Towards on-board synthesis and adaptation of electronic functions: An Evolvable Hardware Approach

Adrian Stoica, Didier Keymeulen, Carlos-Salazar Lazaro, Wei-te Li, Ken Hayworth, and Raoul Tawel
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
818-354-2190
adrian.stoica@jpl.nasa.gov

Abstract—Autonomous space systems will need to adapt to new environments and perform new functions, in addition to those specified at launch. Adaptation supports long-life meaningful survivability and should include both software and hardware. Reconfigurable hardware could speedup computation intensive tasks by orders of magnitude and could ensure fault-tolerance bypassing faulty cells. *Evolvable Hardware* is reconfigurable hardware that self-configures under the control of an evolutionary algorithm. The search for a hardware configuration can be performed using software models or, faster and more accurate, directly in reconfigurable hardware. Initial experiments demonstrate the possibility to automatically synthesize both digital and analog circuits. The paper introduces an approach to automated synthesis of CMOS circuits based on evolution on Programmable Transistor Arrays (PTA). The approach is illustrated by an experiment showing evolutionary synthesis of a circuit with a desired DC characteristic; evolution took ~20 minutes on a supercomputer and is expected to take ~5 seconds on a PTA chip recently fabricated.

TABLE OF CONTENTS

1. INTRODUCTION

Spacecraft autonomy plays a key role in future space missions. Long delays in communications between spacecraft and Earth are inherent for remote missions, and preclude real-time human operator spacecraft control. An intelligent, autonomous spacecraft must be able to cope with unexpected situations and should be able to adapt to new environments. Adaptation includes coping with environmental conditions such as radiation or thermal changes with changes due to aging or faults, with opportunities or imminent dangers, and with mission changes. A high adaptation capability is paramount for long-life meaningful survivability and could enable new classes of missions such as interstellar missions longer than human lifetime, in harsher and highly unknown environments.

Adaptation should be not only in software, which has offered the most flexibility to date, but in hardware as well. In many situations can hardware designs optimized for certain functionality provide orders of magnitude higher processing power than a software solution running on a general purpose central processing unit (CPU). Such increased performance could be achieved using reconfigurable hardware, for example built with field-programmable gate arrays (FPGA). At present, the configurations are designed on ground and up-linked to the spacecraft (for example, a new configuration that bypasses faulty cells). In the future, it is desirable to have the solutions automatically determined on-board the spacecraft.

Recent research has demonstrated the possibility of achieving automatic hardware design and configuration. *Evolvable hardware* is reconfigurable hardware that self-configures under the control of an evolutionary algorithm. The search for a hardware configuration can be made in software, and the final solution can be downloaded to the hardware. Alternatively, evolution in hardware (directly on the chip) can speedup the search for a solution circuit by a few orders of magnitude compared to evolution in software simulations. Moreover, since the software simulation relies on models of physical hardware with certain limited accuracy, 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 is performed through a succession of changes of elementary cell functions and cell interconnectivity pattern, thus obtaining increasingly more fit configurations until a target functionality is reached. As it is the case in nature, evolution results in individuals that are increasingly more adapted to their environments and can change themselves to match changes in environments and modifications of their own goals. Unlike in nature, evolution in silicon has the advantage that could be extremely rapid, with millions of generations of "living" circuits evaluated in only a few seconds.

This paper presents several ideas in Evolvable Hardware and introduces an approach to evolutionary synthesis of CMOS circuits, illustrated with an experiment in evolution of a computational circuit. The paper is organized as follows: Section 2 describes the principles of evolutionary synthesis of electronic circuits, highlighting some important results in the field. Section 3 introduces evolution of CMOS circuits using Programmable Transistor Arrays and proposes a design of hardware reconfigurable at transistor level. Section 4 presents the mechanism used for reconfiguration, detailing the Genetic Algorithm (GA) used in the experiments that follow. Section 5 presents an evolutionary design tool built around a parallel GA implementation and a circuit simulator. The tool was used on a 256-processor machine to simulate evolution of circuits of CMOS transistors. Section 6 describes a hardware testbed and the preparations for replicating directly in hardware the results obtained in simulations. A chip implementing a PTA module was designed and fabricated in 0.5 micron CMOS technology. Finally, Section 7 presents conclusions of this paper

2. Evolutionary Synthesis of Electronic Circuits

This section describes the principles of evolutionary synthesis of electronic circuits and highlights some important results in the field. The idea behind evolutionary synthesis or Evolvable Hardware (EHW) is to employ a search/optimization algorithm that operates in the space of all possible circuits and determines solution circuits with desired functional responses [1], [2], [3]. Most experiments were performed using evolutionary algorithms such as GAs and Genetic Programming . The genetic search is tightly coupled with a coded representation for the circuits. Each circuit is associated with 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. Synthesis is the search in the chromosome space for the solution corresponding to a circuit with a desired functional response. The genetic search follows a "generate and test" strategy: a population of candidate solutions is maintained at each time, the corresponding circuits are evaluated, and the best candidates are selected and reproduced in a subsequent generation until a performance goal is reached. Circuit evaluation can be done on software models using circuit simulators, in which case, evolution is called *extrinsic* evolution, or directly in reconfigurable hardware, in which case, it is called *intrinsic* evolution.

The main steps of evolutionary synthesis are illustrated in Figure 1. First, a population of chromosomes is randomly generated. 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. Preparing for a new iteration loop, a new population of individuals is generated

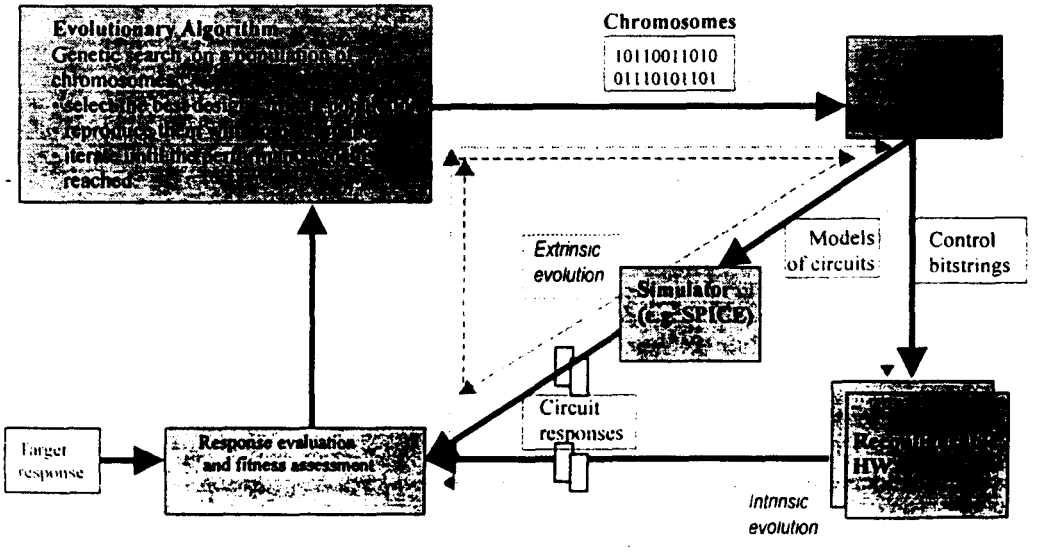

Figure 1: Evolutionary Synthesis of Electronic Hardware

from the pool of best individuals in the previous generation, some individuals being taken as they were and some being modified by genetic operators, such as chromosome crossover and mutation. The process is repeated for many generations and results in increasingly better individuals. The process is usually stopped after a number of generations or when the closeness to the target response has reached a sufficient degree. One or several solutions may be found among the individuals of the last generation.

A variety of circuits have been synthesized this way. Koza, et al, used Genetic Programming to grow an "embryonic" circuit to a circuit that satisfies desired requirements. This approach was used for evolving a variety of circuits, including filters and computational circuits. Koza's evolutions were performed in simulations, without concern for physical implementation, but rather as a proof-of-concept that evolution can lead to designs that compete or even exceed in performance over human designs. 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 application specific integrated circuit (ASIC)), and thus intrinsic evolution was not possible. An alternative encoding technique for analog circuit synthesis, which has the advantage of reduced computational load was used in [4] for automated filter design.

On the other hand, intrinsic evolution was also demonstrated, for the first time by Thompson [5]. Thompson used an FPGA as the programmable (digital) device and a Genetic Algorithm as the evolutionary mechanism to configure a frequency discriminator from the digital gates available on a small part of the FPGA. Although evolution used the circuitry prepared to implement logic gates, the functionality was obtained exploiting more than the underlying physical phenomena at transistor level.

There are currently several directions of research for EHW including automated design of logical circuits, intrinsic evolution in FPGA, FPAA (field-programmable analog arrays) and ASIC, for applications in robotics, adaptive compression, automated parameter tuning for ASICs, etc.[1]

EHW work at JPL focuses on evolvable hardware for space applications [6], such as adaptive signal conditioning and signal processing, adaptive compression, and adaptive robotics. In particular, we are interested in evolution at CMOS transistor level [7]. CMOS transistors are the elementary building blocks of the majority of current microelectronics. Addressing evolution at this low level allows most flexibility for synthesizing analog, digital, and mixed signal designs. Although for many functions it is easier to synthesize based on higher-level dedicated blocks, the lessons we learn in synthesizing at this level can be extended to evolution of circuit systems made of other devices and materials/structures. An important part of our activity is developing dedicated hardware capable of evolution of both analog and digital circuits, directly on the chip

[1] Important international research in this field is concentrated at several universities in UK, in Switzerland at EPFL, and in Japan at ETL (Dr. Higuchi's group is the largest and most prolific group worldwide), ATR, and other places.

3. Reconfigurable Hardware at Transistor Level

This section introduces evolution of CMOS circuits based on PTAs, describing a design for hardware reconfigurable at the transistor level. In the approach proposed here, an evolutionary algorithm searches the space of circuits configurable on a PTAs. The PTA allows synthesis of analog, digital, and mixed-signal circuits being a more suitable platform for synthesis of analog circuitry than the FPGAs used by Thompson and also extending Koza's work and evolving analog circuits directly on the chip.

A Programmable Transistor Array

The proposed PTA 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 one can assign 1 to a switch turned On and 0 to a switch turned Off. The PTA can be a modular architecture, in which a certain module can be cascaded to determine a more complicated circuit topology. 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 transistors arrangement, yet less than the total number of possible connections.

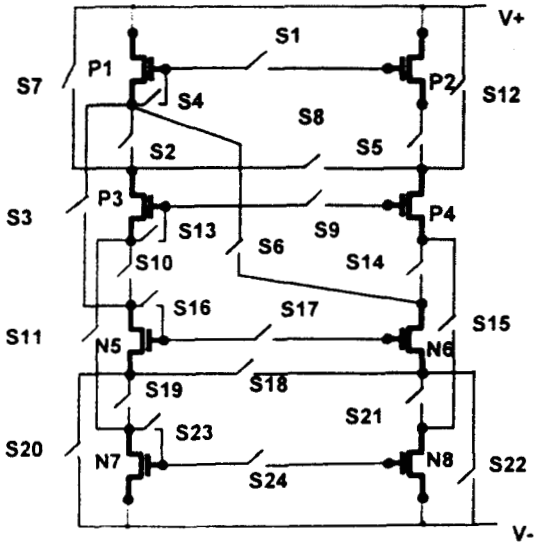

Figure 2. Module of the Programmable Transistor Array

Programming the switches On and Off determines 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. 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 output measuring instrument have been added.

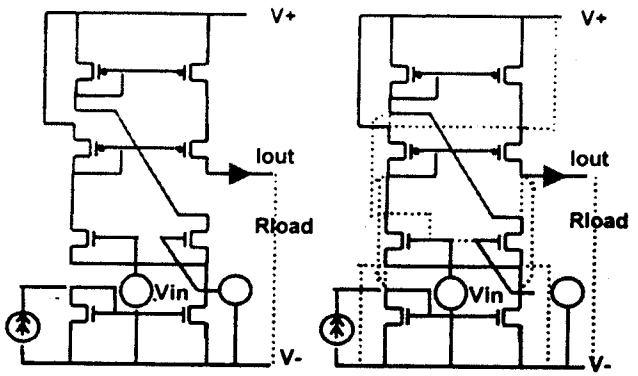

Figure 3. Schematic of a simple circuit implemented on the PTA module (with finite resistance of Off switches as dotted lines on the right figure)

In a realistic model the switches have a non zero resistance/impedance in the On state, and a big, but finite, resistance in the Off state. *While the effects of no perfect switches may be negligible for most common applications of digital circuits, such effects may fundamentally affect analog programmable circuits.* As an example, the circuits shown in Figure 4 are outside normal design practices, e.g., the transistors P1 and P2 have floating gates (in theory infinite resistance between the gates). In reality, when a switch between the two gates is Off, the resistance is not infinite but finite (~ MOhm or GOhm), and the responses of the circuits in Figure 4 are very similar (under certain test conditions) to that of the circuit shown in Figure 3 (the On resistance is also non zero, but rather ~ tens of Ohms). Thicker dotted lines show connections that existed in the circuit in Figure 3 but are missing in the circuits in Figure 4.

4. Reconfiguration Mechanisms

This section presents the mechanisms used for reconfiguration and detail the GA used in the experiments that follow. A variety of evolutionary algorithms (including GAs and Genetic Programming) have been used successfully for evolution of circuits. GAs were chosen here because 1) previous work has demonstrated its efficiency in evolutionary circuit synthesis, 2) the mechanism is simple to understand and implement, 3) public domain software exists and saves development time, and 4) the focus was on the reconfigurable hardware and not on the reconfiguration mechanism.

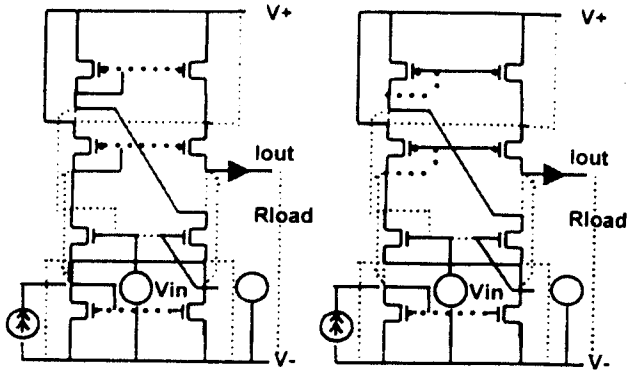

Figure 4. Circuits obtained by evolution: their design is unusual for common practice

It is likely that more intelligence can be inserted into the search mechanism. A simple block diagram of operations taking place in a GA is illustrated in Figure 5.

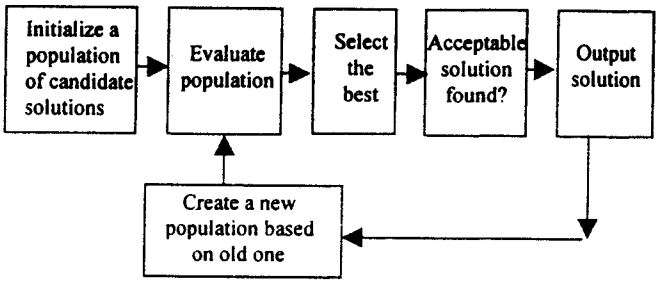

Figure 5. Sketch of a simple Genetic Algorithm

5. Software Implementation Aspects and Simulated Evolution

Section 5 presents an evolutionary design tool built around a parallel GA implementation and a circuit simulator. The tool was used on a 256-processor machine to simulate evolution of circuits of CMOS transistors. This section details the evolution of a circuit with a gaussian I-V DC response. The evolutionary synthesis approach illustrated in Figure 1 was applied to the model of PTA illustrated in Figure 2. The evolution was simulated on a Caltech supercomputer (HP-Exemplar), using an evolutionary design tool developed for this purpose, illustrated in Figure 6.

An Evolutionary Design Tool

An evolutionary design tool was built to facilitate experiments in simulated evolution. The tool can be used for synthesis and optimization of new devices, circuits, or architectures for reconfigurable hardware. These operations get performed before the mission and before any hardware gets fabricated. The tool proved very useful in demonstrating evolution of circuits using the PTA before

the fabrication of a dedicated reconfigurable chip. The tool can also be used in hardware-software co design before the mission. In its current implementation, the tool uses the public domain Parallel Genetic Algorithm package PGAPack and two simulators, the Nanoelectronic Modeling Tool (NEMO), and SPICE. An interface code links the GA with the simulator where potential designs are evaluated, while a graphic user interface (GUI) allows easy problem formulation and visualization of results. Each generation the GA produces a new population of binary chromosomes, which get converted into voltages in SPICE netlists that describe candidate circuit designs. The circuits expressed by netlists are simulated by a public domain version of SPICE 3F5 as the circuit simulator.

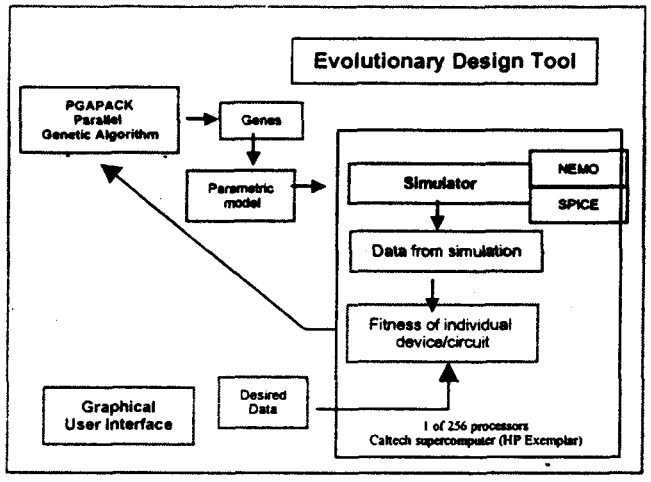

Figure 6. An Evolutionary Design Tool

Evolution of a CMOS circuit with a gaussian response

Evolutionary synthesis of a computational circuit was chosen to illustrate the approach. The goal of evolution is to synthesize a circuit that exhibits a gaussian I-V characteristic. The problem was approached initially by fixing the circuit topology and performing a search/optimization using transistor channel lengths and widths [7]. The initial experiments have shown that such evolution is simple, however, the search for a topology proved a much harder problem and was not achievable in a hardware implementable context before the PTA approach was developed. In the PTA case, the transistor parameters were kept fixed, and the search was performed for the 24 binary parameters characterizing switches status. An important role was the correct specification of the fitness function, for which a set of combined measures (illustrated in Figure 7) was defined. 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 (commonly 64 out of the 256 available), averaging around 20 minutes (the same evolutions took about 2 days on a SUN SPARC 10). The solutions found include the circuits illustrated in Figure 4, which produce the first two responses in Figure 8; some other responses from the same generation are illustrated in Figure 8 for comparison.

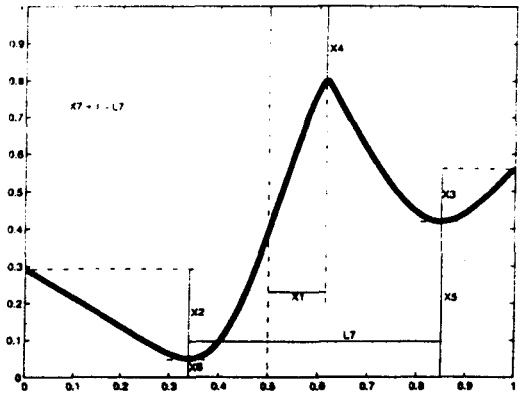

Figure 7. Parameters used for the specification of the fitness function. Fitness =f(x1,...x7)

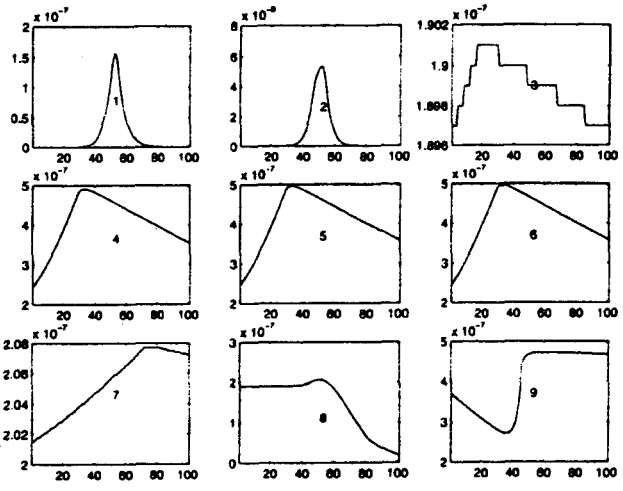

Figure 8. Best circuit responses in a simulated evolution

6. Hardware Implementation Aspects

To fully exploit intrinsic evolution, and to validate in hardware the results of simulations, the chip implementing the PTA has been designed. The chip was called PROTEA (PROgrammable Transistor Evolvable Array) to remind of its morphing capability [2].

[2] Protea – any shrub or small tree of genus Protea, of tropical and southern Africa, having flowers with colored bracts arranged in showy heads (from New Latin, from PROTEUS, referring to the large number of different forms of the plant. In the Greek mythology, PROTEUS was a prophetic sea god capable of changing his shape at will (Collins English Dictionary).

The chip was fabricated as a tiny chip through MOSIS, using 0.5 micron CMOS technology. At the moment of this writing, the chip (Figure 9) has just been received from fabrication. The test board with four chips mounted on it is illustrated in Figure 10.

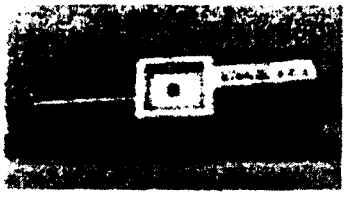

Figure 9. Programmable Transistor Array Chip

Figure 10. A Test Board with Four PTA Chips

7. Summary and Conclusion

In this paper, we have described the following:

- An approach to evolutionary synthesis of electronic circuits consisting of CMOS transistors
- A PTA architecture, proposed for the reconfigurable hardware on which evolutionary synthesis can take place
- An implementation of PTA in a CMOS chip (fabricated and under tests)
- An evolutionary design tool for a parallel machine
- A set of experiments in simulated evolution of analog electronic circuits

Evolutionary algorithms proved to be a powerful technique for automated synthesis of electronic circuits. The feasibility of performing this synthesis directly in reconfigurable hardware creates the possibility of on-board automated synthesis and adaptation of electronics.

Acknowledgements

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] De Garis, H. Evolvable Hardware: Genetic Programming of a Darwin Machine. *Int. Conf. on Artificial Neural Networks and Genetic Algorithms*, Innsbruck, Austria, Springer Verlag, 1993

[2] Higuchi T et al. Evolvable Hardware with Genetic Learning, in Proc. of Simulated Adaptive Behavior, MIT Press, 1993

[5] 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. In *Proceedings of Genetic Programming Conference*, pages 28-31, 1996.

[4] Lohn, J. and Colombano, S. Automated Analog Circuit Synthesis using a linear representation. International Conference on Evolvable Systems, Lausanne, Sept 23-26, 1998, 125-133

[5] 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.

[6] Stoica, A., Fukunaga, A., Hayworth, K., and C. Salazar-Lazaro, Evolvable hardware for Space Applications, *International Conference on Evolvable Systems*, Lausanne, Sept 23-26, 1998

[7] Stoica, A., On hardware evolvability and levels of granularity *International Conference on Intelligent Systems and Semiotics*, NIST Gaithersburg VA, Sept, 1997

Adrian Stoica is a Senior Member of Technical Staff at Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. His research interests include learning and adaptive hardware, evolvable hardware, sensor fusion processors, robot learning, and humanoid robots. He has published more than 30 papers in these areas. He has a Ph.D. in EE from Victoria University of Technology, Melbourne, Australia, and a MSEE from Technical University of Iasi, Romania.

Didier Keymeulen is a Research Engineer at the Jet Propulsion Laboratory of the California Institute of Technology and the National Electrotechnical Laboratory at Tsukuba, Japan. His interests are in complex dynamical systems applied to the design of adaptive embedded systems. He obtained his M.Sc. and Ph.D. in Computer Science from the Artificial Intelligence Laboratory of the Vrije Universiteit Brussel, Belgium.

Wei-Te Li is currently a graduate student at the Electrical Engineering Department at the University of Washington. His research interests focus mainly on EM waves. But while he interned at JPL in the summers of 1997 and 1998, he did VLSI design, layout and simulation for the Multiple Experts Rules Processor (MERPS), fuzzy logic chip components, and an evolvable hardware array.

Ken Hayworth is a Member of Technical Staff at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. He received a BS in Computer Science and Engineering and a BS in Mathematics both from the University of California at Los Angeles. Since joining JPL, his research interests have included artificial intelligence, neural networks, and evolvable hardware.

Raoul Tawel is a Senior Member of Technical Staff at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA. He received his BS in Physics from the University of Western Australia and his Ph.D. in Physics from Brandeis University in 1987. Since joining JPL, his research interests have included neural networks, evolvable hardware, reconfigurable hardware, and parallel processing. He has published more than 40 papers in these areas.