Evolvable Hardware — The Coming Hardware Design Method?

Jim Torresen
NERA Telecommunications, P.O. Box 91, N-1361 Billingstad, NORWAY
E-mail: jim@nera.no

Abstract. This paper introduces a new hardware design scheme inspired by biological systems. The method is named Evolvable Hardware (EHW), since evolutionary algorithms are applied for the circuit design. The idea is that the system should consist of a large set of simple hardware units with connections only to their nearest neighbors. All the units operate concurrently with no global control. However, the local behavior of the complete system determines the global behavior of the system. The connections between the units and the operation of each individual unit are determined using genetic algorithms. Thus, the system is evolved until it attains a structure performing as initially specified.

This paper surveys the undertaken work on EHW. So far few experiments have been reported. One of the reasons to this is that little applicable hardware has been available. The experiments which have been reported are on simple problems. However, the results are promising and many have great expectations on the future of EHW.

1 Introduction

The traditional way of designing hardware has been by drawing schematics or by using a hardware description language. This involves many design considerations to be taken care of like timing constraints and possible glitches. Another possible approach is by *evolving* the circuit description. By this method, first a set of circuits are randomly generated. The behavior of each circuit is evaluated and the best circuits are combined to generate new and hopefully even better circuits. The evaluation is according to a behavior initially specified by the user. After a sufficient number of generations the fittest circuit is to behave according to the initial specification.

Natural evolution is based on Darwin's thinking about development where strong individuals survive and give rise to new generations. In relation to evolution, undertaken in a large population, we speak about collective behavior arising from simple interaction between individuals. One example of collective behavior is an ant society, where the global behavior emerges from local interactions between individual ants. These principles from the nature

inspire researchers to develop new architectures where the configuration is determined by evolution and with no global control of each building block. This in contrast to the general computers used today which are all based on global control mechanisms.

An artificial evolvable system may exhibit several behavior characteristics of natural living systems:

- Emergent behavior: Interaction between a large number of units with no global control determines the behavior of the entire system.
- Adaptation to the environment.
- Self-replication.
- Self-reparation.

Evolvable systems may solve these items better than today's systems. Another word that is used for this research field is Artificial Life, since biological systems are tried to be modeled in a computer.

Evolvable hardware (EHW) is programmable hardware that can be evolved. The idea of EHW was first introduced for about four years ago [5]. Little has happened through the years since then, thus, it is still quite open how evolvable hardware — if any, will be in the future. EHW is based on evolving the circuit configuration instead of designing it the ordinary way. That is, different randomly generated configurations are tested in a programmable hardware device. The configurations that make the device output responses closest to the wanted response are combined to make even better configurations until an usable device is achieved. There has so far been a couple of conferences on evolvable hardware, see [7] and [13]. A short introduction to evolvable hardware is given in [8] and [17].

The word evolution is also used in several other contexts, e.g. how hardware has developed. IBM-PC has evolved throughout the years where each type of 80x86 processor represents one generation. In this paper, this interpretation of evolvable hardware has been completely omitted. Further, note the difference between programming an ordinary microprocessor and programming hardware. The former is based on specifying a set of instructions on fixed hardware, while the latter is based on designing the hardware structure from e.g. gate level. Programmable hardware is often called *reconfigurable* hardware.

The evolution can also be undertaken completely in software. Today, software simulation is by far the most used method, since little EHW has been available.

A large range of applications have been proposed for EHW:

- Autonomous robots:
  - Vacuum cleaners

  - Surveillance robots
  - Automatic De-mining Vehicles
  - Maintenance robots
- Pattern recognition
- Signal processing

One hopes that EHW can solve the difficult problems in these applications better than other methods like ordinary artificial neural networks. Autonomous robots require such complex systems that it has not yet been possible to develop functional systems. By evolving a system instead of designing or programming it, the complexity can be highly increased. To use such a system in real time, special hardware is required.

Evolvable hardware has several advantages:

- Fault tolerant: The system can change its own structure in the case of hardware error.
- The designers abstraction like the synchronous digital model can be abandoned and the dynamics of the reconfigurable hardware can be fully exploited. By trial and error through evolution, a design that performs as specified is found. The knowledge about how the device is internally configured is not required by the user.
- The principle of local interaction instead of global control leads to unlimited scaleability of the system.
- If some objective fitness function can be derived for any complex system, there is a possibility of automatic evolution of the system without explicit design.

The last item in the list is one of the main motivations for the EHW research. Both circuit speed and complexity increases and a limit for designability in the ordinary way may soon arise. Then, other ways of programming like hardware evolution must be investigated.

For hardware evolution to be an applicable design method it must give a better solution — and preferably be faster, compared to software simulations. Furthermore, it is essential to create an evolutionary framework and/or computational mechanisms to guide hardware evolution to fully utilize the hardware device. The design tool must be designed to in the best way utilizing the EHW. Implementation of artificial evolution schemes must be developed such that it is inspired by nature, but suited to the facilities available. The differences between biological systems and the available technology must be accepted. It should be emphasized how the technology can be best utilized instead of copying the biological system. Biological neural networks are highly connected in 3D but slow, while VLSI is fast but with limited connectivity.

1.1 Genetic Algorithms

Genetic Algorithms (GA) are the most commonly used algorithms for evolutionary computing [9]. The algorithm is illustrated in Figure 1.

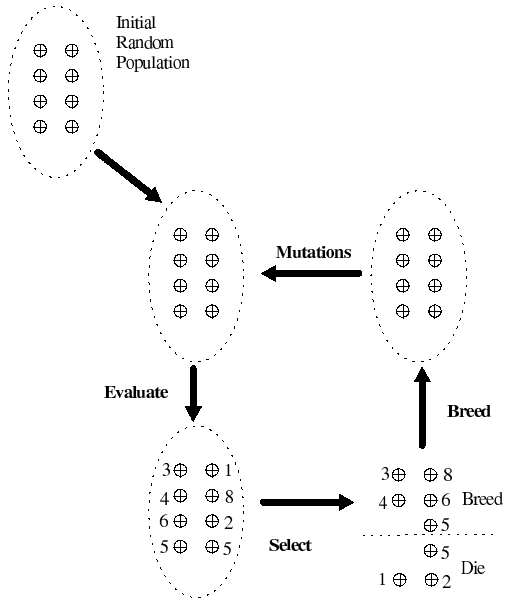

Fig. 1. The genetic algorithm principle.

In GA, the members of a population are initially randomly generated. A member of the population may contain a program, hardware configuration or other kinds of representation and is often called a chromosome or a genotype. The members are evaluated and sorted on fitness according to a given fitness criteria. This is usually a high-level function and may be e.g. the deviation from a specified behavior of a robot. The parameters of chromosomes of the fittest members are exchanged to generate — for each couple, two new offsprings – preferably fitter than the parents, in the next generation. In the figure, no parents are transferred to the next generation. Another variant of GA combines a few of the best members, and the children are exchanged with the lesser fit members of the population. Mutation may also occur and introduces changes in the chromosomes that may not be in the parents.

Evaluation of each member is usually the most time consuming operation

of GA and is the one first to be undertaken in special hardware. GA was initially used for optimizing problems. Later it has also been applied for adaptive systems for robot control.

1.2 Methods in Evolvable Hardware

Hardware evolution can be divided into two sub-groups:

- **Off-line EHW:** The evolution is simulated in software, and only the elite chromosome is written to the hardware device (off-chip evolution).
- **On-line EHW:** The hardware device gets configured for each chromosome for each generation (on-chip evolution). Thus, the genetic operations are done in software, while evolvable hardware is used to test the fitness of each member of the population.

In the future, one may design devices which allow the complete evolution on-chip. Then, it is possible to continue the evolution when the device is operating in its working environment. However, in today's systems the evolution is undertaken once with no continued evolution while the device is in use.

So far, all systems are based on one or a few circuits. In the future the systems can become more complex, where sub-systems are separately evolved and then put together to a larger system. For hardware evolution, building blocks are required and so far gate level (AND,OR,..) is most commonly used, while higher level functions are possible too.

1.3 Technology for EHW

Possible evolvable medium:

- Wetware: Real chemical compounds are used as building blocks.
- Nanotechnology: Molecular scale engineering.
- Silicon hardware:
  - Field Programmable Gate Arrays (FPGA)
  - Custom Application Specific Integrated Circuit (Custom ASIC)
  - Wafer-Scale Integration
- General purpose computer

Much research is going on in the fields of wetware and nanotechnology. However, with few results so far. The most available technology for EHW today is probably FPGA devices. This is programmable chips, where a string of configuration bits determines the internal circuit and behavior of the device. It will be detailed later in this paper.

ASIC is user-specified circuits specially designed for evolution. If the system is to be scaleable to evolve a large network of units, software simulations on a general purpose computer is not an alternative.

2 A survey of EHW Research

This following sections contain a survey of various approaches to evolvable hardware. Most of the research on evolutionary techniques are undertaken using program simulations. It is to be expected that some of these approaches will be implemented in special purpose hardware in the future. Thus, as an introduction, some of the most used evolutionary programming methods will be shortly described. Then, the experiments using hardware evolution is presented.

2.1 Evolution in Software

The main research projects where no special hardware is used:

- **Lindenmayer-systems** (L-systems): A rewriting technique for defining complex objects by successively replacing parts of a simple initial object (mainly used in computer graphics to generate plants).
- **Tierra.** Evolution and optimization of self-replicable computer programs, T. Ray.
- **Genetic Programming.** Computer programs are genetically bred to solve problems, J. Koza.
- **Sub-sumption architecture**, Control of autonomous mobile robots based on layers of simple finite state machines, R.A. Brooks, J. Koza.
- **Cellular Automata.** J. von Neumann, E.F. Codd, H. Garis.
- **Simple artifacts**. A computational modeling method for adaptive self-organization, J. Vario.

The difference between Tierra and Genetic Programming is that the latter is used to solve a specific problem, while the first is used to study aspects of artificial evolution in general.

Cellular Automata (CA) originates back to the 1960's when John von Neumann[1] proposed a self reproducing machine based on simple CA cells. The purpose was to design a machine to cope with the problems of extremely large machines. The machine characteristics were:

- Self-reproduction
- A grid of cells each residing a finite state machine.
- Each cell can only be affected by its four nearest neighbors.

[1] John von Neumann is the originator of the architecture employed today in almost all general purpose computers.

The proposed machine showed to be too complex to be designed. However, the principles were continued by Codd who introduced a simplified Cellular Automata Machine [1]. The system consists of a large number of CA cells. Each cell can be in a one of a set of states. The cell changes states in a synchronous way based on its own state and the state of its neighbors — see Figure 2.

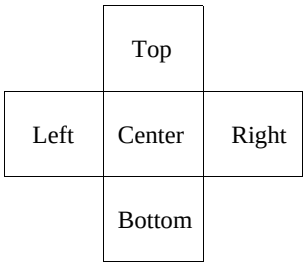

Fig. 2. A CA cell with its four nearest neighbors.

The machine proposed by Codd is the basement for several research projects going on at the moment where the goal is to evolve complex systems based on CA cells. One of these project, going on in Japan, is named the CAM-BRAIN project. Its purpose is to design a "brain" containing billions of artificial neurons based on CA cells [2].

Networks are grown and evaluated at electronic speed in a Cellular Automata Machine (CAM-8). This is a machine designed for various CA simulation and is used for e.g. gas simulations.

The networks are artificial neural networks based on CA cells. After a population of networks has been grown they are evolved by using genetic algorithms. The hope is to successfully evolve systems which function as desired, but which are too complex to be designable. One example of such a system is an autonomous cleaning robot. So far it as been shown that networks can been grown in both 2D and 3D. The 2D model has been evolved to output a sine wave of a desired arbitrary period and amplitude. Furthermore, a simple line detector has been evolved, which outputs the vector velocity of a line moved across an array of "detector" neurons. The future will show if it is possible to evolve systems applicable to real world applications.

2.2 Hardware Evolution

There is a limited number of projects where evolutionary hardware has been part of the evolutionary process. The various projects are quite different from each other and they have got few common properties. The main research projects are:

- Neuromorphs, [12].
- Analog network synthesis, [3, 10].
- Embryonics [11].
- Firefly machine [14].
- Evolving a simple robot controller, [16].
- Evolution by Hardware Description Language (HDL), [4].
- On-line FPGA evolution, [15].
- Evolvable ASIC device, [6].

These project can classified as based on EHW. They will be described below, emphasizing on which parts of the evolution are undertaken in the target hardware. The first two projects in the above list is based on evolution using analog technology, while the latter six projects are using digital hardware.

2.3 Analog Devices

This section contains a description of various evolutionary approaches based on analog technology. All work done so far use off-line EHW. This is due to the overhead required for designing the circuit. However, recently analog programmable Field-Programmable Analog Arrays (FPAA) were introduced from several companies. These should make analog on-line evolution possible.

Neuromorphic Systems. Neuromorphic systems are analog VLSI circuits and systems that mimic the nonlinear and spatiotemporal sensory processing capabilities inherent in dendrite tree [12]. Thus, biological neural networks are tried implemented in silicon. Dendrite trees are built using analog components like capacitors and programmable resistors.

Genetic Algorithms have been used to find a set of parameters that produce the sought behavior. The longer a signal propagates along a dendrite branch the more an input signal is attenuated. Only input spikes are used evolve the system and circuits have not been tested on real world applications.

Analog Circuit Synthesis. Automatic analog network synthesis using genetic algorithms has been proposed by several researchers [3, 10]. Koza et al. generate circuits by using a modified SPICE simulator for measuring fitness of each individual. They apply Koza's proposed genetic programming approach, i.e. the population consists of computer programs of varying sizes and shapes. They have successfully evolved a low pass filter and an Op Amp. Grimbleby generates analog circuits using the ordinary genetic algorithm.

Analog Programmable Devices. As mentioned in the introduction of this section several programmable analog devices have recently been introduced. No experiments using these in evolutionary design schemes are yet reported. The following list includes FPAAs offered by commercial companies:

- IMP Inc. Three devices are available: 50E10, 50E20 and 50E30 each with different analog components inside. The 50E30 contains field-programmable gain and function amplifier. Operation frequency is limited to 100 kHz.
- Motorola. MPAAx020 FPAA is just released and is originally a product Pilkington Microelectronics. It consists of 20 configurable analog blocks with the signal frequency limited to 200 kHz.
- Zetex Semiconductors Ltd. Trac is an FPAA containing 20 configurable analog blocks, each of which can accomplish the following functions: *Add, Negate, Pass, Log, Antilog, Rectify, Aux* and *Off*.

2.4 Digital Approaches to Evolvable Hardware

Programmable semiconductor devices have existed for a fairly long time. One of the first chips was the Programmable Array Logic (PAL). PALs are mainly limited to implement a few logic expressions and a state machine. Moreover, to program a PAL a special programming device is required and the PAL device is not erasable.

About ten years ago the more complex Field Programmable Gate Arrays (FPGAs) were introduced. They consist of a much larger number of gates compared to the PALs and are reprogrammable. A string of configuration bits is downloaded to the FPGA and it determines the connections and functions of the internal gates and thus the operation of the device. The configuration bit string makes the device easily reconfigurable.

However, earlier programmable devices were of limited use as EHW due to several problems:

- Limitation in number of possible re-programming operations.
- Possibility of damaging the device by an in-valid configuration.
- Long down-loading time for the configuration.
- Inhibition of partial downloading of the configuration string.
- Slow speed of operation.
- High cost.

These limitations have severely limited the interest in evolvable hardware research. However, recently new devices have been introduced were these

problems are reduced. The complexity and speed is increasing and the price per gate is descreasing.

The problem of long downloading time for an FPGA configuration is partly because of the inhibition of partial downloading. Recently however, a new device — XC6200, from Xilinx were announced. This device is partial programmable, i.e. each cell configuration can be individually programmed ($ns - \mu s$). This makes mutation-operations simple to implement. Further, the speed is high with 220 MHz flip-flop toggle rate. Figure 3 depicts the architecture of XC6200.

The cells communicate to their nearest neighbors and each cell is of simple nature. Both an advantage for evolution. Any configuration string may be downloaded to the device without risk of damaging the device. However, the configuration must be down-loaded *into* the device, thus, a cell can not reconfigure itself.

Atmel has got a similar device — AT6000, which is also partial programmable. However, there is the limitation that the configuration string has to be a valid one to not damage the device.

In the following sections, the various research projects based on evolvable hardware are described. All the projects apply FPGAs as evolvable hardware.

Embryonics. Embryonics are proposed to be specially designed FPGAs. They behave like biological cell groups, with ability to reproduce, differentiate and self repair [11]:

1. **Reproduction phase:** One cell reproduces itself by duplicating the description of the complete system located in its chromosome. This is is carried out until the full system is populated with cells.
2. **Specialization phase:** Each cell interprets one piece of the chromosome depending on the location in the system.

The computer architecture embryonics are inspired by processes in molecular biology. There are several problems, e.g. a large storage is required to keep the complete chromosome in each cell.

No special FPGAs have been produces so far, but commercial devices have been used to validate the different steps in the scheme, using a prototype design named *biodule*.

The Firefly Machine. An evolving, one-dimensional, nonuniform cellular machine has been implemented [14]. It is based on several XC6200 devices on a single board. The system consists of 56 binary-state cells, each containing a rule table. The evolution is completely on-line, i.e. also the genetic operations are carried out on the board. The system is evolving a configuration where the state of each cell oscillate between all zeros and all ones on successive steps. Thus, it behaves like a swarm of fireflies — flashing on and off in an unison way.

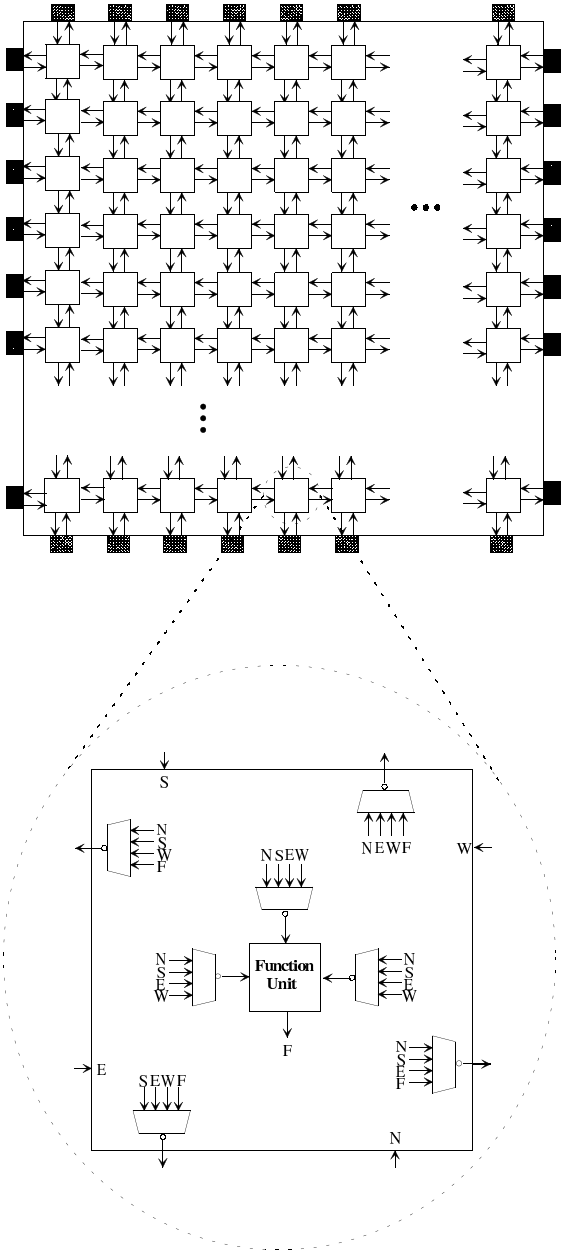

Fig. 3. The XC6200 FPGA architecture consisting of an array of cells. The units on the border of the device are inputs and outputs to the chip.

Evolution by HDL. A Hardware Description Language (HDL) is a programming language and a hardware design tool. A system is proposed where HDL programs are evolved to generate hardware structures and behavior [4]. The HDL language is defined by a rewriting system — i.e. grammar, similar to other programming languages like C. The HDL circuit description can be drawn like a tree where the leaf nodes represent the hardware units with the tree representing the connections between them.

Evolution is based on Genetic Algorithms. The chromosome of the system is represented by the rewriting system, where the terminal symbols are HDL building blocks. Genetic operations like crossover and mutation are performed on the tree-structured design. From a chromosome a program/HW configuration can be generated deterministically. The target architecture for an evolved circuit is an FPGA.

The approach has been successfully used to evolve a binary adder performing summation free of error after 251 generations. Only off-line evolution is used.

On-line Evolution of FPGAs. Evolving an FPGA on-line usually contains the following:

- An evolutionary algorithm operates on a population of configuration bit strings, *each* of which determines the architecture of the FPGA.
- The behavior is evaluated by down-loading one configuration into the FPGA at a time and generating a fitness score.

The initial configuration strings may contain random values. After evolution the goal is to have a configuration string making the FPGA to perform as the user has specified. Since the fitness is measured by downloading each bit string to the FPGA this approach is classified as on-line EHW.

A few experiments have been conducted with evolving FPGAs and the first based on on-chip evolution was conducted by Thompson [16]. He refrains from the digital aspects of a XC6200 FPGA and treats it like an analog, dynamic and asynchronous device. The evolution is allowed to explore the chip without concern about normal design restrictions like switching transients and clock delay. The chromosome was defined by 100 gates and their input connections. Thus, this is gate level evolution. No flip-flops were used in the evolution.

The purpose of the first experiment was to evolve an oscillator without external signal input. One of the things to be investigated was to see if it was possible to evolve an oscillator of fast logic gates to generate a relatively low output frequency. This showed to be possible with fairly high precision and oscillators with frequencies of 10 Hz and 1 kHz were evolved. Thus, it was shown that it is possible to learn a device to do something useful by allowing the hardware to freely evolve without the ordinary restrictions a designer would have used.

In a later experiment, Thompson has evolved a tone-discriminator in the XC6216 FPGA [15]. A 10 x 10 corner of the FPGA were used and the configuration was evolved to discriminate between square waves of 1 kHz and 10 kHz presented at the single input. The single output should go to +5V for one frequency and 0V for the other. The population of size 50 and evolution started by an initial random strings of 1800 bits. The final circuit appeared to be perfect when observed by eye on an oscilloscope. It was the result of evolving 5000 generation. There were initially some unwanted spikes in the output, but these were eliminated around generation 4100.

Functional Level Evolution. If hardware is genetically synthesized from higher level hardware functions — higher than gate level, more complex applications can be evolved [6]. The basis for the work undertaken by Higuchi is earlier EHW gate level experiments using a GAL chip. This is a device similar to a PAL, but it is erasable. It was used in a system evolved to do simple pattern recognition.

Thus to conduct experiments on functional-level evolution, a new FPGA model — F^2PGA, is proposed by Higuchi and shown in Figure 4.

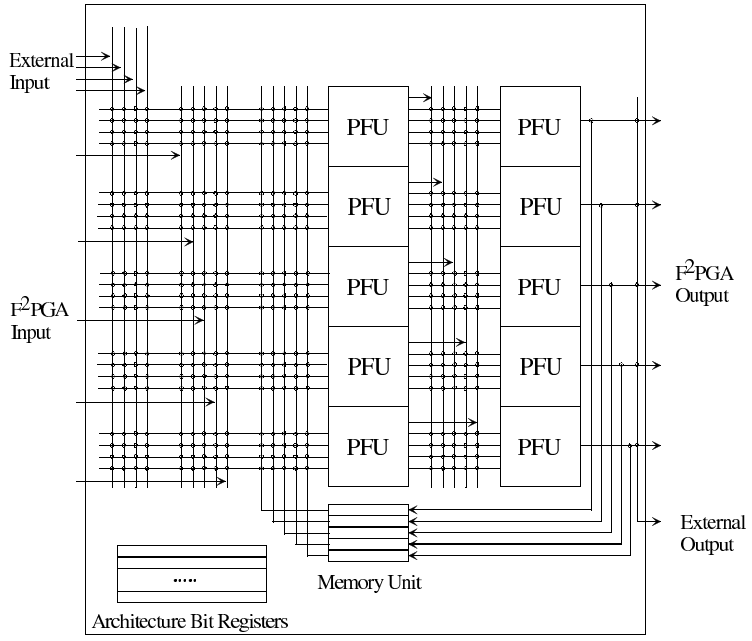

Fig. 4. Function based FPGA.

It consists of a number of Programmable Floating Units (PFUs) interconnected by crossbar switches. The function of the device is dependent on the PFU's function and the crossbar switch settings, which are represented in the chromosome. The length of the chromosome is variable to obtain faster GA execution and allow for large scale evolution. The Memory unit stores the "state" of the system. The Architecture Bit Registers store the configuration bits for the F^2PGA. Each PFU can perform functions like adding, subtracting, multiplication, cosine and sine using floating point numbers. Thus, each PFU includes a floating point ALU and a floating point multiplier. Each PFU attains comparable performance to a DSP. Trigonometric functions are implemented with ROM tables.

The GA operations are selection and mutation, and no crossover is used. After the fittest chromosomes have been selected they all undergoes mutation. This may be:

1. Mutation of the operand — i.e. mutation of a crossbar setting.
2. Mutation of a function — i.e. the operation of a PFU is mutated.

The proposed scheme will be tested by a newly designed ASIC. This will be one of the few devices specially designed for evolution. Simulations have been conducted and indicates performance equal to artificial neural networks.

3 Discussion

Evolvable hardware is a young research field with few experiments on real applications. Furthermore, the number of researchers who have undertaken experiments using evolvable hardware is small. However, the founding of several conferences on the topic should lead to more widespread interest. The results so far of applying evolvable techniques are promising. The next obvious step seems to be testing them on real applications. This will probably require larger scale systems.

The commercial vendor refrain from saying too much about their future devices. However, programmable devices will exist in the future and probably with more gates per chips than today. Thus, this invites researchers to experiment with new design schemes like evolution. So far it seems that FPGAs are the most promising technology for evolution. Later nanotechnology will be introduced and allow for larger scale designs.

It seems like most researchers in the field of evolvable systems are using software simulation, while it is probably the development in hardware that will determine how large the progress will be.

4 Conclusions

Research on evolvable hardware can be summarized in the following conclusions:

- Evolvable hardware is proposed to be a method for designing complex systems.
- Several different evolutionary methods are proposed and under development at the time being.
- Few experiments involving on-chip evolution have been undertaken.
- Both evolutionary techniques and evolvable hardware circuits have to be further improved to be used for complex real world applications.

Today the achievements are few and the small number of experiments undertaken by EHW show that it is still a way to go before we have an evolved vacuum cleaner moving around in our homes doing the cleaning while we are at work.

References

1. E. F. Codd. *Cellular Automata.* Academic Press, New York, 1968.
2. H. de Garis. Cam-brain project: The evolution of a billion neuron artificial brain by 2001. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware: The evolutionary Engineering Approach".* Springer-Verlag, 1996. Lecture Notes in Computer Science, vol. 1062.
3. J. B. Grimbleby. Automatic analogue network synthesis using genetic algorithms. In *Proc. of the 1st IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA95)*, pages 53–58. IEE Conf. Publ. No. 414, 1995.
4. Hitoshi Hemmi et al. Development and evolution of hardware behaviors. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 371–376. MIT Press, 1994.
5. T. Higuchi et al. Evolvable hardware : a first step towards building a darwin machine. In *Proc. of the 2nd Intl. Conf. on Simulated Adaptive Behaviour*, pages 417–424. MIT Press, 1993.
6. T. Higuchi et al. Evolvable hardware with genetic learning. In *Proc. of IEEE Int. Symp on Circuits and Systems (ISCAS96).* Atlanta, 1996.
7. T. Higuchi and others (eds.). *Evolvable Systems: From Biology to Hardware. First Int. Conf., ICES 96.* Springer-Verlag, 1997. Lecture Notes in Computer Science, vol. 1259.
8. A. J. Hirst. Notes on the evolution of adaptive hardware. In *Proc. of the 2nd Int. Conf. on Adaptive Computing in Engineering Design and Control (ACEDC96).* University of Plymouth UK, 1996.
9. J. H. Holland. *Adaption in Natural and Artificial Systems.* The University of Michigan Press, 1975. Ann Arbor, Michigan.

10. J. R. Koza. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In J. S. Gero and F. Sudweeks, editors, *Artificial Intelligence in Design '96*. Dordrecht: Kluwer Academic Publishers, 1996.
11. P. Marchal et al. Embryological development on silicon. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 371–376. MIT Press, 1994.
12. D. P. M. Northmore and J. G. Elias. Evolving synaptic connections for a silicon neuromorph. In *Proceedings of the IEEE Conference on Evolutionary Computation, Orlando*, volume 2, pages 753–758, 1994.
13. E. Sanchez and M. Tomassini (eds.). *Towards Evolvable Hardware: The evolutionary Engineering Approach*. Springer-Verlag, 1996. Lecture Notes in Computer Science, vol. 1062.
14. M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag, 1997. Lecture Notes in Computer Science, vol. 1194.
15. A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware: The evolutionary Engineering Approach*, pages 136–165. Springer-Verlag, 1996. Lecture Notes in Computer Science, vol. 1062.
16. A. Thompson. Silicon evolution. In *Proc. of Genetic Programming 1996*, 1996.
17. J. Torresen. Evolvable hardware — A short introduction. In *Proc. of International Conference On Neural Information Processing (ICONIP'97, Dunedin, New Zealand)*. Springer-Verlag, 1997.

This article was processed using the LaTeX macro package with LLNCS style