Scalable Evolvable Hardware Applied to Road Image Recognition

Jim Torresen
Department of Informatics
University of Oslo
P.O. Box 1080 Blindern
N-0316 Oslo, Norway
E-mail: jimtoer@ifi.uio.no
Phone: +47 2285 2454

Abstract

Evolvable Hardware (EHW) has the potential to become a new target hardware for complex real-world applications. However, there are several problems that would have to be solved to make it widely applicable. This includes the difficulties in evolving large systems and the lack of generalization of gate level EHW. This paper proposes new methods targeting these problems, where a system is evolved by evolving smaller sub-systems. The experiments are based on a simplified image recognition task to be used in a roadway departure prevention system and later in an autonomous driving system. Special concern has been given to improve the generalization of the system. 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 reduction of the performance in the final system. Improvement in the generalization is shown as well.

1 Introduction

Evolvable hardware (EHW) has recently been introduced as a new scheme for designing systems for real-world applications [17]. So far the number of applications is limited.

One of the main problems in evolving hardware systems seems to be the limitation in the chromosome string length [11, 19]. A long string is normally required for representing a complex system. However, a larger number of generations is 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 [3]. The schemes involve fitness computation in parallel or a partitioned population evolved in parallel. When small applications require weeks of evolution time, 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 [8]. Another option, called function level evolution, is to evolve at a higher level than gate level [12]. Most work is based on fixed functions. However, there has been work in Genetic Programming for *evolving* the functions [10]. The method is called Automatically Defined Functions (ADF) and is used in software evolution.

Another improvement to artificial evolution — called co-evolution, has been proposed [7]. In co-evolution, data which defines the problem co-evolves simultaneously with a population of individuals solving the problem. This could lead to a solution with a better generalization than a solution evolved based on the initial data. A variant of co-evolution — called cooperative co-evolutionary algorithms, has been proposed by De Jong and Potter [9, 14]. It consists of parallel evolution of sub-structures, which interact to perform more complex higher level structures. Darwen and Yao have proposed a co-evolution scheme where the subpopulations are divided without human intervention [5].

Task decomposition for robot controllers have been proposed in several different ways. Chavas et al [4] have developed a two-stage system using incremental evolution for a neural network based controller. Another method, proposed by Lee et al [11], evolves a distributed control architecture. Every module only deals with the sensory information directly related to its particular need.

Incremental evolution for EHW was first introduced in [16] 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*. The goal is to develop a scheme that could evolve complex systems for real applications. In this paper, it is applied to a new application and several improvements in the EHW architecture are introduced. These should improve the generaliza-

tion of gate level EHW. The application is to compute the orientation of line markings in road images. This is of interest to vehicle safety systems and autonomous driving.

The next section introduces the *increased complexity evolution*, the new application and the proposed improvements in the EHW architecture. Experimental results are given in Section 3. Finally, Section 4 concludes the paper.

2 Increased Complexity Evolution

In this section, the *increased complexity evolution* scheme is proposed for an EHW-based road image recognition system. The system is evolved to detect the position of a car in the road. The evolution is based on a set of simplified road images, similar to highly pre-processed real images. The purpose of applying *increased complexity evolution* is to overcome the problem of a long chromosome string. Most of the derived principles in this section are general and could be applied to many different problems. The idea is to evolve a system gradually as a kind of divide-and-conquer method. Evolution is first undertaken individually on a set of basic units. These could be gates or higher level functions. 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.

2.1 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. It may be looked at as a division of the problem domain. 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.

Two alternatives seem possible:

- **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. In this method, the fitness function is given explicitly as a subset of the complete fitness function.
- **Partitioned training set.** A second approach is to divide 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. The fitness function would have to be designed for each task individually and used together with a global fitness function, when the tasks are operating together. This may not be a trivial problem.

2.2 Image Recognition

Image recognition is a large research field including numerous problems. Several issues make it an interesting problem to be solved by EHW:

- Operations on many pixels in an image that can be performed in parallel.
- Large amount of computation is required in real-time.
- On-line adaptation to change in light conditions etc.

The systems in mind here are for real-time image recognition. That is, images arrive to the classification system at a set frequency. Thus, a decision would have to be made on the amount of time spent for analyzing each image. If each image is going through a detailed analysis, a smaller number of images could be processed compared to a less detailed analysis.

Images input by a camera are digitized and then put through a large number of processing steps (e.g. noise removal and thinning) before the recognition can be performed. Each digitized pixel in an image has a fixed number of levels to represent its color and intensity. For many applications this could be well below 32 bits, which is the default floating point representation as further detailed in the next section.

2.3 Vehicle Safety Systems and Autonomous Driving

The target for the research presented in this paper is in the near future to design a warning system for preventing roadway departures – e.g. caused by driver inattention. In the more far future, the approach could be applied in a system for autonomous steering of a vehicle. The rest of the paper focus mainly on the latter system, as the former system may be regarded as a subset of the latter. However, some comments are included on how to apply the approach for a roadway departure warning system.

Several requirements exist for such systems [1]:

- Adaptable to different road and weather conditions.
- Reliable
- Compute a lot of data rapidly (real-time computing system).
- Low-cost

All these items are properties that can be provided by EHW. Through the use of the optimization by GA, EHW could have the potential of providing better solutions to the requirements above, than what has been possible with traditional methods and technologies. E.g., the two last items

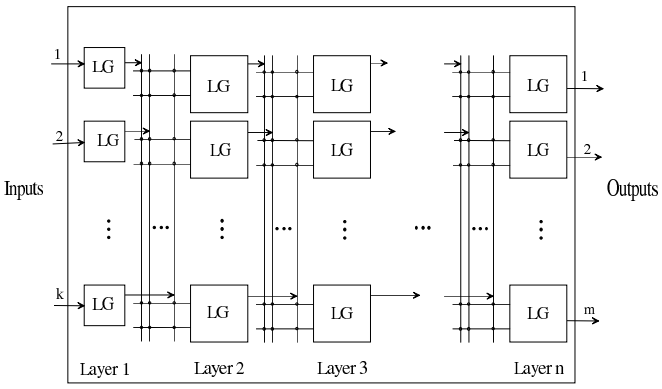
are often conflicting requirements. Traditional systems are normally based on integer (16 bit) or floating point computation (32 bits). EHW, on the other hand, could be based on one bit signals. Several such signals can be assembled into a signal with a higher resolution. This implies that a system could be designed with an adapted signal resolution. Thus, it must be shown that an application can perform well with a processing based on signals with less resolution than standard values. If this is possible, EHW-based systems could be built with less hardware than traditional systems and then provide a lower cost in commercial systems.

2.4 An EHW-based Steering Control System

A system for autonomous driving would have to consist of many sub-processing systems, since several different input sensors and outputs to units to be controlled are required. The vision system analyses the inputs and makes decisions on how the driving should proceed. One approach to driving control is partially based on *line-marking* detection, analyzing pre-processed input images where mainly only line marking are left [2]. Outputs go to steering-wheels, cruise control and braking system. This is a highly complex system. It would not be realistic to be designing such a system in one step using EHW.

The work would have to start by some limited and possible simplified part of the system. Thus, the work to be presented herein is based on a very simplified steering control system. It inputs images of lane markings. These are processed by EHW to decide the orientation of the line markings. This information can then be used to steer the wheels. No concern about braking and cruise control are given here. Further, it is assumed that there are no obstacles or other cars present in the road. The goal of these initial experiments is to show that applying EHW is of interest to the given real-world application. If appropriate, future work would have to be on more complex road images. Similar systems, e.g. ALVINN [13], have been designed using neural networks. An EHW system may not necessarily outperform these systems in performance. However, it may very well outperform them on issues like adaptability, real-time performance and cost.

The evolution will be based on gate level. However, as several output bits will be used to represent an input image, this increases the signal resolution above the normal two levels. The target EHW is an array of logic gates as illustrated in Figure 1. This array can be configured in the Xilinx XC6200. The array consists of *n* number of layers of gates from input to output. Based on earlier experience, *n* equal to four is used in the following experiments [18]. Further, 32 gates are applied in each layer of the array. Except for layer 1, the Logic Gate (LG) is either a *Buffer*, *Inverter*,



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 encoding of each logic gate in the binary chromosome string is as follows:

Input 1 (5 bit)	Input 2 (5 bit)	Function (2 bit)

The gates in layer 1 use a simplified coding consisting of only one bit to distinguish if the function is a buffer or inverter. One of the main goals of this work is to reduce the evolution time for complex problems. The number of generations required for evolving a gate array N_g could be represented as follows:

$$N_g = f(N_t, N_i, N_o, N_c) \quad (1)$$

where N_t, N_i, N_o and N_c are the number of training vectors, number of input gates, number of output gates and number of chromosome bits, respectively. In the results section, a discussion of this equation is included.

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 of such a digital system should equal the simulation of it. Any spikes could be removed using registers on the output gates.

The application to be used in the experiments is the problem of recognizing the *orientation* of line markings in road images of 8 x 4 pixels size, where each pixel can be “0” or “1”. Examples of images are seen in Figure 2. A hatched square indicates a “1”, while a non-hatched square indicates a “0”. These images are hand coded and it is assumed that

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

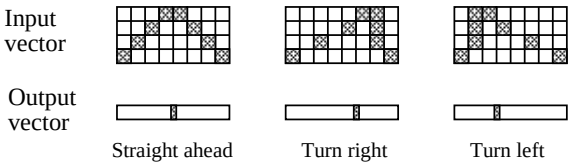


Figure 2. Examples of three training vectors.

all objects other than line markings are already removed from the images. This has shown to be possible in other autonomous driving systems, based on pre-processing the input images [2]. The set of images used in the evolution consists of 13 images. In a real system, both the image size and the gate array size would be much larger and require a much larger number of training images. Each of the input pixels is connected to *one* input gate. The output vector indicates the direction to turn the steering wheels for the corresponding input image. That is, if the line markings in the input image indicate a turn to the right, we would like the system to indicate a turn to the right on its outputs. To apply the system for roadway departure warning, a set of abnormal output vectors could lead to the activation of an alarm.

2.5 Clustered Outputs.

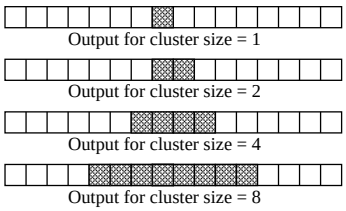


Figure 3. An input image gives a response on a cluster of outputs, when the output should present "straight ahead".

Earlier experiments — classification of characters, have shown the limitation in noise robustness of gate level circuits, when only *one* output gate is assigned to each character the system is trained to recognize [18]. Thus, it is here proposed that for each input vector, a *cluster* of nearby outputs are set to "1". Different cluster sizes are illustrated in Figure 3 for the 16 outputs used in the road image experiment. As above, a hatched square indicates that this output is "1", while a non-hatched one is "0". This figure illustrates driving straight ahead, while turn right and turn left would have a cluster shifted to the right and left, respectively. When the cluster size is larger than one, an average output value is computed based on the outputs equal to "1". Thus, when slightly different images are input — as a test of generalization, it is estimated that better performance is obtained, when more than one output gate are to be set to "1". Still, the performance will probably be below what could be obtained by floating point computation (e.g. artificial neural networks). However, a gate array hardware is a very much smaller than a floating point processor and therefore, it could provide a low-cost system. Further, the parallelized operations on pixels in an image make such a system scalable and it could process a lot more data than a program running on a single processor. By increasing the amount of data analyzed — both the number of pixels in an image and the number of images per second, it may show to be possible to use a limited precision in the computation of each pixel in the image. Thus, using a large number of two-level pixels could be as valuable as processing a lesser number of multilevel pixels. The precision level required for acceptable performance can only be found by undertaking experiments on real data.

In the automatic steering application, we do not need to classify an image into a unique class, but rather indicate the approximate orientation of the steering wheels on the road. Thus, a small deviation from the correct output (according to the training set) value is acceptable. Further, in this application, the *next* output value will always be similar to the *present* one. Therefore, to further improve the robustness of the system, a comparison of the last *previous* output values could be undertaken. This is illustrated in Figure 4, where the steering is based on the short history of former outputs.[2] In a roadway departure warning system this part will be important as well, to *predict* the probability of a roadway departure.

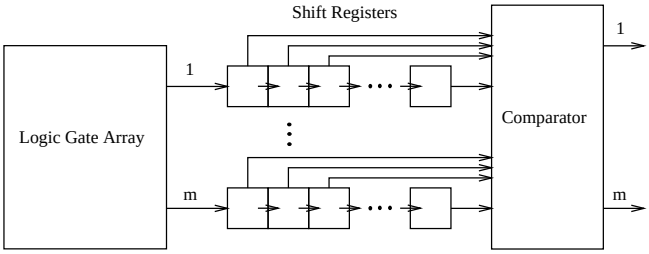


Figure 4. Including previous outputs in the present output computation.

Each output from the gate array is fed into a shift register. When the outputs of a new image are available at the outputs of the gate array, the values are shifted into the leftmost bin of the shift register. All the other stored values are simultaneously shifted one position to the right. In this operation, the value in the rightmost bin is removed from the shift register storage. Each bit in the shift register is further input to a comparator, which computes the output value mainly

[2] This feature has not been included in the following experiments.

based on the present[3] input, but can compare it with the previous outputs. Therefore, the comparator would have to include some weight values to indicate how much old outputs should be allowed to influence the present output. A strong influence prevents rapid (and maybe wrong) turns of the steering wheels, but may lead to a slow response, when a rapid turn is demanded. Another possible improvement is to weight the *inputs*. Some pixels — e.g. those in the area corresponding to the line markings, would be more important than other pixels in an image.

2.6 Increased Complexity Evolution Applied to the Road Image Experiment

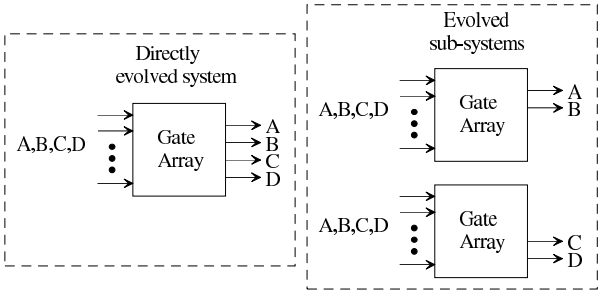


Figure 5. The increased complexity method applied for designing systems for recognizing four different images A, B, C and D. This is for an output cluster size equal to 1.

The aspect of *increased complexity evolution* is introduced by the way the evolution is undertaken as seen in Figure 5. This corresponds to the partitioned training vector approach presented in Section 2.1. 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 vectors – as shown in Figure 5. That is, each subsystem input *all* the 32 input pixels and *all* training vectors are applied during fitness evaluation. However, each subsystem has a limited number of *outputs*. 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 more easily become evolved to perform the correct operation. The assignment of output gates to a subsystem is shown in Figure 6, where the output vector is represented in the same way as in Figure 2. For this application, the integration of sub-systems are straightforward by running them in parallel. Dividing a task into simpler ones has earlier been proposed for parallel processing applied to neural networks [15]. However, the major issue in this work is that by using EHW, it is possible to make a scalable low-cost system. This is for the reasons discussed in Section 2.3.

[3]Given in the leftmost position of the shift register.

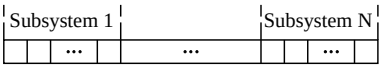


Figure 6. Assignment of outputs to subsystems.

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 complex system design issues, more than the lack of large gate arrays. In each new generation of CMOS chips, the number of digital gates available on a single chip increases substantially. 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 style – given by Goldberg [6], was applied for the evolution with a population size of 50. For each new generation an entirely new population of individuals is generated. Elitism is used, thus, the best individual from the previous generation is carried over to the present generation if no individual in the present generation is any better. The (single point) crossover rate is 0.9, thus the cloning rate is 0.1. Roulette wheel selection scheme is applied. The mutation rate - the probability of bit inversion for each bit in the binary chromosome string, is 0.001.

The fitness function is equal to the sum of the number of correct outputs for each training vector, summed for all the vectors in the training set. That is, for each output gate of the gate array that matches the output specified in the training set, 1 is added to the fitness function.

For each experiment presented in the next section, ten different runs of GA were performed. For each run, when a circuit for which the maximum score is obtained, the evolution is stopped and the number of generations is stored for comparison with the other runs. For the ten runs, quite often only a few of them resulted in a correctly classifying circuit. This was due to GA reaching the set maximum number of generations before a correctly classifying circuit had been found. Thus, it is impossible to compute any realistic average or variance analysis for the ten runs. Therefore, only the number of generations for the run requiring the least number of generations of the ten different runs is reported in the result section. This circuit may not be providing the

same (or as good as) performance as a circuit evolved by a larger number of generations. However, this topic is to be considered in future work.

3 Results

In this section, various aspects of evolvable hardware are investigated through different experiments. The results presented herein are the first to be published on the *increased complexity evolution* applied to road image detection. Thus, there are three goals of the work. First, to show the benefits of the *increased complexity evolution* method and second, to show the benefit of applying clusters of outputs on the generalization of gate level EHW. Third, to see if EHW seems applicable for the real-world application of autonomous driving.

Cluster size	1	2	4	8
# of generations	13174	14466	>30000	>30000

Table 1. The results of evolving circuits for classifying the road image training set (direct evolution).

Initially, we would like to see if there are any difference in the number of generations required for different output cluster sizes. Table 1 shows the number of generations required for direct evolution of a system, when the cluster size varies. The evolution stops when a circuit outputs correct values according to the training set. The number of generations increases with the cluster size. This is understandable since the gate array would have to overlap the output values equal to "1" for a larger number of different images as the cluster size increases. Still, we would like to apply the system with the best performance rather than the one that could be evolved in fewest possible generations.

Next, we would like to find out if the *increased complexity evolution* results in a lower number of generations. The results are given in Table 2 for a cluster size equal to four. If the evolution is undertaken in one operation no circuit is found[4] that correctly classifies all training images. A system based on real images would definitely not be evolvable in one operation, since there are difficulties in evolving a system in one operation for this simplified application. As the number of outputs to be evolved is reduced, the number of generations is decreased. In average, for each diving by two undertaken, a ratio of about four times less generations is required. This is the same as for an earlier conducted character recognition experiment [18]. A larger number of generations is required for sub-systems classifying center outputs than those on the left side and right side. This is due to the fact that many training vectors have their output clusters in the center area.

When an input image, which should not lead to a response on any of the outputs from a sub-system, the subsystem 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.

Equation (1) gives a relation between various parameters and the number of generations required. In the experiments conducted here, only N_o and N_c varies. Thus, they are included Table 2. The column with N_oN_c lists the product of these two numbers. An interesting observation from the experiments are that:

$$N_g \approx \frac{1}{4} 2^{ln_2 N_o} N_o N_c \tag{2}$$

In these experiments, the N_o is the major contributor to N_g. Thus, the chromosome string needs to be evolved to solve a more complex problem as N_o increases. Further, the chromesome string length N_c increases as well.

Cluster size	1	2	4	8
Percentage error	39.0	4.0	4.0	19.0

Table 3. The results of applying a test set to the evolved systems.

An important feature of an image recognition system is the generalization ability. To test this aspect, a separate test set was made. It consists of 26 images (twice the number used for the evolution). It is made by modifying 1 to 5 pixels near the line markings in the training set images. The test set was applied to test how well the systems classify slightly different patterns from those used in the evolution. Only if the output cluster[5] is more than one output away from the correct one, the image is said to give a wrong output. This is possible since we aim at having a response in the correct area, rather than at the exact output. Table 3 shows the importance of applying the clustered output method, when the test set is employed. A cluster equal to one gives the worst performance. Thus, demanding a single output to be one is vulnerable. For some images *all* pixels became "0", when the cluster size was equal to one and two. This often leads to a large (and unacceptable) error as seen in Table 4. This is a major failure that could not be accepted in reliable system. The best cluster size is equal to four, where the *maximum* error is 1.5. The computation of the cluster average could probably be improved in such a way that larger cluster sizes could provide better performances in the cases when the output cluster sizes are larger or smaller than expected. Thus, these results indicate that it should be worth testing the scheme on real images.

[4] The maximum number of generations was set to 30000.

[5] A cluster avarage value is computed based on the nearby outputs equal to "1". A "0" in between two "1"s are regarded as a "1".

Number of (sub)systems	N_c	N_o	N_oN_c	Number of generations	Total (N_g)
One (1,...,16)	992	16	15872	>30000	>30000
Two (1,...,8),(9,...,16)	896	8	7168	8579,4749	13328
Four (1,...,4),...,(12,...,16)	848	4	3392	59,1389,2720,58	4226
Eight (1,2),...,(15,16)	824	2	1648	9,16,111,179,475,163,35,1	989
Sixteen (1),...,(16)	812	1	812	1,1,1,1,10,44,15,21,44,32,1,42,1,1,1,1	217

Table 2. The results of evolving circuits for classifying the road image training set, for an output cluster size of four.

	Difference between correct and predicted output values											
Cluster size	0	±0.5	±1.0	±1.5	±2.0	±2.5	±3.0	±6.0	±7.0	±8.0	±9.0	±10.0
1	13		3		2			2	1	1	4	
2	15	7	3									1
4	14	2	9	1								
8	15	4	2		2	1	2					

Table 4. The number of test vectors with a certain deviation from the correct output value.

4 Conclusions

In this paper, a scheme, called *increased complexity evolution*, is proposed for an EHW-based system for road image classification. The system is to be applied in a roadway departure warning system and later in an autonomous driving system. Clusters of outputs are used to improve the generalization ability of gate level circuits. The method is based on sub-system evolution for the design of complex systems.

The experiments are based on simplified images. There are several important results from the simulations. First, the total number of generations can be substantially reduced by evolving sub-systems instead of a complete system in one operation. When there are difficulties in finding a correct working system for this simplified application, a system based on real data will be even worse to evolve in one operation. Second, by applying clusters of outputs instead of single outputs, the generalization and reliability of gate level circuits are improved. The robustness could be further improved as indicated in the paper. Third, the road image detection scheme is promising and experiments on real images should be undertaken next.

Acknowledgements

The author would like to thank the group leader Dr. Higuchi and the researchers in the Evolvable Systems Laboratory, Electrotechnical Laboratory, Japan for inspiring discussions and fruitful comments on my work, during my visit there in January-April 2000.

References

[1] M. Bertozzi and A. Broggi. Vision-based vehicle guidance. *Computer*, 30(7):49–55, July 1997.

[2] A. Broggi et al. Argo and the millemiglia in automatico tour. *IEEE Intelligent Systems*, pages 55–64, Jan-Feb 1999.

[3] E. Cantu-Paz. A survey of parallel genetic algorithms. *Calculateurs Parallels*, 10(2), 1998. Paris: Hermes.

[4] J. Chavas et al. Incremental evolution of neural controllers for robust obstacle-avoidance in khepera. In P. Husbands and J.-A. Meyer, editors, *Proc. of the First European Workshop on Evolutionary Robotics, EvoRobot98.* Springer, 1998.

[5] P. Darwen and X. Yao. Automatic modularization by speciation. In *Proc. of 1996 IEEE International Conference on Evolutionary Computation*, pages 88–93, 1996.

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

[7] W. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In *Physica D*, volume 42, pages 228–234. 1990.

[8] 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.

[9] K. D. Jong and M. Potter. Evolving complex structures via co-operative coevolution. In *Proc. of Fourth Annual Conf. on Evolutionary Programming*, pages 307–317. MIT Press, 1995.

[10] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. The MIT Press, 1994.

[11] W.-P. Lee et al. Learning complex robot behaviours by evolutionary computing with task decomposition. In A. Brink and J. Demiris, editors, *Learning Robots: Proc. of 6th European Workshop, EWLR-6 Brighton.* Springer, 1997.

[12] 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.

[13] D. Pomerleau. *Neural network perception for mobile robot guidance.* Kluwer Academic Publishing, 1993.

[14] M. Potter and K. D. Jong. Evolving neural networks with collaborative species. In *Proc. of Summer Computer Simulation Conference*, pages 340–345, 1995.

[15] N. Sundararajan and P. Saratchandran. *Parallel Architectures for Artificial Neural Networks.* IEEE CS Press, 1998. ISBN 0-8186-8399-6.

[16] 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.

[17] 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.

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

[19] X. Yao and T. Higuchi. Promises and challenges of evolvable hardware. 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.