Evolvable Hardware as a New Computer Architecture

Jim Torresen

Abstract— **Evolvable Hardware (EHW) is a new scheme - inspired by natural evolution, for designing hardware systems. By exploring a large design search space, EHW may find solutions for a task, unsolvable, or more optimal than those found using traditional design methods. The paper introduces this new approach and outlines how it can be applied for hardware design of real-world applications. It continues by including proposals of how a variety of principles from biology can be applied to improve the scheme.**

I. Introduction

Moores law seems to be valid for the development of new computer hardware. This implies that a larger number of transistors implementing digital logic gates are becoming available for designers. Earlier we have seen a limit in the *size* of hardware devices. However, we may very well soon see a limit in *designability*. That is, designers are not able to apply all the transistors in the largest integrated circuits becoming available. To overcome this problem, new and more automatic design schemes would have to be invented. One such method is evolvable hardware.

It was introduced for about eight years ago [1] as a new way of designing electronic circuits. 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.

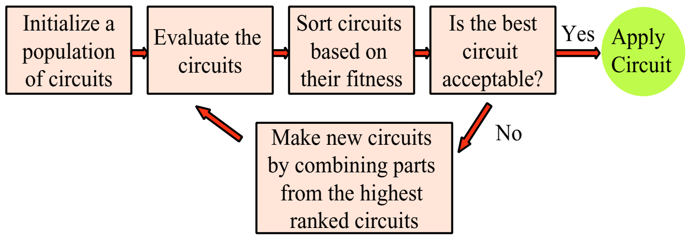
Initialize a population of circuits
Evaluate the circuits
Sort circuits based on their fitness
Is the best circuit acceptable?
Yes
Apply Circuit
No
Make new circuits by combining parts from the highest ranked circuits

Fig. 1. The algorithm for evolving circuits.

In this algorithm, a set (population) of circuits – i.e. circuit representations, are first randomly generated. The behavior of each circuit is evaluated and the best circuits are combined to generate new and hopefully better circuits. The evaluation is according to the behavior initially specified by the user. After a number of iterations, the fittest circuit is to behave according to the initial specification. The most commonly used evolutionary algorithm is genetic algorithm (GA) [2]. The algorithm – which follows the steps described above, contains important operators like crossover and mutation for making new circuits.

Jim Torresen is with the Department of Informatics, University of Oslo, PO Box 1080 Blindern, N-0316 Oslo, Norway.
E-mail: jimtoer@ifi.uio.no, Web: http://www.ifi.uio.no/~jimtoer.

These are very similar to those found in natural evolution as seen in Fig. 2.

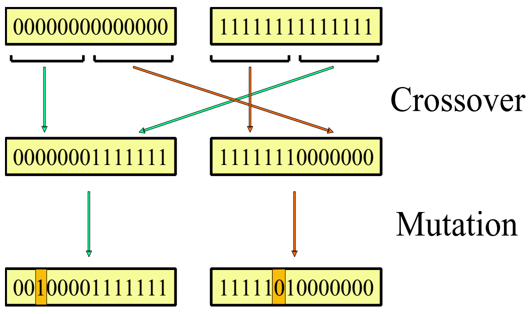

Fig. 2. The genetic algorithm operators.

Each individual in the population is often named *chromosome* or genotype and is represented by an array of bits. Each bit in the array is often called a *gene*. Thus, each chromosome contains a representation of a circuit with a set of components and their interconnections. In crossover, the parameters of the pairwise selected circuits are exchanged to generate – for each couple, two new offspring – preferably fitter than the parents. As an alternative to crossover operation, there is usually some propabilty for conducting cloning instead. Then, the two offspring circuits are equal to the two parent circuits. Further, the *best* circuit may as well be directly copied into the next generation (called elitism). Mutations may also occur and involves inverting a few genes in the chromosome. This make the chromosomes slightly different from what could be obtained by only *combining* parent chromosomes.

When the number of offspring circuits equals the number of circuits in the parent population, the new offspring population is ready to become the new parent population. The original parent population is deleted. Thus, one loop in Fig. 1 is named one *generation*. Randomness is introduced in the selection of parents to be mated. Not only the fittest circuits are selected. However, the probability of a circuit being selected for breeding decreases with decreasing fitness score.

A circuit can be represented in several different ways. 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 bitsteam 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 often

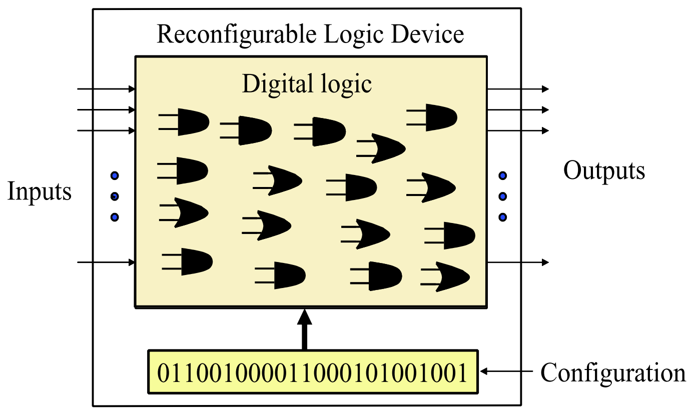

Fig. 3. Illustration of Field Programmable Logic Device (FPLD).

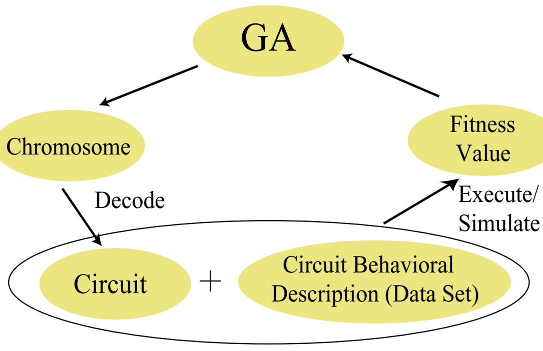

Fig. 4. The cycle of evolving a circuit.

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. These are classified in this paper. The paper further includes an overview and proposals of how biological principles can be applied to improve the present EHW. The target is to find new schemes making EHW applicable to complex real-world applications. Much work has been undertaken on various topics related to EHW. Some consider *modeling* of biological systems without any specific application in mind — e.g. artificial life research. Such studies are not considered in this paper.

The next section contains a classification of EHW research based on a given classification framework [3]. This is followed by a discussion of biological aspects in Section III. Conclusions are given in Section IV.

II. A Framework for Classifying EHW

EHW research is rapidly diverging. Thus, to understand the EHW field of research, a classification framework would be beneficial. This is presented below. The many degrees of freedom in EHW could be represented in a multi-dimensional space. However, here a list format is prefered.

Evolutionary Algorithm (EA). 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 contraints on the representation, uses mutation only.

Technology (TE). Technology for the target EHW:
- **Digital**
- **Analog**

Architecture (AR). The architecture applied in evolution.
- **Complete Circuit Design (CD)** Complete circuit evolution where building block functions (see below) and their interconnections are evolved.
- **Circuit Parameter Tuning (CT)** The architecture is designed in advance and only a set of configurable parameters are evolved.

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

• **Analog comp. level.**	E.g. transistors, resistors, inductors and capacitors.
• **Gate level**	E.g. OR and AND gates.
• **Function Level**	E.g. sine generators, adders and multipliers.

Target Hardware (THW). 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 (FC). Degree of fitness computation in hardware:
- **Offline EHW (OFL).** The evolution is simulated in software, and only the elite chromosome is written to the hardware device (sometimes named extrinsic evolution).
- **Online EHW (ONL).** The hardware device gets configured for each chromosome for each generation (sometimes named intrinsic evolution).

Application	EA	TE	AR	BB	THW	FC	EV	SC
Adaptive Equalizer [4]	GA	D	CD	Neuron	Custom	ONL	On-chip	S
Ampl. and Filter Design [5]	GA	A	CD	T/R/L/C	Custom	OFL	Off-chip	S
Analog Circuit Synthesis [6]	GP	A	CD	R/L/C	Custom	OFL	Off-chip	S
Character Recognition [7]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S
Clock Adjustment [8]	GA	D	CT	Gate	Custom	ONL	Off-chip	S
Digital Filter Design [9]	GA	D	CD	Gate	–	OFL	Off-chip	S
IF Filter Tuning [10]	GA	A	CT	Filter	Custom	ONL	Off-chip	S
Image Compression [11]	GA	D	CT	Pixel	Custom	ONL	On-chip	D
Image Compression [12]	GA	D	CD	Gate	Comm	ONL	On-chip	D
Multi-spect. Image Rec. [13]	GA	D	CT	Function	Comm.	OFL	Off-chip	S
Number Recognition [14]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S
Prosthetic Hand [15]	GA	D	CD	Gate	Custom	ONL	Complete	S
Road Image Rec. [16]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S
Robot Control [17]	GA	D	CD	Gate	Comm.	ONL	Complete	D
Robot Control [18]	GA	D	CD	Gate	Comm.	ONL	Off-chip	S
Sonar Classification [19]	GA	D	CD	Gate	Comm.	OFL	Off-chip	S

TABLE I
CHARACTERISTICS OF EHW APPLIED TO REAL-WORLD APPLICATIONS.

Evolution (EV). 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.
- **Complete HW evolution.** The evolutionary algorithm is implemented in special hardware – i.e. not running on a processor.

Scope (SC). 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.

Table I summarizes the characteristics of the published work on EHW applied to real-world applications. The applications are mainly in the areas of classification and control when complete circuit design is applied. There are also some examples of circuit parameter tuning. A major part of them are based on digital gate level technology using GA as the evolutionary algorithm. However, promising results are given for analog designs, where evolution is used to find optimal parameters for analog components. About half of the experiments are based on custom hardware – or simulation of such. It is more common to put only the fitness evaluation (ONL), than the whole evolution (On-chip/Complete), on the *same* chip as the target EHW. This is reasonable, since the fitness evaluation is – as mentioned earlier, the most computational demanding part of the evolution.

EHW is inspired by biology. Probably several principles could be included from biology to improve the performance when evolving systems. Several of these are discussed in the next section.

III. Evolvable Hardware Inspired by Biology

A. Evolution and Learning

The idea behind artificial evolutionary schemes is to make models of biological systems and mechanisms. From nature the following two laws seem to be present:
1. *Law of Evolution.* Biological systems develop and change during generations by combination and mutation of genes in chromosomes. In this way, new behavior arises and the most competitive individuals in the given environment survive and develop further.
2. *Law of Development.* By cell division a multi-cellular organism is developed.
3. *Law of Learning.* All individuals undergo learning through their lifetime. In this way, they learn to better survive in their environment.

These laws are concerning different aspects of life and should be distinguished, when studying artificial evolution. Most work on genetic algorithms is inspired by the Law of Evolution, while artificial neural networks are considering learning. Examples of artificial development are L-systems and embryological development (see section III-B.1). Biological memory – used store knowledge when learning, is still not fully understood. However, a part of the memory is in the biological neural networks. An interesting approach to design artificial evolutionary systems would be to combine the first law and last law into one system. One approach to this is Evolutionary Artificial Neural Networks (EANNs), which adapt their *architectures* through simulated evolution and their *weights* through learning (training) [20]. Neural networks are based on *local* learning, while the evolutionary approach, on the other hand, is based on *global* operations [21]. While artificial neural networks are normally trained only ones, their biological conterparts undertake life-long learning. This would probably improve many applications if the systems were able to adapt to changes automatically. This is further discussed in Section III-B.3.

B. Aspects of Biological Representation and Development

B.1 The Basic Building Blocks and Their Interconnections

An evolutionary system would have to be based on some kind of building blocks. The model of a multi-cellular living organism would be to use a cell, which is able to reproduce itself by duplicating the full description of the complete organism. This corresponds to *developing* an individual (phenotype) from a chromosome (genotype). One cell could then be used to start reproduction to be carried out until the full organism is populated with cells [22]. Then, a specialization phase starts, where each cell interprets one piece of the chromosome depending on the location in the system. The major problem of this approach – named embryological development, is the tremendous amount of memory required within each cell for storing the complete chromosome. Thus, simpler digital logic gates or higher level functions have been used as the basic building blocks for evolution.

Most research on digital evolvable hardware is based on gate level evolution. There are several reasons for this. One is that the evolved circuit can be partly inspected by studying the evolved Boolean expressions. However, the complexity of the evolved circuit is limited. Functional level evolution has been proposed as a way to increase the complexity. This has been conducted using either ASIC or FPGA technology.

Reconfigurable hardware like FPGAs are normally slower than microprocessors. However, they have an inherently parallel structure with flexible communication. These aspects are similar to those in the biological neural networks, where the neurons are massively interconnected. However, their speed of operation is slow. This indicates that reconfigurable technology should be an interesting approach for solving problems, where the biological brain is superior to ordinary computers.

One area within EHW research is analog design [23]. As the world is analog in nature there would always be a need for interfacing the analog environment. However, in contrast to digital design, most of the analog circuits are still handcrafted by the experts of analog design [24]. Unfortunately, the number of people with knowledge about analog design has diminished as the vast field of digital design and computer science has appeared. Thus, EHW could be a means for making analog design more accessible.

B.2 Local Interactions

The internal operation of FPGAs available today is given by the configuration bitstream loaded from an external source. This is in conflict with the wish that the operation of each cell in a device should only be based on local interactions with no global control. This is not a problem during runtime, but rather during evolution. To make an FPGA device evolvable based on local interaction, it is possible to group FPGA cells together into subsets of cells [25]. Each subset should represent a circuit element as well as the configuration bits for this element. In this scheme, the normal FPGA configuration bitstream should not have to be changed during evolution, but rather the newly defined configuration bits for each circuit element. Thus, such an FPGA could be conducting self-reconfiguration.

Cellular Automata (CA) has been introduced to design systems based on local interactions. An experimental system based on FPGAs has been built for CA and local learning [26], [27]. The states of the cells in the system are evolved to oscillate between all zeros and all ones on successive time steps like a swarm of fireflies. The inherent parallelism in programmable hardware should be applied not only for runtime, but also for the evolution.

B.3 On-line Adaptation in Real-Time.

Most living creatures are able to learn to live in an environment and adapt if some change occur. Artifical systems are far from such an adaptivity. Research on evolutionary methods have – with a few exceptions, been based on one-time evolution. However, there is a wish of being able to design on-line adaptable evolvable hardware. The hardware would then have to be able to reconfigure its configuration dynamically and autonomously, when operating in its environment [28].

One possible approach to such a system is to use *two* parallel units. A primary unit is applied for normal runtime operation of an application, while a secondary unit keeps evolving in parallel. If the performance of the secondary unit becomes better than the primary unit, they are exchanged. Thus, the unit giving best performance at the moment is enabled.

There could be mutual benefits between evolutionary computation and neural network. Neural networks – which are not normally on-line adaptable, could benefit from being adaptable to changing environments. This also includes a changing noise level. However, if they are retrained the first learning may vanish. Thus, we could represent a system by a *set* of neural networks and select the best to apply at a given moment. A strategy is possible where we in parallel to the normal runtime operation evaluate each net by thorough computation. The networks in the set are modified if they are rarely used. This could be either by continued neural network training or evolution of the weights/neurons.

B.4 Generalization

Biological signals are multi-level and the processing of inputs are providing generalization. That is, the input would not have to be equal to a *unique* value to trigger a correct response. This makes biological systems robust.

The gate level version of EHW is basically applying two-level signals. In comparison to neural network modeling using 32-bit floating point values, digital EHW could not with its two-level signal format provide the same noise robustness and generalization [7]. To improve the representation ability of EHW, it is hereby proposed that in order to evolve systems interfacing a real-world environment, multi-level signals and non-linear functions should be applied. If the input patterns to the system are digital, it would probably be interesting to investigate an architecture where an increased number of bits is used towards the output of the system. That is, the *number* of bits used for representing signals in each layer increases from input to output. This would correspond to providing more accuracy for the higher levels of the system. Another option to improve the signal is to include time in the coding approach. That is, to attain the value of a signal, it must be observed for a certain time.

B.5 Multi-Environment Training.

So far, artificial systems have mainly been trained by a single training set or in a limited environment. This is in contrast to biological organisms trained to operate in different and changing environments. When training in a limited environment, the system with maximum fitness is selected. This fitness value does not have to represent the system with the best generalization [21]. It would be an interesting approach to investigate if an artificial system trained to operate in one environment could be combined with similar systems trained in other environments. If the resulting system were able to contain knowledge about the different environments, it would be able to operate in various environments *without* retraining. A possible implementation could include a soft switching mechanism between control systems, when the system enters a new environment. Another approach would be to separately evolve systems for different environments, and then afterwards, evolve them together into a final system. This is a part of the scheme to be presented in the next section.

B.6 Gradual Development

Biological evolution, development and learning are all based on gradual development of structures and properties of these. This idea is applied in incremental evolution for EHW. It was first introduced in [25] for a character recognition system. The approach is a divide-and-conquer on the evolution of the EHW system, and thus, named *increased complexity evolution*. It consists of a division of the *problem* domain together with incremental evolution of the hardware system. Evolution is first undertaken individually on a set of basic units. The evolved units are the building blocks used in further evolution of a larger and more complex system. The benefits of applying this scheme is both a *simpler* and *smaller* search space compared to conducting evolution in one single run. The goal is to develop a scheme that could evolve systems for complex real-world applications. Successfull results have been achieved for several different classification problems [16], [29].

IV. Conclusions

This paper has introduced EHW and contained a study of the characteristics of EHW applied to real-world applications. Further, applying new principles from biology in evolving systems have been discussed. This is required to make it possible to evolve systems for complex applications.

Acknowledgments

The author would like to thank Lukas Sekanina (Brno University of Technology, Czech Republic) for inspiring and fruitful discussions during this work.

References

[1] T. Higuchi et al., “Evolvable hardware: A first step towards building a Darwin machine,” in *Proc. of the 2nd Int. Conf. on Simulated Behaviour*. 1993, pp. 417–424, MIT Press.

[2] D. Goldberg, *Genetic Algorithms in search, optimization, and machine learning*, Addison Wesley, 1989.

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

[4] M. Murakawa et al., “The GRD chip: Genetic reconfiguration of dsps for neural network processing.,” *IEEE Transactions on Computers*, vol. 48, no. 6, pp. 628–638, June 1999.

[5] J.D. Lohn and S.P. Colombano, “A circuit representation technique for automated circuit design,” *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 3, pp. 205–219, September 1999.

[6] J. R. Koza et al., *Genetic Programming III*, San Francisco, CA: Morgan Kaufmann Publishers, 1999.

[7] J. Torresen, “Increased complexity evolution applied to evolvable hardware.,” in *Smart Engineering System Design: Neural Networks, Fuzzy Logic , Evolutionary Programming, Data Mining, and Complex Systems, Proc. of ANNIE’99*, Dagli et al., Eds. ASME Press, November 1999.

[8] E. Takahashi et al., “An evolvable-hardware-based clock timing architecture towards GigaHz digital systems,” in *Proc. of the Genetic and Evolutionary Computation Conference*, 1999.

[9] J. F. Miller, “Digital filter design at gate-level using evolutionary algorithms,” in *Proc. of the Genetic and Evolutionary Computation Conference*, 1999.

[10] M. Murakawa et al., “Analogue EHW chip for intermediate frequency filters.,” in *Evolvable Systems: From Biology to Hardware. Second Int. Conf., ICES 98*, M. Sipper et al., Eds., pp. 134–143. Springer-Verlag, 1998, Lecture Notes in Computer Science, vol. 1478.

[11] Sakanashi et al., “Evolvable hardware chip for high precision printer image compression,” in *Proc. of 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.

[12] L. Sekanina, “Toward uniform approach to design of evolvable hardware based systems.,” in *Field-Programmable Logic and Applications: 10th International Conference on Field Programmable Logic and Applications (FPL-2000)*, R.W. Hartenstein et al., Eds., pp. 814–817. Springer-Verlag, 2000, Lecture Notes in Computer Science, vol. 1896.

[13] R. Porter et al., “An applications approach to evolvable hardware,” in *Proc. of the First NASA/DoD Workshop on Evolvable Hardware*, 1999.

[14] M. Iwata et al., “A pattern recognition system using evolvable hardware,” in *Proc. of Parallel Problem Solving from Nature IV (PPSN IV)*. September 1996, Springer Verlag, LNCS 1141.

[15] I. Kajitani and other, “An evolvable hardware chip and its application as a multi-function prosthetic hand controller,” in *Proc. of 16th National Conference on Artificial Intelligence (AAAI-99)*, 1999.

[16] J. Torresen, “Scalable evolvable hardware applied to road image recognition.,” in *Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware*. July 2000, Silicon Valley, USA.

[17] D. Keymeulen et al., “On-line model-based learning using evolvable hardware for a robotics tracking systems,” in *Genetic Pro-*

gramming 1998: Proc. of the Third Annual Conference. 1998, pp. 816–823, Morgan Kaufmann.

[18] A. Thompson, "Exploration in design space: Unconventional electronics design through artificial evolution," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 3, pp. 171–177, September 1999.

[19] M. Yasunaga et al., "Evolvable sonar spectrum discrimination chip designed by genetic algorithm," in *Proc. of 1999 IEEE Systems, Man, and Cybernetics Conference (SMC'99)*, 1999.

[20] X. Yao and Y. Liu, "Evolutionary artificial neural networks that learn and generalize well.," in *Proc. of IEEE Int. Conf. on Neural Networks, Washington DC.* 1996, IEEE Press.

[21] 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.

[22] P. Marchal et al., "Embryological development on silicon.," in *Artificial Life IV*, R. Brooks and P. Maes, Eds., pp. 371–376. MIT Press, 1994.

[23] J. Torresen, "Evolvable hardware — The coming hardware design method?," in *Neuro-fuzzy techniques for Intelligent Information Systems*, N. Kasabov and R. Kozma, Eds., pp. 435 – 449. Physica-Verlag (Springer-Verlag), 1999.

[24] O. Aaserud and I.R. Nielsen, "Trends in current analog design: A panel debate," *Analog Integrated Circuits and Signal Processing*, vol. 7, no. 1, pp. –, 1995.

[25] J. Torresen, "A divide-and-conquer approach to evolvable hardware.," in *Evolvable Systems: From Biology to Hardware. Second Int. Conf., ICES 98*, M. Sipper et al., Eds., pp. 57–65. Springer-Verlag, 1998, Lecture Notes in Computer Science, vol. 1478.

[26] M. Sipper et al., "A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 1, pp. 83–97, April 1997.

[27] M. Sipper, *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*, Springer-Verlag, 1997, Lecture Notes in Computer Science, vol. 1194.

[28] T. Higuchi et al., "Evolvable hardware and its applications to pattern recognition and fault-tolerant systems," in *Towards Evolvable Hardware: The evolutionary Engineering Approach*, E. Sanchez and M. Tomassini, Eds. Springer-Verlag, 1996, Lecture Notes in Computer Science, vol. 1062.

[29] J. Torresen, "Two-step incremental evolution of a digital logic gate based prosthetic hand controller.," in *Evolvable Systems: From Biology to Hardware. Fourth Int. Conf., ICES'01).* Springer-Verlag, 2001, Lecture Notes in Computer Science, vol. 2210.