Preface

At the beginning of the 1990s research started in how to combine soft computing with reconfigurable hardware in a quite unique way. One of the methods that was developed has been called *evolvable hardware.* Thanks to evolutionary algorithms researchers have started to evolve electronic circuits routinely. A number of interesting circuits – with features unreachable by means of conventional techniques – have been developed. Evolvable hardware is quite popular right now; more than fifty research groups are spread out over the world. Evolvable hardware has become a part of the curriculum at some universities. Evolvable hardware is being commercialized and there are specialized conferences devoted to evolvable hardware.

On the other hand, surprisingly, we can feel the lack of a theoretical background and consistent design methodology in the area. Furthermore, it is quite difficult to implement really innovative and practically successful evolvable systems using contemporary digital reconfigurable technology.

This monograph deals with evolvable hardware and its mathematical generalization: evolvable computational machines. The aim is to introduce the concepts (missing in the field nowadays) that designers as well as theoreticians could benefit from. Nowadays it is more than evident that component technology plays a crucial role in engineering. And it seems that evolvable hardware could benefit from the component approach as well. Hence an original idea, a theory, and implementations of the *evolvable component* are developed and presented in this book. We do believe that the concept of the evolvable component will lead to a much deeper understanding of computer systems designed by methods inspired by evolutionary biology.

Structure of the Book. Because the book deals with various scientific areas such as evolutionary algorithms, hardware design and theoretical computer science, a relatively large portion of the text is devoted to the introductory chapters. We will start with a general overview of soft computing and bioinspired hardware in the first chapter, which – together with Chaps. 2, 3, and 4 – could also serve as an introduction to the field of (digital) evolvable hardware for those who are interested in this popular and topical area of computer engineering and computer science and who are now meeting evolvable hardware for the first time.

Chapter 2 briefly introduces the contemporary paradigm of digital circuit design (i.e. conventional circuit design). In particular, the basic characteristics of field programmable gate arrays (FPGA) and their reconfiguration options will be discussed. We have also included sections on reconfigurable computing and nanotechnology, which represent two recent stages of computational system design and implementation and which are closely related to evolvable hardware.

Chapter 3 provides a survey of evolutionary algorithms. We will emphasize evolutionary design in contrast to evolutionary optimization. A formal definition of an evolutionary algorithm is introduced because it will be utilized in Chap. 6.

It is possible to find in the literature a suitable survey of reconfigurable hardware and evolutionary algorithms for the purposes of this book. It is also the case of evolvable hardware. For instance, Gordon's and Bentley's paper [57] provides a good introduction to the field. According to our knowledge there exist only two books fully devoted to evolvable hardware. The first is Adrian Thompson's dissertation from 1996 [187] and the second is "Evolutionary Electronics" written by Zebulum, Pacheco and Vellasco [230]. However, we have decided to summarize the state-of-the-art of digital evolvable hardware in Chap. 4. This survey is mainly based on papers published during the last seven years on two evolvable hardware-related conference series (the Evolvable Systems: From Biology to Hardware Conferences and the NASA/DoD conferences on Evolvable Hardware), some journal papers, and our insight into the problem.

The component approach to evolvable hardware is introduced in Chap. 5. We will show how evolvable components support dynamic environment and reusability concepts. An automaton, which controls the behavior of the evolvable component, will be derived and considered as a basic reusable unit.

The formal definition of a general evolvable computational machine will be formulated in Chap. 6. Then some properties of evolvable computational machines will be investigated formally. Evolvable machines will be related to an emerging field of theoretical computer science: hypercomputation (or super-Turing computation), where well-defined computations are not exhausted by the computations that can be carried out by a Turing machine. In particular, we will prove that evolvable systems exhibit a super-Turing computational power. In this case some concepts and results of theoretical computer science (such as Turing machines, the Church–Turing thesis, cellular automata, computability, etc.) will be utilized and recalled.

We will develop the evolvable component for image pre-processing in Chap. 7 in order to illustrate that the component approach is beneficial for the solution of real-world problems. The various image operators evolved in using the component will be compared with conventional operators in terms of quality and implementation costs.

Chapter 8 describes an implementation of a virtual reconfigurable circuit utilized in the evolvable component for image pre-processing. This virtual reconfigurable circuit enables us to speed up the component adaptation substantially. We will show that it is possible to synthesize the virtual reconfigurable circuit into an ordinary FPGA.

Finally, Chap. 9 summarizes the results we present in this book and gives some directions for future research.

Readers. The reader is assumed to be familiar with the basics of theoretical computer science, software design and hardware design in the scope of a B.Sc. degree in computer engineering or equivalent. All readers will benefit from a unified and coherent approach to the evolvable systems presented in this book. In contrast to the mainstream single-orientation publications, the book is based on formal, software and hardware viewpoints. Academic and scientific readers should benefit from the system-level approach and formal concepts introduced for evolvable systems in the book. Practitioners, evolutionary designers, and hardware designers will probably like the results of experiments and the methods which can directly be applied in research or industry. The concept of the virtual reconfigurable circuit should be important for people from the reconfigurable computing community. Finally, teachers and students can utilize the first chapters on evolutionary algorithms, reconfigurable circuits and evolvable hardware during lectures and home reading.

Acknowledgments. This work was carried out in the period 1998–2002 at the Department of Computer Science and Engineering of the Faculty of Electrical Engineering and Computer Science and, after its transformation in 2002, at the Faculty of Information Technology of the Brno University of Technology, Czech Republic.

I am grateful to Vladimír Drábek for his continuous support of my research during recent years. I would also like to thank my colleagues at the Faculty of Information Technology, namely Alexander Meduna, Otto Fučík, Zdeněk Kotásek, Richard Růžička, Azeddien M. Sllame, Jiří Očenášek, Tomáš Dulík, and Josef Strnadel.

I am greatly indebted to Jim Tørresen for his help during my work with the Department of Informatics, University of Oslo, funded by the Research Council of Norway in 2001.

I would like to acknowledge the Springer-Verlag editor Ingeborg Mayer, the series editor A. E. Eiben and the Le–TeX editor Peggy Glauch for their close cooperation.

Last, but not least, I am very grateful to three beautiful women: my mother, my sister and my girlfriend. I would like to thank them for their patience, tolerance, and love.

My research was partly supported by the following projects: *Formal Approach in Digital Circuit Diagnostics – Testable Design Verification* and *Evolvable Hardware-Based Application Design Methods* (financed by the Grant Agency of the Czech Republic under numbers 102/01/1531 and 102/03/P004) and by the Research Intention No. MSM 262200012 – *Research in Information and Control Systems*. I also greatly appreciate the financial support that I have received from Siemens (the Siemens Awards in 1999 and 2000) and from the Brno University of Technology.

Brno, August 2003 *Lukáš Sekanina*

1. Introduction

Everyday we use computers whose architecture is in principle derived from the models introduced more than half a century ago. Of course, our computers exhibit more computing resources, operate faster, utilize deeper parallelism, pipelining and virtualization and so on than former models could provide (as seen in classical textbooks on computer architectures [68, 78, 170]). Computer engineering, which deals with constructing computers and their applications, is a well-established field as well as commercial business. Computer science investigates the computational limits of computers as well as the limits of various (sometimes impracticable) computational systems [59]. Recall that computer science is primarily about two important things:

- the representation and processing of *information*,
- the *complexity* of computers and problems.

Modern (or post-modern) science and engineering have discovered that the most interesting and important things appear at the borders of research fields. Certainly, this is also the case for computers. We can mention some typical "partners" of computer science and engineering: biology, medical science, chemistry, physics, sociology, psychology, engineering of all types, and so on. In most interactions of these fields computers were utilized to "help" people from the "other" domains and to do hard mechanical work.

In recent years we have observed the development of approaches that do something else. They are exploiting the other fields in order to change and improve the fundamental architectonic concepts of computers. However, these approaches originated in the minds of A. M. Turing, J. von Neumann, S. W. McCulloch, W. Pitts, H. J. Bremermann, I. Rechenberg, L. Fogel, J. Holland, R. Feynman, D. Deutsch and many others many years ago. They represent attempts to go much deeper into nature in order to reveal its potential for information processing and storage. They could lead to a significant increase of our computational power. The principles can be embodied into software in order to improve its quality. The real challenge is to change and redefine the nature of computing hardware. It should have an impact not only on software and hardware computational platforms but also on the application domain of computers. Computer engineers have started to construct the systems which we could only dream about a few years ago.

1.1 Natural Computing

Natural computing is a term that is usually introduced to integrate the approaches where something unconventional and unorthodox for computers is introduced into computer science and engineering. We try to understand and harvest the computational power of nature in a more basic and direct way than so far. Natural computing includes a number of research fields: evolutionary computing, neural computing, fuzzy computing, cellular computing, DNA computing, quantum computing, membrane computing and some others. These approaches are characterized not only by the special way in which the computation is performed but also by the special "hardware" platforms where the computation is carried out. As this book mainly deals with evolutionary computation and bioinspired hardware, the following sections are devoted to a bird's eye view of the whole field. We can expect that new computational models will be introduced in the near future, based on principles we have never heard of in a computer context.

1.1.1 Soft Computing

Computer engineers have always been attracted by the design of *intelligent programs* – programs which can solve problems without explicitly programming solutions for these problems. A certain kind of success was achieved by means of methods which borrowed some concepts from biology. A *genetic algorithm* is a typical example [71]. In the 1960s J. Holland was one of the people who discovered that Darwinian theory of evolution can be utilized to provide a powerful way to perform optimization on a computer. He recognized evolution as a creative process, the essence of which is making *something out of nothing* automatically. In this book, we will utilize this method for design at the hardware level directly.

Traditional artificial intelligence has studied the possibilities of creating programs and machines that can rival human abilities in some tasks by means of heuristics, encapsulated knowledge, symbolic logic and many other methods. In contrast, modern *computational intelligence* (or *soft computing*) has explored the potential for creating intelligent machines by modeling the behaviors and mechanisms that underlie biologically intelligent organisms. In general we can identify the following sources of inspiration [12, 108, 172, 174]:

- *Phylogeny* concerns the temporal evolution of a genetic program, the hallmark of which is the evolution of a species. The emergence of living organisms is based upon the reproduction of the program (genotype), subject to an extremely low error rate at the individual level. This ensures that the identity of the offspring remains practically unchanged. However, mutation and recombination give rise to new genetic material that is superfluous for the survival of species and for their continuous adaptation to a changing environment. The evolutionary algorithm is a basic model inspired by this process in computational intelligence [7, 8, 44, 121]. Note that the evolutionary algorithm is a general term. A number of variants of the evolutionary algorithm exist – genetic algorithms [56, 71], genetic programming [95], evolutionary programming [46] and evolutionary strategy [156] are the most well-known ones. It is more than evident that the field is very much inspired by Dawkin's popular books on evolution [31, 32].

- *Embryology* and *ontogeny* deal with the development of multicellular organisms. It is based on the successive *division* of the mother cell, with each newly formed cell possessing a copy of the original genome, followed by a *specialization* of the daughter cells in accordance with their surroundings. We can mention *cellular automata* and *Lindenmayer systems* (*L-systems*) as probably the most well-known computational models inspired by ontogeny. Cellular automata were originated by Ulam and von Neumann in the 1940s to provide a formal framework to study the behavior of complex systems, especially the questions of whether computers can self-replicate. Then properties such as reversibility, universality and emergent behavior have been studied [173, 193, 219]. L-systems were intended to model cellular division; nowadays they are popular in many areas, including computer graphics and theoretical computer science [59, 110].
- *Epigenesis* is the process emerging upon reaching a certain level of complexity that permits the individual to integrate the vast quantity of interactions with the outside world and to learn. Epigenesis primarily includes the nervous system, the immune system and the endocrine system. While popular *artificial neural networks* are inspired by the nervous system [119], there exist (not so widespread) computational systems based on the principles of the immune system [29]. Note that the very first model of the neuron was published by W. S. McCulloch and W. Pits in 1943. Practical applicable learning algorithms were initially defined by D. Hebb in 1949.
- Other approaches are based on principles observable in human society, colonies of interacting individuals or predator–prey conflicts. We have to mention Artificial Life, in which artificial digital creatures with the features we understand as the expression of life, live, see, die, reproduce, compete, fight, emerge and so on in computers. The stuff of this life is nonorganic matter, and its essence is information [108]. Artificial Life is devoted to the creation and study of these lifelike organisms.

The above-listed models are often combined together. From a scientific viewpoint, we can learn about biological principles via modeling of these principles. From an engineering viewpoint, which is relevant for this book, we can utilize the above-mentioned principles to design more powerful, adaptive, effective and competitive engineering products.

1.1.2 Quantum Computing

Our world is quantum mechanical [34]. If computers become smaller, they will operate in the world of quantum effects. R. Feynman asked whether quantum physics could be simulated on classical computers and vice versa in 1982. The first step was done by Deutsch, who designed a (theoretically) physically realizable model for a *quantum computer* and developed its elementary theory [33]. The challenge is to exploit quantum parallelism in practical applications. It was shown that it is possible to factor integers and to compute discrete logarithms in polynomial time on a quantum computer. These two important problems of cryptography require exponential time on conventional computers. However, there is a problem with the scale of quantum computers. Only a few bits are available for computation today. The open question remains as to whether one can build a practically successful quantum computer [60].

1.1.3 DNA Computing

As shown originally by L. M. Adleman in 1994 [3], computation can be performed on molecules directly. The best-studied molecules for this purpose to date have been DNA [139] and *bacteriorhodopsin* [54]. While DNA computing is based on the massive parallelism of DNA strands and Watson–Crick complementarity, the protein bacteriorhodopsin – which contains the light-sensitive rhodopsin – offers the potential for optical computing. As with quantum computing, the concept of DNA computing is well formalized [139, 224]. It could be beneficial for some specific tasks that require exponential time on conventional computers. Here the problem is with the reliability of the computational process and with the amount of DNA that must be supplied.

1.1.4 Membrane Computing

Membrane computing is another field related to biology. G. Paun initiated the research on membrane computing in 1998. He was inspired by the structure and functioning of the living cell and devised distributed parallel computing models in the form of membrane systems [140]. Such computing devices operate with the regions defined by a membrane structure. The regions contain multisets of objects which can evolve according to given evolution rules. The rules are applied in a maximally parallel manner and nondeterministically. The objects can interact and can pass through membranes. The computation is defined as a transition from a configuration of a system to another configuration. However, only software implementations exist. The formal approach to membrane computing is based on the theory of *P systems.*

1.2 Bioinspired Hardware

The previous approaches have dealt with special sorts of hardware. In this book we will consider only the devices which are currently available and which belong to traditional and well-explored implementation platforms – digital circuits.

The methods of artificial intelligence have also supported *hardware design* for a long time. However, ten years ago computational intelligence combined with reconfigurable circuits gave rise to a new engineering field – *biologically inspired hardware* – in which biological principles are used during the circuit design phase or during the operational time of a physical electronic circuit. Reconfigurable hardware is a special architecture whose function (physical internal interconnection of elements) can be programmed from outside.

Nowadays we can identify several research areas including evolvable hardware, embryonics, immunotronics, evolvable sensors or neural hardware in which the usage of biological principles has led to the design of circuits that are "better" than conventional circuits[1] in terms of quality, implementation cost, fault tolerance, learning and adaptability to a changing environment.

Combining evolutionary algorithms with reconfigurable circuits in the areas of *evolvable hardware* (EHW) and *evolutionary electronics* has attracted the greatest attention of designers in recent years. *Embryonics* (embryonic electronics) tries to employ some of the developmental processes of multicellular organisms to design fault-tolerant and robust circuits with features such as self-repair and self-replication [114, 115]. *Immunotronics* (i.e. immunological electronics) refers to the usage of the principles of immune systems to implement fault tolerance and circuit protection [15, 29]. By *neural hardware* we usually mean a hardware implementation of a model of the nervous system [51, 136, 155].

It is natural to place all bioinspired systems introduced in the previous paragraphs to the *POE* model, which was established for their classification [172]. As seen in Fig. 1.1, *P* (phylogeny), *O* (ontogeny) and *E* (epigenesis) indicate three orthogonal axes that define the space where the bioinspired systems are situated. For instance, evolvable hardware occupies the *P* axis, embryonics is on the *O* axis, and neural hardware and immunotronics take place on the *E* axis. Systems that show a higher degree of hardware implementation and a deeper level of inspiration in biology are plotted upward along a given axis. Combining two biological principles we define the *PE* plane, the *OE* plane, and the *PO* plane. Finally, the development of an artificial neural network, implemented on a self-replicating multicellular automaton whose genome is subject to evolution constitutes a possible example situated in the *POE* space [172].

We will mainly be interested in the evolutionary approach in this book. *Evolutionary electronics* is a term that was introduced in 1997 to cover all the applications of evolutionary techniques to electronic system design. Note that evolutionary algorithms have already been applied to optimization problems related to digital chip design (like placement, routing, and high-level synthesis [35, 118, 175]) since the middle 1980s.

[1] By *conventional* or *traditional* we mean a product or a design approach that is carried out using common techniques that do not employ artificial intelligence.

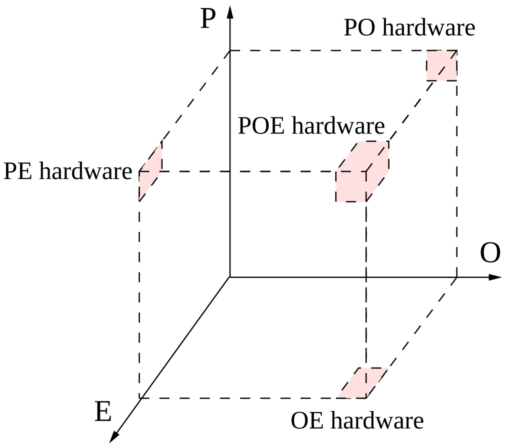

Fig. 1.1. The POE model for the classification of bioinspired systems (developed in [172])

Evolvable hardware refers to the evolutionary design of electronic circuits directly at the hardware level. The concept will be explained in detail in Chap. 4. For now we just say that the circuits are encoded in the chromosomes of an evolutionary algorithm. The chromosomes are uploaded into reconfigurable hardware and evaluated to obtain their quality (fitness). The evolutionary algorithm generates new populations of circuits and when it is terminated we obtain the target circuit.

Higuchi's [69] and de Garis's [49] papers are some of the first reports where it is possible to find the term *evolvable hardware* explicitly. Although the term *adaptive hardware* is sometimes considered as more suitable for the method, the term evolvable hardware was chosen in the early 1990s because it directly emphasizes the evolutionary approach, which is utilized to achieve adaptation. Up to now, evolvable hardware has been successfully applied for the design of a number of digital as well as analog circuits. Furthermore, online evolution has enabled the emergence of high performance and adaptive systems for the applications in which the problem specification is unknown beforehand and can vary with time. In addition to the evolutionary circuit design carried out in physical hardware, some authors regard the presence of continual adaptation to an environment as the second crucial feature of evolvable hardware [172, 227]. This will be discussed in more detail in Sect. 4.5.3.

Recently, Miller has used the more general term *evolvable matter* to address the usage of evolutionary algorithms for the design on any physical reconfigurable platform (e.g. chemical) [127]. The idea behind the concept is that applied voltages may induce physical changes that interact in unexpected ways with other distant voltage-inducted configurations in a rich physical substrate. In other words, it should theoretically be possible to perform the evolution directly *in materio* if the platform is configurable in some way. Miller and Downing reviewed this promising technology in [130]. Tour's group has presented a concrete working example – the Nanocell [205].

The authors of [172] concluded their report: "Looking (and dreaming) toward the future, one can imagine nanoscale (bioware) systems becoming a reality, which will be endowed with evolutionary, reproductive, regenerative, and learning capabilities. Such systems could give rise to novel species which will coexist alongside carbon-based organisms."

1.3 Motivation for Research

The book is primarily devoted to digital evolvable hardware. Although some circuits have been successfully evolved using evolvable hardware, the standard routine application of evolvable hardware is practically unexplored. The problem is that designers do not perceive what evolvable hardware actually means for a system in which evolvable hardware should be applied. We can ask with a designer the following questions. Is evolvable hardware hardware or software? How shall I integrate evolvable hardware to my system under design? Can I reuse evolvable hardware in some other applications? Does there exist any general architecture or template which reflects a typical evolvable hardware-based application? Why is it difficult to apply evolvable hardware right now?

It is difficult to find a general answer. Furthermore, it seems that nobody is noticeably interested in a conceptual approach now. We can only guess the reasons for doing so: (1) evolvable hardware is a very young field and there is still some space for experimental design approaches, (2) evolvable hardware has not become commercially attractive yet, and (3) there are not widely accessible physical platforms suitable for the design of evolvable hardware-based applications. Hence evolvable hardware remains an approach for a group of experts who perform ad hoc application designs.

We argue that a conceptual approach more efficient than the ad hoc approach is important for evolvable hardware right now, since the first commercial applications of evolvable hardware have been developed [70]. We can observe that a *component approach* plays a dominant role for software as well as for hardware design. Hence it seems natural to introduce the component approach to evolvable hardware-based systems as well. The systematic application of such an approach usually introduces a unified view on all systems that are designed. Hence a semi-automatic design can be utilized for routine application design. Recall that Stoica et al. mentioned three years ago that "the path leads to the IP level and evolvable hardware solutions will become an integrated component in a variety of systems that will thus have an evolvable feature" [182]. However, no other (implementation) details were given there.

When a unified view on the evolvable hardware-based systems is established, it is natural to apply formal methods in order to define the systems in terms of theoretical computer science and to investigate the properties of the systems. Nothing is practically known about evolvable hardware (and evolvable computational machines) from a theoretical viewpoint [227].

As we have noted, there is not a widely applicable hardware platform for the design of evolvable hardware-based applications. Most research is carried out on reconfigurable circuits like field programmable gate arrays, which are not usually suitable for real-world applications of evolvable hardware because of various problems, mainly to do with the method of reconfiguration. On the other hand, these devices are relatively cheap in comparison with ASIC designs. It would be beneficial to introduce an approach suitable for the design of real-world applications of evolvable hardware in common FPGAs.

Now we can formulate the goals we would like to reach in this book. The primary *objective of the book* is to show possible solutions to at least some of the problems, which appear when we want to perform a routine design of evolvable hardware-based applications using common FPGAs. The partial goals are to:

1. introduce a component approach to evolvable hardware,
2. define an "evolvable system" and investigate its properties formally,
3. show the applicability of the component approach for the design of a practical real-world application, and
4. introduce a method for the implementation of real-world applications of evolvable hardware in common FPGAs.

In order to accomplish these objectives we start with the following introductory chapters on reconfigurable hardware and evolutionary algorithms, which are essential prerequisites for the heart of the matter.