# EHWPack: a Parallel Software/Hardware Environment for Evolvable Hardware

**Didier Keymeulen, Gerhard Klimeck, Ricardo Zebulum, Yili Jin*, Adrian Stoica and Carlos Salazar-Lazaro**

Jet Propulsion Laboratory
*California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91101
Email: didier.keymeulen@jpl.nasa.gov

## Abstract

This paper describes the EHWPack development system, a tool that performs the evolutionary synthesis of electronic circuits, using the SPICE simulator and the Field Programmable Transistor Array hardware (FPTA) developed at JPL. EHWPack integrates free and commercial software packages such as PGAPack for the evolutionary algorithm, Spice for the circuit evaluation, Tcl-Tk for the graphic interface, and LabView for the hardware evaluation. The paper investigates the performance of the tool in two typical problems of EHW: evolutionary synthesis of a Gaussian computational function and the evolution of a band-pass filter.

## 1 INTRODUCTION

This paper presents the tool EHWPack, a multi-tasking parallel software package targeted for Evolvable Hardware (EHW) experiments. EHW involves the investigation of the automatic synthesis of electronic circuits through evolutionary systems (Thompson, 1998). This field of research constitutes a new approach for automatic circuit design, where the process of circuit conception is interpreted as a search task. The use of evolutionary systems as a search technique provides an efficient way to sample the large search space associated with problems of electronic circuit synthesis. EHW can afford the synthesis of novel circuits that are comparable to the human designed ones in terms of such aspects as size, noise, power consumption and others (Koza, 1998) (Zebulum, 1999). Nevertheless, EHW experiments need powerful computer resources and evolution-oriented programmable devices.

The large search space associated with EHW applications requires the sampling of more individuals compared to ordinary applications of evolutionary computation. Extrinsic EHW, where circuit simulators are used to evaluate electronic circuits, needs non-conventional computer resources, such as multi-computers or multiprocessors to speed up the experiments. Circuit simulation is a very time consuming process, particularly in the case of analog circuits. For example, Bennett et al. built and used a parallel cluster of 1,000 Pentium processors for the evolutionary design of analog filters and amplifiers (Bennett, 1999).

Instead of using simulators, programmable chips (Stoica, 1999a) can be used to assess each circuit synthesized by the evolutionary algorithm (intrinsic approach). In this case, the circuits are downloaded into the reconfigurable chip and their performance is measured based on their actual behavior. Besides being used as a platform for evolution, these reconfigurable chips are important to validate solutions evolved in simulation. It may be noted, however, that commercially programmable chips are not particularly suitable for evolutionary applications (Zebulum, 2000). Evolution-oriented programmable devices must be included in the experimental setup.

For several reasons (including mismatches between models and physical HW, limitations of the simulator and testing system, etc.) circuits evolved in SW may not perform the same way when implemented in HW, and vice-versa. This *portability problem* limits the applicability of SW evolved solutions, and on the other hand, prevents the analysis (in SW) of solutions evolved in HW. A third approach to EHW called *mixtrinsic EHW (MEHW)* was presented in Stoica, 2000. In MEHW evolution takes place with hybrid populations in which some individuals are evaluated intrinsically and some extrinsically, within the same generation or in consecutive ones.

In order to handle the above issues, the EHWPack tool has been developed. The EHWPack is a distributed parallel software-hardware environment for evolutionary circuit design. It runs on the HP Exemplar Caltech parallel supercomputer and is remotely controlled from a local workstation. It has been developed to facilitate experiments, both in simulated as well as hardware evolution, using SPICE circuit simulator and a Field Programmable Transistors Array (FPTA) respectively

(Stoica, 1999a). The tool is used for the evolutionary synthesis, optimization and on-line adaptation (Keymeulen, 2000) of electronic circuits in *extrinsic*, *intrinsic* and *mixtrinsic* mode. Furthermore, it has also been used as a test-bed for new architectures of reconfigurable hardware (Stoica 1999b) and nano-electronic devices (Stoica, 1999c).

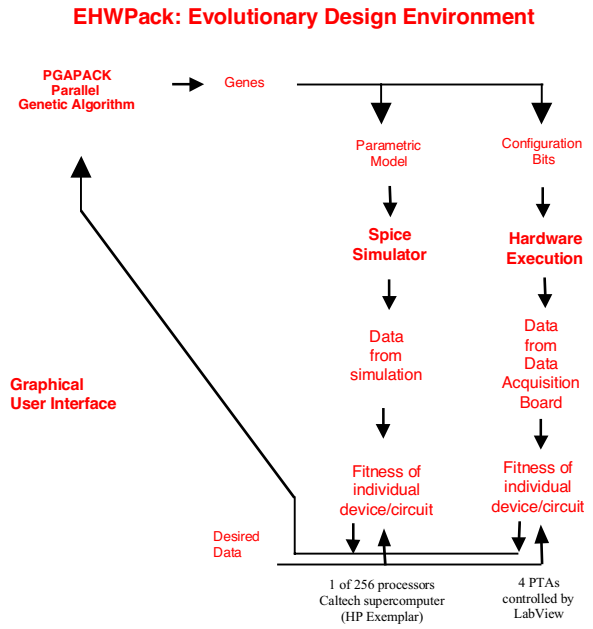


**Figure 1** - EHWPack Environment

Although a variety of evolution-based software environments have successfully been developed for evolutionary designs (Levine, 1994) (Heitkötter, 1997) (van Lent, 1999) (Wall, 1999) (Bennett, 1999), a tool like EHWPack was needed due to a number of factors: (a) the currently available evolutionary software packages implement general-purpose genetic algorithms running on various workstations and under different operating systems, but a dedicated genetic algorithm is needed for circuit design; (b) public domain software is available for genetic algorithm, circuit simulation, graphical interface, PC-board control and network communication, however no software integrates all these components into a single environment; (c) the genetic algorithm for circuit design using both software simulation and hardware implementation must be evaluated on a single platform; (d) the tool must be user friendly and transparent, such that experimentalists (not necessarily experts in software simulation on supercomputer) located at different sites can use it; and finally, (e) the evolutionary design of portable circuits can only be achieved by integrating results from software simulation and hardware execution in the same experimental environment (Stoica, 2000).

The paper is organized as follows: Section 2 describes the EHWpack environment. It starts with a general view of the system, followed by a description of its window interface and its implementation on the supercomputer. Section 3 presents an analysis of the EHWPack performance in terms of speed to evolve circuits. It begins with a short overview of the FPTA architecture, and shows time statistics associated with the synthesis of the Gaussian and a band-pass filter tuned to the AM band. The section concludes with a comparison of the intrinsic and extrinsic evolution and an illustration of the circuits achieved by EHWPack for the Gaussian and band path filter. The conclusions are discussed in section 4.

## 2 EHWPACK ENVIRONMENT

EHWPack implements the three main steps of an evolutionary design of electronic circuits. In the first step, a population of chromosomes is randomly generated and the chromosomes are converted into circuit models (for extrinsic EHW) or control bit strings, downloaded to programmable hardware (intrinsic EHW). In the second step, circuit responses are compared against specifications of a target response, and individuals are ranked based on how close they come to satisfying it. In the third step, preparing for a new iteration loop, a new population of individuals is generated from the pool of best individuals in the previous generation. Some individuals are token as they were and some are modified by genetic operators, such as chromosome crossover and mutation. The process is repeated for many generations, which results in increasingly better individuals. The process is usually stopped after a number of generations, or when the closeness to the target response has been reached to a sufficient degree. One or several solutions may be found among the individuals of the last generation.

In its current implementation, the tool uses the public domain Parallel Genetic Algorithm package, PGAPack, (Levine, 1994) a public domain version of SPICE 3F5 as circuit simulator and a FPTA (Stoica, 1999a) evolvable hardware test bed built around LabView (figure 1). The FPTA is an array of transistors interconnected by programmable switches and will be discussed in details later. An interface code links the GA with the circuit simulator and with the hardware where potential designs are evaluated, while a graphic user interface (GUI) allows easy problem formulation.

At each generation, the genetic algorithm (GA) produces a new population of binary chromosomes, which get converted into voltages in netlists that describe candidate circuit designs, and into configuration bits for the FPTA devices. Netlists are further simulated by SPICE while configuration bits are downloaded into the hardware device by LabView. The output signals are compared with the target and the fitness is sent back to the GA. The process continues until the output response is close to target output.

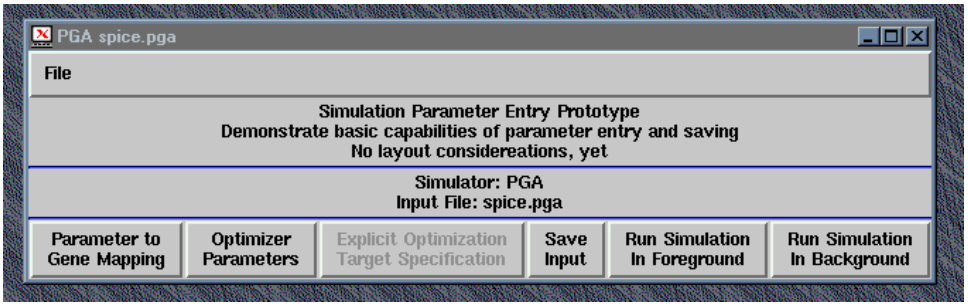


**Figure 2** - EHWPack Interface Main Window

## 2.1 INTERFACE

A graphical interface remotely controls the evolutionary process running on the supercomputer and the reconfigurable hardware. The user specifies on his local workstation the parameters of the evolutionary design through a series of windows. After calling the EHWPack tool from the user local workstation, the main window, shown in figure 2, appears in the user's screen. By clicking the buttons on the main window, the user will be connected to a series of sub-windows to graphically set up the PGAPack parameters, SPICE parameters, and supercomputer parameters, before starting the execution on the supercomputer or the reconfigurable hardware. In the next paragraphs we review some of the windows.

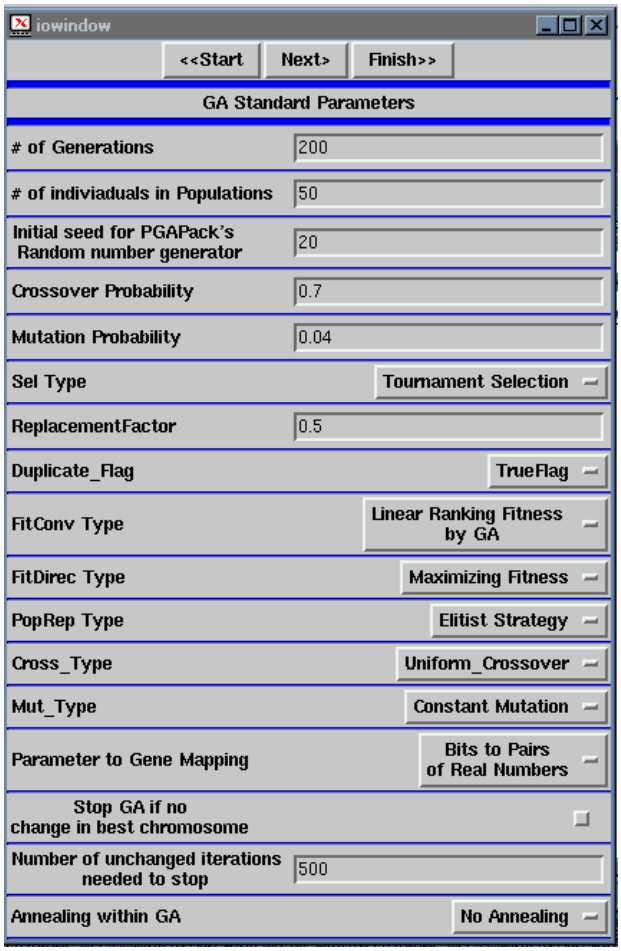


**Figure 3** - EHWPack Interface: GA parameters

The first window obtained by clicking on Parameter to Gene Mapping button, allows the user to specify the high and low values for the gene associated with a substitute string in the SPICE netlist. For example, the genes are associated with resistance strings in the netlist, which represent the behavior of opened and closed switches. The user has also the possibility to include or take genes away. The netlist should be changed in accordance with the substitute strings added or deleted.

By clicking on the Optimizer Parameters button, the interface runs through five windows where the user selects the optimization method (Michalewicz, 1999) (genetic algorithm in the current version, Tabu search (Glover 1997) and simulated annealing in later versions), the fitness function (mean square error or control points), set up the parameters of the GA, the parameters of the SPICE simulator and the names of the result text files where the EHWpack output is saved. We show in figure 3 the parameters of the genetic algorithm.

Once the user has entered the optimizer parameters, he saves all the settings in a file by clicking the Save Input button. Finally, the execution of the EHWPack is launched in interactive mode (Run Simulation in Foreground) or in batch mode (Run Simulation in Bacground). The execution in foreground runs on one processor and is used for debugging. When the execution runs in background, the user must specify the number of processors and the estimated time of the job. When the job has started on the supercomputer or on the reconfigurable hardware, a window pops up showing the intermediate fitness of the best individual. In the current stage of the EHWPack, the outcome of the experiment is a text file that can be read by other visual tools, not integrated into the package.

## 2.2 IMPLEMENTATION

The EHWPack was implemented on the HP Exemplar shared-memory supercomputer at the California Institute of Technology with 256 CPUs and 64 GB of memory. The exemplar is composed of 16 nodes, each node having 16 processors as shown in figure 4. One node, the single-node system sub-complex, is dedicated to the users' login sessions and compilation jobs. All the other nodes are

reserved for batch or interactive jobs requiring input-output interface with the user during execution.

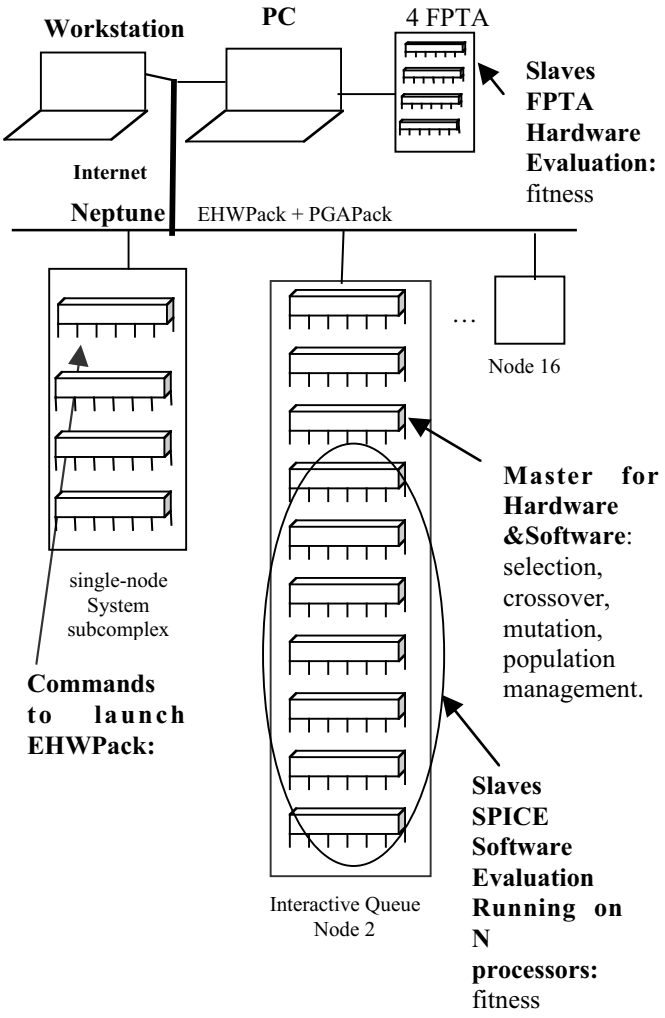


**Figure 4**. - Software and Hardware Implementation

The parallel programming model used by EHWPack is the one implemented by the PGAPack. It is a message passing model, in particular a Single Program Multiple Data (SPMD) model using the Message Passing Interface (MPI) standard implemented by the Neptune software libraries. The EHWPack parallel implementation uses a master/slave algorithm, in which one process, the master, executes all steps of the GA except the function evaluations (SPICE simulation or on-chip evaluation). The function evaluations are executed by slave processes. The master is running on one processor of the HP exemplar and the slaves are running on the other processors allocated for the job or on the FPTA hardware through an internet communication.

The hardware configuration is a board mounted with four FPTAs. The board is controlled by National Instruments data acquisition hardware and software (LabView). The LabView Software implements a TCP/IP client-server system where the server is the LabView and the master processor running on neptune is the client. LabView receives the configuration bits from the master and returns the sampling data of the output responses back to the master.

The graphical display is executed on a local UNIX workstation through a Xwindows interface.

## 3 EXPERIMENTS/PERFORMANCE

This section presents an analysis of the EHWPack performance in terms of speed of circuit evolution. We start by reviewing the basic features of the architecture of the FPTA.

### 3.1 FPTA

The FPTA cell 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. Figure 5 illustrates the schematic of the FPTA cell consisting of 8 transistors and 24 programmable switches.

In this implementation, 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 arrangements, and yet less than the total number of possible connections. To offer sufficient flexibility, the cell has all transistor terminals connected via switches to expansion terminals which allows the implementation of bigger circuits by cascading FPTA cells.

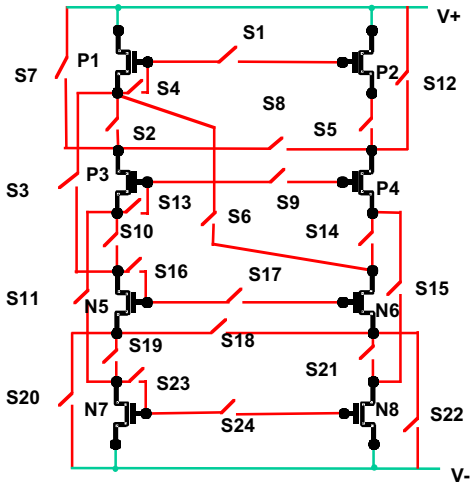


**Figure 5** – Schematic of an FPTA cell

The FPTA cell was manufactured using 0.5 micron CMOS technology. The chip allowed us to use circuits obtained through evolution in simulations and validate them by downloading and evaluating their performance in hardware.

### 3.2 SPEED ANALYSIS

The objective of this investigation is to assess the performance of the tool in two typical problems of EHW: the evolutionary synthesis of a Gaussian computational function (Stoica, 1999b) and the evolution of a band-pass filter tuned to the AM band. The fitness value is the mean square error between the desired DC/AC signals and the DC/AC signals obtained by the circuit. We investigate the impact of varying the number of FPTA cells and the number of processors in the elapsed time.

We start by describing extrinsic EHW experiments, where the SPICE simulator is used to evaluate the circuit performance. In this particular case a netlist file can be selected by the user from the Graphical User Interface of EHWPack.

Table 1 shows time statistics associated with the synthesis of the Gaussian. We compare the tool speed when using 1, 16, 32, 64 and 128 processors of the supercomputer. We also vary the number of FPTA cells used in the netlist: 1, 2 and 4 cells.

| # Cells | Pop. x Gen. | #Proc | Elapsed Time (sec) | Norm. Time (msec) | #Trans |
|---|---|---|---|---|---|
| 1 | 50 x 200 (25) | 1 | 407 | 81.4 | 8 |
| | | 16 | 56 | 11.2 | 8 |
| | | 32 | 46 | 9.2 | 8 |
| | | 64 | 49 | 9.8 | 8 |
| | | 128 | 63 | 12.6 | 8 |
| 2 | 400 x 200 (200) | 1 | 5860 | 146.5 | 16 |
| | | 16 | 491 | 12.3 | 16 |
| | | 32 | 321 | 8.0 | 16 |
| | | 64 | 240 | 6.0 | 16 |
| | | 128 | 222 | 5.6 | 16 |
| 4 | 100 x 200 (50) | 1 | 2971 | 297.1 | 32 |
| | | 16 | 288 | 28.8 | 32 |
| | | 32 | 184 | 18.4 | 32 |
| | | 64 | 146 | 14.6 | 32 |
| | | 128 | 154 | 15.4 | 32 |

**Table 1** – EHWPack performance in the evolution of a Gaussian circuit . *Pop* is the population size, *Gen* is the number of generations; *#Proc* stands for the number of processors; *Norm. Time* is the normalized time; and *#Trans* gives the number of transistors in the netlist.

The table 1 shown above provides the following information: number of FPTA cells; number of processors; population size; number of generations; elapsed time; normalized time; and the number of transistors in the SPICE netlist. The elapsed time is the total time taken to run the experiment. The normalized time is the elapsed time divided by the number of individuals evaluated (half of the product population x generations shown in the above table, since we replace only 50% of the population at each generation). The number of individuals evaluated per generation is given within parenthesis in the second column. As shown in the table, we used different population sizes in the experiments in order to assess the influence of this parameter in the performance. As expected, the normalized time decreases as we increase the number of processors. We can further observe that, as we increase the number of FPTA cells and, as a consequence, the number of transistors in the netlist, the normalized time increases as well. Finally, the fitness of the best circuit using 1,2 or 4 cells and running on 1, 16 to 128 processors are the same.

It is interesting to observe that for small populations (50 individuals in the single cell experiments), increasing the number of processors improves the speed only to a certain extent: for more than 32 processors, the communication overhead produces an increase in the time for evolution. This stems from the fact that using the master/slaves algorithm only 25 individuals are evaluated in parallel per generation. In order to fully explore the parallelism, large populations should be used, as in the experiment with two cells (400 individuals). In this case, 128 circuits can be evaluated in parallel on 128 processors. We can see from Table 1 that, for more than 32 processors, the time for evolution in the two-cell experiment is smaller than the one observed in the single-cell experiment. For maximum efficiency, we should use the number of individuals evaluated at each generation equal to the number of processors used.

Table 2 is analogous to Table 1, displaying the evolution time for the synthesis of a band-pass filter. While the Gaussian circuit requires a DC transfer analysis, filters require the use of the frequency domain analysis of the SPICE simulator.

| # Cells | Pop x Gen | #Proc | Elapsed Time (sec) | Norm. Time (msec) | #Trans |
|---|---|---|---|---|---|
| 1 | 200 x 200 (100) | 1 | 1445 | 72.3 | 8 |
| | | 16 | 144 | 7.2 | 8 |
| | | 32 | 101 | 5.1 | 8 |
| | | 64 | 72 | 3.6 | 8 |
| | | 128 | 79 | 4.0 | 8 |
| 2 | 200 x 200 (100) | 1 | 2034 | 101.7 | 16 |
| | | 16 | 214 | 10.7 | 16 |
| | | 32 | 145 | 7.3 | 16 |
| | | 64 | 115 | 5.8 | 16 |
| | | 128 | 94 | 4.7 | 16 |
| 4 | 200 x 200 (100) | 1 | 4895 | 244.8 | 32 |
| | | 16 | 402 | 20.1 | 32 |
| | | 32 | 250 | 12.5 | 32 |
| | | 64 | 146 | 7.3 | 32 |
| | | 128 | 119 | 6.0 | 32 |

**Table 2** – EHWPack performance in the evolution of a bandpass filter.

In the case of the filter experiment, we kept the number of evaluated individuals constant, so that we can clearly observe the increase in evolution time as we include more cells in the netlist. In this case also the fitness of the best circuit using 1, 2 or 4 cells and running on 1, 16 to 128 processors are the same. If we compare the statistics shown in Tables 1 and 2, it can be observed that the filter experiment consumes less time than the Gaussian experiment. This is consistent with the fact that the AC analysis is less time consuming than the DC analysis.

Further, we compare the statistics shown for the *extrinsic* evolution in Table 1 with the ones measured in an *intrinsic* experiment, where two FPTA cells were used as a hardware evolution platform. A total of 20 individuals have been evaluated along 200 generations. The elapsed time was 272 seconds. The total time spent in the TCP/IP connection between the supercomputer and the PC was 5 seconds. The computation time spent on the PC is 180 seconds and on the supercomputer is 87 seconds. We should mention that most of the evaluation time on the PC is used by LabView to download the configuration bits into the chip and to acquire the data from the chip (153 seconds). The time needed to obtain a DC transfer analysis of the circuit is reduced to 27 seconds. This value can be normalized, dividing it by the number of individuals evaluated (4000), giving 6.75 ms. This number is one order of magnitude less than that obtained for the simulated experiments with 2 FPTA cells and using only one processor, and it is equivalent to the time observed when using 128 processors. Another advantage of hardware evolution is that the elapsed time does not increase with the number of cells. We are currently working on an enhanced version of the chip that will integrate 36 FPTA cells. The chip will be mounted on a dedicated board with processor, memory and analog/digital converter to accelerate the chip reconfiguration and the data acquisition.

Finally, we illustrate two circuits achieved by EHWPack in these experiments. Figure 6A depicts the schematic of a band-pass filter evolved in the extrinsic experiment, using two cells. Figure 6B shows its frequency response. Figure 7A depicts the schematic of a Gaussian circuit evolved in hardware, using one FPTA cell. Figure 7B plots its DC transfer function.

# 4 CONCLUSIONS

A parallel evolutionary software/hardware environment, EHWPack, was developed around PGAPack, SPICE as a simulator, and FPTA as reconfigurable VLSI chip to facilitate experiments in simulated and hardware evolution on a single platform. It allows experimentalists located at different sites, to design, optimize and test circuits using evolutionary algorithms in a user friendly, transparent and expeditious manner. Using the EHWPack, we were able to synthesize a Gaussian computational function and a band-pass filter tuned to the AM band in less than 4 minutes using 1, 2 or 4 FPTA cells. The same job took 1 hour 30 minutes in simulation on a 1 processor. The experiments confirm that, to obtain maximum efficiency, the number of processors should be equal to the number of individuals evaluated at each generation. Finally, we observe that the time needed to evaluate one individual using the reconfigurable hardware (6.75 ms) is as fast as the time observed when using 128 processors and twenty times (2 FPTAs: 146.5 ms) to fifty times (4 FPTAs: 297.1 ms) faster that the SPICE simulator running on one processor.

**Acknowledgment**
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 Defense Advanced Research Projects Agency (DARPA) under the Adaptive Computing Systems Program and the National Aeronautics and Space Administration.

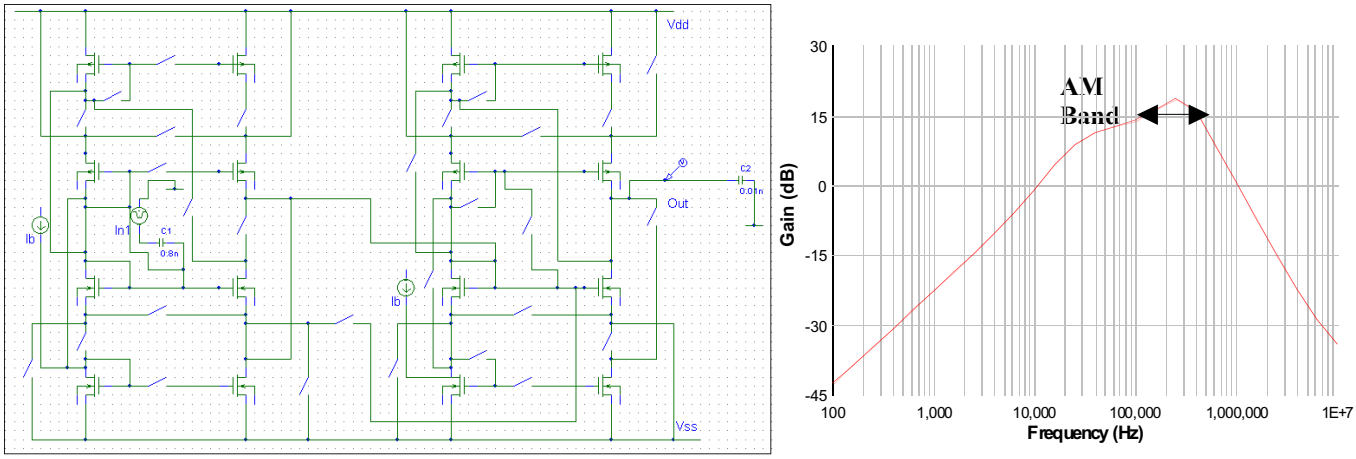


**Figure 6**: (A) – Schematic of the band-pass filter evolved in simulation using EHWPack; (B)- Frequency response

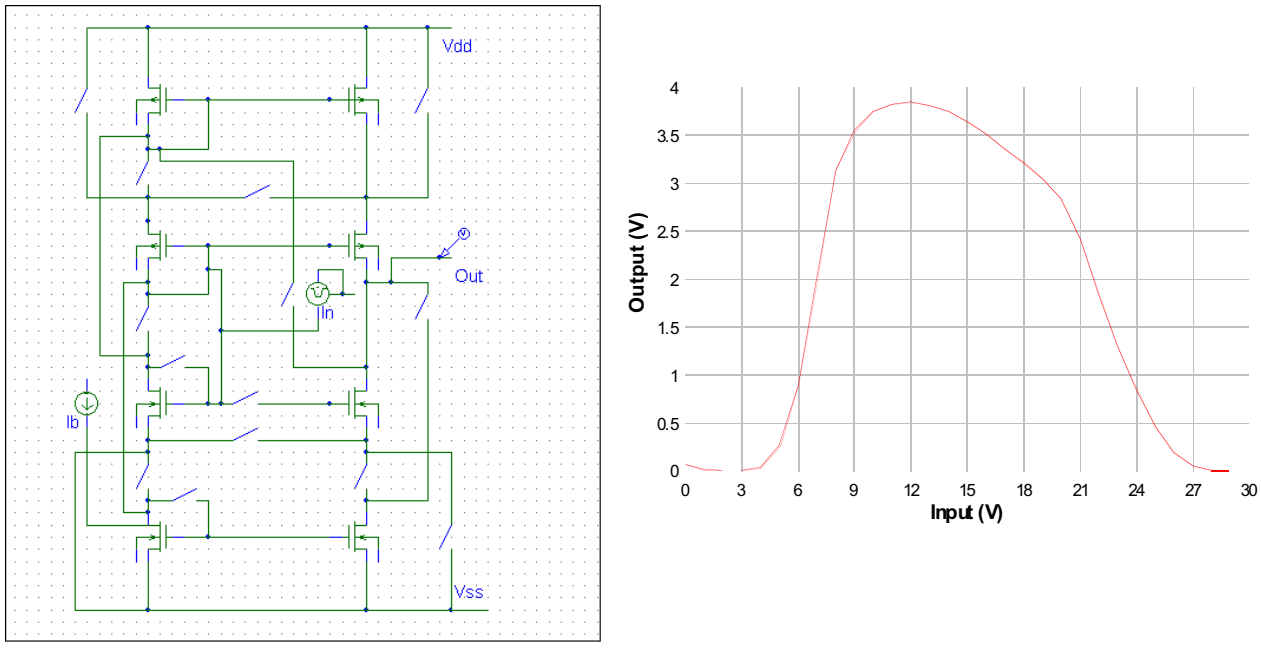


**Figure 7** - (A) – Schematic of the Gaussian circuit evolved in hardware using EHWPack; (B)- DC transfer response

**References**


Bennett, Forrest H III, Koza, John R., Shipman, James, and Stiffelman, Oscar. (1999). Building a parallel computer system for $18,000 that performs a half peta-flop per day. In. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, July 13-17, 1999, Orlando, Florida USA. San Francisco, CA: Morgan Kaufmann. Pages 1484 - 1490. (http://www.genetic-programming.com/machine1000.html)

Heitkötter, J. (1997), *The Hitch-Hiker's Guide to Evolutionary Computation*, at http://gnomics.udg.es/~encore/www/

D. Keymeulen, A. Stoica, R. Zebulum (2000). Fault-Tolerant Evolvable Hardware using Field Programmable Transistor Arrays. In *IEEE Transactions on Reliability, Special Issue on Fault-Tolerant VLSI Sytsems*, vol. 49, No. 2. IEEE Press

Koza, John R., Bennett III, Forrest H, Andre, David, Keane, Martin A, and Dunlap, Frank (1998), Automated synthesis of analog electrical circuits by means of genetic programming, In *Journal of IEEE Transactions on Evolutionary Computation*. 1(2). pp. 109-128.

Levine, D (1994), *PGAPack: Parallel Genetic Algorithm Library* http://www-unix.mcs.anl.gov/~levine/PGAPACK/index.html,.

Stoica, A (1999a). Toward evolvable hardware chips: experiments with a programmable transistor array. In *Proceedings of 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems,* Granada, Spain, IEEE Computer Science Press.

Stoica, A, et al. (1999b). Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digitl CMOS circuits. In, *Proceedings of the First NASA/DoD workhop on Evolvable Hardware*. Los Alamitos, CA: IEEE Computer Society Press.

Stoica, A et al. (1999c). Evolutionary Design of Electronic Devices and Circuit. In *Proceedings of the 1999 Congress on Evolutionary Conputation,* July 6th-9th, 1999, Washington DC. IEEE press.

Stoica, A, Zebulum R. and Keymeulen D. (2000) Mixtrinsic Evolution. In *Proceedings of the third International Conference on Evolvable Systems: from Biology to Hardware,* Edinburgh, Scotland, UK, April 17-19. Springer-Verlag.

Thompson, A. (1998). On the Automatic Design of Robust Electronics Through Artificial Evolution. In *Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES98)*, vol. 1478, pp. 13-24, LNCS, Springer-Verlag.

van Lent, M. (1999), *PARAGenesis: a Genesis on the CM-200 in C**, at ftp://ftp.aic.nrl.navy.mil/pub/galist/src/ga/paragenesis.tar.Z

Wall M. (1999), *C++ library for Genetic Algorithm*, at http://lancet.mit.edu/ga/

Whitley L. (1999), *Genitor is a modular GA package for floating-point, integer, and binary representations*, at ftp://ftp.cs.colostate.edu/pub/GENITOR.tar

Zebulum, R.S., Pacheco, M.A., Vellasco, M., (1999) Artificial Evolution of Active Filters: A Case Study, Proc. of the First NASA DoD Workshop on Evolvable Hardware, pp.66-75, Los Alamitos, CA: IEEE Computer Society Press.

Zebulum, R.S., Stoica, A., Keymeulen, D. (2000) A Flexible Model of a Field Programmable Transistor Array Targeted For Hardware Evolution, In the *Proc. of the Third International Conference on Evolvable Systems (ICES2000)*, Edinburgh, Scotland, April 17-20. Springer Verlag.

Glover, F. and Laguna M. (1997). *Tabu Search*. Boston, Kluwer Academic Publishers.

Michalewicz, Z and Fogel, D.B. (1999). *How to Solve It: modern Heuristics*. New York, Springer-Verlag.