

INCREASED COMPLEXITY EVOLUTION APPLIED TO EVOLVABLE HARDWARE

JIM TORRESEN

Department of Informatics, University of Oslo,
PO Box 1080 Blindern, N-0316 Oslo, Norway
E-mail: jimtoer@ifi.uio.no

ABSTRACT

Evolvable Hardware (EHW) has been proposed as a new method for designing systems for complex real world applications. One of the problems has been that only small and simple systems have been evolvable. This paper highlights some of the reasons that can explain why EHW has not yet been widely applied. Further, to make EHW more applicable, an increased complexity scheme is proposed, where a system is evolved by evolving smaller sub-systems. Experiments show that the number of generations required for evolution by the new method can be substantially reduced compared to evolving a system directly. This is with no lack of performance in the final system. Another experiment shows that gate level EHW lacks noise robustness.

INTRODUCTION

Evolvable hardware (EHW) has recently been introduced as a new scheme for designing systems for real world applications. So far the number of applications is highly limited.

One of the main problems in evolving hardware systems seems to be the limitation in the chromosome string length [7]. A long string is required for representing a complex system. However, a larger number of generations are required by genetic algorithms (GA) as the string increases. Thus, work has been undertaken to try to diminish this limitation.

Various experiments on speeding up the GA computation have been undertaken [2]. The schemes involve fitness computation in parallel or a partitioned population evolved in parallel. Experiments are based on speeding up the GA computation, rather than dividing the application into subtasks. This approach requires that GA finds a solution if it is allowed to compute enough generations. When very small applications require weeks of evolution time [12], there would probably be strict limitations on the systems evolvable even by parallel GA.

Other approaches to the problem have been by using variable length chromosome [4]. Another option, called function level evolution, is to evolve at a higher level than gate level [9]. Most work is based on fixed functions. However, there has been work in Genetic Programming for *evolving* the functions [6]. The method is called Automatically Defined Functions (ADF) and is used in software evolution.

Both gate and function level of EHW evolution have been applied to real applications. To control an artificial hand, a gate-level EHW chip has been designed [5]. The GA operations are undertaken in a separate unit within the chip. Simulations of data compression using function level evolution indicate performance comparable to other compression methods like JPEG compression [11]. The scheme is designed for implementation in a custom ASIC device. A function based FPGA has been proposed for applications like ATM cell scheduling [8] and adaptive equalizer in digital mobile communication [10]. Except for the limited number of real problems studied, there is a larger range of small and artificial problems, see [13, 15]

The idea of gradual increase in complexity has been introduced for AI systems using software models [1]. In the work it is stated that humans undergo a process of development where we are able to perform more difficult tasks in more complex environments en route to the adult state. It is foreseen that the construction of such an AI system could scale autonomously. Which learning structures and organizational principles to apply is still an open question.

Increased complexity evolution for EHW was first introduced in [14]. The approach is a divide-and-conquer on the evolution of the EHW system. In, this paper more details about the scheme are given including experimental results showing the benefits of applying such a method. The goal is to make a system that could evolve complex systems for real applications.

The next section introduces some aspects of the noise robustness of EHW. This is followed by a presentation of the increased complexity evolution method and experimental results. Discussion and conclusions are given in the last part of the paper.

NOISE ROBUSTNESS AND GENERALIZATION

The gate level version of EHW is basically applying two-level signals. In comparison to neural network modeling using 32-bit floating point values, EHW could probably not provide the same noise robustness and generalization. Results from experiments undertaken to demonstrate these issues are given in the result section. To improve the representation ability of EHW, it is here proposed that each signal could be coded by a variable number of bits. 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 the representation of signals in each layer increases from input to output. This would correspond to providing more accuracy for the higher levels of the system. Detecting the values of a small number of pixels in a picture could be undertaken with a coarse accuracy compared to detecting larger objects in an image. 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.

There could be mutual benefits between evolutionary computation and neural network. Neural networks – which are not normally online 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 apply evolutionary computation to select the best to apply at a given moment. A strategy is possible where we in parallel to the normal 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 to modify for long term changes.

INCREASED COMPLEXITY EVOLUTION

In this section an evolution scheme – called *increased complexity evolution*, is proposed to overcome the problem of a long chromosome string. The idea is to evolve a system gradually as a kind of divide-and-conquer method. Evolution is first undertaken individually on a large number of simple cells. The evolved functions are the basic blocks used in further evolution or assembly of a larger and more complex system. This may continue until a final system is at a sufficient level of complexity.

Approaches to increased complexity evolution

The main advantage of the method is that evolution is not performed in one operation on the complete evolvable hardware unit, but rather in a bottom-up way. The chromosome length can be limited to allow faster evolution. The problem of the approach would be how to define the fitness functions for the lower level sub-systems. Below, two alternatives are presented: Partitioned training vectors and Incremental training vector complexity.

Partitioned training vectors.

A first approach to incremental evolution is by partitioning the training vectors. For evolving a truth table - i.e. like those used in digital design, each separate output could be evolved separately.

Incremental training vector complexity.

For many applications it is impossible to use the *complete* input set and a *limited* number of the outputs to the environment in the evolutionary system design. For those instances one must look for other ways to building a complex system. One

approach could be by dividing the training set into several subsets. This corresponds to the way humans learns: Learning to walk and learning to talk are two different learning tasks. In this way a robot could in the first step be taught each subtask individually. In the second step all sub-learned systems are integrated for working together. Each of these two steps could be based on evolution or other appropriate methods (e.g. artificial neural networks).

The complexity of the basic building block

Most work in the field of EHW is undertaken using gate level evolution. However, experiments with higher level functions have been undertaken as well. Based on the above described limitations of logic level EHW, it is here proposed that to evolve systems interfacing a real-world environment, multilevel signals and non-linear functions should be applied. As indicated earlier in the paper, this could include a variable number of bits used for representing signals.

Increased complexity evolution applied to FPGAs

Experiments have shown that the evolution time is very long even for a small application, when gate level evolution is applied to FPGAs [12]. Thus, it is interesting to apply the increased complexity scheme to FPGAs.

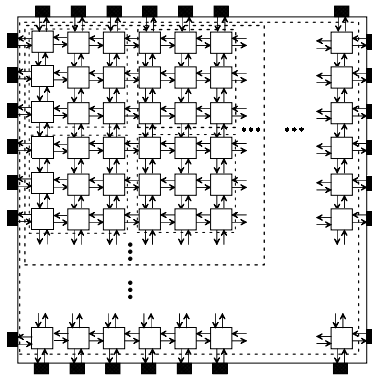


Figure 1: Evolving an FPGA at higher level than gate level.

To make an FPGA device evolvable at higher than gate level, it is possible to arrange sets of FPGA cells into subsets of cells, see Figure 1. The function of each subset could be either evolved or designed manually. Groups of these subsets are then in the next level assembled or evolved into more complex functions. This continues and for each iteration the basic functions are getting more complex than in the previous iteration. There are two aspects to develop:

- The function of each subset (different functions are possible).
- The networks connecting the subsets.

An approach similar to evolutionary artificial neural networks could be applied where the *functions* are learned by local learning and the *connections* are evolved. We would have to find the optimal macro-cell size compared to the network size for evolution at a higher level than gate level. A large or complex cell structure would lead to a smaller number of such cells within a single FPGA device. Thus, the network will be small compared to using a simple macro-cell. This approach concentrates on utilizing the available hardware technology rather than trying to copy a biological system.

The experiment

In this paper, the increased complexity principle will be illustrated by a character recognition system. The evolution will be based on gate level, but can easily be

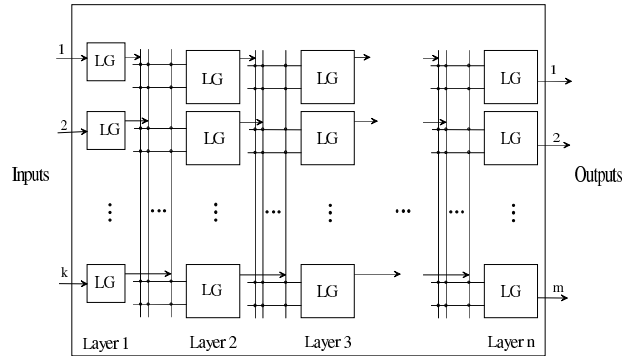


Figure 2: An array of gates. “LG” indicates a logic gate and can be *Buffer Inverter*, *AND* or *OR* gate.

changed to function level. The target EHW is an array of logic gates as illustrated in Figure 2. This array can be configured in the Xilinx XC6200.

The array consists of n number of layers of gates from input to output. Except for layer 1, the Logic Gate (LG) is either a *Buffer*, *Inverter*, *AND* or *OR* gate. In layer 1, only *Buffer* and *Inverter* gates are available. Each gate’s two inputs¹ in layer l is connected to the outputs of two gates in layer $l - 1$. The function of each gate and its two inputs are determined by evolution.

The evolution is undertaken off-line using software simulation. However, since no feed-back connections are used and the number of gates between the input and output is limited to n , the real performance should equal the simulation. Any spikes could be removed using registers on the output gates.

The application to be used in the experiments is the problem of recognizing characters of 5×6 pixels size, where each pixel can be 0 or 1. Each of the pixels is connected to *one* input gate. In addition to the 30 pixel inputs, two extra inputs are included as a bias of value 0 and 1, respectively. The output of the gate array consists of one output gate for each character the system is trained to recognize. During recognition, the output gate corresponding to the input character should be 1, while the other outputs should be 0. Thus, if the training set consists of m *different* characters, the gate array should consist of m output gates.

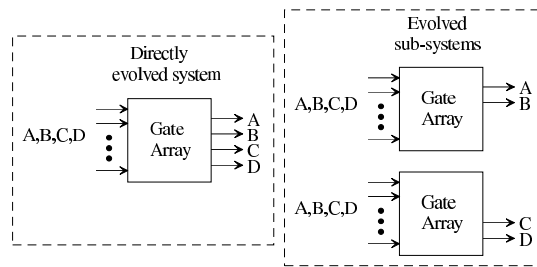


Figure 3: The increased complexity method applied for designing systems for recognizing four characters.

The aspect of *increased complexity evolution* is introduced by the way the evolution is undertaken as seen in Figure 3. We compare evolving a system directly to evolving sub-systems. In the former case, the system is evolved to classify all training vectors in the training set. In the latter case, an evolved sub-system is able to classify a *subset* of the training characters. That is, each subsystem input

¹*Buffer* and *Inverter* gates have only one input.

all the 30 input pixels and *all* training vectors are applied during fitness evaluation. However, each subsystem has a limited number of *output* gates. In this way, the sub-systems are evolved without lack of generalization. That is, when all evolved sub-systems are operated in parallel, they provide the same generalization as the directly evolved system. The benefit is that each gate array is smaller and thus, should easier become evolved to perform the correct operation. For this application, the integration of sub-systems are straightforward by running them in parallel. For more complex applications, like speech recognition, a next level of evolution could be applied, where the sub-systems are the basic blocks in the evolution. The number of gate layers in the array is flexible and different numbers will be tested in the experiments. A large number would allow for more complex logic expressions. However, the chromosome becomes longer and obtaining a correctly evolved circuit could be more difficult.

Except for the output layer in the gate array, 32 gates are applied in each layer in the array. Thus, the *complete* system will be larger as the number of sub-systems increases. The main motivation for this work is to allow for evolution of complex systems and limiting the number of gates is not regarded as an important topic. The reason for this is that the main problem of today's research seems to be the lack of evolutionary schemes overcoming the chromosome length problem, more than the lack of large gate arrays. In fact, by reducing the amount of information to be represented in a digital circuit, the number of generations required by GA could be reduced as well.

The simple GA – detailed by Goldberg [3], was applied for the evolution with a population size of 50. For each new generation an entirely new population of individuals is generated. The mutation rate - the probability of bit inversion for each bit in the binary chromosome string, is 0.001. For each test, ten circuits were evolved and the circuit requiring the least number of generations was picked as the best.

RESULTS

In this section various aspects of evolvable hardware are investigated through different experiments. This includes the benefits of the increased complexity evolution method and the limitations of evolvable hardware in generalization and noise robustness.

Type of system	Number of gen.
16 characters (A,...,P)	> 30000
8 characters (A,...,H)	> 20000
8 characters (I,...,P)	18442
4 characters (A,B,C,D)	1975
4 characters (E,F,G,H)	3997
4 characters (I,J,K,L)	185
4 characters (M,N,O,P)	3626
2 characters (A,B)	323
2 characters (C,D)	623
2 characters (E,F)	76
2 characters (G,H)	163
2 characters (I,J)	23
2 characters (K,L)	36
2 characters (M,N)	66
2 characters (O,P)	1059

Table 1: The results of evolving a circuit for classifying 16 patterns, for four layers of gates in the array.

The first experiment is applying the increased complexity method for designing a system for recognizing 16 characters. Four layers of gates in the logic gate arrays are applied. Table 1 shows the required number of generations used for obtaining a circuit that correctly classifies the patterns in the training set. If the evolution is undertaken in one operation no circuit is found² that correctly classifies all training patterns. As the training set is partitioned into smaller sets evolved separately, the

²The maximum number of generations was set to 30000.

number of generation is decreased. The total number of generations for the two-character systems are about 2400, while the four-character systems require about 9,800. This is a ratio of about four in difference. Each of the sub-systems recognizes a limited number of characters. When other characters in the training set are input, the system is evolved to output the value 0 on each output. Thus, the final systems still has the same classification ability as the system evolved in one operation.

Training set size	Dir. evolved	2 sub-sys.	4 sub-sys.	8 sub-sys.
4 characters	101	16	–	–
8 characters	3,514	1,120	225	–
16 characters	>30,000	>38,442	9,783	2369

Table 2: The total number of generations used for of evolving a circuit for classifying from 4 to 16 patterns, for four layers of gates in the array.

Earlier experiments with a smaller number of characters (four and eight) were undertaken for both three and four layers of gates [14]. Almost without exception the four layer array converged faster to a solution than the array with three layers of gates. Since the number of characters are larger here, it is assumed that at least four layers of gates are required for representing the recognition system. A larger number of gates could be beneficial, especially for the sub-systems recognizing many characters. However, the chromosome becomes longer. For the directly evolved system, the chromosome length is 990 bits and 1374 bits for four and five layers of gates, respectively. Experiments conducted using five or six layers did not perform better than four layers for direct evolution.

The experiments with the smaller training sets give the same results as above on the convergence when comparing evolving systems of different sizes [14]. The total number of generations required to find correct classification circuits for three different training set sizes are given in Table 2.

Without exception, a system is evolved with fewer generations, when dividing it into as many subsystems as possible. It is required in average a factor of four less generations for each diving by two undertaken.

These results also indicate the substantial increase in generations required as the training set size increases. Over ten times as many generations are required as the training set size is doubled. This demonstrates the importance of not undertaking the evolution in one single operation

For some populations there are very slow convergence compared to others due to different initial conditions. Thus, when conducting experiments it could be effective to terminate slowly converging experiments and restart the experiment with other initial conditions to look for a faster converging population.

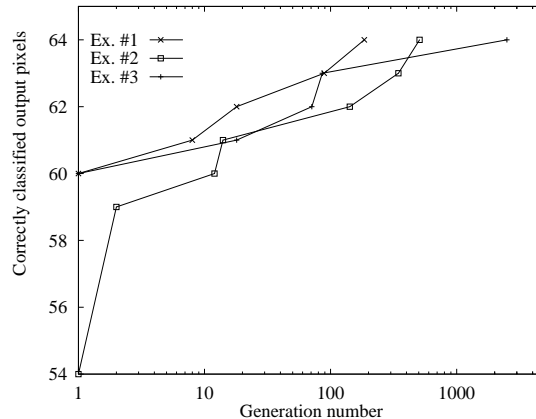


Figure 4: Comparison of convergence for the same kinds of experiments – evolving a sub-system for the characters (I,J,K,L)), with three different initial conditions.

Figure 4 plots the convergence for three different initial conditions. There is not a consistent relation between the initial convergences and total number of generations required for each of them. However, for the fastest converging experiment, no other experiments converge better in the initial generations. Thus, if several contiguous experiments are run, some termination feature for the slowly converging ones could be implemented.

Noise robustness

Experiments to test the robustness to noise for gate level EHW were undertaken by introducing noise in the inputs during recognition. To evaluate the noise robustness and generalization ability, a comparison were made against an artificial neural network (ANN). It is a two weight layer network with the 24 hidden neurons and the same number of inputs (32) and outputs (8) as the EHW system for recognizing characters. The training set is the same as for the EHW as well and consists of eight characters.

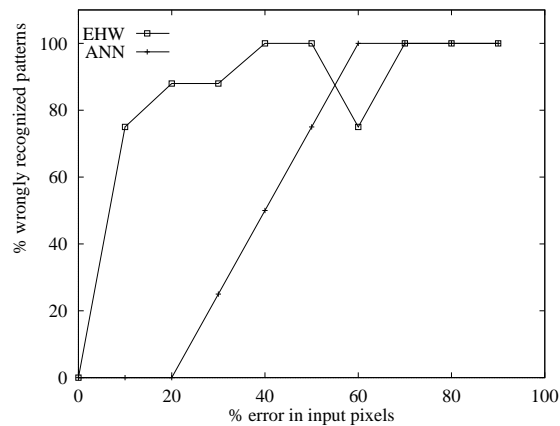


Figure 5: Comparison of error in recognition, when noise is introduced in the inputs to the EHW and ANN, respectively.

Figure 5 shows the results where the error in the input pixels are from 0% up to 90%. The performance of the ANN system outperforms the EHW system for error rates below 60%. Similar results were found for systems recognizing four characters.

This is understandable because every output of the EHW system has to give an exact correct response. This is much harder than for the neural network system, where the output with the largest floating point value is selected. To improve the noise robustness for EHW, higher resolution in signals and functional units must be applied as outlined earlier in this paper.

DISCUSSION

The results indicate a large increase in the number of generations, as the training set size increases. By evolving sub-systems instead of the complete system in one operation, the total number of generations required can be highly limited. Thus, the increased complexity method is a promising way of designing more complex systems. However, as the experiments have shown, the gate level EHW do not provide the same noise robustness as neural networks based on multilevel-signals and non-linear functions. This emphasizes the importance of applying more complex basic building blocks when demanded.

CONCLUSIONS

This paper has reviewed some of the present weaknesses of EHW and proposed possible improvements. A scheme, called *increased complexity evolution*, is introduced. The method is based on sub-system evolution for the design of complex systems.

A character recognition application is used to present one implementation of the scheme. It is shown that the number of generations can be substantially reduced by evolving sub-systems instead of a complete system in one operation. The lack of noise robustness shown could be dealt with in future systems by introducing higher signal precision and non-linear functions in the systems to be used in evolution.

References

- [1] Rodney A. Brooks et al. Alternative essences of intelligence. In *Proc. of the 15th National Conf. on Artificial Intelligence (AAAI-98)*. AAAI Press, 1998.
- [2] E. Cantu-Paz. A survey of parallel genetic algorithms. *Calculateurs Parallels*, 10(2), 1998. Paris: Hermes.
- [3] D. Goldberg. *Genetic Algorithms in search, optimization, and machine learning*. Addison Wesley, 1989.
- [4] M. Iwata et al. A pattern recognition system using evolvable hardware. In *Proc. of Parallel Problem Solving from Nature IV (PPSN IV)*. Springer Verlag, LNCS 1141, September 1996.
- [5] I. Kajitani et al. A gate-level ehw chip: Implementing GA operations and reconfigurable hardware on a single LSI. In M. Sipper et al., editors, *Evolvable Systems: From Biology to Hardware. Second Int. Conf., ICES 98*, pages 1–12. Springer-Verlag, 1998. Lecture Notes in Computer Science, vol. 1478.
- [6] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. The MIT Press, 1994.
- [7] W-P. Lee et al. Learning complex robot behaviours by evolutionary computing with task decomposition. In Andreas Brink and John Demiris, editors, *Learning Robots: Proc. of 6th European Workshop, EWLR-6 Brighton*. Springer, 1997.
- [8] W. Liu et al. Atm cell scheduling by function level evolvable hardware. In T. Higuchi et al., editors, *Evolvable Systems: From Biology to Hardware. First Int. Conf., ICES 96*, pages 180 – 192. Springer-Verlag, 1997. Lecture Notes in Computer Science, vol. 1259.
- [9] M. Murakawa et al. Hardware evolution at function level. In *Proc. of Parallel Problem Solving from Nature IV (PPSNIV)*. Springer Verlag, LNCS 1141, September 1996.
- [10] M. Murakawa et al. Evolvable hardware for generalized neural networks. In *Proc. of Fifteenth Int. Joint Conf. on AI (IJCAI-97)*. Morgan Kaufmann Publishers, 1997.
- [11] M. Salami et al. Lossless image compression by evolvable hardware. In *Proc. of 4th European Conf. on Artificial Life (ECAL97)*. MIT Press, 1997.
- [12] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In T. Higuchi et al., editors, *Evolvable Systems: From Biology to Hardware. First Int. Conf., ICES 96*. Springer-Verlag, 1997. Lecture Notes in Computer Science, vol. 1259.
- [13] J. Torresen. Evolvable hardware — A short introduction. In *Proc. of International Conference On Neural Information Processing (ICONIP'97, Dunedin, New Zealand)*. Springer-Verlag, November 1997.
- [14] J. Torresen. A divide-and-conquer approach to evolvable hardware. In M. Sipper et al., editors, *Evolvable Systems: From Biology to Hardware. Second Int. Conf., ICES 98*, pages 57–65. Springer-Verlag, 1998. Lecture Notes in Computer Science, vol. 1478.
- [15] J. Torresen. Evolvable hardware — The coming hardware design method? In N. Kasabov and R. Kozma, editors, *Neuro-fuzzy techniques for Intelligent Information Systems*, pages 435 – 449. Physica-Verlag (Springer-Verlag), 1999.