INTRODUCTION TO EVOLVABLE HARDWARE
A Practical Guide for Designing Self-Adaptive Systems

GARRISON W. GREENWOOD
Portland State University

ANDREW M. TYRRELL
University of York

IEEE Computational Intelligence Society, Sponsor

IEEE Press Series on Computational Intelligence
David B. Fogel, Series Editor

IEEE PRESS

WILEY-INTERSCIENCE
A JOHN WILEY & SONS, INC., PUBLICATION
INTRODUCTION TO EVOLVABLE HARDWARE
INTRODUCTION TO
EVOLVABLE HARDWARE
A Practical Guide for Designing
Self-Adaptive Systems

GARRISON W. GREENWOOD
Portland State University

ANDREW M. TYRRELL
University of York

IEEE Computational Intelligence Society, Sponsor

IEEE Press Series on Computational Intelligence
David B. Fogel, Series Editor

IEEE PRESS

A JOHN WILEY & SONS, INC., PUBLICATION
To my wife Linda
and my two children
Matthew and Sarah, who
all mean the world to me
—GWG

To my wife
Maggie for her everlasting
support and love
—AMT
CONTENTS

PREFACE xi
ACKNOWLEDGMENTS xiii
ACRONYMS xv

1 INTRODUCTION 1
  1.1 Characteristics of Evolvable Circuits and Systems / 1
  1.2 Why Evolvable Hardware Is Good (and Bad!) / 5
  1.3 Technology / 6
  1.4 Evolvable Hardware vs. Evolved Hardware / 9
  1.5 Intrinsic vs. Extrinsic Evolution / 10
  1.6 Online vs. Offline Evolution / 12
  1.7 Evolvable Hardware Applications / 13
       References / 15

2 FUNDAMENTALS OF EVOLUTIONARY COMPUTATION 17
  2.1 What Is an EA? / 17
  2.2 Components of an EA / 18
       2.2.1 Representation / 18
       2.2.2 Variation / 21
       2.2.3 Evaluation / 23
       2.2.4 Selection / 26
       2.2.5 Population / 28
3 RECONFIGURABLE DIGITAL DEVICES

3.1 Basic Architectures / 35
  3.1.1 Programmable Logic Devices / 38
  3.1.2 Field Programmable Gate Array / 39
3.2 Using Reconfigurable Hardware / 45
  3.2.1 Design Phase / 49
  3.2.2 Execution Phase / 50
3.3 Experimental Results / 53
3.4 Functional Overview of the POEtic Architecture / 55
  3.4.1 Organic Subsystem / 59
  3.4.2 Description of the Molecules / 59
  3.4.3 Description of the Routing Layer / 62
  3.4.4 Dynamic Routing / 62
3.5 Remarks / 64
  References / 64

4 RECONFIGURABLE ANALOG DEVICES

4.1 Basic Architectures / 67
4.2 Transistor Arrays / 70
  4.2.1 The NASA FTPA / 72
  4.2.2 The Heidelberg FPTA / 82
4.3 Analog Arrays / 89
4.4 Remarks / 93
  References / 93

5 PUTTING EVOLVABLE HARDWARE TO USE

5.1 Synthesis vs. Adaption / 95
5.2 Designing Self-Adaptive Systems / 96
  5.2.1 Fault Tolerant Systems / 96
  5.2.2 Real-Time Systems / 101
5.3 Creating Fault Tolerant Systems Using EHW / 102
5.4 Why Intrinsic Reconfiguration for Online Systems? / 103
5.5 Quantifying Intrinsic Reconfiguration Time / 104
5.6 Putting Theory Into Practice / 108
  5.6.1 Minimizing Risk With Anticipated Faults / 109
  5.6.2 Minimizing Risk With Unanticipated Faults / 111
5.6.3  Suggested Practices / 113
5.7  Examples of EHW-Based Fault Recovery / 113
  5.7.1  Population vs. Fitness-Based Designs / 114
  5.7.2  EHW Compensators / 117
  5.7.3  Robot Control / 125
  5.7.4  The POEtic Project / 138
  5.7.5  Embryo Development / 156
5.8  Remarks / 172
    References / 172

6  FUTURE WORK

  6.1  Circuit Synthesis Topics / 177
    6.1.1  Digital Design / 178
    6.1.2  Analog Design / 184
  6.2  Circuit Adaptation Topics / 185
    References / 187

INDEX

ABOUT THE AUTHORS
The complexity of electronic and computer systems continues to increase at a rate that one might consider too fast for designers to cope with. This increase in complexity enables us to produce impressive engineering systems: aircraft, cars, mobile phones, the internet, “intelligent” homes, for example. However, there are negative aspects to this increased complexity most obviously how to design, manage and reason about such complex systems. Biological systems are many orders of magnitude more complex than anything we can currently produce. Unfortunately as complexity increases so do the chances of faults and errors occurring in these systems. For non-maintainable systems, those that are not available to repair, such as satellites, deep-sea probes or long-range spacecraft, faults and errors can damage the system making the device useless. Are there different ways to produce our current and future complex engineering systems?

Evolvable hardware (EHW) is a dynamic field that brings together reconfigurable hardware, artificial intelligence, fault tolerance and autonomous systems. EHW uses simulated evolution to search for new hardware configurations. The evolution is performed by a variety of different stochastic search algorithms such as genetic algorithms, evolutionary programming or evolution strategies. The evolved hardware is implemented on reconfigurable devices such as field programmable gate arrays (FPGAs), field programmable analog arrays (FPAAs) or field programmable transistor arrays (FPTAs). Each device is configured to define its architecture (and thus function) and the purpose of this evolution is to find the best performing architecture for the given application.

EHW techniques have been successfully used for both original system design and online adaptation of existing systems. It is in latter application area that has generated the most interest. EHW allows systems to self-adapt to compensate for failures or unanticipated changes in the operational environment. This capability
has attracted the attention of NASA and the military because operating in extreme physical environments, coupled with the need for high reliability, is the norm for systems deployed by these agencies. It is for this reason this book places a special emphasis on using EHW in fault tolerant applications.

This book provides a comprehensive view of this growing field for researchers, engineers, designers and managers involved in the design of adaptive and high reliability systems. The reader is introduced to the basic terminology and principles of EHW, reconfigurable hardware, algorithms that conduct the simulated evolution, and system integration concepts. Background information is included on real-time systems and fault-tolerant principles. Several real-world application examples (both digital and analog) are included to teach the basic concepts and to illustrate the power and versatility of EHW.

The motivation behind this book comes from the realization that all of the EHW literature is scattered in a variety of journal articles and conference proceedings. The authors have attempted to bring together, under one cover, the main concepts behind EHW, which will allow the reader to begin applying EHW techniques in a short period of time.

G. W. Greenwood
A. M. Tyrrell
ACKNOWLEDGMENTS

The authors wish to thank the following people for their help, whether implicitly or explicitly, in the production of this book: Cesar Ortega, Gordon Hollingworth, Renato Krohling, Will Barker, Crispin Cooper, Andy Greensted, Mic Lones, Alex Jackson, Richard Canham, Steve Smith, Gianluca Tempesti, Julian Miller, Yann Thoma, Manuel Moreno, Daniel Mange, Adrian Stoica, Ricardo Zebulum, David Fogel, Karlheinz Meier, and Joerg Langeheine. A particular thanks goes to Hugo de Garis, for his thorough review of the first draft of this book, and to Paul Garner, for his patience with the numerous changes to the figures!

A.M.T

G.W.G
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>application specific integrated circuit</td>
</tr>
<tr>
<td>CLB</td>
<td>configurable logic block</td>
</tr>
<tr>
<td>CPLD</td>
<td>complex programmable logic device</td>
</tr>
<tr>
<td>COTS</td>
<td>commercial-off-the-shelf</td>
</tr>
<tr>
<td>CGP</td>
<td>cartesian genetic programming</td>
</tr>
<tr>
<td>DFT</td>
<td>design-for-test</td>
</tr>
<tr>
<td>EA</td>
<td>evolutionary algorithm</td>
</tr>
<tr>
<td>EDA</td>
<td>electronic design automation</td>
</tr>
<tr>
<td>EHW</td>
<td>evolvable hardware</td>
</tr>
<tr>
<td>FPAA</td>
<td>field programmable analog array</td>
</tr>
<tr>
<td>FPGA</td>
<td>field programmable gate array</td>
</tr>
<tr>
<td>FPTA</td>
<td>field programmable transistor array</td>
</tr>
<tr>
<td>FSM</td>
<td>finite state machine</td>
</tr>
<tr>
<td>HDL</td>
<td>hardware description language</td>
</tr>
<tr>
<td>OTP</td>
<td>one time programmable</td>
</tr>
<tr>
<td>PLD</td>
<td>programmable logic device</td>
</tr>
</tbody>
</table>
INTRODUCTION

Aims: We aim here to give a brief overview of evolvable hardware (EHW), to explain what it is and how it is used. Many more details will be given in subsequent chapters, but here we aim to give the reader a taste of things to come. We will also give some basic characteristics and examples of good and not so good features of evolvable systems.

1.1 CHARACTERISTICS OF EVOLVABLE CIRCUITS AND SYSTEMS

In the last ten years the complexity of electronic and computer systems has increased dramatically. As more power is required more complex systems have been created to fulfill these demands. This increase in complexity enables us to produce, what to most of us are, impressive engineering systems: aircraft, cars, mobile phones, the internet, “intelligent” homes, to name just a few. However, there are potentially negative aspects to this increased complexity. Most obviously how to design, manage and reason about such complex systems. It is interesting to make the point now that biological systems are many orders of magnitude more complex than anything we can currently produce. In addition, unfortunately as complexity increases so do the faults and errors seen in these systems. For non-maintainable systems, those that are not available to repair due to the expense, such as satellites, deep-sea probes or long-range spacecraft, faults and errors can damage the system making the device useless.
Let’s consider in a little more detail here the ideas behind faults in systems; we will talk much more about this in later chapters. Faults in high integrity computer system can be detected and fixed through maintenance. Fault tolerant techniques are provided to allow the system to continue working even when one or more of its redundant parts have failed. In non-maintainable systems this is not an option; once faults have occurred and been masked through fault tolerance techniques, the specific element of the system is no longer used and it cannot be repaired to be used again. Once all of the redundant elements have been destroyed through different faults the system will fail.

One extreme example of this is the one hundred year mission to the nearest star. Over the course of one hundred years an electronic system will have degraded to the point where very little of the original hardware will still be operational. In this case, if the system fails, the whole mission will fail. There is no chance of sending out maintenance engineers to fix the craft.

Consider Figure 1.1, a rather stylized model of the evolutionary cycle that occurs for all life. Details of this cycle as applied to engineering systems, and particularly evolvable hardware are given in later chapters. But for now let’s consider this cycle. We will start with parents (we have to start somewhere and we don’t wish to get into the chicken and egg argument!). These parents will produce children, who will exist in an environment to which they are better or worse suited than others. Selection will occur to determine if these children will (or will not) become parents for the next generation.

The important points here are that we have a population of parents, who produce a new population of children that form the next generation. Some form
of selection will take place and success in this selection process will enable members of the population to participate in the creation of the next generation (non-selection means obscurity for eternity!)

How might we now move this towards our evolvable hardware system? Consider Figure 1.2. Our population is now a set of electronic systems attempting to solve the same problem. If we assume that we can transform the problem solution into a measurable number—called “fitness” in Figure 1.2—we can use this number to help us with the selection stage of our cycle. Again, those selected will be involved in forming the next generation of possible solutions to the problem. DNA from Figure 1.1 has now been transformed into a binary string (it does not have to be a binary string, but at this early stage in our discussions this is a good example). This binary string is used to “configure” our system. As illustrated in Figure 1.2, this cycle continues, maybe forever, or more typically until the fitness achieves its maximum value or until we get bored!

Figure 1.3 illustrates this process again. This time its representation is more closely aligned to electronic hardware. It also illustrates some of the basic ideas used in the field to introduce some randomness into the whole process: In this case crossover and mutation (again much more will be said about these in Chapter 2).

These are the basics of evolvable hardware. Of course, there are many details we have not mentioned here, but these will be covered in later chapters. This quick overview is intended to just cover the underlying ideas behind the subject.

What are the characteristics that one might expect to find in an evolvable hardware system? It is difficult to generalize because the answer depends on a