Encyclopedia of Information Science and Technology, Third Edition

Mehdi Khosrow-Pour
Information Resources Management Association, USA

A volume in the

Information Science REFERENCE
An Imprint of IGI Global

Managing Director: Lindsay Johnston
Production Editor: Jennifer Yoder & Christina Henning
Development Editor: Austin DeMarco & Jan Travers
Acquisitions Editor: Kayla Wolfe
Typesetter: Mike Brehm, John Crodian, Lisandro Gonzalez, Deanna Zombro
Cover Design: Jason Mull

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA, USA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Encyclopedia of information science and technology / Mehdi Khosrow-Pour, editor.
pages cm
Includes bibliographical references and index.
ISBN 978-1-4666-5888-2 (hardcover) -- ISBN 978-1-4666-5889-9 (ebook) -- ISBN 978-1-4666-5891-2 (print & perpetual access) 1. Information science--Encyclopedias. 2. Information technology--Encyclopedias. I. Khosrow-Pour, Mehdi, 1951-
Z1006.E566 2015
020.3--dc23

2014017131

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Evolvable Hardware

André Macário Barros
Universidade Tecnológica Federal do Paraná - UTFPR, Brazil

Heitor Silvério Lopes
Universidade Tecnológica Federal do Paraná - UTFPR, Brazil

INTRODUCTION

Evolvable Hardware (EHW) is an intelligent technology which belongs to a large area called Evolutionary Electronics (EEL), which includes applications of Evolutionary Computation (EC) in the domain of electronics. EHW combines concepts from Evolutionary Computation (EC) and Electronic Design (ED), and it has been an active area of research in the last years. This article provides an updated review of this area. First, EEL with focus on EHW is overviewed, pointing out the applicability of this technology. A brief review of two important EC paradigms is provided: GAs (Genetic Algorithms) and Genetic Programming (GP), since they are the most used in EHW. On the other hand, the basic aspects of ED are also shown with focus on Reconfigurable Computing (RC) and the Field Programmable Gate Arrays (FPGAs) technology. Next, the usual taxonomy found in the literature is presented: by design type - analog and digital, and by evolution type: extrinsic, intrinsic and mixtrinsic. Some relevant applications are presented and discussed. Finally, future trends are presented.

BACKGROUND

As previously described, this subject merges two areas of study. A brief background of both areas is presented here to provide the necessary comprehension.

Evolutionary Computation

Problems considered complex and hard (e.g., multimodality, large search space, large number of constraints) to be solved using conventional optimization techniques of Computer Science are being addressed by EC, which employs a collection of algorithms called Evolutionary Algorithms (EAs). EAs imitate the nature, specifically the Darwinian principle of survival of the fittest. In EC, an individual is the representation of a possible solution. A set of individuals forms the initial population, randomly created. The individuals of this population eventually produce descendants by means of selection, reproduction and mutation, according to the survival rule in which the best fitted individuals are those who will have more chances to reproduce. The descendants form a new population and the process is repeated for many generations. When a stopping criterion is met, the evolution stops, and the best individual ever found is the solution for the problem being handled.

The main EAs are: Evolutionary Programming (EP), (Fogel, 1962), (Fogel, Owens & Walsh, 1966), Evolution Strategies (ES) (Rechenberg, 1965, 1973), Genetic Algorithms (GAs) (Holland, 1975) (Goldberg, 1989), and Genetic Programming (GP) (Koza, 1992, 1999). GAs and GP are the most frequently used in EHW.

Considering P(t) a population of individuals at time t, based on the concepts previously presented, GAs and GP can be represented by the algorithm shown in Figure 1 and described as follows (Bäck, Fogel & Michalewicz, 2000a), Goldberg (1989):

Representation

Before running the EA, it is necessary to choose the representation, i.e., how the candidate solutions are represented in during the execution of an EA. Data structures are the commonly representations used by computer programs. Depending on the problem, the data structure itself can be the real-world solution. But there are applications in which such a direct representation is not possible, for instance, a list of tasks, a mechanism, or an electronic circuit. Binary strings, finite state

DOI: 10.4018/978-1-4666-5888-2.ch703

Figure 1. An evolutionary algorithm

```
1- t=0
2- initialize P(t)
3- evaluate P(t)
4- while termination condition is false do
5-    select P(t+1) from P(t)
6-    crossover P(t+1)
7-    mutate P(t+1)
8-    evaluate P(t+1)
9-    t = t + 1
```

representations and parse trees are the data structures processed by the EAs. The term used for this indirect representation in EC is genotype, usually composed by one or more chromosomes and the one employed for the solution in the real-world is phenotype.

First Generation and Evaluation

For algorithm presented on Figure 1, the initial population of individuals is randomly generated. Each individual of the population needs to be evaluated to inform the EA how good the individual is. This is called fitness evaluation and can be processed in two steps: a) the genotype to phenotype conversion – in the case of an indirect representation; and b) the fitness computation of the phenotype related to that individual. This computation is a procedure using expressions related to the problem to produce usually a number, which will be used as a measurement of quality. An example is an individual such as a sequence of cities to deliver a product. This sequence is used in an expression looking for higher profits. A sequence that provides a good profit receives a better evaluation than another individual that provides lower profits. Each individual of the population enters the while loop of the algorithm after have being evaluated.

Selection

In this phase, the individuals of the population are selected for crossover according to their fitness value. This selection uses probability. So, the selected individuals can be the ones with higher fitness values, but there can be too some individuals with lower fitness values. The most known selection mechanisms are: proportional (or roulette wheel), tournament, truncation, linear rank, and exponential rank.

Crossover

This operation consists of the exchange of genetic information between the selected individuals. They are called parents. For instance, a crossover between two parents "111111" and "000000" can produce two possible offspring such as "001100" and "110011." The two bits in their middle portion were changed. This operation is probabilistic. Crossover types most known are: one point, two point and uniform. A chromosome is composed of genetic information (subsets of bits or numbers) called building blocks. It is desirable that crossover operations allow the exchange of information between parents preserving their building blocks, promoting the positive evolution, i.e., descendants with better values of fitness then their ancestors. Crossover is an operator that performs local search in the search space of solutions.

Mutation

After crossover, the descendants are probabilistically submitted to the mutation operator, which consists in changing the value of a single locus (a locus is a position where the symbol is in a chromosome) to another value. Example: a mutation at locus 0 in the binary chromosome "111000111," results in "111000110." Mutation provides diversity in the search for the solution, since it performs global search, an attempt to avoid the AE to be stuck at local maxima.

Genetic Algorithms

GAs (Holland, 1975), (Goldberg, 1989), (Bäck, Fogel, & Michalewicz, 2000a), are the most popular EAs and all the features described previously are found in it.

Genetic Programming

GP (Koza, 1992, 1999) is an EA that usually evolves computer programs. The chromosome is encoded in a tree structure. The tree is composed by function nodes and terminal nodes. The function nodes are, in most cases, arithmetic or mathematical operations and Boolean or conditional or iteration functions. The function nodes are responsible for connecting the terminal nodes, forming the tree, thus generating an evolved program in a LISP-like form.

Figure 2. A LEA template

```
 1- pop <- initialize (par)
 2- repeat
 3-    newpop1 <- cooperate(pop, par)
 4-    newpop2 <- improve(newpop1, par)
 5-    pop <- compete(pop, newpop2)
 6-    if converged(pop) then
 7-       pop <- restart(pop, par)
 8-    end if
 9- until termination_criterion(par)
10- return best_individual(pop)
```

Lamarckian Evolutionary Algorithms

Lamarckian Evolutionary Algorithms (LEA) are population based EAs, but they emphasize exchange of learning between individuals to produce the next generation. LEA are also referred in the literature as Memetic Algorithms (MA), Baldwinian EAs, or social algorithms (Neri, Cotta, & Moscato, 2011).

Considering a *pop* population with *par* parameters, the main LEA operators responsible for exchange of individual learning information and their evolution are: cooperation (cooperate), improvement (improve), and competition (compete). To avoid premature convergence, it is often employed some kind of restart procedure. Figure 2 shows a basic template of a MA based on these concepts.

EA Applications

Some areas of application are: Mathematics (optimization, time-series), Biology (biochemistry, ecology), Physics (nuclear, optics), Engineering (mechanics, aerospace, robotics, electronics, signal processing), Computer Science (data mining, image processing), fault-tolerant systems (Zebulum, Pacheco, & Vellasco, 2001). More examples can be found at Bäck, Fogel, Michalewicz (2000). Examples of applications with LEAs are: Artificial Neural Network training, pattern recognition, robotic motion planning, medical systems, and VLSI design.

Electronic Design

ED (Whitaker, 2005), this great area of electronics, accumulates at least one hundred years of engineering knowledge of electronic systems development. This knowledge is based on human efforts, i.e., some developers created a set of rules to be followed during a project, all of them based on analog and digital circuit theory. Electronic systems are conceived through the observation of these rules. Belonging to this wide area, the technology of programmable devices is of special interest for EHW.

FPAA

FPAA (Field Programmable Analog Array) (Pierzchala, Gulak, Chua, & Rodrigues-Vásquez, 1998) is a programmable device composed of operational amplifiers and passive components. It can be programmed by software and this configuration can be transferred to them through a computer interface. The configuration contains an analog circuit created by the designer.

PLD

A PLD (Programmable Logic Device) (Pedroni, 2008) is a digitally programmable device that can be categorized in SPLDs (Simple PLD) - PLAs, PALs, or GALs, CPLDs (Complex PLDs), and FPGAs.

The FPGA is composed of thousands of logic elements called CLBs (Configurable Logic Blocks) or LABs (Logic Array Blocks), depending of the vendor. Each of these logic blocks is composed of elementary units such as Lookup Tables, "D" flip-flops, fast carry chain, adders, and multiplexers. The configuration that the FPGA receives corresponds to specific forms of associate these logic elements in order to build a specific digital circuit.

It is possible to program a FPGA in two ways. Using HDL (Hardware Description Language, such as Verilog and AHDL, or an IEEE standard HDL called VHDL (Very High Speed Integrated Circuits HDL). Alternatively, a graphic interface can be used, in which the designer draws the schematic of the desired digital circuit.

The design flow (Pedroni, 2010, pp. 3-10) to put an FPGA to work implemented with some idealized digital circuit is through an EDA (Electronic Device Automation) PC tool supplied by the vendor: a) project designer register his or her specifications; b) schematic or VHDL code containing the circuit described by the designer is entered in the EDA tool; c) analysis and

synthesis is processed. This is called code compilation; d) Fitting the compiled VHDL code into a specific space (in the PC) related to a specific FPGA device is processed. This is called place and routing; e) bitstream file (the file that contains the configuration) is generated; and f) download the bitstream file to the FPGA device. Between steps (b) and (c), (c) and (d), (d) and (e) it is possible to run simulations to check the consistency of the digital circuit to be implemented.

This flexibility provided by FPGAs allowed to grow an area of research called RC (Reconfigurable Computing) (Zebulum, Pacheco, & Vellasco, p. 241). Basically, RC runs an application in two parts: a) a conventional computer running the ordinary parts of one program; and b) a FPGA implemented with a specialized circuit running a specific part of the program providing acceleration.

EVOLVABLE HARDWARE

Zebulum, Pacheco & Vellasco (2001) define EEL as a research area that covers all the applications involving EC for electronic design. To design an electronic circuit using EC it is necessary to model the problem taking into account the characteristics presented in the EC section, such as the representation. Considering the algorithm presented at Figure 1, the population of individuals here is a population of possible candidate-circuits (in genotype form, e.g., bit strings) that satisfies the necessary requisites of a design specification.

Phenotype-Genotype (Electronic Circuit-Chromosome) Mapping

The representation (the phenotype to genotype mapping) will be demonstrated using a second order Butterworth low-pass filter presented as an example in Figure 3.

Four relevant aspects of Figure 3: 1) following this kind of modeling, the size of the chromosome has a relationship with the size of the circuit; 2) it is up to the designer the investigation of the estimated size of the search space and the respective specification of the bit widths for of each gene (a gene for EC here is a sequence of bits or symbols that represent together something meaningful – in this case *ctype*, *cvalue*, $term_1$ and $term_2$); 3) to run the EA it is necessary to build a full chromosome, with all genes of the circuit-candidate; and iv) it is possible to conclude that a digital system can follow this netlist style of modeling at gate level. A subprogram can do this conversion (chromosome to netlist file).

With the candidate-solutions mapped, it is now necessary to establish the evaluation process (see EC section). In this example a good chromosome is the one that maps component values that provides a frequency response acceptably close to the filter parameters presented at Figure 3. This is done in the circuit-candidate with a sweep signal for V_{in} and a corresponding observation of V_{out}. Receiving a netlist file, a circuit simulator can produce a value useful for this evaluation process. Observe that it is not necessary to handle the Butterworth formulas.

Figure 3. Phenotype to genotype mapping for a circuit

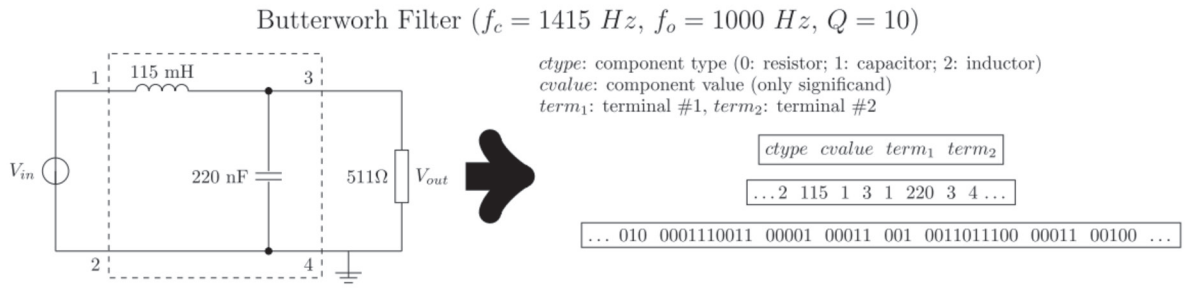
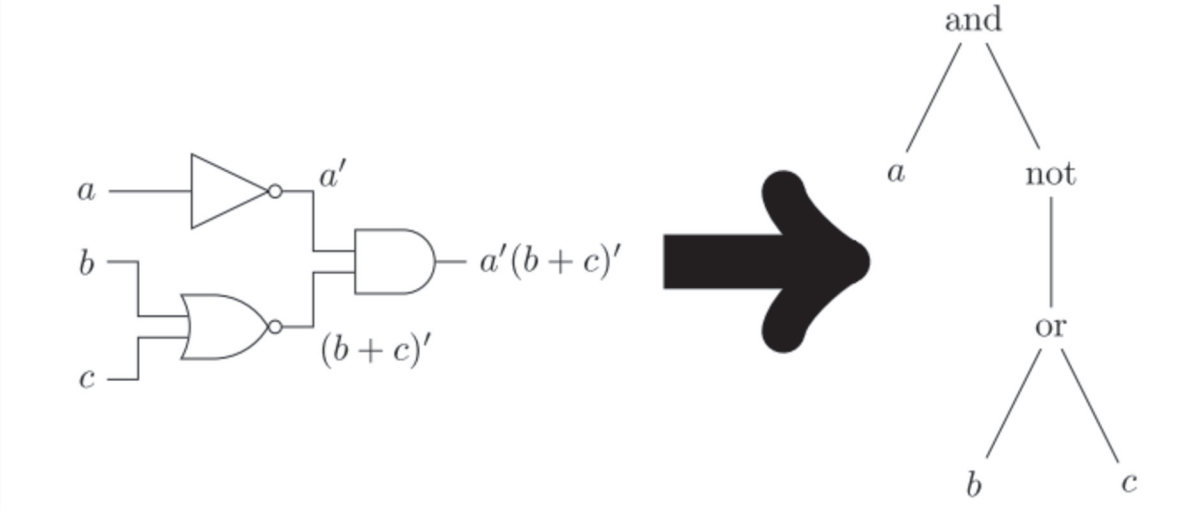


Figure 4. Tree-structure representation for GP



This lack of dependency of all formulas involved in a classical design gives novelty for the solutions. Consider three sets: 1) *A*: all the solutions provided by classic ED; 2) *B*: all the solutions provided by EEL; and 3) *C*: all the possible solutions. Empirical results demonstrated empirically that $A \in B$ (Lohn & Hornby, 2006). But it is not possible to affirm that $B = C$.

Consider a problem of GP where there is a search for a VHDL code. An example can be the one presented in Figure 4, where a digital circuit can be expressed in a chromosome represented in a tree structure.

The tree structure which represents the chromosome is then mapped to the VHDL code (phenotype) presented at Figure 5.

Figures 4 and 5 show how a chromosome can be transformed to a real VHDL code. This transformation is lexically possible because the VHDL code presents three fixed sections: 1) library declarations (lines 1 and 2 of Figure 5); 2) pin declarations (lines 3 to 6 of Figure 5); and 3) chip behavior (lines 7 to 10 of Figure 5), i.e., what the chip to be evolved must process digitally to produce the desired input/output relationship. For digital combinational circuits this is almost a direct conversion for GP and GA (in cases of binary representation). After the code is converted it can be evaluated in two ways: 1) through logic simulator software which works with VHDL; or 2) through real circuit evaluation. In this case the code is synthesized and transferred to a FPGA (see Electronic Design/PLD section). The fitness evaluation can be in this case a measurement of how close the result is from the truth-table.

Figure 5. VHDL code mapped from GP

```
01- library ieee;
02- use ieee.std_logic_1164.all;
03- entity basic_gate is
04-    port(a, b, c:  in std_logic;
05-         y:       out std_logic);
06- end entity;
07- architecture behavior of basic_gate is
08- begin
09-    y <= not(a) and (not(b or c));
10- end behavior;
```

Each chromosome of each population at each generation must pass to this conversion to be calculated a fitness value. With this measurement the EA algorithm proceeds with its loop (Figure 1) and the process is repeated until its termination condition.

Evolvable Hardware Definition

To understand the definition of EHW it is necessary to explain evolution. In the case of a circuit evolution, it is possible in EHW to evaluate the candidate-solutions in software simulation or in a real circuit-on-the-loop. It is called extrinsic evolution (or offline fitness computation – OFC) when all the EA runs in software and only the best chromosome of the last generation

is implemented in hardware. It is called intrinsic evolution when each chromosome of each population at each generation is evaluated in a real circuit during the EA loop. This kind of evolution is also called real-time evolution, hardware in the loop, or EHW (online fitness computation – ONL) (Stoica, Zebulum, & Keymeulen, 2000, p.4). A third kind of evolution mixes the two previous types in the called mixtrinsic evolution (Stoica, Zebulum, & Keymeulen, 2000).

Floreano and Mattiussi (2008, p. 58) follow this concept, but they alert for the fact that EHW has also been used to cover all the EEL area, for example, in Torresen (2004).

Fault-tolerance systems has gained so much attention that another concept for EHW has appeared: according to IEEE Task Force on Evolvable Hardware (2012): the objective of EHW is to design systems that can self-adapt as necessary to compensate for changing operational environments or to survive and recover from faults using simulated evolution to search for new hardware configurations.

Professor Adrian Thompson, the pioneer of EHW, recommends in his home-page that an electronic search for EEL must be made using the following keywords: Hardware Evolution, Evolvable Hardware (EHW, E-Hard), Evolutionary Electronics, EvolWare, and bio-inspired electronics (Thompson, nd), i.e., it is possible to use the terms EEL or EHW to express the same concept.

Pioneer works of EHW are: Louis & Rawlins (1991), which was the first to introduce the idea of using EAs to perform structure design; the first to use FPGAs in structure design (De Garis, 1993); and the first to apply intrinsic evolution (Thompson, 1996).

EHW Specific Danger

When an application runs intrinsic evolution there is a dangerous situation. The circuit could be damaged due to an incorrect connectivity between the components. This is fixed with two actions: the generation of chromosomes that are not capable of produce incompatible connections; or to apply penalties during the EA evaluation phase for individuals that present such anomalies.

Taxonomy

A taxonomy proposed by Torresen (2004) for EHW with slight modifications is subdivided in eight fields and repeated here.

- **Algorithm (EA):** the application can be developed using one or more of the four EA previously presented – GA, GP, EP, ES; or combined with other techniques: a HA (Hybrid Algorithm);
- **Technology (TE):** the system can be analog (A), digital (D);
- **Architecture (AR):** fixed circuit with parameter tuning (CT) or complete circuit design (CD);
- **Building Block (BB):** analog (ANLG) component level (transistor, passive componets), gate (GATE) level (AND, OR, etc), or function level (FUNC) level (multiplexes, adders, etc);
- **Target Hardware (THW):** the circuit to be evolved can be a custom (CST) hardware (e.g. ASIC), a commercially (COM) available PLD (e.g. FPGA), or the experiment was completed only with simulation results (SIM);
- **Fitness Computation (FC):** this category was previously presented;
- **Evolution (EV):** Off-chip (OFCP) evolution (the EA is performed on a separate processor); on-chip (ONCP) evolution (the EA is performed on a separate processor incorporated into the chip containing the target EHW); and complete HW (CHWD) evolution (the EA runs on a special hardware, i.e., not a traditional processor);
- **Scope (SC):** if the circuit is evolved and, at end, it is put to work, this is called static (S) evolution. On the other hand, if each evolved circuit replaces the circuit in operation, this is called dynamic (D) evolution.

EHW Applications

Taking the previous taxonomy as a basis, Table 1 presents a sample of the most important contributions of EHW.

Table 1. Sample of more than 20 years of EEL applications

Application	EA	TE	AR	BB	THW	FC	EV	SC
Digital Basic functions evolution Louis & Rawlins (1991), Barros, Lopes & Stelle (2007)	GA	D	CD	GATE	SIM	OFL	OFCP	S
Amplification and Filter Design London & Colombano (1999)	GA	A	CD	ANLG	CST	OFL	OFCP	S
Analog Circuit Synthesis Koza, Bennett III, Andre, & Keane (1999)	GP	A	CD	ANLG	CST	OFL	OFCP	S
Robot Navigation Keymeulen, Durantez, Konaka, Kuniyoshi & Higuchi (1996)	GA	D	CD	GATE	CST	ONL	CHWD	S
Adaptive Image Compression Salvador, Vidal, Moreno, Riesgo & Sekanina (2012)	ES	D	CD	FUNC	COM	ONL	ONCP	D
Spread Spectrum Balaji & Rao (2012)	HA	D	CT	GATE	COM	ONL	ONCP	D
Automatic Fault Detection and Self-Repair Oreifej & DeMara (2012)	GA	D	CD	GATE	COM	ONL	ONCP	D
Circuit Design Ong & Keane (2004)	HA	D	CD	GATE	COM	OFL	OFCP	S

FUTURE RESEARCH DIRECTIONS

One of the great challenges of EHW is to provide scalability to work with larger circuits. One countermeasure is to enlarge the size of the chromosome to be proportional to larger search spaces. An analogous effect can be obtained through another technique called VLR (Variable Length Representation) (Zebulum, Pacheco, & Vellasco, 2001, p. 53). What encourages this direction is that recent experiments had shown that only 3% of the available slices in an academic FPGA were used to fit a modified GA (Balaji & Rao, 2012).

An important contribution is morphogenesis for EHW (Lee & Sitte, 2004, 2006). With this strategy it is possible to design hardware considering scalability as well as novelty that EAs can provide for solutions.

The possibility of an application to self-adapt and self-repair is a key component for mission-critical systems that can be damaged by radiation, noise, and other sources. NASA/DoD Conference on Evolvable Hardware is a Conference that encourages academic productivity in this direction.

Some challenges are still present. The design flow for an application run in a FPGA (ED section) is time-consuming. Even with more memory or clock in the computer that runs the EDA tool, it is necessary to consider a relevant time to implement just one chromosome. Considering g as the number of generations, p the size of a population and ni the number of individuals in a population, any intrinsic evolution consume a total time of approximately $g \times p \times ni$. This does not take account the size of the chromosome that usually is proportional to the level of the complexity, enlarging the search space. Therefore, high computational resources are mandatory for evolving real-world circuits. These problems are being addressed by machine parallelism and FPGAs with DPR (Dynamic Partial Reconfiguration). Works considering this problem are an open highway.

CONCLUSION

EHW and a brief background necessary to its comprehension were presented.

As just a simple sample of the universe of applications, it is possible to conclude that EHW is continuously presenting advances in the level of complexity of applications, regarding the parameters considered in the taxonomy during these 22 years of academic productivity.

Evolvable Hardware is a fascinating area of research due to its flexibility for several and distinct applications.

REFERENCES

Anadigm. (n.d.). *Anadigm Development Kits*. Retrieved September 12, 2013, from http://www.anadigm.com/devkits.asp

Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.). (2000a). *Evolutionary computation 1: basic algorithms and operators*. Bristol, UK: Institute of Physics Publishing. doi:10.1887/0750306645

Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.). (2000b). *Evolutionary computation 2: advanced algorithms and operators*. Bristol, UK: Institute of Physics Publishing. doi:10.1887/0750306653

Balaji, N., & Rao, K. S. (2012). VLSI-based real-time signal processing solution employing four-phase codes for spread-spectrum applications. *Journal of the Institution of Electronics and Telecommunication Engineers*, *58*(1), 57–64. doi:10.4103/0377-2063.94083

Barros, A. M., Lopes, H. S., & Stelle, A. L. (2007). Automatic FIR filter design method and tool based on genetic algorithms. In *Proceedings of the 1st IEEE Symposium on Computational Intelligence in Image and Signal Processing* (pp. 151-156). Piscataway, NJ: IEEE Computer Society Press.

De Garis, H. (1993). Evolvable hardware: genetic programming of a Darwin machine. In R. F. Albretch, C. R. Reeves, & N. C. Steele (Eds.), *Artificial Neural Nets and Genetic Algorithms* (pp. 441–449). New York: Springer-Verlag. doi:10.1007/978-3-7091-7533-0_64

Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: theories, methods, and technologies*. Cambridge: The MIT Press.

Fogel, L. J. (1962). Autonomous Automata. *Industrial Research Magazine*, *4*(2), 14–19.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*. New York: Wiley.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Boston: Addison-Wesley Longman Publishing, Inc.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Keymeulen, D., Durantez, M., Konaka, K., Kuniyoshi, Y., & Higuchi, T. (1996). An evolutionary robot navigation system using a gate-level evolvable hardware. In T. Higuchi, M. Iwata, & W. Liu (Eds.), *Proceedings of the First International Conference of Evolvable Hardware: from biology to hardware (ICES96)* (pp. 195-209). New York: Springer-Verlag.

Koza, J. R. (1992). *Genetic programming*. Cambridge, MA: The MIT Press.

Koza, J. R., Bennett, F. H. III, Andre, D., & Keane, M. A. (1999). *Genetic programming III: Darwinian invention and problem solving*. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Lee, J., & Sitte, J. (2004, December). A gate-level model for morphogenetic evolvable hardware. In *Proceedings of the International Conference on Field Programmable Technology (ICFPT´2004)* (pp. 113-119). Brisbane, Australia.

Lee, J., & Sitte, J. (2006, June). Gate-level Morphogenetic Evolvable Hardware for Scalability and Adaptation on FPGAs. In *Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems (AHS´2006)* (pp.145-152). Istambul, Turkey.

Lohn, J. D., & Colombano, S. P. (1999). A circuit representation technique for automated circuit design. *IEEE Transactions on Evolutionary Computation*, *3*(3), 205–219. doi:10.1109/4235.788491

Lohn, J. D., & Hornby, G. S. (2006, February). Evolvable hardware: using evolutionary computation to design and optimize hardware systems. *IEEE Computational Intelligence Magazine*.

Louis, S. J., & Rawlins, J. E. (1991). Designer-genetic algorithms: genetic algorithms in structure design. In R. K. Belew, & L. B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)* (pp. 53-60). San Francisco, CA: Morgan Kaufmann.

Neri, F., Cotta, C., & Moscato, P. (2011). *Handbook of Memetic Algorithms*. New York: Springer-Verlag.

Ong, Y. S., & Keane, A. J. (2004). Meta-lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, *8*(2), 99–110. doi:10.1109/TEVC.2003.819944

Oreifej, R. S., & DeMara, R. F. (2012). Intrinsic evolvable hardware platform for digital circuit design and repair using genetic algorithms. *Applied Soft Computing*, *12*, 2470–2480. doi:10.1016/j.asoc.2012.03.032

Pedroni, V. A. (2008). *Digital electronics and design with VHDL*. Burlington, MA: Elsevier, Inc.

Pedroni, V. A. (2010). *Circuit design and simulation with VHDL* (2nd ed.). Cambridge: The MIT Press.

Pierzchala, E., Gulak, G., Chua, L. O., & Rodrigues-Vásquez, A. (Eds.). (1998). *Field-programmable analog arrays. Dordrectht, NL.* Kluwer Academic Publishers. doi:10.1007/978-1-4757-5224-3

Rechenberg, I. (1965). *Cybernetic solution path of an experimental problem (Technical Report Library Translation No. 1122)*. Farnborough, UK: Royal Aircraft Establishment.

Rechenberg, I. (1973). *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Stuttgart, Germany: frommann-holzboog.

Salvador, R., Vidal, A., Moreno, F., Riesgo, T., & Sekanina, L. (2012). Accelerating FPGA-based evolution of wavelet transform filters by optimized task scheduling. *Microprocessors and Microsystems*, *36*, 427–438. doi:10.1016/j.micpro.2012.02.002

Stoica, A., Zebulum, R. S., & Keymeulen, D. (2000). Mixtrinsic Evolution. In *Proceedings of the 3rd International Conference on Evolvable systems: from biology to hardware (ICES'2000)* (pp. 208-217). Edinburgh, Scotland, UK.

Thompson, A. (1996). Silicon Evolution. In J. R. Koza, D. E. Goldberg, & R. L. Riolo (Eds.), *Proceedings of Genetic Programming (GP96)* (pp. 444–452). Cambridge: The MIT Press.

Thompson, A. (n.d.). Evolutionary Electronics at Sussex. Retrieved September 12, 2013, from http://www.sussex.ac.uk/Users/adrianth

Torresen, J. (2004, September). An evolvable hardware tutorial. In *Proceedings of the 14th International Conference on Field Programmable Logic and Applications (FPL'2004)* (pp. 821-830). Antwerp, Belgium.

Whitaker, J. C. (2005). *The electronics handbook*. Boca Raton, FL: CRC Press.

Zebulum, R. S., Pacheco, M. A. C., & Vellasco, M. M. B. R. (2001). *Evolutionary electronics: Automatic design of electronic circuits and systems by genetic algorithms*. Boca Raton, FL: CRC Press. doi:10.1201/9781420041590

ADDITIONAL READING

Altera. (2013). All Development Kits. Altera Corporation. Retrieved September 12, 2013 from http://www.altera.com/products/devkits/kit-dev_platforms.jsp

Boylestad, R. L. (2000). *Electronic devices and circuits theory* (10th ed.). Boston: Pearson Education.

Chu, P. P. (2008). *FPGA prototyping by VHDL examples*. New Jersey: John Wiley & Sons, Inc.

Goodman, E. (2002). GALOPPS 3.2.4 - the "Genetic ALgorithm Optimized for Portability and Parallelism System." *MSU GARAGe Software*. Retrieved September, 12, 2013, from http://garage.cse.msu.edu/software/galopps/index.html

Higuchi, T. Liu, Yong, & Yao, X. (2006). Evolvable hardware. New York: Springer.

Koza, J. R. (1994). *Genetic programming II: automatic discovery of reusable programs*. Cambridge, MA: The MIT Press.

Langdon, W. B., & Poli, R. (2002). *Foundations of genetic programming*. New York: Springer-Verlag. doi:10.1007/978-3-662-04726-2

Mentor Graphics. (2013). ModelSim: leading simulation and debugging. Retrieved September 12, 2013 from http://www.mentor.com/products/fpga/model

Punch, B., & Zongker, D. (1998). lil-gp Genetic Programming System, version 1.1. Retrieved September, 12, 2013, from http://garage.cse.msu.edu/software/lil-gp/index.html

Quagliarella, D., Périaux, J., Poloni, C., & Winter, G. (1998). *Genetic algorithms and evolution strategy in engineering and computer science: recent advances and industrial applications*. Chichester: John Wiley & Sons.

Taub, H. (1982). *Digital circuits and microprocessors*. New York: McGraw-Hill.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, *4*, 65–850. doi:10.1007/BF00175354

Whitley, D., Gordon, V. S., & Mathias, K. (1994). Lamarkian evolution: the Baldwin effect and function optimization. *Paper presented at 3rd International Conference on Evolutionary Computation - Parallel Problem Solving (PPSN III)*. Jerusalem, Israel, p. 6-15.

Xilinx. (2013). All Programmable FPGAs. *Xilinx Inc*. Retrieved September 12, 2013 from http://www.xilinx.com/products/silicon-devices/fpga/index.htm

KEY TERMS AND DEFINITIONS

Evolutionary Algorithms (EAs): Algorithms which are based on evolutionary computation principles.

Evolutionary Computation (EC): A research area that consists of computational optimization search methods based on heuristics which imitates nature to find optimized solutions.

Evolutionary Electronics (EEL): Also known as Hardware Evolution, Evolvable Hardware, EvolWare, or bio-inspired electronics, is a research area which covers all the applications involving the use of Evolutionary Computation in electronic systems' design.

Field Programmable Analog Array (FPAA): A programmable chip that contains operational amplifiers and passive components that can be arranged in different ways. To receive different configurations, the FPAA is connected through an interface to a computer which contains vendor specific software that offers to the developer a specification language usually graphical.

Field Programmable Gate Array (FPGA): A programmable chip that contains logic gates as well as that can be arranged in different ways. To receive different configurations, the FPGA is connected through an interface to a computer which contains vendor specific software that offers to the developer two kinds of specification languages: one graphical and the other textual. VHDL (IEEE Standardized), Verilog, and AHDL are examples of the textual description languages.

Genetic Algorithm (GA): An evolutionary algorithm that works with encoded population-based search imitating nature through the following mechanisms: selection, reproduction and mutation. The encoding consists of mapping each possible solution belongs to the search space to a symbol string, usually bits. Each encoded string is called chromosome. A population of chromosomes which is a sample of the search space is the base for the heuristic search.

Genetic Programming (GP): A variation of genetic algorithm in which the members of the population are parse trees. GP usually is used to evolve symbolic information (examples are: computer code, symbolic regression, and automatic electrical circuit design).