Design and Implementation of Scalable Online Evolvable Hardware Pattern Recognition Systems

Ph.D. thesis

Kyrre Glette

October 13, 2008

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
Nr. 790

ISSN 1501-7710

Cover: Inger Sandved Anfinsen.
Printed in Norway: AiT e-dit AS, Oslo, 2008.

Produced in co-operation with Unipub AS.
The thesis is produced by Unipub AS merely in connection with the
thesis defence. Kindly direct all inquiries regarding the thesis to the copyright
holder or the unit which grants the doctorate.

Unipub AS is owned by
The University Foundation for Student Life (SiO)

Abstract

Evolvable hardware (EHW) is a method where hardware is designed and/or modified automatically by optimization algorithms called evolutionary algorithms (EAs). The results so far are promising but somewhat limited partly because of a scalability problem of the EHW design method. It is hard to generate circuits which are capable of handling large real-world problems in a competitive manner. Schemes for dealing with the scalability problem have been proposed earlier, however the application of these to run-time adaptive EHW systems has been limited.

This thesis addresses the challenge to generate autonomous run-time adaptive digital EHW systems for solving large real-world problems. The challenge consists of dealing with the lack of scalability in EAs, combined with the challenge of designing an adaptive hardware architecture for the evolution.

Specifically, hardware classifiers are developed for accurate classification of inputs with a large number of features. For experimentation with online adaptation, an on-chip evolutionary system has been proposed and implemented, making use of an on-chip processor. In order to overcome the scalability problem, the use of data buses and high-level building blocks has been investigated. Further, these elements have been combined with incremental evolution into a specialized high-speed classifier architecture for online evolution. The architecture has been applied to several difficult application benchmarks and compared to traditional approaches as well as previously presented EHW approaches.

The work has resulted in a flexible system for on-chip evolution. The proposed online architecture is capable of classifying problems with a larger number of inputs than previous online EHW classifiers, and it gives a higher accuracy for these problems than previously presented EHW systems. These systems have often been based on offline evolution – regarded as less challenging than online evolution. In addition, the amount of evaluations needed for the evolutionary search is low compared to previously presented systems. The system has also shown to be competitive to traditional classification approaches for the applied benchmarks.

Preface

This thesis has been submitted to the Faculty of Mathematics and Natural Sciences at the University of Oslo in partial fulfillment of the requirements for the degree of Philosophiae Doctor (Ph.D.).

The research project has been funded by the Research Council of Norway through the project *Biological-Inspired Design of Systems for Complex Real-World Applications* with project number 160308/V30. The research has been conducted at the Microelectronics group and later the Robotics and Intelligent Systems group at the Department of Informatics during the period 2004-2008, under the supervision of Prof. Jim Tørresen and the co-supervision of Assoc. Prof. Mats Høvin. In 2006 the author stayed for six months as a visiting researcher at the Integrated Systems Laboratory, University of Tsukuba, Japan. The head of the laboratory, Prof. Moritoshi Yasunaga, functioned as a host and a co-supervisor during this period. In 2007 a cooperation with Prof. Marco Platzner and Mr. Paul Kaufmann at the University of Paderborn was initiated. The cooperation was funded by the German Academic Exchange Service / Research Council of Norway through the collaborative project *Autonomously Adaptable System-on-Chip* under project number D/06/12740. During 2007 and 2008 several research meetings of short duration have been conducted in both Germany and Norway.

Acknowledgements

Firstly, I would like to thank my supervisor Prof. Jim Tørresen for all the valuable advice and support given during the research process. I would also like to thank my co-supervisor Assoc. Prof. Mats Høvin for constructive feedback and interesting discussions. Further, I wish to thank Prof. Moritoshi Yasunaga for hosting and supervising me during my stay at the University of Tsukuba, and Assoc. Prof. Yoshiki Yamaguchi for giving valuable feedback on my research. I would also like to thank Prof. Marco Platzner and especially Paul Kaufmann at the University of Paderborn for a very good collaboration. Further, I wish to thank Dr. Bernhard Sick and Thiemo Gruber at the University of Passau for contributing with their expertise on classifiers.

Moreover, I would like to thank my colleagues at the Nanoelectronics and the Robotics and Intelligent Systems groups at the University of Oslo for providing a friendly working environment, and the people at the Integrated Systems Laboratory at the University of Tsukuba for welcoming and inspiring me. In particular I would like to mention Marcus Furuholmen and Hiroyuki Kawai, with whom I have spent many office and lab hours. Finally I would to thank my friends, my family and Natalia for optimism and support.

Contents

Chapter 1

Introduction

This chapter introduces the thesis. The motivation for the research is presented, and research objectives are formulated. An outline of the thesis is presented.

1.1 Introduction

Evolvable hardware (EHW) is a relatively young research field concerning automatic tuning or design of electrical circuits or other kinds of hardware. By specifying a target behavior, good circuits according to this specification can be found through the means of a search. Candidate circuits are defined, through mappings, by strings upon which the search is performed. Although in principle any kind of optimization algorithm could be employed, evolutionary algorithms (EAs) have been found efficient for this kind of search, giving rise to the name evolvable hardware.

By employing EHW as a design method, new and original solutions can be found to problems, as a product of the search exploring methods not thought of by human designers. These solutions could prove to give higher performance than traditional human-designed solutions in aspects such as circuit speed, resource usage or functionality. Such a search would also be useful and efficient for tuning or optimizing systems with many (interdependent) parameters, where the alternative is a lengthy manual tuning process with no established approach. In addition, EHW gives the possibility of systems which at run-time, that is, while the system is operational, can adapt themselves to changes in requirements or the environment. In a similar manner, fault-tolerant systems could be devised, repairing themselves if malfunctions occur.

Several applications of EHW have been proposed, of which some have been very successful. Examples include data compression for printers [33], analog filters [18], evolved image filters [28], and evolved shapes for space antennas [22]. However, many of the applications so far have been on limited, simple problems, with little comparison to traditional approaches. One of the reasons for this is the well-known scalability problem of the evolutionary search. As the circuits to be generated become larger, the string describing the circuit also grows, effectively increasing the search space dimensionality. This gives rise to an above-linear increase in computational effort with the circuit size.

Several methods have been proposed for dealing with the scalability problem. An obvious but limited approach is to increase the amount of computing power available for the search, by the use of multiple processors or dedicated hardware. Some approaches try to make the bit string length variable, only coding for the active parts of a circuit, and therefore speeding up the search. Another approach is to make high level building blocks available to the evolution, instead of logic gates. These building blocks could be pre-specified or automatically defined. The approach of incremental evolution consists of dividing the application into smaller sub-problems, which can be more easily evolved, before assembling the subsystems into a complete system. Further, by inputting a priori knowledge to the evolutionary process, by limiting the freedom of the search, desired solutions could be found more easily. The study of artificial development, which consists of an advanced mapping process from the bit string to the final circuit, could be a promising approach for larger circuits. However, few "useful" circuits have been found by this method so far.

This thesis aims at developing systems for large and/or difficult real-world problems. This includes investigating feasible methods for scalable evolution *and* run-time adaptation for hardware systems. These two requirements impose limitations on what kind of schemes that can be employed for dealing with the scalability, and much of the work in this thesis deals with investigations in this field. Finally, it is a goal to make systems which are better than previous EHW approaches and competitive with traditional methods. Experiments are conducted by means of software simulations and hardware implementations with prototyping on field programmable gate arrays (FPGAs). Standard realistic benchmarks for traditional solutions are employed in order to get impressions of performance for real-world systems and to compare with the performance of traditional solutions.

1.2 Research Objectives

The main research objective for this thesis is:

> *Develop online evolvable hardware schemes that could evolve systems for large real-world applications.*

This can be divided into several sub-goals:

- Develop scalable evolvable hardware schemes for solving large and/or difficult problems.
- Experiment with on-line evolvable systems
- Demonstrate that the developed systems are *competitive* for selected applications.

1.3 Thesis Outline

The thesis is a *collection of papers*. The research contribution of this thesis is thus constituted of the seven included research papers, in their original publication format. The rest of the thesis is organized as follows: Chapter 1 (this chapter) gives an introduction to the thesis. Chapter 2 presents relevant background information for the research. Chapter 3 summarizes the research process and gives an overview over the papers constituting the research contribution. Chapter 4 presents conclusions and proposes future work.

Chapter 2

Background

This chapter describes background information relevant to the work presented in the thesis.

2.1 Evolutionary Algorithms

EAs are a set of optimization algorithms inspired by evolution in nature. The most commonly known variants are genetic algorithms (GAs), genetic programming (GP), evolution strategies (ESs) and evolutionary programming (EP). EAs are inspired by natural evolution in that they are often population-based, adopting heritance, selection and variation mechanisms.

2.1.1 Algorithm Outline

An outline of an EA can be described with the following steps:

1. Generate a population of individuals with random genomes
2. Perform a genotype-phenotype mapping process on each individual
3. Evaluate the obtained phenotypes with regard to a fitness function and assign scores to individuals
4. If the termination criterion is reached, terminate the algorithm
5. Select individuals according to fitness scores and apply genetic operators in order to create a population of new individuals
6. Repeat from step 2.

Possible solutions, also called *individuals*, are usually coded in a string of symbols, called a *genome* or *chromosome*. At the starting point of the algorithm, a collection of individuals – a *population* – is generated with random genomes. In order to evaluate the solutions coded in the population, a mapping or decoding process is usually performed. This process transforms the genome into a candidate solution which can be evaluated, also called a *phenotype*.

The *fitness function* gives a measure of how good the solution is performing for the target task. In order for the search to perform well, it should be possible to grade different solutions with a high resolution, i.e. the output should not just be "good" or "bad". If the fitness of the solutions is measured with regard to multiple qualities, it is called *multi-objective optimization*.

There are several ways to select individuals for creating a new population. A common way is to use fitness-proportionate selection, where an individual is given a probability of being drawn for reproduction proportional to the individual's fitness score. An alternative to this is tournament selection where a number of individuals are drawn randomly and the individual with the highest fitness score of this selection is chosen for reproduction.

The selected individuals are subject to variation operators before they are inserted into the population of individuals constituting the new generation: The most common form of variation operator is called *mutation*, where a small change of the original genome is performed. Recombination is simulated using the *crossover* operator - where parts of one individual are exchanged with another individual. A common way of speeding up the EA is to include *elitism* – this consists of directly transferring the fittest individual of the current generation to the next generation without any variation operators applied. This strategy of always keeping the best solution found so far often speeds up the search considerably. However, this depends on the context and there could be situations where elitism is not advantageous, or even contributing to a too early convergence.

One of the most popular variants of EAs, GA, was introduced in [13]. For this approach, the genome consists of a string of consecutive genes, often single bits, which can be mapped into a phenotype. In this case mutation consists of inverting one or a few bits. However, it is also possible to operate on higher-level components such as integer or floating point data types, with more advanced mutation operators. The crossover operation has several variations on exchanging chunks of genes between two genomes.

Another popular version of EAs is the family of algorithms known as ESs. ESs were originally defined for numerical optimization over real numbers. The algorithms differ from GAs in that only the best solution from the current generation is kept and used as a basis for the new generation. The new generation is filled up with individuals which are mutated variations of the best individual. In addition an elitist strategy of keeping the best individual is applied. Mutations are decided by adding normally distributed values to genes, however in simpler variants where only bits are used, bit inversion is applied like with GAs. ESs can also have the property of self-adaptation, where parameters (typically mutation parameters) are adapted during the evolution.

2.2 Evolvable Hardware

This section gives relevant background information on the field of EHW.

2.2.1 Principle

EHW originated in [4, 10] as a method for automatically designing hardware by the use of EAs. The principle involves decoding the genome bit string from an EA into a hardware circuit phenotype, and then testing the resulting circuit on a fitness function. The fitness function dictates the desired functionality of the circuit, and would typically contain desired input/output behavior. It is however possible to also include other aspects in the fitness function, for example circuit size, fault-tolerance, or power consumption.

The research field of EHW can be categorized into two main interest fields, presented below:

- *The biology-focused view*. This view concentrates on the biological aspects and possibilities offered by modeling biological systems in HW. This includes using EHW to build biologically-inspired structures such as virtual cells [41], artificial immune systems [2] and artificial brains [5]. In addition, this view experiments with possible substrates for hardware evolution [9, 34]. The development stage of hardware systems is also under research [40]. Research in this area focuses primarily on biological mechanisms which can eventually lead to techniques for improved HW design, and focus on real-world applications is thus so far limited.

- *The application-focused view*. The other field of interest for EHW researchers is to create EHW systems which can perform tasks better than traditional hardware or software approaches. This area of interest thus focuses on real-world applications and evolution is therefore more directly employed as a search/invention tool in the engineering of hardware systems. This includes using evolution for invention of new hardware (circuits) to be included in conventional systems [20, 22, 25]. In the other end of the scale, evolution is used as a search algorithm for modifying or tuning parameters of already deployed, reconfigurable, hardware [12, 32].

2.2.2 EHW Variants

Approaches for EHW fitness evaluation can be divided into two main categories: *Offline* and *online* fitness evaluation. *Offline fitness evaluation*, also called *extrinsic EHW*, evaluates the fitness of the hardware by simulating it on a host computer, which could be the same computer as the one running the EA. Actual hardware is produced only after evolution is terminated and a best solution is found. *Online fitness evaluation*, also called *intrinsic EHW*, performs fitness evaluation on the target hardware. This usually requires having some kind of reconfigurable hardware, such that several possible solutions can be evaluated on the same physical piece of hardware, one at a time. The online approach has three possible advantages over the offline approach: Firstly, evaluating real hardware could significantly reduce the time needed for evaluation compared to a simulation. Secondly, by evaluating on real hardware, physical aspects not covered by the simulator could be explored by evolution. Thirdly, the evolved solution is known to be valid since it is evaluated on the real system and not on a possibly inaccurate model or simulator.

When evolving hardware for a chip, online evolution can be further divided into *off-chip evolution* and *on-chip evolution*. In the case of off-chip evolution, the EA is running on a system separated from the device containing the reconfigurable circuit. However, with on-chip evolution the EA is implemented to run on the same chip as the reconfigurable circuit itself, giving a self-contained EHW chip. This gives the possibilities of having a compact system which could constantly re-evolve and adapt during the application lifetime of the system. In this case it would be possible to implement the EA completely in hardware – giving complete

HW evolution, or the EA could be running in SW on an embedded processor with optional HW acceleration of time-consuming tasks.

In order to have online fitness evaluation, reconfigurable hardware is the only practical approach. In addition, it is desirable to have a short reconfiguration time in order to not slow down the evolutionary process. Although FPGAs are popular reconfigurable devices, going from a high-level circuit description to an FPGA configuration bitstream is normally a lengthy process. This involves invoking vendor-specific tools for implementing the design by a multi-step process including steps such as synthesis, place and route, mapping and floorplanning. Earlier FPGA models (such as the XC6200 series by Xilinx) allowed direct bitstream manipulation, where no combination could be harmful for the FPGA. In addition, the coding of the bitstream was publicly available. This made it possible to run evolution directly on FPGA bitstreams, allowing quick reconfiguration and evaluation [34]. However, most modern FPGAs do not allow this kind of direct bitstream manipulation, and have a proprietary, undisclosed bitstream format[1].

In some cases it is still possible to modify the contents of lookup tables (LUTs) inside the FPGA, by reverse engineering of the bitstream, by utility functions, or even by configuring the LUTs while in a special shift register mode. The first two approaches can be combined with the use of the Internal Configuration Access Port (ICAP) found in recent Xilinx Virtex FPGAs. This allows for one part of the design to reconfigure another part of the chip with a partial bitstream, and is an interesting possibility for EHW. Work on reverse engineering the bitstream in order to reconfigure LUTs through the ICAP in EHW systems has been performed in [42]. Another EHW approach using the ICAP and Xilinx utility functions in order to reconfigure LUTs can be found in [23]. The third EHW approach shifts configuration data into LUTs by using the shift register behavior of Xilinx Virtex LUTs [39].

An alternative approach to modifying the configuration of the FPGA itself is to implement a circuit with user-defined reconfigurability on the FPGA. Functionality could in this case be modified by for example using registers to control the multiplexing of different outputs. By writing to these registers fast reconfiguration can be obtained. This virtual reconfiguration approach also has the advantage of being less technology-dependent than direct bitstream manipulation methods, and a higher reconfiguration flexibility can be obtained. A disadvantage of the approach is the increased resource usage overhead associated with having to implement all possible connections and functionality. This method has been termed virtual reconfigurable circuit (VRC) [30] or virtual FPGA [8] in earlier work.

Another approach to reconfigurable HW in the EHW context is to design a custom reconfigurable ASIC. This allows for full control of the reconfigurability and the reconfiguration process, making it possible to have chips tailored for online evolution. Custom ASIC implementations of reconfigurable circuits are proposed in [27] and [31].

2.2.3 Scalability

An often pointed-out challenge about EHW is what is referred to as the *scalability problem*. Especially with direct genotype-phenotype mappings, an increase in the size of the circuit requires

[1]Detailed bitstream documentation is available for some recent devices, such as Atmel FPSLIC FPGAs, allowing for direct bitstream manipulation [24].

a proportional increase in the size of the genome. This is equivalent to an increase of the search space for the EA. Thus, a linear increase in circuit size gives an exponential increase in possible searchable solutions. Although EAs may have a better performance on searches of high dimensionality than other metaheuristics, it is evident that the described phenomenon limits the efficiency of the search for larger circuits. Consequently, few circuits of large size have been evolved. As an example, the experiments with evolving digital multiplier circuits by Miller et al. [25] were successful with evolving relatively small circuits, such as 2-bit multipliers. 3-bit multipliers were also successfully evolved, however this required a much higher computational effort. The practical limit to the size of directly evolvable multiplier circuits seems, from the experiments in [43], to be 4 bits.

Given the scalability limitation, researchers have experimented with several approaches which could evolve larger EHW systems. A brute-force method would be to make more *computational power* available to the EA, by techniques such as parallelization [3]. However, given the high increase in computing time observed even for small problems, such an approach would probably still be limited by practical reasons.

Experiments have been conducted using a *variable length genome* [14]. This approach consists of storing in the genome only the connections which contribute to the functional circuit. The idea is to keep the genome shorter and has shown through experiments to evolve large circuits more quickly than a traditional circuit representation.

Another approach consists of using *function level evolution* as opposed to gate level evolution. By evolving circuits with higher level building blocks than logic gates, less information is required in the genome in order to achieve the same level of functionality. This approach was employed for online EHW in [26]. Here, an array of fixed functions such as floating point addition and multiplication was available for the evolution of various test circuits. Function level evolution has also been applied successfully to the online evolution of image filters and other image operators [28]. A set of functions which operated on 8-bit integer pixel values were organized in an array-style configuration.

Experiments have also been conducted on using *automatically* defined building blocks. The principle of automatically defined functions (ADFs) from GP [21] has been applied to the evolution of analog circuits [19]. An alternative approach, called embedded cartesian genetic programming (ECGP), has been applied to the evolution of digital circuits [44]. Both approaches select portions from a solution at random and create building blocks from them. These building blocks are then available to evolution as single primitives. Both of the approaches to automatically defined building blocks have concentrated on offline evolutionary experiments.

Another approach to the scalability problem has been to limit evolution to subsystems which are evolved one at a time before assembling them into a final system [37]. This approach, called *incremental evolution*, has the effect of shortening the genome compared to what would have been required in order to evolve a complete system directly. Thus, the dimensionality of the search space can be reduced. In addition, by letting subsystems concentrate on different subtasks the search space can also be *simplified* (i.e. making it easier to find a solution) compared to a system which needs to be evolved with all functionality simultaneously [35]. The incremental approach requires that the systems to evolve are divisible, i.e. their functionality can be partitioned and measured at a subsystem level. Such a partitioning can be decided manually and will in some cases be a logical consequence of the application, e.g. for classification applications, where one category to be classified could be the task of one subsystem. However, work has also been undertaken on automatic partitioning [17]. This process consists of breaking down circuits into sub-circuits and combining them again in order to achieve not only evolvable but also compact circuits. Incremental evolution has been applied to the digital multiplier problem [36], resulting in multipliers having more inputs than those evolved with a direct evolutionary approach.

Another approach to the scalability problem is called *artificial development*. This approach is inspired by the development from genomes to multicellular bodies (phenotypes) observed in nature, and consists of having indirect mapping processes from the genome to the candidate solution. For EHW one possible way of applying development would be to evolve programs at the genotype level. By executing the program (the genotype) a circuit (the phenotype) is constructed. Such an approach, where the genotype is a GP program, can be seen in [20]. One property which could be desirable for the evolution of digital circuits would be the possibility to generate large circuits consisting of recurring elements and regular patterns, such that circuits for an arbitrary number of input elements could be constructed. Developmental processes could be useful for evolving these kinds of structures. Research has been undertaken in [1, 29] where development is employed in order to evolve arbitrarily large sorting networks and multipliers. An alternative approach to artificial development for EHW concentrates on models closer to biology [38].

2.2.4 EHW for Pattern Recognition

Pattern recognition consists of classifying input data into different categories (classes). This process, also referred to as classification, has often been an application of EHW. Pattern recognition systems could benefit from EHW in having a high speed of classification, as well as being contained in a compact and energy-saving circuit. Further, pattern recognition systems could benefit from run-time adaptation, where EHW could have an advantage. Another reason for EHW researchers of applying EHW to classification could be the use of classification as a *benchmark*: Classification problems come in a large variety of sizes and difficulties in terms of number of inputs, number of classes and the difficulty of separating these. It is thus a possible application for comparing EHW methods by looking at the size and difficulty of the problems the circuits can handle. Further, classification benchmarks exist which make it more practical to compare with results from traditional approaches.

An early use of EHW for pattern recognition was reported in [11]. Originally, the architecture was applied to character classification but later on used for classification in a prosthetic hand controller [15, 16]. The classifier architecture is a programmable logic array (PLA)-like structure of AND gates followed by OR gates. The configuration of the architecture was evolved using a GA implemented on the same chip as the classifier, resulting in an online adaptable system. The system had 16 input feature bits, could classify six different categories, and its classification accuracy was shown to be competitive to an artificial neural network (ANN).

Experiments on two-phase incremental evolution of an EHW architecture applied to prosthetic hand control (PHC) were presented in [35]. The two-phase approach consists of first evolving category subsystems separately and then assembling them, through evolution, in a second phase. The results showed that the approach can lead to a better generalization performance than both traditional direct evolution and ANNs.

EHW classification architectures applied to domains other than PHC include, for example, the function level evolution of [26]. This architecture was applied to typical ANN applications, however, with fewer inputs and outputs, and attained accuracies comparable to ANNs.

A different EHW pattern classification system, Logic Design using Evolved Truth Tables (LoDETT), was presented in [45, 46]. LoDETT allows for high accuracy classification on problems with a much higher number of inputs and outputs. An example is face image recognition with 512 inputs and 40 different categories. Although providing high classification accuracy, also outperforming an ANN, the system lacks the ability of online evolution and relies on synthesis in software before the circuit is implemented on an FPGA. The approach utilizes incremental evolution, i.e., sub-circuits are evolved separately before being assembled into a final system.

Chapter 3

Research Summary

This chapter summarizes the research conducted in the proposed thesis. First, a brief overview of the research process is given. Then motivations and abstracts for each included paper are presented. Finally, the publications produced during the project are listed.

3.1 Overview

The work started with the sub-goal of *experimenting with online systems*. Paper I addresses the establishment of an on-chip self-adaptable evolutionary platform. Next, focus was shifted to the sub-goal of *developing schemes for solving large problems*. Here, preliminary explorations were undertaken in Paper II, while a more scalable solution was presented in Paper III. Paper IV focused on an improved evolutionary platform for the architecture presented in Paper III. While Paper III initially covered a realistic application, further applications were explored in Papers V and VI. At this stage, the focus was also shifted towards evaluating the *competitiveness* of the proposed system, and Paper VI included a comparison with state-of-the-art conventional methods. Paper VII further focused on evaluation of the system, this time against other EHW schemes.

3.2 Papers

This section presents details on the motivations and contributions for each paper together with the paper abstracts.

3.2.1 Paper I

A Flexible On-Chip Evolution System Implemented on a Xilinx Virtex-II Pro Device

As online EHW systems for real-world applications were one of the main goals of the research, it was desirable to perform experiments with real hardware. This paper addresses the development of an FPGA-based hardware platform which could serve as a base for further experiments. With evolution speed and embedded applications in mind, a single-chip solution, capable of performing evolution and fitness evaluation on the same chip, was developed. While having an on-chip system with online fitness evaluation it was still desirable to have the possibility for flexible experimentation with the EA itself. It was therefore chosen to implement parts of the EA in SW running on an embedded processor core on the target FPGA.

The work resulted in a novel on-chip EHW system which allowed for flexibility by combining the use of an on-chip processor for the EA and reconfigurable logic for online fitness evaluation.

Abstract

There have been introduced a number of systems with evolvable hardware on a single chip. To overcome the lack of flexibility in these systems, we propose a single-chip evolutionary system with the evolutionary algorithm implemented in software on a built-in processor. This architecture is implemented in a Xilinx Virtex-II Pro FPGA with an embedded PowerPC processor. This allows for a rapid processing of the time consuming parts in hardware and leaving other parts to more easily modifiable software. This platform will be beneficial for future work regarding both cost and compactness. Experiments have been performed on the physical device with software running in parallel with fitness computation in digital logic. The results show that the system uses only twice as much time when compared to a PC running at 10 times faster clock speed.

3.2.2 Paper II

On-Chip Evolution Using a Soft Processor Core Applied to Image Recognition

With the experimental platform from Paper I in place, research could focus more on challenges in applications and EHW schemes. Pattern recognition was recognized as an application with properties of having important real-world applications as well as being a good candidate for testing the scalability of an architecture/method. Several pattern recognition benchmarks built from realistic data exist, which vary in number of input features, number of classes, and difficulty. In this paper, an architecture consisting of functional units arranged in a rectangular array, popular from earlier EHW approaches, was employed to a face recognition application. However, the architecture operated with high-level functions on picture elements (pixels) in an attempt to make evolution feasible on the high number of inputs, as well as to achieve a good classification accuracy. In addition, it was experimented with using a "soft" processor core instead of an embedded "hard" core, in order to potentially expand the platform to FPGA devices not containing embedded cores.

The work resulted in a step towards an online EHW pattern recognition system capable of many inputs by using function level evolution.

Abstract

To increase the flexibility of single-chip evolvable hardware systems, we explore possibilities of systems with the evolutionary algorithm implemented in software on an on-chip processor. This gives higher flexibility compared to implementing an evolutionary algorithm directly in hardware, since the parameters and behavior of the algorithm can easily be changed, and complex operators are more feasible to implement. In this paper a Xilinx MicroBlaze soft core processor is used, and the system is implemented in a Xilinx FPGA. A suitable hardware architecture for image recognition has been proposed, and it is applied to a face recognition task. Data buses and higher level functions have been utilized in order to reduce the search space for the evolutionary algorithm. Experiments have been performed on the physical device, with software running in parallel with fitness computation in digital logic. Results show that the MicroBlaze system evolves at half the speed of a Pentium M system running at 17 times the FPGA clock frequency. The distinction of a certain face from others is performed at 94.9% accuracy. In addition, the possibilities for evolutionary adaptation over time are explored by introducing changes in the training set. The system shows ability to adapt to these changes.

3.2.3 Paper III

Online Evolvable Pattern Recognition Hardware

This book chapter will be included in the book *Evolutionary Image Analysis and Signal Processing (provisional title)* and is an extension of the paper *An Online EHW Pattern Recognition System Applied to Face Image Recognition* [6].

Although the architecture proposed in Paper II was successful for classification on a single category, it is not straightforward to scale it to classify several categories. Firstly, the resource usage for the virtual circuit proposed is high, and thus an increase up to the number of categories required (40) would not be very practical for the image recognition task described. The array-like structure employed is very general and many functional units would be unused in each circuit. Secondly, even if the circuit could be duplicated to the number of categories required, there would be a problem distinguishing the outputs since the nature of the utilized functions lead to each circuit outputting values equivalent to "true" or "false". Research effort was therefore put into a more scalable architecture. Inspired by Yasunaga's LoDETT system, with several possible matching rules for each category, a specialized classification architecture for online evolution, the *functional unit row (FUR)* architecture, was developed. Although bearing similarities to the LoDETT system, it also distinguishes itself in several ways. Firstly, the proposed system allows for online evolution while LoDETT is based on offline evolution. Secondly, the classification "rules"/rows in the proposed system are based on functions different from the LoDETT system. Thirdly, the "rules" are not derived directly from the training data, and therefore any number of rows is allowed, allowing for a more flexible system. The classification results in this paper were based on software simulations.

The work resulted in an online EHW pattern recognition architecture capable of having many inputs, more than previous online EHW systems, and classifying into more categories. Classification accuracy was better than the best previously proposed (offline) EHW system and comparable to several traditional approaches.

Abstract

We propose an evolvable hardware (EHW) architecture for pattern recognition. Compared to an earlier offline EHW pattern recognition approach, the proposed architecture is advantageous in its suitability for online adaptation, while a very high recognition speed is maintained. With its support for virtual run-time reconfiguration, the architecture is suitable for implementation in an on-chip evolution system. Function level modules and data buses are employed in the

architecture in order to reduce the search space. Furthermore, incremental evolution is applied, which shortens evolution time and allows for the evolution of a larger system. The use of incremental evolution also has the advantage of reducing the size of the evolution hardware in an on-chip evolution system. Variations in architecture parameters are explored, and it is found that the size of the system can be reduced at the cost of longer evolution time or lower recognition accuracy. The architecture is applied to a face image recognition task, for which a recognition accuracy of 96.25% is demonstrated. This result is better than previously proposed offline EHW architectures.

3.2.4 Paper IV

Online Evolution for a High-Speed Image Recognition System Implemented On a Virtex-II Pro FPGA

Having developed the architecture presented in Paper III, it was in this paper chosen to look further into the hardware issues, focusing in particular on the aspects of online evolution. Using the same application as in Paper III, work was undertaken on developing a fitness evaluation module for the evolutionary process which could function in parallel with the classification system. In addition to uninterrupted classification operation, another motivation for developing an autonomous fitness evaluation module was to speed up the task of feeding training vectors to the system and reading back the results, taking much time in the initial system presented in Paper I.

The work demonstrated the implementation of a fitness evaluation module for the FUR architecture in hardware for a significant speedup of the evolution time. Fitness evaluation in parallel and little resource utilization compared to the total classifier system were demonstrated.

Abstract

Online incremental evolution for a complex high-speed pattern recognition architecture has been implemented on a Xilinx Virtex-II Pro FPGA. The fitness evaluation module is entirely hardware-based in order to increase the speed of the circuit evaluation which uses a large training set (360 images/23040 bytes). The fitness evaluation time for 1000 generations consisting of 16 individuals is 623ms, twice as fast as software fitness evaluation performed on a workstation running at a 30 times higher clock frequency. The rest of the genetic algorithm (GA) runs in software on a PowerPC 405 processor core on the FPGA. The total evolution time for 1000 generations is 1313ms, equivalent to the total time used by the workstation. Resource utilization for the fitness evaluation module is 1393 slices (10%) of a XC2VP30 device.

3.2.5 Paper V

An Online EHW Pattern Recognition System Applied to Sonar Spectrum Classification

While showing good accuracy on the face image recognition application, it was desirable to test the developed architecture on more problems. The sonar spectrum data set [7] is a widely known benchmark, and has been shown to be more difficult for the LoDETT system to classify than the face image recognition task [46]. It would therefore be an interesting test for the developed classifier system. Given the high feature variation for vectors in each category, an enhanced fitness function was developed which stimulated the recognition of distinct features within each category. In addition, experiments were conducted on stopping evolution earlier in order to reduce overfitting.

The work demonstrated that a high classification accuracy could be achieved for the FUR architecture on a difficult signal classification task. This was significantly better than a previous EHW approach and an ANN approach, and the high accuracy was partly obtained by training different submodules on different training set partitions.

Abstract

An evolvable hardware (EHW) system for high-speed sonar return classification has been proposed. The system demonstrates an average accuracy of 91.4% on a sonar spectrum data set. This is better than a feed-forward neural network and previously proposed EHW architectures. Furthermore, this system is designed for online evolution. Incremental evolution, data buses and high level modules have been utilized in order to make the evolution of the 480 bit-input classifier feasible. The classification has been implemented for a Xilinx XC2VP30 FPGA with a resource utilization of 81% and a classification time of 0.5μs.

3.2.6 Paper VI

Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control

A third application, PHC, was selected as a base for further experiments. In cooperation with Paul Kaufmann and Marco Platzner at the University of Paderborn, a dataset of muscle electromyographic (EMG) signals were collected and processed in order to serve as a base for evaluation and comparison of our two different EHW approaches. While my approach was the FUR architecture presented in paper III, the approach of Kaufmann and Platzner was an ECGP-based model. With one of the research goals of the thesis being to demonstrate competitive schemes, it was desirable to compare our EHW approaches with state-of-the-art conventional classifiers. In cooperation with Thiemo Gruber and Bernhard Sick at the University of Passau, comparison experiments with three conventional classification techniques were undertaken. Experiments showed that the FUR architecture delivered a high classification performance on the signals, close to that of support vector machines (SVMs).

The work addressed the need for a direct comparison of EHW with state-of-the-art traditional classifier approaches, and showed that the FUR architecture had competitive performance to these classifiers.

Abstract

Evolvable hardware has shown to be a promising approach for prosthetic hand controllers as it features self-adaptation, fast training, and a compact system-on-chip implementation. Besides these intriguing features, the classification performance is paramount to success for any classifier. However, evolvable hardware classifiers have not yet been sufficiently compared to state-of-the-art conventional classifiers. In this paper, we compare two evolvable hardware approaches for signal classification to three conventional classification techniques: k-nearest-neighbor, decision trees, and support vector machines. We provide all classifiers with features extracted from electromyographic signals taken from forearm muscle contractions, and try to recognize eight different hand movements. Experimental results demonstrate that evolvable hardware approaches are indeed able to compete with state-of-the-art classifiers. Specifically, one of our evolvable hardware approaches delivers a generalization performance similar to that of support vector machines.

3.2.7 Paper VII

A Comparison of Evolvable Hardware Architectures for Classification Tasks

With competitiveness still in mind, it was desirable to compare the proposed methods with earlier presented EHW classification architectures. The collected EMG dataset provided a good basis for a comparison with earlier EHW classifiers targeting EMG signals. By simulating the architectural principles of previously presented EHW EMG classifiers, a comparison between different architectural concepts could be made. The concepts to investigate included various degrees of incremental evolution and the use of building blocks, as well as the application-specific features of the approaches. It was of interest to see if the application-specific features and the use of building blocks of the ECGP-based model and the FUR architecture could increase classification accuracy. It was also of special interest to see how the FUR architecture would compare in terms of number of evaluations required for the search, as this can indicate the suitability for run-time adaptation applications.

The work sought to compare the FUR architecture with other EHW approaches on the same application, and showed that the FUR approach gives a high recognition accuracy significantly faster than the other approaches.

Abstract

We analyze and compare four different evolvable hardware approaches for classification tasks: An approach based on a programmable logic array architecture, an approach based on two-phase incremental evolution, a generic logic architecture with automatic definition of building blocks, and a specialized coarse-grained architecture with pre-defined building blocks. We base the comparison on a common data set and report on classification accuracy and training effort. The results show that classification accuracy can be increased by using modular, specialized classifier architectures. Furthermore, function level evolution, either with predefined functions derived from domain-specific knowledge or with functions that are automatically defined during evolution, also gives higher accuracy. Incremental and function level evolution reduce the search space and thus shortens the training effort.

3.3 List of Publications

Papers Included in Thesis

I K. Glette and J. Torresen. A Flexible On-Chip Evolution System Implemented on a Xilinx Virtex-II Pro Device. In *International Conference on Evolvable Systems (ICES)*, pages 66–75, Springer 2005.

II K. Glette and J. Torresen and M. Yasunaga and Y. Yamaguchi. On-Chip Evolution Using a Soft Processor Core Applied to Image Recognition. In *Adaptive Hardware and Systems*, pages 373–380, IEEE 2006.

III K. Glette and J. Torresen and M. Yasunaga. Online Evolvable Pattern Recognition Hardware. To appear in *Evolutionary Image Analysis and Signal Processing (provisional title)*, Studies in Computational Intelligence series, Springer 2008 (est.).

IV K. Glette and J. Torresen and M. Yasunaga. Online Evolution for a High-Speed Image Recognition System Implemented On a Virtex-II Pro FPGA. In *Adaptive Hardware and Systems*, pages 463–470, IEEE 2007.

V K. Glette and J. Torresen and M. Yasunaga. An Online EHW Pattern Recognition System Applied to Sonar Spectrum Classification. In *International Conference on Evolvable Systems (ICES)*, pages 1–12, Springer 2007.

VI K. Glette and J. Torresen and T. Gruber and B. Sick and P. Kaufmann and M. Platzner. Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control. In *Adaptive Hardware and Systems*, pages 32–39, IEEE 2008.

VII K. Glette and J. Torresen and P. Kaufmann and M. Platzner. A Comparison of Evolvable Hardware Architectures for Classification Tasks. In *International Conference on Evolvable Systems (ICES)*, Springer 2008.

Other Papers

- K. Glette and J. Torresen and M. Yasunaga. An Online EHW Pattern Recognition System Applied to Face Image Recognition. In *Applications of Evolutionary Computing (EvoWorkshops2007)*, pages 271–280, Springer 2007.

- J. Torresen and J. Norendal and K. Glette. Establishing a New Course in Reconfigurable Logic System Design. In *Proceedings of the 10th IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 1–4, IEEE 2007.

- J. Torresen and K. Glette. Making Hardware Soft in Intelligent Systems. In *ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security (BLISS)*, pages 87–90, IEEE 2007.

- J. Torresen and K. Glette. Improving Flexibility in On-Line Evolvable Systems by Reconfigurable Computing. In *International Conference on Evolvable Systems (ICES)*, pages 391–402, Springer 2007.

- M. Furuholmen and M. Hovin and J. Torresen and K. Glette. Continuous Adaptation in Robotic Systems by Indirect Online Evolution. In *ECSIS Symposium on Learning and Adaptive Behavior in Robotic Systems (LAB-RS)*, IEEE 2008.

- M. Furuholmen and K. Glette and J. Torresen and M. Hovin. Indirect Online Evolution - a Conceptual Framework for Adaptation in Industrial Robotic Systems. In *International Conference on Evolvable Systems (ICES)*, Springer 2008.

- H. Kawai and Y. Yamaguchi and M. Yasunaga and K. Glette and J. Torresen. An Adaptive Pattern Recognition Hardware with On-chip Shift Register-based Partial Reconfiguration. To appear in *International Conference on Field-Programmable Technology (ICFPT)*, IEEE 2008.

- J. Torresen and G. A. Senland and K. Glette. Partial Reconfiguration Applied in an On-line Evolvable Pattern Recognition System. To appear in *26th Norchip Conference*, IEEE 2008.

Chapter 4

Discussion

This chapter discusses the approach followed in the thesis. In addition, the results are summarized and discussed. Finally, a conclusion is given and directions for possible future work are suggested.

4.1 Discussion of Approach and Results

The primary goal of this thesis was to *develop online evolvable hardware schemes that could evolve systems for large real-world applications*. A run-time adaptable on-chip EHW classifier architecture has been developed which allows for a large number of categories and inputs, capable of good classification performance on well-known classifier benchmarks including both image recognition and signal classification.

4.1.1 Scalability

The following schemes have been adopted and combined in the pursuit of designing scalable systems for large applications:

- *High-level building blocks with predefined functions, operating on input elements rather than single bit signals.* When the number of input lines to an EHW system becomes high, operating with fine-grained logic on single bit lines quickly makes the search space very large. This is due to the large amount of possible routing configurations of single lines and the amount of possible function configurations. By grouping the input lines according to signal-specific elements (such as image pixels of 8 bits) and operating with pre-defined functions on these elements, the search space is greatly reduced. However, this approach requires domain knowledge both in terms of suitable data elements and operators on these. While suitable data elements are often given by only a little knowledge about the input, the operators can be found through experimentation and also design experience. The first experiments on using building blocks and data buses for pattern recognition were conducted in paper II while more refined building blocks were proposed in paper III.

- *An application domain-specific architecture.* While the architecture in paper II was a relatively general array of functional units, the architecture presented in paper III is customized. By tailoring the EHW architecture to the evolution of classifier systems, the search is more directed towards working classifiers. The inclusion of counter modules inputting a number of "rules" forces the solutions to include the property of gradual distinction of different categories. Such a property could be hard for the search to discover by itself. The number of elements and the way elements can be connected are also limited. From an online EHW and virtual reconfiguration (see Sec. 2.2.2) perspective, these limitations allow a large amount of hardware resources to be saved. This is therefore a critical property when large circuits are to be evolved.

- *Incremental evolution of subsystems before assembling them into a final system.* By designing for modularity in the EHW system, like in paper III, incremental evolution can be applied. A direct advantage of incremental evolution is that the search space for each evolution run is reduced significantly, due to the decreased genome size. As an example the genomes for the subsystems in papers III – VII have been between 44 and 120 bits long. Further, as shown in paper IV, the incremental approach makes it possible to have an evaluation module in parallel with the classification module without much resource overhead, which is an advantage for online adaptable systems. In addition, a significant increase in recognition accuracy was observed in paper V when subsystems were trained with different partitions of the training set. The use of incremental evolution has been facilitated through the pre-defined and relatively fine-grained modularity of the specific architecture. For applications where architectural modularity is unclear, automatically defined modules could be an interesting approach. It is however not clear at the moment how such an approach, with changing and different subsystem sizes, could be mapped to an online evolutionary system in a practical way.

These schemes have the effect of reducing the genome length, and thus the search space, as well as eliminating a number of non-working solutions. As a consequence, it becomes possible to evolve, in a realistic timeframe, a system which is larger and has a higher performance compared to more direct evolutionary approaches. In addition, when compared with other EHW approaches on smaller problems, Paper VII shows that the proposed system provides significantly shorter evolution time: At 2^{13} evaluations, the FUR architecture obtains a better recognition accuracy than what the other architectures obtain after at least 32 times more evaluations.

The employed schemes are all relying more or less on *a priori* knowledge in order to direct evolution towards working systems. It can be argued that this approach reduces the possibilities for evolution to find new and inventive designs. However, in this context, evolution is used not so much as an invention machine but rather as a tool for achieving an adaptable and self-optimizing system.

4.1.2 Online Evolution

It should be noted that the system has been designed with *online* evolution, and thus quickly reconfigurable hardware architectures, in mind. A consequence of this is that much of the structural flexibility allowed by a synthesis/place and route process is not available, implying restricted freedom for evolution. In addition, a virtual reconfiguration approach in general implies high resource usage since all possible connections and functions have to be implemented. This size-limiting factor has been approached by limiting the number of available functions to the FUs. Further, the use of time multiplexing of the input data saves resources required for multiplexers. This could possibly be extended with a more sequential use of the available hardware resources, where the number of rows in the actual classifier is higher than the number of rows available in hardware. Such resource-saving methods will of course come at the expense of slower classification, however this is a tradeoff which could be considered for each application.

The on-chip online EHW experimental platform developed has shown in papers I and IV that it is feasible to do on-chip evolution also when offloading some parts of the EA to an on-chip processor. Results show that the total evolution time for the on-chip system is the same as for a workstation with a more powerful processor running at 10 to 30 times the clock speed of the on-chip system. However, the results indicate that in the current on-chip system, the processor seems to be a bottleneck and therefore, when fast training is a priority, further optimization needs to be performed. This could be either in terms of software optimizations or simplifications, or by implementing more of the algorithm in hardware.

4.1.3 Competitiveness

The goal of developing a system for real-world applications has been pursued. However, at this early stage of research it has been judged too time-consuming to target operational systems for specific real-world applications. Instead, standard benchmarks for traditional classifiers with difficult and relatively realistic data have been employed. In this way, the problem difficulty is high, while at the same time possibilities for comparison with other approaches are maintained. It should be noted that in a final operational system extra work will be needed in terms of preprocessing of the input data. Further, other themes than those covered by the employed benchmarks may be more suited to the architecture's advantages.

One of the sub-goals was to demonstrate that the developed schemes are *competitive* for selected applications. The competitiveness against other offline EHW approaches capable of the same number of inputs has been covered in papers III, V, and VI, showing accuracy improvements. Specifically, the FUR architecture achieved 96.2% accuracy versus 94.7% accuracy for the LoDETT architecture for the face image recognition task. Further, an accuracy of 91.4% versus the 83% for the LoDETT system was achieved for the sonar return recognition benchmark. Finally, the FUR system obtained an accuracy of 91.6% versus 89.4% for the ECGP system on the EMG signal classification task.

The competitiveness against earlier proposed online and online-friendly EHW systems has been studied in paper VII. Since no other online systems designed for the same number of inputs as the FUR system are known, the comparisons were performed on an application with a reduced number of inputs. The results indicate that the FUR architecture indeed gives better accuracy. Specifically, the FUR architecture obtained an accuracy of 90.0% compared to 74.1% for the PLA-based approach and 79.8% for the ICE architecture.

The competitiveness against traditional classifiers (including state-of-the-art) has been addressed in papers III and V, while a more thorough comparison has been performed in paper VI. Accuracies equal to or better than Eigenfaces, Fisherfaces, and ANNs have been indicated in papers III and V. Specifically, the FUR architecture achieved an accuracy of 95.1% for the conditions where Eigenfaces scored 90.1% and Fisherfaces scored 95.0% for the face image recognition task. Further, FUR obtains 91.4% for the sonar return classification task, where the original ANN approach scored 90.4%. Papers III, V and in particular VI address the comparison with SVM, today regarded as the best of the traditional classifier methods. The results show somewhat lower accuracies for the FUR classifier: 96.2% (FUR) versus 98.0% (SVM) for the image recognition task, 91.4% versus 95.2% for the sonar classification task, and 91.6% versus 94.6% for the EMG classification task. This indicates that at present, SVMs will most likely be a better choice for applications where classification accuracy is the most important requirement. The EHW classifier system is, at its present stage, better suited for applications which also have requirements such as high classification speed (classification in 0.5μs is possible) or possibly fast training and/or adaptation.

4.2 Conclusion

The main contributions of this thesis can be summarized as follows:

- It has been demonstrated how schemes like incremental evolution, high-level building blocks, and a specialized architecture can be combined and make it possible to evolve relatively large online EHW systems for relatively large applications. This makes it possible to evolve systems for relatively large applications in a reasonable amount of time and systems for small applications with fewer evaluations than other known comparable EHW approaches.

- Possible solutions for on-chip run-time adaptive EHW systems, including the use of an on-chip processor and HW/SW co-design, have been presented. It is demonstrated how flexibility can be achieved by implementing parts of the EA in SW, while fitness evaluation can be accelerated in HW for training speed. On-chip evolution speed equal to the speed of a workstation with a much higher clock rate was shown. It is demonstrated how a small hardware fitness evaluation module can run in parallel with the full-size operational module in an incremental EHW system.

- A high-performance online EHW classifier architecture has been presented, which classifies realistic standard benchmarks more accurately than earlier presented EHW architectures. Further, it handles larger applications with more inputs and classes than previous *online* EHW systems. The classification accuracy is competitive to traditional classification methods, while the classification speed is very high because of the parallel implementation.

It can be concluded that this thesis has contributed to the initial goal of developing schemes for evolving scalable EHW for real-world applications, with a focus on pattern recognition systems. Hopefully some of the methods and results could also be of use for the evolution of other types of systems.

4.3 Future Work

Improvements to the presented results could be made, and the following is a list of possible focus areas of future work:

- A prototype system for a real-world application should be developed. This includes a search for a suitable application, which would preferably have requirements such as high-speed classification and/or online adaptation. In addition, further optimization of the EA SW running on the on-chip processor could be performed in order to reduce bottleneck problems, and a suitable HW/SW balance could be investigated.

- Improvements to the classifier architecture could be investigated, in the search for even better classification or less resource usage. In this theme, more flexible or hierarchical classification rules could be an interesting direction. In addition, better partitioning of the training set for each FU row could be investigated.

- It could be desirable to classify more categories than what is actually implemented in hardware. In light of this, a system with "virtual" categories would be interesting, where classification is partly serialized depending on the available hardware, and the configuration of this is automated.

- Further experiments on online adaptation could be conducted. This could include investigating applications which would benefit from online adaptation and see how the architecture performs with regard to changes in the training set.

- The mapping of more flexible schemes into online EHW, such as automatically defined building blocks, could be an interesting and challenging avenue to investigate.

Bibliography

[1] M. Bidlo. Evolutionary design of generic combinational multipliers using development. In *Evolvable Systems: From Biology to Hardware*, volume 4684 of *Lecture Notes in Computer Science*, pages 77–88, 2007.

[2] R. O. Canham and A. M. Tyrrell. A hardware artificial immune system and embryonic array for fault tolerant systems. *Genetic Programming and Evolvable Machines*, 4(4): 359–382, 2003. ISSN 1389-2576. doi: http://dx.doi.org/10.1023/A:1026143128448.

[3] E. Cantu-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10(2):141–171, 1998.

[4] H. de Garis. Evolvable hardware: Genetic programming of a Darwin machine. In C. R. R. et al., editor, *Artificial Neural Nets and Genetic Algorithms - Proc. of the International Conference in Innsbruck, Austria*, pages 441–449. Springer-Verlag, 1993.

[5] H. de Garis. Cam-brain project: The evolution of a billion neuron artificial brain by 2001. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware: The evolutionary Engineering Approach*. Springer-Verlag, 1996. Lecture Notes in Computer Science, vol. 1062.

[6] K. Glette, J. Torresen, and M. Yasunaga. An Online EHW Pattern Recognition System Applied to Face Image Recognition. In *Proceedings Applications of Evolutionary Computing (EvoWorkshops2007)*, volume 4448 of *Lecture Notes in Computer Science*, pages 271–280. Springer, 2007.

[7] R. P. Gorman and T. J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1(1):75–89, 1988.

[8] P. Haddow and G. Tufte. Bridging the genotype-phenotype mapping for digital FPGAs. In *Proc. of the Second NASA/DoD Workshop on Evolvable Hardware*, 2001.

[9] S. Harding and J. Miller. Evolution in materio: a real-time robot controller in liquid crystal. *Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on*, pages 229–238, June-1 July 2005. ISSN 1550-6029.

[10] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya. Evolving hardware with genetic learning: a first step towards building a darwin machine. In *Proceedings of the second international conference on From animals to animats 2 : simulation of adaptive behavior*, pages 417–424, Cambridge, MA, USA, 1993. MIT Press. ISBN 0-262-63149-0.

[11] T. Higuchi, M. Iwata, I. Kajitani, H. Iba, Y. Hirao, B. Manderick, and T. Furuya. Evolvable Hardware and its Applications to Pattern Recognition and Fault-Tolerant Systems. In *Towards Evolvable Hardware: The evolutionary Engineering Approach*, volume 1062 of *LNCS*, pages 118–135. Springer, April 1996.

[12] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, M. Salami, N. Kajihara, and N. Otsu. Real-world applications of analog and digital evolvable hardware. *IEEE Trans. Evolutionary Computation*, 3(3):220–235, 1999.

[13] J. H. Holland. *Adaption in Natural and Artificial Systems*. The University of Michigan Press, 1975. Ann Arbor, Michigan.

[14] I. Kajitani, T. Hoshino, M. Iwata, and T. Higuchi. Variable Length Chromosome GA for Evolvable Hardware. In *International Conference on Evolutionary Computation (ICEC)*, pages 443–447. IEEE, 1996.

[15] I. Kajitani, T. Hoshino, D. Nishikawa, H. Yokoi, S. Nakaya, T. Yamauchi, T. Inuo, N. Kajihara, M. Iwata, D. Keymeulen, and T. Higuchi. A Gate-Level EHW Chip: Implementing GA Operations and Reconfigurable Hardware on a Single LSI. In *Proceedings 2nd International Conference on Evolvable Systems (ICES)*, volume 1478 of *LNCS*, pages 1–12. Springer, 1998. ISBN 3-540-64954-9.

[16] I. Kajitani, I. Sekita, N. Otsu, and T. Higuchi. Improvements to the Action Decision Rate for a Multi-Function Prosthetic Hand. In *Proceedings 1st International Symposium on Measurement, Analysis and Modeling of Human Functions (ISHF)*, pages 84–89, 2001.

[17] T. Kalganova. Bidirectional incremental evolution in extrinsic evolvable hardware. In J. L. et al., editor, *Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware*, pages 65–74. IEEE Computer Society, Silicon Valley, USA, July 2000.

[18] J. Koza, M. Keane, and M. Streeter. Routine high-return human-competitive evolvable hardware. *Evolvable Hardware, 2004. Proceedings. 2004 NASA/DoD Conference on*, pages 3–17, June 2004.

[19] J. Koza et al. The importance of reuse and development in evolvable hardware. In J. L. et al., editor, *Proc. of the 2003 NASA/DoD Conference on Evolvable Hardware*, pages 33–42. IEEE, 2003.

[20] J. R. Koza. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In J. S. Gero and F. Sudweeks, editors, *Artificial Intelligence in Design '96*. Dordrecht: Kluwer Academic Publishers, 1996.

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

[22] J. Lohn, G. Hornby, and D. Linden. Evolutionary antenna design for a NASA spacecraft. In U.-M. O'Reilly, T. Yu, R. L. Riolo, and B. Worzel, editors, *Genetic Programming*

Theory and Practice II, chapter 18, pages 301–315. Springer, Ann Arbor, 13-15 May 2004. ISBN 0-387-23253-2.

[23] S. Lynch. A platform for intrinsic evolution of digital circuits on Virtex II pro. B.E. electronic and computer engineering project report, National University of Ireland, Galway, 2006.

[24] A. Megacz. A library and platform for FPGA bitstream manipulation. In *FCCM '07: Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 45–54, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2940-2.

[25] J. Miller, D. Job, and V. Vassilev. Principles in the evolutionary design of digital circuits – Part I. *Journal of Genetic Programming and Evolvable Machines*, 1(1):8–35, 2000.

[26] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi. Hardware Evolution at Function Level. In *Proceedings 4th Parallel Problem Solving from Nature (PPSN)*, volume 1141 of *LNCS*, pages 62–71. Springer, September 1996.

[27] M. Murakawa et al. The grd chip: Genetic reconfiguration of dsps for neural network processing. *IEEE Transactions on Computers*, 48(6):628–638, June 1999.

[28] L. Sekanina. *Evolvable Components: From theory to Hardware Implementations*. Springer-Verlag, 2004. ISBN 3-540-40377-9.

[29] L. Sekanina and M. Bidlo. Evolutionary design of arbitrarily large sorting networks using development. *Genetic Programming and Evolvable Machines*, 6(3):319–347, 2005. ISSN 1389-2576.

[30] L. Sekanina and R. Ruzicka. Design of the Special Fast Reconfigurable Chip Using Common FPGA. In *Proceedings of the IEEE Conference on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pages 161–168, 2000.

[31] A. Stoica, D. Keymeulen, R. Tawel, C. Salazar-Lazaro, and W. te Li. Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digital cmos circuits. In *Proceedings of the 1st NASA/DOD workshop on Evolvable Hardware*, page 76, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0256-3.

[32] A. Stoica, D. Keymeulen, and R. Zebulum. Evolvable hardware solutions for extreme temperature electronics. In *Proceedings of the 3rd NASA/DOD workshop on Evolvable Hardware*, page 93, Los Alamitos, CA, USA, 2001. IEEE Computer Society. ISBN 0-7695-1180-5.

[33] M. Tanaka, H. Sakanashi, M. Salami, M. Iwata, T. Kurita, and T. Higuchi. Data compression for digital color electrophotographic printer with evolvable hardware. In M. Sipper et al., editors, *Evolvable Systems: From Biology to Hardware. Second International Conference, ICES 98*, volume 1478 of *Lecture Notes in Computer Science*, pages 106–114. Springer-Verlag, 1998.

[34] A. Thompson. Silicon evolution. In *Proc. of Genetic Programming*, 1996.

[35] J. Torresen. Two-Step Incremental Evolution of a Digital Logic Gate Based Prosthetic Hand Controller. In *Proceedings 4th International Conference on Evolvable Systems (ICES)*, volume 2210 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2001.

[36] J. Torresen. Evolving multiplier circuits by training set and training vector partitioning. In P. H. A. Tyrrell and J. Torresen, editors, *Evolvable Systems: From Biology to Hardware. Fifth International Conference, ICES'03*, volume 2606 of *Lecture Notes in Computer Science*, pages 228–237. Springer-Verlag, 2003.

[37] J. Torresen. A divide-and-conquer approach to evolvable hardware. In M. Sipper et al., editors, *Evolvable Systems: From Biology to Hardware. Second International Conference, ICES 98*, volume 1478 of *Lecture Notes in Computer Science*, pages 57–65. Springer-Verlag, 1998.

[38] G. Tufte and P. C. Haddow. Building knowledge into development rules for circuit design. In *Proceedings of 5th International Conference on Evolvable Systems: From Biology to Hardware, ICES2003*, volume 2606, pages 117–128, Trondheim, Norway, 2003. Springer-Verlag.

[39] G. Tufte and P. C. Haddow. Biologically-inspired: A rule-based self-reconfiguration of a virtex chip. In M. Bubak, G. D. van Albada, and P. M. A. S. et al., editors, *Proc. of International Conference on Computational Science 2004*, volume 3038 of *Lecture Notes in Computer Science*, pages 1249–1256, May 2004.

[40] G. Tufte and P. C. Haddow. Towards development on a silicon-based cellular computing machine. *Natural Computing: an international journal*, 4(4):387–416, 2005. ISSN 1567-7818.

[41] A. M. Tyrrell, E. Sanchez, D. Floreano, G. Tempesti, D. Mange, J. M. Moreno, J. Rosenberg, and A. E. P. Villa. POEtic tissue: An integrated architecture for bio-inspired hardware. In *ICES*, pages 129–140, 2003.

[42] A. Upegui. *Dynamically Reconfigurable Bio-inspired Hardware*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne (EPFL), 2006. Thesis No. 3632.

[43] D. J. V. Vassilev and J. Miller. Towards the automatic design of more efficient digital circuits. In J. L. et al., editor, *Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware*, pages 151–160. IEEE Computer Society, Silicon Valley, USA, July 2000.

[44] J. A. Walker and J. F. Miller. Evolution and acquisition of modules in cartesian genetic programming. In *Genetic Programming, 7th European Conference, EuroGP2004*, pages 187–197, 2004.

[45] M. Yasunaga, T. Nakamura, and I. Yoshihara. Evolvable Sonar Spectrum Discrimination Chip Designed by Genetic Algorithm. In *Systems, Man and Cybernetics*, volume 5, pages 585–590. IEEE, 1999.

[46] M. Yasunaga, T. Nakamura, I. Yoshihara, and J. Kim. Genetic Algorithm-based Design Methodology for Pattern Recognition Hardware. In *Proceedings 3rd International Conference on Evolvable Systems (ICES)*, volume 1801 of *Lecture Notes in Computer Science*, pages 264–273. Springer, 2000.