# Evolvable Hardware for Space Applications

Adrian Stoica, Alex Fukunaga, Ken Hayworth, Carlos Salazar-Lazaro

Center for Integrated Space Microsystems
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena CA 91109, USA

**Abstract.** This paper focuses on characteristics and applications of evolvable hardware (EHW) to space systems. The motivation for looking at EHW originates in the need for more autonomous adaptive space systems. The idea of evolvable hardware becomes attractive for long missions when the hardware looses optimality, and uploading new software only partly alleviates the problem if the computing hardware becomes obsolete or the sensing hardware faces needs outside original design specifications. The paper reports the first intrinsic evolution on an analog ASIC (a custom analog neural chip), suggests evolution of dynamical systems in state-space representations, and demonstrates evolution of compression algorithms with results better than the best-known compression algorithms.

## 1 Introduction

Spacecraft autonomy plays a key role in future space missions. During remote missions, spacecraft are separated from Earth by distances that delay communications by many minutes (e.g. ~ 10 minutes one-way in communications with the Mars rover), which precludes real-time human operator control of the spacecraft. An intelligent, autonomous spacecraft must be able to cope with unexpected situations, and should be able to adapt to new environments. Spacecraft adaptation is largely controlled by on-board electronic hardware, hence a special need exists for adaptive electronic hardware.

Evolvable hardware (EHW) is adaptive hardware that self-organizes/reconfigures under the control of an evolutionary algorithm [1]. *Extrinsic* EHW refers to evolution in a software simulation (using models of the hardware behavior), followed by a download of the configuration of the most fitted solution to a programmable hardware. In *intrinsic* EHW the configuration bits are downloaded from the beginning to hardware, and the degree of adaptation/fitness is evaluated by observing the behavior of the real hardware. Successful evolution has been reported in simulations of analog (e.g. [2], [3]) and digital (e.g. [4]) circuits and in real digital hardware (e.g. [5] [6]). No intrinsic EHW on analog chips has been reported, an important reason being that the lack of commercial programmable analog chips suitable for EHW. The analog circuits evolved in simulations (e.g. [2], [3]) can not be extended directly to practical HW implementations. On the other hand, some researchers believe that the analog domain would be more suitable for evolution, and there were interpretations that even evolution on (digital) FPGA may have benefited from effect of analog underlying circuitry [5].

This paper addresses some EHW issues relevant to space applications. The paper is organized as follows: Section 2 discusses characteristic aspects of space-oriented EHW. Section 3 describes experiments in intrinsic evolution on application specific analog chips: a test in evolving a function approximator, and evolution of a vision-based tracking behavior for a mobile robot. Section 4 introduces a novel approach to EHW representing the system to be evolved in a behavioral AHDL (Analog Hardware Descriptive Language), in a state-space description. Section 4 presents an application of EHW to adaptive compression.

## 2 Space-oriented evolvable hardware

There are several characteristic aspects that need to be considered when addressing space-oriented EHW. It is very important to have a systems approach, understanding clearly that EHW is part of a bigger system for which optimality is sought. One needs to understand who/what provides the means for calculating a fitness function for candidate solutions, whether there is a target functionality or reward mechanism stored in some memory on-board, or reinforcement comes from the environment. Also, of most importance is to know how safe is EHW for the space system and also if evolution can provide a response in useful time.

The safety of space systems (such as satellites, spacecraft, planetary probes or rovers) being so critical, our current focus is on evolving adapted sensors and sensory information processing systems, rather than, for example, spacecraft control. The operations from the moment signals reach the sensors until a decision is made, or a coded signal is sent to ground, are fully inter-related and ultimately could be co-evolved in their ensemble to a global optimal signal processing efficiency. In practice, it may be simpler to consider them separately, and evolve independently. The operations could be, for example, signal acquisition (e.g. sensor adaptation in terms of sensor sensitivity domain/profile, focus of attention, etc.), signal pre-processing (e.g. filtering, amplification), extraction of information for on-board decisions (such as sensor-pointing), and preparation of a signal for transmission to Earth (e.g. compression).

The EHW directions we have explored aim to address some aspects from each of the above operations. We performed experiments in intrinsic evolution on analog ASICs, trying to understand more about intrinsic EHW and integration of such chips into higher level systems such as control of sensor arrays, antennas and solar panels, instrument pointing (in this sense we evolved circuits with desired I/O characteristic). We addressed the evolution of electronic circuits, which can be used for filtering or other signal transformations, exploring the design of evolvable CMOS chips based on transistor and elementary circuit blocks (current mirrors, differential pairs, etc) (which will be described in another paper). We addressed evolution of complex dynamic systems, which can be used to learn decision mechanisms or system behaviors. Thus, we developed a novel approach to EHW, which relies on a state-space representation of systems. We developed a simple test to explore the capability of evolving autonomous vision-based navigation. Finally, we approached evolution of algorithms for on-board signal processing, more precisely compression algorithms. In the context of space applications, compression is a very important problem because of the limited communications bandwidth between a spacecraft and the ground. EHW has already shown to be capable of deriving efficient adaptive compression [14].

## 3 Intrinsic evolution on programmable analog ASICs

This section reports results of intrinsic evolution on dedicated (special purpose) analog chips (ASIC), more precisely analog neural chips. The domain of evolutionary neural networks [5], as well as various analog neural chips have existed for several years, but no results on evolving on the chip have been reported. Previously, our group has explored other approaches for hardware-in-the-loop and on-chip learning, including gradient-descent approaches [7]. A main reason for performing intrinsic (hardware-in-the-loop) learning on an analog chip is that, unlike the digital case where very good models exist in advance, in analog there is always a slight discrepancy between a model and the physical implementation, and therefore a system evolved in software (extrinsic) may exhibit an offset behavior when downloaded to hardware. The tasks described in this section are small and should be regarded as demonstrating an idea rather than applications.

The chip used in the experiments, code-named NN-64, belongs to a family of programmable analog neural network chips developed at JPL [7] [8]. The chip consists of 64 neurons, each with 64 digitally programmable synapses and performs analog processing on analog input signals. The synapses have analog inputs received from chip inputs or from other neurons on the chip, which they multiply (using a multiplying DAC) with a digital weight, providing an analog signal to the somatic level. At the somatic level the analog contributions of the synapses are summed and passed through a sigmoid non-linearity, providing analog neural outputs. Signal processing from synaptic input to neural output takes ~250ns, while reprogramming the weights requires loading in rows of 64, 8 bits at a time, 64 rows for the full chip (or in random access). Loading at 33 MHz takes less than 2 microseconds per neuron, and about 120 microseconds for the full chip; the speed in the current setup where the download is controlled by software is about 3 orders of magnitude lower.

**Test 1**. The purpose of this test was to evolve on-chip a neural functional approximation. A feedforward, three layer 5-3-1 network was used to learn a simple function of one variable. The target was a bell shape Gaussian response at a linear increasing input. The genome was 23 bytes long, coding the values for the 23 8-bit synaptic weights. Each neuron was pre-biased to have a 2V output in the absence of the input signal. The fitness function was determined based on the sum of the squared errors between the calculated target function, and the circuit response, as measured at 15 input values. We used the Population-Based Incremental Learning (PBIL), an algorithm that "evolves" a probability vector that biases a randomized generation of bit strings representing candidate solutions, and has been found to be competitive with genetic algorithms for a wide variety of optimization problems [9]. A population of 200 networks was evolved for 160 generations. The result can be compared to the target in Fig. 1 (left). The response at a ramp signal is illustrated in the oscilloscope caption in Fig. 1 (right).

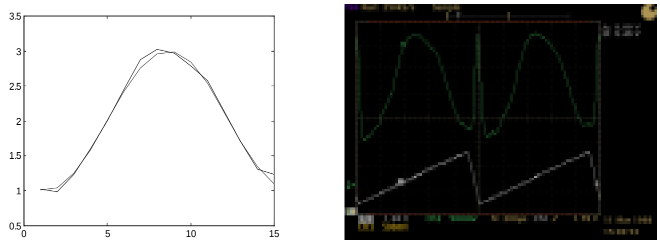

**Fig. 1.** A function learned on the chip (intrinsic EHW): (left) closeness to target; (right) response on the oscilloscope.

**Test 2.** The purpose of this test was to evolve visuo-motor tracking behavior for a mobile robot. A single neuron mapped low-resolution visual images to steering controls. The video input was preprocessed to provide a low-resolution, 3x3 image. The neuron output was a value in the [-1, 1] interval. In terms of steering controls [-1, 1] mapped to [-90, 90] degrees turn in respect to robot's frontal direction (-90 signifying a 90 degrees anti-clockwise turn, +90 signifying a 90 degrees clockwise turn, etc.). A training set collected in a human-controlled driving session was simplified to obtain 12 training patterns like the ones shown in Fig. 2 (input: pixel image, output: steering value). Evolution took place on the chip (intrinsic), the fitness being measured against the stored desired behavior (stored training set). The problem can be seen as evolving the weights for a 9 to 1 neural function approximation. Again, we used PBIL with a population of 200 individuals for 160 generations. The resulting neural controller had an approximation error below 5% (on the training set), which proved sufficient for driving the robot around the track.

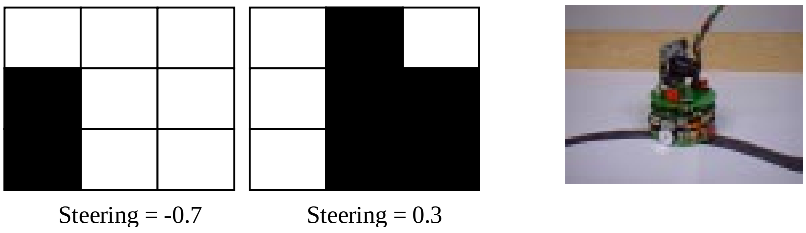

**Fig. 2.** Examples from the training set used for learning, and Khepera robot following a marked trail.

## 4 Evolution of dynamical systems in state-space representations

The behavior of systems, including electronic systems, can be described in terms of an analog descriptive language. Different levels of design abstractions appear in an analog modeling hierarchy (see for example [10]): primitive (device), functional (macromodel), and behavioral (high-level language description) level. The representations commonly used for evolving hardware are primitive or functional. The approach briefly exposed here, and treated in more detail in [11], relates to a behavioral description: a state-space representation expressed by differential equations. Moreover, an intrinsic evolution is proposed, using specially designed hardware that implements this representation: an analog computing machine, which we built and tested in a simple prototype form. In brief the representation we refer to is the state-space representation:

$$\dot{\vec{q}}(t) = \vec{f}(\vec{q}(t); \vec{x}(t))$$
$$\vec{y}(t) = \vec{g}(\vec{q}(t); \vec{x}(t))$$

where x(t) is a vector of continuous signal values coming into the system, y(t) is a vector of continuous output signal values, and q(t) is a vector of continuous internal state values, the “memory” of the system. The functions f() and g() are vector valued and in general non-linear. Figure 3 illustrates an example of the equivalence between a circuit in its schematic description and the state-space representation, graphically displayed by drawing the vector field f().

The prototype programmable analog computer implements with enough flexibility the description in terms of differential equations. Fast context switching allows the state-space of a dynamic system to be decomposed into a lookup-table of smaller vector representations. A search is employed in terms of modifications of the vector field towards a target that ensures certain optimality. This technique called the “modeling clay” approach to bio-inspired hardware is described in detail in [11].

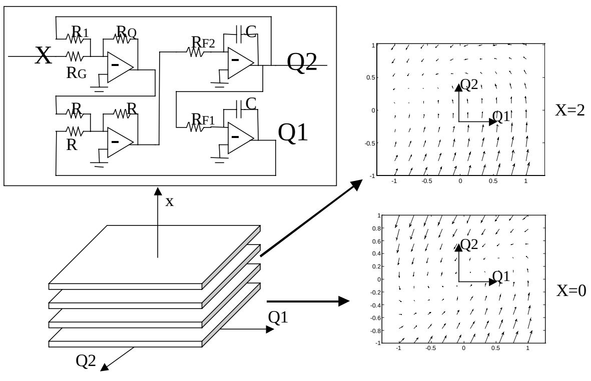

**Fig. 3** An active filter circuit from [12] and the three-dimensional state-space of the circuit dynamics with two Q1xQ2 vector field planes plotted at different points along the x-axis

## 5 Evolution of algorithms for on-board signal processing: results in lossless compression

In space applications compression is necessary in order to enable the downlink of massive amounts of science data (images). (The application of EHW for image compression was pioneered by Salami et al [13]). Because image compression is extremely computationally intensive, a low-power, fast, hardware implementation of a compression algorithm is desirable. An EHW system could be used to automatically generate a hardware-based image compression algorithm specially adapted for the class of images captured by the spacecraft. Image compression for space communications can be approached both in intrinsic and extrinsic EHW mode. For example, suppose a deep space probe needs to send thousands of similar images (e.g., atmospheric images) from the mission target (say, Pluto) back to Earth. The spacecraft could send several exemplar images back to the ground, where an FPGA configuration adapted for the class of images is evolved and uploaded to the spacecraft (extrinsic EHW). Alternatively, the spacecraft could evolve image-specific compression strategies directly using on-board hardware (intrinsic EHW). The work presented in the following relates to the first alternative. A nonlinear model used by the compression algorithm is evolved, which can be then compiled to an FPGA configuration, and finally downloaded (up-link to the spacecraft) to the real FPGA.

A genetic programming (GP) system was developed to perform adaptive image compression based on predictive coding. Predictive coding uses a compact model of an image to predict pixel values of an image based on the values of neighboring pixels. A model of an image is a function model(x,y), which computes (predicts) the pixel value at coordinate (x,y) of an image, given the (known) values of some neighbors of pixel (x,y). Typically, when processing an image in raster scan order (left to right, top to bottom), neighbors are selected from the pixels above and to the left of the current pixel. To complete the compression, the error image (the differences between the predicted pixel value and the actual pixel value) is compressed using an entropy coding algorithm such as Huffman coding or arithmetic coding. If we transmit this compressed error signal as well as the model and all other peripheral information, then a receiver can reconstruct the original image by applying an analogous decoding procedure.

The GP system evolves s-expressions that represent nonlinear predictive models for lossless image compression. The error image is compressed using a Huffman encoder. Because the computational cost of evolving nonlinear predictive models using standard GP systems would be prohibitively expensive, we have implemented a highly efficient, genome-compiler GP system which compiles s-expressions into native (Sparc) machine code to enable the application of GP to this problem. The terminals used for genetic programming were the values of the four neighboring pixels (Image[x-1,y-1],Image[x,y-1], Image[x+1,y-1], Image[x-1,y]), and selected constant values: 1, 5, 10, 100. The functions used were the standard arithmetic functions (+,-,*, %), and MAX/MIN (which return the max/min of two arguments). A detailed presentation of this system and of the results obtained is reported in [14].

The system was evaluated comparing the size of the compressed files with a number of standard lossless compression algorithms on a set of gray scale images. The images used were science images of planetary surfaces taken from the NASA Galileo Mission image archives. The compression results (file size) of the following algorithms are shown in Table 1:

- evolved - the evolved predictive coding compression algorithm.
- CALIC - a state-of-the art lossless image compression.
- LOCO-I - recently selected as the new ISO JPEG-LS (lossless JPEG) baseline standard.
- gzip, compress, pack - standard Unix string compression utilities (*gzip* implements the Lempel-Ziv (LZ77) algorithm, *compress* implements the adaptive Lempel-Ziv-Welch (LZW) algorithm, and *pack* uses Huffman coding).
- szip - a software simulation of the Rice Chip, the current standard lossless compression hardware used by NASA.

It is important to note that in our experiments, a different model was evolved for each image. In contrast, the other approaches apply a single model to every image. Thus, the time to compress an image using the genetic programming approach is several orders of magnitude greater than the time it takes to compress an image using other methods. (This may be reduced if one can evolve models that perform well for a class of images, as opposed to models specialized for individual images). However, the time to decompress an image is competitive with other methods.

**Table 1.** Compression ratios of various compression techniques applied to set of test images.

| Image Name | Original size | evolved | CALIC | LOCO-I | Com-press | gzip | pack | szip |
|---|---|---|---|---|---|---|---|---|
| Earth | 72643 | 30380 | 31798 | 32932 | 42502 | 40908 | 55068 | 40585 |
| Earth4 | 11246 | 5513 | 5631 | 5857 | 7441 | 6865 | 8072 | 7727 |
| Earth6 | 20400 | 9288 | 10144 | 10488 | 11339 | 10925 | 13264 | 12793 |
| Earth7 | 21039 | 10218 | 11183 | 11476 | 13117 | 12520 | 15551 | 13269 |
| Earth8 | 19055 | 9594 | 10460 | 10716 | 11699 | 11350 | 13298 | 12465 |

The results obtained show that for science data images, an evolvable-hardware based image compression system is capable of achieving compression ratios superior to that of the best known lossless compression algorithms.

## 5 Summary

In this paper we have discussed characteristic aspects of space-oriented evolvable hardware, and identified a set of specific applications. The paper contains the first reported intrinsic analog EHW results. A novel approach to EHW based on a representation of systems in terms of state-space was introduced. An EHW based image compression system was described, which achieves compression ratios superior to that of the best known lossless compression algorithms.

**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. The authors wish to thank Drs. A. Thakoor, T. Daud, B. Toomarian, S. Thakoor, C. Assad for the ideas shared during discussions on evolvable hardware, and to the anonymous reviewers for their comments and suggestions.

## 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. Grimbley, J. B. Automatic Analogue Network Synthesis using Genetic Algorithms, 1st IEE/IEEE Conf: Genetic Algorithms in Engineering Systems, UK, 1995
3. Koza, J., Bennett III, F. H., Lohn J., Dunlap, F., Keane M. A., and Andre, D. “Automated Synthesis of Computational Circuits Using Genetic Programming”. In Proc. of Second Annual Genetic Programming Conference, Stanford July 13-16, 1997
4. Hemmi, H., Hikage, T. and Shimohara, K. AdAM: A Hardware Evolutionary System , In Proc. of ICEC, (193-196), 1997
5. Thompson, A. Silicon Evolution. In: Proceedings of Genetic Programming 1996 (GP96), J.R. Koza et al. (Eds), pages 444-452, MIT Press 1996
6. Higuchi, T., Murakawa, M., Iwata, M., Kajitani, I., Liu, W. and Salami, M. , “Evolvable Hardware at Function Level.” *In Proc. of ICEC*, (187-192), 1997
7. Duong, T. A. et al., "Learning in neural networks: VLSI implementation strategies," In: Fuzzy Logic and Neural Network Handbook, Ed: C.H. Chen, McGraw-Hill, 1995
8. Eberhardt, S. et al, "Analog VLSI Neural Networks: Implementation Issues and Examples in Optimization and Supervised Learning," IEEE Trans. Indust. Electron. v39 (6):p. 552-564, Dec. 1992.
9. Baluja. I. Genetic Algorithms and Explicit Search Statistics. In Advances in Neural Information Processing Systems 9. Proceedings of the 1996 Conference. 1997. p.319-25
10. 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
11. Hayworth, K., The “Modeling Clay” approach to bio-inspired electronic hardware, To appear in Proc. ICES98, 1998.
12. Horowitz, P., Winfield, H.: The Art of Electronics 2nd ed Cambridge Univ. Press 1989
13. Salami, M., Murakawa, M., Higuchi, T., Data compression based on evolvable hardware, Proc. Evolvable Systems Workshop, International Joint Conference on Artificial Intelligence, 1997
14. Fukunaga A, Stechert A. Evolving nonlinear predictive models for lossless image compression with genetic programming. To appear in Proceedings of 3rd Annual Genetic Programming Conference (GP-98) , Madison, Wisconsin USA , July 22 – 25, 1998