolution on a simulated PTA

4.1 Evolution using binary switches

The following experiment illustrates the evolutionary synthesis of a computational circuit. The goal of evolution was to synthesize a "Gaussian" circuit: a circuit that exhibits a Gaussian I-V (current-voltage) input-output characteristic. In a previous experiment [8] the circuit topology was fixed and the search/optimization addressed transistor parameters (channel length and width). Such evolution proved quite simple. The search for a topology turned out to be a much harder problem and several architectures were unsuccessfully attempted before the PTA was conceived. In the case of PTA, the transistor parameters were kept fixed and the search was performed for the 24 binary parameters characterizing the status of the switches. The fitness function was specified as a weighted combination of parameters x1,...,x7 in Figure 6. In some experiments the fitness was simply determined by an Euclidian distance between the candidate circuit response and the target response.

The evolution was simulated on a Caltech supercomputer (HP-Exemplar), using the Evolutionary Design Tool. Successful evolution was demonstrated on multiple runs with populations between 50 and 512, evolving for 50 or 100 generations. The execution time depends on the above variables and on the number of processors used (usually 64 out of the 256 available), averaging around 20 minutes (the same evolutions took about 2 days on a SUN SPARC 10). In some runs the solution circuit shown in Figure 3 (human-designed) was rediscovered by evolution.

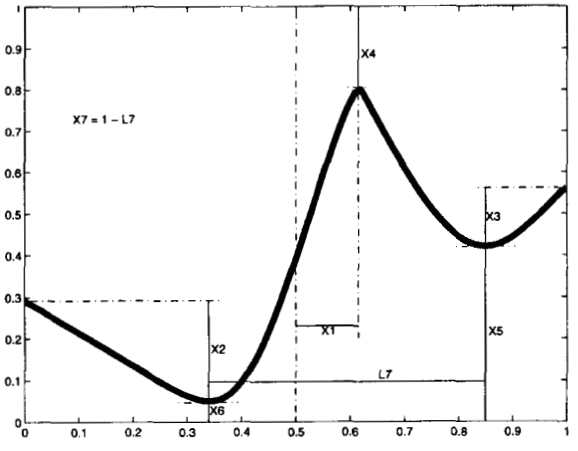

Figure 6. Parameters used for the specification of the fitness function. Fitness = f(x1,...,x7)

Besides rediscovering a human-designed solution, simulated evolution leads to several circuits that are

unusual from the perspective of current design practice, which were facilitated by the characteristics of the switches. While those observed so far were, in fact, mutants (2-4 bits away) of the same solution humans would design, it is expected that quite different solutions can be obtained. Even if this were not the case, mutants increase the number of valid solutions, and ease the search for a solution.

Examples of solutions found include the circuits illustrated in Figure 7, which produce the first two responses in Figure 8; some other responses from the same generation are illustrated in Figure 8 for comparison. It is interesting to analyze in more detail the unusual solutions found by evolution. Circuits like those illustrated in Figure 7 resulting from evolutionary synthesis are very similar (under certain test conditions) to that of the circuit shown in Figure 3. Thicker dotted lines show connections that existed in the circuit in Figure 3, but are missing in the circuits in Figure 7. As one can see these circuits are outside normal design practices, e.g., the transistors P2, P4 and N8 on the left circuit in Figure 7 have floating gates. The reality is that the switches have a big, but finite, resistance in the OFF state (~MOhms or GOhms) and a non-zero resistance/impedance in the ON state (~ tens of Ohms). An immediate observation is that while the effects of non-perfect switches may be negligible in a first approximation for many digital circuits, such effects may fundamentally affect analog programmable circuits.

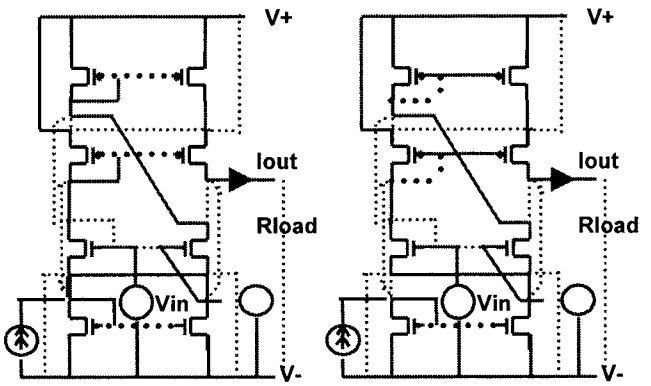

Figure 7. Circuits obtained by evolution; their design is unusual for common practice

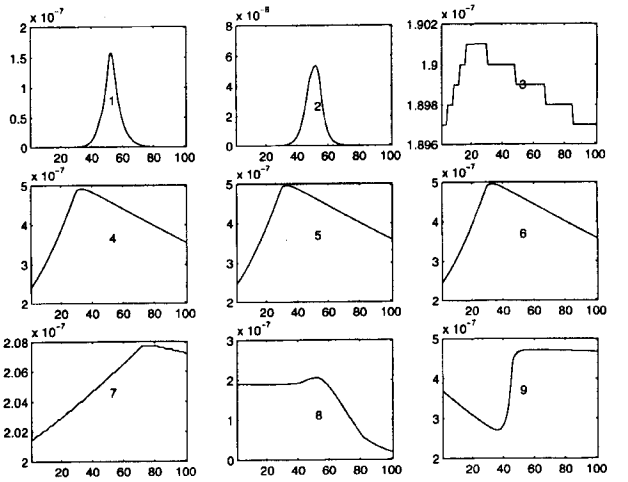

Figure 8. Best circuit responses in a simulated evolution

4.2 Evolution using an annealing effect on gray-scale switch conductance: morphing through fuzzy topologies

Two effects are explored in this section: the use of gray-level switches (with controllable conductance) and an annealing-like effect introduced to control switch conductance during evolution. A topology with gray-level switches is named here a fuzzy topology, because it blurs the borders between distinctive circuit topologies: the resulting circuits belong only to certain degrees to fixed, standard topologies in which two components are either connected or not. This diffuse, fuzzy topology resembles having many seeding topologies simultaneously co-existing, with superimposing effects, the role of evolution being to isolate the most promising topology. In a sense, evaluation of a fuzzy topology is equivalent to simultaneous concurrent evaluation of several superimposed circuit configurations.

Instead of being only ON/OFF, the switches were considered as having a Low/High resistance (Low for ON state). The binary genetic code would thus specify if the switch is Low or High, but the numerical meaning of this qualitative code would change gradually as a function of a temperature-like parameter. Initially the temperature is high, and Low and High switch status have values close to each other (2M for Low, 20M for High). Gradually the temperature goes down and the switch resistance polarizes to the extremes (10s of Ohms for Low and 10s of GOhms for High). The number of generations (100 in most runs) was chosen to ensure some quasi-static behavior (the response of the same best individuals from older generation differed in the newer generation because the circuit had

different resistance for switches). The annealing effect induces a modification in the circuit to be evolved rather than a modification in the fitness function or in a parameter of the search algorithm directly (e.g. the mutation rate). This technique, which we refer to as "morphing through fuzzy topologies" was evaluated in conjunction with the fitness function from Figure 6, which considered shape information. The technique proved about one order of magnitude more efficient than the search with binary switches. Promising individuals (with higher fitness) have shown-up much earlier in the search. This is probably because of the richer set of effects due to the active contribution of all switch transistors (not only for signal passing), but also because through the gray-switches signals get to the output test/probe points (albeit attenuated) even through "closed" switches along the path.

Many solutions were actually observed while running through this "annealing". If the goal is to design a blueprint "binary" topology (a wire connecting two components either exists or does not) the annealing technique could be used as a catalyst to accelerate evolution. If on the other hand evolution takes place on hardware that supports gray-level switches, then the degree of opening of the switches could be an extra degree of freedom for the problem, enabling an increased number of solutions. It is possible that these solutions are more sensitive (to various drifts, etc.) than solutions with binary switches.

5 Evolution on PTA chips

5.1 Evolution of a Gaussian circuit on a single PTA and on cascaded PTAs

The same evolutionary experiment, aiming at the synthesis of a DC circuit with a Gaussian response, was performed in hardware on the PTA chips. Four chips were programmed in parallel with bit-string configurations corresponding to four individuals of a population of 1000; then, the next four were programmed, and so on until all 1000 in one generation were tested. Evolution led to "Gaussian" circuit solutions within 20-30 generations. The current speed of evaluation is 1000 circuits in 8.25 seconds using the four PTA chips in parallel; another order of magnitude speed-up is expected when some existing data acquisition bottlenecks will be solved.

An example of GA parameters in one of the runs is:
Population: 1000, Chromosome size: 24 bits for 1 PTA, and 52 to 88 bits for 2 PTAs (the number depends on interconnection schemes), Evaluation samples: 30, Mutation rate: 4%, Crossover rate: 70%, Tournament Selection: 20 individuals, Elit Strategy: 9% population size (88 individuals), Fitness Function: Square Root Mean Error.

The response of four mutants is illustrated in the screen capture shown in Figure 9 (LabView display of the signals captured by the data acquisition boards). Notice the "mutations" in the genetic code of the solutions obtained by evolution (vertical chromosomes R24 to R1 reading from top to bottom, corresponding to switches S24 to S1 in Figure 2) compared with the human-designed circuit.

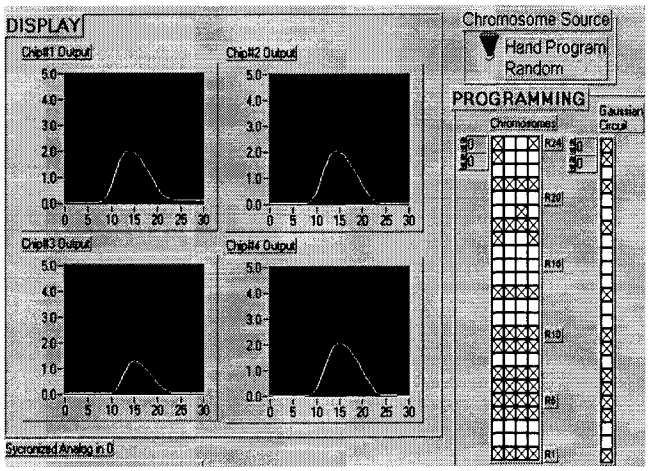

Figure 9. The "Gaussian" response of four "mutants" and their "genetic code" compared to the code of a human-designed circuit

5.2 Evolution of an inverter

The PTA should allow evolutionary synthesis of a variety of a variety of analog and digital circuits. The current setup allowed only one variable inputs, therefore only the evolution of an inverter was attempted. The characteristic of the evolved circuit for the inverter is shown in Figure 10, and is compared with the target response.

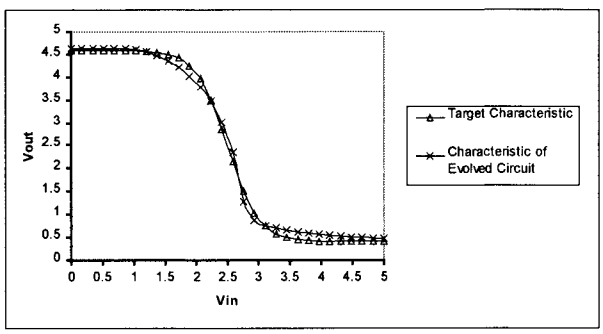

Figure 10. The characteristic of an evolved inverter

6 Lessons learned

6.1 Limitation of the software models

An interesting observation was that, other than the "correct" human-designed solution rediscovered by evolution, the solutions evolved on the PTA chip were different than those evolved in simulations. (At least the few of them that were tested; additional circuit solutions may exist that lead to the same response both in the SPICE simulation and programmed on the chip). It would thus appear that different effects are exploited to lead to solutions in the model and in the silicon implementation. More precisely, the circuit solutions evolved in simulations (with SPICE resistive models for ON/OFF switches) did not prove to be solutions when programmed on the PTA chip, and vice-versa.

The configuration solutions that evolved directly on the PTA chip (e.g. those in Figure 9) did not produce a Gaussian output when simulated in SPICE. (Further experiments using more accurate models of the PTA silicon implementation are in progress). Thus, it appears justifiable to express reservation on the validity of a solution obtained by "extrinsic" evolution of analog circuits until it is verified in hardware (at least for particular PTA discussed here and with the limited accuracy model used).

6.2 Effects of switches

Switches were implemented with transistors, whose switching characteristic differs from that of an ideal switch (zero resistance/impedance when ON, infinite when OFF). However, instead of being a drawback, the effects of non-perfect binary switches appear to be exploitable in favor of evolution. The OFF state is in fact a high-impedance connection which still allows leakage currents that can affect the CMOS circuit behavior. The effect of partly opened switches (variable resistance) leads to an extremely rich set of behaviors, including solutions to the target problem. Transistors that operate like partly open switches become active components. Thus it is more suitable to look at the circuit as having 32 transistors than 8 transistors interconnected by 24 switches.

One should mention here that the richer set of behaviors possible using transistors as variable resistance elements rather than switches makes the testability problem harder. However the architecture has an intrinsic higher degree of fault-tolerance.

6.3 Speed-up by evaluations in hardware

Hardware evaluation can produce a speed-up, especially when one simulates large, complex analog circuits, and the circuit response is rapid. One aspect that can however be easily overlooked is the frequency of operation for which a certain circuit is designed. For example, the DC circuit evolved in Section 3 was evaluated obtaining results of DC SPICE analysis (which in principle can be accelerated by more powerful simulation platforms). There are however limitations to increasing the speed of configuration and test in hardware. For example, the output of the Gaussian circuit on the PTA started attenuation when the input ramp signals were exceeding 1kHz. Thus, no more than 1000 circuits per second (of desired low frequency response) could be reliably evaluated. Even though some artifacts of the particular PTA design and load choice may be involved, it appears natural that evaluating the circuits at a different frequency than that of intended functioning may introduce errors. Evaluation in parallel is an alternative speed-up technique, and at least in the experiments with the PTA chips no significant differences were noted between the implementation of the same circuit on different chips.

6.4 Effects of fitness function and algorithm characteristics

The use of the problem-oriented fitness function (Figure 6) combined with a mechanism for enforcing diversity in the population lead to evolutions an order of magnitude faster than the use of fitness defined as Euclidian distance and parameters for the GA illustrated in Section 4. Future research should explore how problem-specific fitness functions can be automatically determined from user's desired characteristic.

7 Related work

Evolution of circuits reconfigurable at transistor level (in particular CMOS) was proposed and demonstrated for parameter optimization in [11]. Evolution at transistor level became more recently one of the focuses at Sussex University, in particular reflected in the work of Layzell [12]. Our focus on CMOS has some important consequences including the existence of the effects presented in this paper as associated with the OFF switches. The small leakage current through OFF switches is sufficient for CMOS but does not affect bipolar transistors, hence some mutant solutions appear only in CMOS for the described topology.

8 Conclusions

Automatic synthesis/self-configuration of an analog and of a digital circuit were demonstrated on an experimental CMOS chip implementing a PTA architecture proposed as reconfigurable hardware platform for evolutionary synthesis experiments. Comparative software and hardware

evolutionary experiments indicate that only evolution in hardware guarantees a valid solution to the evolutionary synthesis. Solutions evolved on the chip proved robust when transferred to other chips from the same fabrication lot. Controlling the resistance of transistors rather than using them as binary ON/OFF switches were shown to improve circuit evolvability.

Acknowledgments

The research described in this paper was performed at the Center for Integrated Space Microsystems, Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the National Aeronautics and Space Administration.

References

[1] J. Koza, F.H. Bennett, D. Andre, and M.A Keane, "Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming", *Proceedings of Genetic Programming Conference*, Stanford, CA , pp. 28-31, 1996

[2] J. Lohn, J. and S. Colombano, "Automated Analog Circuit Synthesis using a linear representation", M. Sipper, D. Mange and A. Perez-Uribe (Eds.) *Evolvable Systems: From Biology to Hardware*, Springer-Verlag Lecture Notes in Computer Science Berlin 1998, pp. 125-133

[3] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined in physics". In *International Conference on Evolvable Systems*. Springer-Verlag Lecture Notes in Computer Science, 1996, pp. 390-405.

[4] E. Sanchez and M. Tomassini (Eds.) *Towards Evolvable Hardware*, LNCS 1062, Springer-Verlag, 1996

[5] T. Higuchi, M. Iwata, and W. Liu (Eds.) Evolvable Systems: From Biology To Hardware, *Proc. of the First International Conference, ICES 96*, Tsukuba, Japan, Springer-Verlag Lecture Notes in Computer Science, 1997.

[6] M. Sipper, D. Mange, A. Perez-Uribe (Eds.) Evolvable Systems: From Biology To Hardware, *Proc. of the Second International Conference, ICES 98*, Lausanne, Switzerland, Springer-Verlag Lecture Notes in Computer Science, 1998.

[7] J. R. Koza, F. H. Bennett III,, D. Andre and M. A. Keane, Genetic Programming III – Darwinian Invention and Problem Solving, Morgan Kaufman, San Francisco, 1999

[8] E. Vitoz, Analog VLSI Processing: Why, Where and How, *Journal of VLSI Processing*, Kluwer, 1993

[9] Stoica, A. On hardware evolvability and levels of granularity. *Proc. of the International Conference "Intelligent Systems and Semiotics 97: A Learning Perspective,* NIST, Gaithersburg, MD, Sept. 22-25, 1997

[10] Stoica, A. Klimeck, G. Salazar-Lazaro, C. Keymeulen, D. and Thakoor, A. Evolutionary design of electronic devices and circuits, *Proc. of the 1999 Congress on Evolutionary Computation,* Washington, DC, July 6-9, 1999

[11] Stoica, A. Toward evolvable hardware chips: experiments with a programmable transistor array. *Proceedings of 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems,* Granada, Spain, April 7-9, IEEE Comp Sci. Press, 1999.

[12] Layzell, P. " A New Research tool for Intrinsic Hardware Evolution" , *ICES'98*, Lausanne, Switzerland, 1998