Introduction to Evolvable Hardware Design

Jong-O, Kim *, Duk-Soo, Kim ** and Young-Gun, Kim ***
* ,** Dept of Electronics, Dongyang Technical College, Seoul, 152-714, Korea
Tel : +82-2-2610-1778 Fax : +82-2-2688-5494
E-mail: jokim@dongyang.ac.kr, dskim@dongyang.ac.kr
*** Dept of C.I.P., Ansan College, Ansan-Si, Korea E-mail : ygkim@ansan.ac.kr

Abstract: An area of research called evolvable hardware (EHW) has recently emerged which combines aspects of evolutionary computation with hardware design and synthesis. The features that can be used to identify and classify evolvable hardware are the evolutionary algorithm, the implementation and the genotype representation. This paper gives an introduction to the field. It continues by including classifying the EHW and the applications of the area.

Keywords : Evolvable hardware, genenetic algorithm, genetic programming, chromosome.

1. INTRODUCTION

Evolvable hardware is currently used as an general term to characterize applications of evolutionary techniques to hardware design and synthesis. The history of EHW is not that old. It started simultaneously in Switzerland and Japan around 1992 and approximately 1995 in the UK. The first International Conference on Evolvable System(ICES 96) was held in Japan 1996 [1]. Instead of manually designing a circuit, only input/output-relations are specified. The circuit is automatically designed using an adaptive algorithm which is illustrated in Fig. 1.

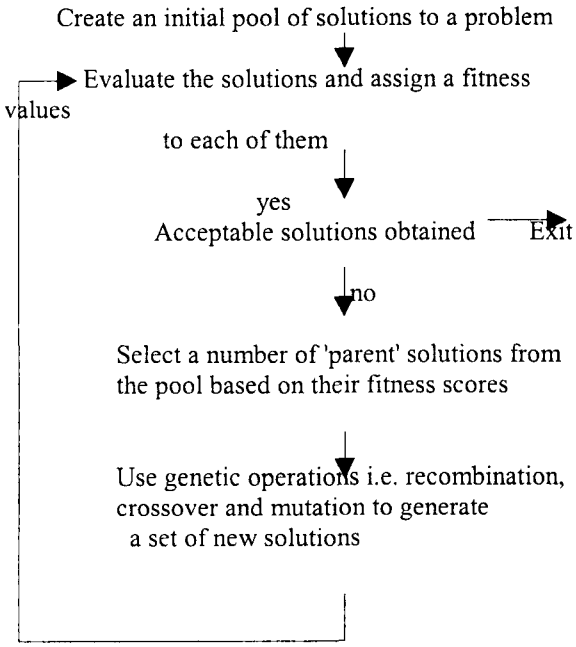

Fig. 1. Standard genetic algorithm.

Genetic algorithms, and more generally evolutionary algorithms (EAs) find solutions to problems e. g. complex optimization problems by emulating the way nature uses biological evolution. They operate on a population (potential solutions) of constant size, and uses the principle of survival of the fittest to produce increasingly better solutions. Many of the terms used in genetics are also used to specify the operation of genetic algorithms. A genetic algorithm starts with a set of random candidate solutions. This set is refered to as a population, and each solution within the population is known as a 'chromosome.' (In genetics chromosomes are strings of DNA, each cell in a living organism contains the same set of chromosomes.) These chromosomes are evaluated and a fitness score is assigned to each of them, the fitness score is some predefined criterion that satisfies.

Next individuals from this graph of chromosomes are selected based on their fitness scores. A number of selection procedures are available, among them Holland's assigned fitness-proportional selection is one of the simplest in which individuals with good fitness scores have a higher probabilities of being selected. The selection process alone can't generate the new individuals into the group of chromosomes. In other words, individuals with better fitness have to be 'reproduced' from the selected individuals.

New individuals are created using genetic operators such as crossover and mutation. Crossover recombines two selected individuals, called parents, by exchanging parts of their encodings to form two new individuals off-springs.

One approach to implement crossover operation between two selected binary strings(chromosomes) is to partition the strings about a randomly chosen point, and exchange substrings. To illustrate let us consider the following two strings, '1' is the crossover point.

Chromosome 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0
Chromosome 2 0 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1
The crossover operation results in the following ff-springs.

Off-spring 1 0 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1
Off-spring 2 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0

It should be emphasized the selection of a specific crossover point can improve the efficiency of the genetic algorithm in solving a particular problem. Therefore, crossover depends on the binary string corresponding a particular solution.

Mutation is the random flipping with a very small probability of any bit in a string e. g. from a 0 to a 1, vice versa altering its value. For example, the sixth bit in string A below is flipped to form a new string B

A = 1 0 1 1 0 1 0 0 1 0

B = 1 0 1 1 0 0 0 0 1 0

The mutation operator performs the important role of injecting diversity into a small set of population. In other words it allows creation of a 'better' gene (binary string) which then becomes part of the rest of the population.

However, many genetic algorithm based techniques do not implement mutation since it has a very low impact on the search for a solution. This is because, as stated earlier mutation has a very low probabilities of changing a bit string corresponding to a solution. Fig.1 presents the structure of the standard genetic algorithm. An initial pool of solution is first selected, this initial selection is usually random.

When the number of offspring circuits equals the number of circuits in the parent population, the new offspring po pulation is ready to become the new parent population. The original parent population is deleted. Thus, one loop
in Fig.1 is named one generation.

A circuit can be represented in several different ways[2]. For digital circuits however, gate level representation is most commonly used. That is, the representation contains a description of what kind of gates are applied and their interconnections. This is coded into a binary configuration bit stream applied to configure a reconfigurable logic device as seen in Fig. 3. This is usually either a commercial device like a Field Programmable Gate Array (FPGA) or a part of an Application Specific Integrated Circuit (ASIC). In addition to the evolutionary algorithm (GA), a circuit specification would have to be available. This is

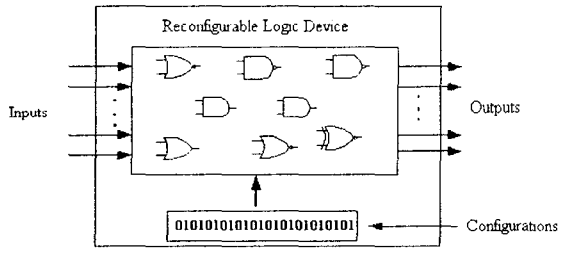

Fig. 3. Illustration of Field Programmable Logic Device (FPLD).

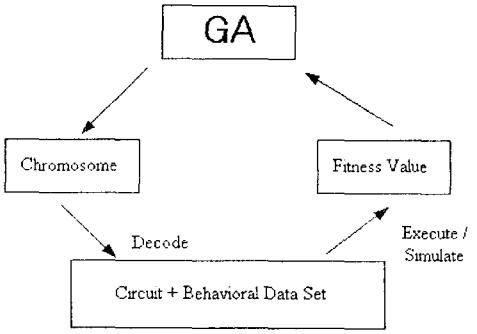

Fig. 4. The cycle of evolving a circuit.

often a set of training vectors (input/output mappings) assembled into a data set. The operation of GA together with the data set are given in Fig. 4. The most computational demanding part of GA is usually the evaluation of each circuit typically named fitness value computation. This involves inputing data to each circuit and computing the error given by the deviation from the specified correct output. A number of industrial applications has arrived based on EHW. The target is to find new schemes making EHW applicable to complex real-world applications.

2. CLASSIFYING EVOLVABLE HARDWARE

EHW research is rapidly diverging. Thus, to understand the EHW field of research, a classification framework would be beneficial [2].

Evolutionary Algorithm A set of major algorithms exists:

- Genetic Algorithm (GA)
- Genetic Programming (GP)
- Evolutionary Programming (EP)

The major difference between GA and GP is the chromosome representation. GA organizes the genes in an array, while GP applies a tree of genes. Both schemes apply both crossover and mutation, while EP- which has no constraints on the representation, uses mutation only.

Building Block The evolution of a hardware circuit is based on connecting basic units together. Several levels of complexity in these building blocks are possible:

- Analog component level: transistors, resistors, inductors and capacitors.
- Gate level : OR and AND gates.
- Function Level: sine generators, adders and multipliers.

Target Hardware. In EHW, the goal is to evolve a circuit. The two major alternatives for target hardware available today are:

•Commercially available devices. FPGAs (Field Programmable Gate Arrays) are most commonly used.

They consist of a number of reconfigurable digital gates, which are connected by entering a binary bitstring into the device. This string specifies how the gates are connected.

For evolution, normally only a subset of the configuration bitstring is used. Field-Programmable Analog Arrays (FPAA) are available as well. They use the same programming principle as FPGAs, but they consist of reconfigurable analog components instead of digital gates.

•Custom hardware. ASIC (Application Specific Integrated Circuit) is a chip fully designed by the user.

Fitness Computation. Degree of fitness computation in hardware:

•Offline EHW . The evolution is simulated in software, and only the elite chromosome is written to the hardware device (sometimes named extrinsic evolution).

• Online EHW . The hardware device gets configured for each chromosome for each generation (sometimes named intrinsic evolution).

Evolution. Degree of evolution undertaken in hardware:

•Off-chip evolution. The evolutionary algorithm is performed on a separate processor.

• On-chip evolution. The evolutionary algorithm is performed on a separate processor incorporated into the chip containing the target EHW.

Scope The scope of evolution:

•Static evolution. The evolution is finished before the circuit is put into normal operation. No evolution is applied during normal operation. The evolution is used as a circuit optimizing tool.

• Dynamic evolution. Evolution is undertaken while the circuit is in operation and this makes the circuit online adaptable.

3. APPLICATIONS OF EVOLVABLE HARDWARE

Using evolution to design circuits in this way brings a number of important benefits to electronics, allowing design automation and innovation for an increasing range of applications. Some of the more important areas where evolvable hardware can be applied include:

- **Automatic design of low cost hardware;**

Automation has been used in circuit synthesis for many years. Traditional digital design involves the mapping of an abstract human-designed circuit to a specific technology through the application of simple minimization, placement and routing rules. As our capability for synthesizing more complex circuits has grown, so has the need for more resourceful processes to handle the complex mapping procedures. Evolvable hardware allows us to take the automation of circuit production a step further, automating how to generate the actual circuit design from a behavioural specification at the same time as automating the circuit synthesis process. The behavioural specification presented to the evolvable system may be as simple as a series of circuit input signals that the system must match to a corresponding predefined set of output signals, although other representations of circuit behaviour may be used, often including environmental conditions or simulated error test cases or depending on the requirements of the circuit

- **Coping with poorly specified problems**

For some problems it is difficult to specify their functionality succinctly, but easy to specify a behavioural description of the problem. Computer scientists have used evolution for to handle problems with such poor specifications for many years. For instance Higuchi et al. have evolved high-speed robust classifiers [3]. Good generalisation characteristics were incorporated into the solutions by specification of a bias based on machine learning theory. More recently do Amaral et al. evolved fuzzy functions that can be used as building blocks in the construction of fuzzy logic controllers [12].

- **Creation of adaptive systems;**

With sufficient automation (i.e., real-time synthesis provided by PLDs), evolvable hardware has the potential to adapt autonomously to changes in its environment. This can be very useful in situations where real-time manual control over systems is not possible, such as on deep space missions. It could be particularly useful when unexpected conditions are encountered.

Many other adaptive filters have been evolved, including digital finite impulse response (FIR) filters, commonly used in audio applications such as noise and echo cancellation [13] and their more complex but less reliable counterparts, infinite impulse response (IIR) filters [14]. Analogue adaptive filters have also been evolved. For example Zebulum et al. presented signal extraction filters capable of adaptively amplifying the strongest component of the input signal whilst attenuating others, thus improving a hypothetical signal/noise ratio [15]. Through evolution these circuits could be adapted to new input profiles. On-line scheduling hardware has also been developed, most notably adaptive cell scheduling systems for ATM networks, that respond to changes in traffic flow [16]. In a related field, Damiani et al. have developed an on-line adaptive hashing system that could be used to map cache blocks to cache tags dependent on the access patterns of the data over time [17].

- **Creation of fault tolerant systems**

Ongoing advances in component miniaturisation have not been complemented by improvements in fabrication reliability. This means that many modern VLSI circuit designs must be tolerant to fabrication faults. It is expected that this will become even more of an issue in future circuit technologies. Miniaturisation also exposes components to a greater risk of operational faults, for instance due to the effects of power fluctuations or ionising radiation. Reliability is of paramount importance for many systems such as medical equipment and transport control systems. Military and spacecraft systems are particularly susceptible to reliability problems as they are regularly subjected to harsh conditions. Current techniques for fault tolerance rely on the presence of additional redundant components and thorough testing either at the point of manufacture or on-line, and add considerable cost and design complexity. Fortunately evolvable hardware provides a number of mechanisms to introduce fault tolerance into circuits.

An early demonstration of this ability was that of Higuchi et al.[3], where an adaptive hardware system that learned the behaviour of an expert robot controller by example using a genetic algorithm. More recently Vigander demonstrated that a simple evolutionary system could restore most but not all functionality to a 4bitx4bit multiplier that had been subjected to random faults [18]. Zebulum et al. demonstrated evolutionary recovery with a 4 bit DAC that had initially been evolved using traditionally-designed operational amplifiers and smaller DACs evolved in earlier experiments as building blocks. A number of other bio-inspired on-line autonomous hardware fault tolerance mechanisms have been developed for both fault detection [19] and recovery [20]. Although these have been proposed as a platform for evolutionary experiments, they do not use evolution as an adaptive repair mechanism, and so will not be considered further here.

- **Innovation in poorly understood design spaces.**

Traditional circuit designers tend to work on a problem from the top down, decomposing the problem in to smaller sub-problems that have limited interactions, and repeating the process that until only a number of small problems remain that are well understood in the field of circuit design, and have known solutions. Evolution works differently. It works from the bottom up, adding components together to make partial solutions to the design problem, which are in turn combined and tinkered with, until the solution meets the design criteria. Perhaps the most successful application of evolution to complex design spaces is the automatic design of antennas. Traditional antenna designs are based on a handful of known, regular topologies. Beyond these the interactions between elements become too complex to abstract. Linden has demonstrated that evolution is capable of discovering an array of highly unconventional, irregular antenna designs [21] and has shown that evolved antennas can be evolved and operate effectively in real-world settings using transmission of real data [22] and where the signal path is obstructed [22]. Such is evolution's performance when applied to antenna design that an evolved antenna is undergoing flight qualification testing for NASA's upcoming Space Technology 5 mission [23] and if successful will be the first evolved hardware in space.

4. CONCLUSIONS

The problems of electronic circuit design are increasing as demand for improvements increases. In this review we have introduced a promising new type of solution to these difficulties - evolvable hardware. This emerging field exists at the intersection of electronic engineering, computer science and biology. The benefits brought about by evolvable hardware are particularly suited to a number of applications, including the design of low cost hardware, poorly specified problems, creation of adaptive systems, fault tolerant systems and innovation.

Evolvable hardware is still a young field. It does not have all the answers to the problems of circuit design and there are still many difficulties to overcome. Nevertheless, these new ideas may be one of the brightest and best hopes for the future of electronics

References

[1] X. Yao and T. Higuchi, Promises and challenges of evolvable hardware," in Evolvable Systems: From Biology to Hardware. First Int. Conf., ICES 96, T. Higuchi et al., Eds. Springer-Verlag, 1997, Lecture Notes in Computer Science, vol. 1259.

[2] Jim Torresen. Evolvable Hardware as a New Computer Architecture. *ICAIEB (SSGRR 2002W)*, January 2002, L'Aquila, Italy.

[3] T Higuchi et al., Evolvable hardware: A firrst step towards building a Darwin machine," in Proc. of the 2nd Int. Conf. onSimulated Behaviour. 1993, pp. 417-424, MIT Press.

[4] Goldberg, Genetic Algorithms in search, optimization, andmachine learning, Addison Wesley, 1989.

[5] J. Torresen, Possibilities and limitations of applying evolvable hardware to real-world application.," in Field-Programmable Logic and Applications: 10th International Conference on FPL-2000, R.W.Hartenstein et al., Eds., pp. 230-239. Springer-Verlag, 2000, Lecture Notes in Computer Science, vol. 1896.

[6] Sechen (1988). *VLSI Placement and Global Routing Using Simulated Annealing*. Boston, MA, U.S.A, Kluwer Academic Publishers.

[7] Yih, J. S. and P. Mazumder (1990). "A Neural Network Design for Circuit Partitioning." *IEEE Transactions on Computer Aided Design* **9**(10): 1265-1271.

[8] Mazumder, P. and E. M. Rudnick (1999). *Genetic Algorithms for VLSI Design, Layout and Test Automation*. Upper Saddle River, NJ, U.S.A., Prentice-Hall.

[9] Rumelhart, D. E., B. Widrow, et al. (1994). "The Basic Ideas in Neural Networks." *Communications of the ACM* **37**(3): 87-92.

[10] Higuchi, T., M. Iwata, et al. (1996). *Evolvable hardware and its application to pattern recognition and fault-tolerant systems.* Proceedings of Towards Evolvable Hardware: An International Workshop. 2 3 Oct. 1995 Lausanne, Switzerland, Springer-Verlag, Berlin, Germany.

[11] Iwata, M., I. Kajitani, et al. (1996). *A pattern recognition system using evolvable hardware.* 4th International Conference on Parallel Problem Solving from Nature PPSN IV, Berlin, Germany, Springer-Verlag, Berlin, Germany.

[12] do Amaral, J. F. M., J. L. M. do Amaral, et al. (2002). *Towards Evolvable Analog Fuzzy Logic Controllers.* 2002 NASA/DoD Conference on Evolvable Hardware, Alexandria, VA, U.S.A., IEEE Press.

[13] Tufte, G. and P. C. Haddow (2000). *Evolving an adaptive digital filter.* Proceedings. The Second NASA/DoD Workshop on Evolvable Hardware. 13 15 July 2000 Palo Alto, CA, USA, IEEE Comput. Soc, Los Alamitos, CA, USA.

[14] Sundaralingam, S. and K. C. Sharman (1998). *Evolving Complex Adaptive IIR Structures.* 9th European Signal Processing Conference, Rhodes, Greece.

[15] Zebulum, R. S., D. Keymeulen, et al. (2003). *Experimental results in evolutionary fault-recovery for field programmable analog devices.* 2003 NASA/DoD Conference on Evolvable Hardware, Chicago, IL, USA, IEEE Comput. Soc, Los Alamitos, CA, USA.

[16] Liu, W., M. Murakawa, et al. (1997). *ATM cell scheduling by function level evolvable hardware.* 1st International Conference on Evolvable Systems, Tsukuba, Japan, Springer-Verlag, Berlin, Germany.

[17] Damiani, E., V. Liberali, et al. (2000). *Dynamic Optimisation of Non-linear Feed-Forward Circuits.* 3rd International Conference on Evolvable Systems, Edinburgh, U.K.

[18] Vigander, S. (2001). Evolutionary Fault Repair of Electronics in Space Applications. Trondheim, Norway, Norwegian University Sci. Tech.

[19] Bradley, D. W. and A. M. Tyrrell (2001). *The architecture for a hardware immune system.* Proceedings Third NASA/DoD Workshop on Evolvable Hardware. EH 2001. 12 14 July 2001 Long Beach, CA, USA, IEEE Comput. Soc, Los Alamitos, CA, USA.

[20] Macias, N. J. and L. J. K. Durbeck (2002). *Self-assembling circuits with autonomous fault handling.* 2002 NASA/DoD Conference on Evolvable Hardware. 15 18 July 2002 Alexandria, VA, USA, IEEE Comput. Soc, Los Alamitos, CA, USA.

[21] Linden, D. S. and E. E. Altshuler (1999). *Evolving wire antennas using genetic algorithms: a review.* Proceedings of the First NASA/DoD Workshop on Evolvable Hardware. 19 21 July 1999 Pasadena, CA, USA, IEEE Comput. Soc, Los Alamitos, CA, USA.

[22] Linden, D. S. (2002). *An evolvable antenna system for optimizing signal strength in-situ.* IEEE Antennas and Propagation Society International Symposium. vol.1 16 21 June 2002 San Antonio, TX, USA, IEEE, Piscataway, NJ, USA.

[23] Lohn, J., G. Larchev, et al. (2003). *A Genetic Representation for Evolutionary Fault Recovery in Virtex FPGAs.* 5th International Conference on Evolvable Systems, Trondheim, Norway, Springer-Verlag.