GENETIC AND EVOLUTIONARY COMPUTATION

Series Editors: David E. Goldberg and John R. Koza

Edited by

Tetsuya Higuchi

Yong Liu

Xin Yao

Evolvable Hardware

Springer

Tetsuya Higuchi, Yong Liu, Xin Yao (Eds.)

Evolvable Hardware

Genetic and Evolutionary Computation Series

Selected titles from this series:

David E. Goldberg
The Design of Innovation: Lessons from and for Competent Genetic Algorithms, 2002
ISBN 1-4020-7098-5

John R. Koza, Martin A. Keane, Matthew J. Streeter, William Mydlowec, Jessen Yu, Guido Lanza
Genetic Programming IV: Routine Human-Computer Machine Intelligence
ISBN: 1-4020-7446-8 (hardcover), 2003; ISBN: 0-387-25067-0 (softcover), 2005

Carlos A. Coello Coello, David A. Van Veldhuizen, Gary B. Lamont
Evolutionary Algorithms for Solving Multi-Objective Problems, 2002
ISBN: 0-306-46762-3

Lee Spector
Automatic Quantum Computer Programming: A Genetic Programming Approach, 2004
ISBN: 1-4020-7894-3

William B. Langdon
Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming! 1998
ISBN: 0-7923-8135-1

For a complete listing of books in this series, go to http://www.springer.com

Tetsuya Higuchi
Yong Liu
Xin Yao
(Eds.)

Evolvable Hardware

Springer

Tetsuya Higuchi
National Institute of Advanced Industrial Science and Technology, Japan

Yong Liu
The University of Aizu, Japan

Xin Yao
The University of Birmingham, United Kingdom

Library of Congress Control Number: 2006920799

ISBN-10: 0-387-24386-0 e-ISBN-10: 0-387-31238-2
ISBN-13: 978-0387-24386-3 e-ISBN-13: 978-0387-31238-5

Printed in the United States of America

9 8 7 6 5 4 3 2 1

springer.com

PREFACE

Evolvable hardware refers to hardware that can learn and adapt autonomously in a dynamic environment. It is often an integration of evolutionary computation and programmable hardware devices. The objective of evolvable hardware is the autonomous reconfiguration of hardware structure in order to improve performance over time. The capacity for autonomous reconfiguration with evolvable hardware makes it fundamentally different from conventional hardware, where it is almost impossible to change the hardware's function and architecture once it is manufactured. While programmable hardware devices, such as a PLD (Programmable Logic Device) and a FPGA (Field Programmable Gate Array), allow for some functional changes after being installed on a print circuit board, such changes cannot be executed without the intervention of human designers (i.e., the change is not autonomous). With the use of evolutionary computation, however, evolvable hardware has the capability of autonomously changing its hardware architectures and functions.

The origins of evolvable hardware can be traced back to Mange's work and Higuchi's work, which were conducted independently around 1992. While Mange's work has led to bio-inspired machines that aim at self-reproduction or self-repair of the original hardware structure rather than evolving new structures, Higuchi's work has led to evolvable hardware research utilizing evolutionary algorithms for the autonomous reconfiguration of hardware structures. This book focuses primarily on the second line of research.

Departing from the initial interest of the artificial intelligence and artificial life communities in evolvable hardware to autonomously evolve hardware structures, recent evolvable hardware research has come to address some important topics for semiconductor engineering and mechanical engineering, such as:

- post-fabrication LSI adjustment,
- LSI tolerance to temperature change,
- self-testing/self-repairing LSI,
- human-competitive analog design,
- MEMS fine-tuning,
- adaptive optical control with micron-order precision, and
- evolvable antenna for space missions.

Research activities relating to evolvable hardware are mainly reported at two international conferences. The first one is the series of International Conferences on Evolvable Systems (ICES). The second is the NASA-DoD Evolvable Hardware Conferences that have been held every year in the USA since 1999. While it is rather difficult to neatly classify this body of research activities, this book adopts the following three categories:

1. Digital hardware evolution
   (1-1) Digital evolvable hardware based on genetic algorithms
   (1-2) Bio-inspired machines
2. Analog hardware evolution
   (2-1) Analog evolvable hardware based on genetic algorithms
   (2-2) Analog circuit design with evolutionary computation
3. Mechanical hardware evolution

This book brings together 11 examples of cutting-edge research and applications under these categories, placing particular emphasis on their practical usefulness.

Tetsuya Higuchi
Yong Liu
Xin Yao

CONTENTS

Chapter 1

INTRODUCTION TO EVOLVABLE HARDWARE

Tetsuya Higuchi[1], Yong Liu[2], Masaya Iwata[1], and Xin Yao[3]
[1]Advanced Semiconductor Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Email: t-higuchi@aist.go.jp; [2]The University of Aizu, Fukushima, Japan; [3]The University of Birmingham, UK.

Abstract: This chapter provides an introduction to evolvable hardware. First, the basic idea of evolvable hardware is outlined. Because evolvable hardware involves the integration of programmable logic device and evolutionary computation, these are both explained briefly. Then, an overview of current research on evolvable hardware is presented. Finally, the chapter discusses some directions for future research.

Key words: genetic algorithm, genetic programming, FPGA, programmable logic device.

1. INTRODUCTION

This book describes a new hardware paradigm called "Evolvable Hardware" and its real-world applications.

Evolvable hardware is the integration of evolutional computation and programmable hardware devices. The objective of evolvable hardware is the "autonomous" reconfiguration of hardware structure in order to improve performance. The capacity for autonomous reconfiguration with evolvable hardware makes it fundamentally different from conventional hardware, where it is almost impossible to change the hardware's function once it is manufactured. While programmable hardware devices, such as a PLD (Programmable Logic Device) and a FPGA (Field Programmable Gate Array), allow for some functional changes after being installed on a print circuit board, such changes cannot be executed without the intervention of human designers (i.e., the change is not autonomous). With the use of evolutional computation, however, evolvable hardware has the capability to autonomously change its hardware functions.

The origins of evolvable hardware can be traced back to two papers by Daniel Mange (Mange, 1993a, 1993b) and a paper by Tetsuya Higuchi (Higuchi, 1992), which were written independently around 1992. While the Mange papers led to bio-inspired machines that aim at self-reproduction or self-repair of the original hardware structure rather than evolving new structures, the Higuchi paper led to evolvable hardware research utilizing genetic algorithms for the autonomous reconfiguration of hardware structure. This book focuses primarily on the second line of research.

Departing from an initial interest in artificial intelligence and artificial life for hardware to autonomously evolve its own structure, recent evolvable hardware research has come to address important topics for semiconductor engineering and mechanical engineering, such as:

- post-fabrication LSI adjustment
- LSI tolerance to temperature change
- self-testing/self-repairing LSI
- human-competitive analog design
- MEMS (Micro Electro Mechanical System) fine-tuning
- adaptive optical control with micron-order precision
- evolvable antenna for space missions

Research activities relating to evolvable hardware are mainly reported at two international conferences. The first one is the series of International Conferences on Evolvable Systems (ICES) held in 1996 (Japan), 1998 (Switzerland), 2000 (UK), 2001 (Japan), 2003 (Norway), 2005 (Spain), 2007 (to be in China) and 2008 (to be in Czecho-Slovakia). The second is the NASA-DOD Evolvable Hardware Conferences that have been held every year in the USA since 1999. While it is rather difficult to neatly classify this body of research activities, this book adopts the following three categories:

1. (Category 1) Digital hardware evolution
   (1-1) Digital evolvable hardware based on genetic algorithms
   (1-2) Bio-inspired machines
2. (Category 2) Analog hardware evolution
   (2-1) Analog evolvable hardware based on genetic algorithms
   (2-2) Analog circuit design with evolutionary computation
3. (Category 3) Mechanical hardware evolution

This book brings together nine examples of cutting-edge research and applications under these categories, placing particular emphasis on their practical usefulness.

Category 1-1 of digital evolvable hardware based on genetic algorithms includes a data compression method for print images proposed by H. Sakanashi that has been accepted by the International Standards Organization (ISO) (Chapter 2), the first evolvable hardware chip developed for myoelectric hand control by I. Kajitani (Chapter 3), and E. Takahashi's work on clock-timing adjusting (Chapter 4). According to a report, a similar method to Takahashi's was used in the clock-timing adjustment for engineering samples of the Intel Pentium 4.

Under Category 1-2, the pioneering work on bio-inspired machines is discussed by D. Mange (Chapter 5), while Chapter 6 by A. Tyrrell describes an LSI implementation of a bio-inspired machine.

As representative work on analog evolvable hardware based on genetic algorithms (Category 2-1), M. Murakawa describes the analog intermediate filter LSI used in commercial cellular phones (Chapter 7) and A. Stoica outlines NASA's FPTA (Field Programmable Transistor Array) (Chapter 8). Employing a FPTA, Chapter 9 by R. Zebulum demonstrates how adaptation to extreme low temperatures is possible with the genetic algorithm approach.

In Category 2-2, Chapter 10 presents J. Koza's work which highlights the great potential of evolvable hardware for innovative analog circuit design.

Two pioneering works in mechanical hardware evolution are described within Category 3: H. Nosato's work on evolvable femtosecond laser systems (Chapter 11) and D. Keymulen's evolvable gyro system (Chapter 12). These works suggest that optimization by genetic algorithms can open up a new paradigm of mechanical evolutions that has a potentially wide application for MEMS.

Before each of these eleven works are described in detail in the following chapters, the remainder of this chapter briefly discusses three aspects of evolvable hardware research: (1) the basic idea of evolvable hardware, (2) an overview of evolvable hardware research being conducted around the world, and (3) directions for future research on evolvable hardware.

2. BASIC IDEA OF EVOLVABLE HARDWARE

As previously noted, two concepts are combined in realizing evolvable hardware: programmable hardware devices such as FPGA, and evolutionary computation such as genetic algorithms and genetic programming. In this section, these two concepts are briefly explained.

2.1 Programmable Hardware Devices

The advantage of programmable devices is that the hardware architecture can be changed any number of times by loading software string bits that determine the specific hardware structure. Because this advantage leads to fast prototyping and reductions in repair costs, the market for FPGAs as programmable hardware devices has grown rapidly over the last two decades. Typically, a FPGA has a structure, as depicted in Figure 1-1, consisting of functional blocks (FBs) and interconnections among the FBs. The function of an FB can be specified by setting some bit string into the block. The interconnections to FBs are also determined by setting the bit strings. These bit strings, shown as dots in the figure, are called configuration bits. Thus, the specific hardware structure of a programmable hardware device is actually determined by downloading the configuration bits.

Figure 1-2 is a more specific example of hardware reconfiguration with a programmable logic array (PLA) which is an early PLD product of the 1980's. A PLA consists of a Boolean AND part (left side of Figure 1-2) and a Boolean OR part (right side of Figure 1-2). In Figure 1-2, there are four inputs (X0 to X4) and two outputs (Y0 and Y1). By setting the switch dots, various hardware functions can be realized.

For example, on the first row of the switch setting for the AND part, only one switch dot is *on*, which means that negation of input X0 is allowed to enter the Boolean AND part of the PLA. So, the output from the first row of the AND part becomes $\overline{X0}$. On the second row, four switch dots are *on*, which means that X0 and X3, as well as the negations of X1 and X2, can enter the second row of the AND part. So, the output from the second row becomes $X0\overline{X1}\,\overline{X2}X3$. Similarly, the outputs for the third and fourth rows would be $X0\overline{X1}X2\overline{X3}$ and $X0\overline{X3}$, respectively.

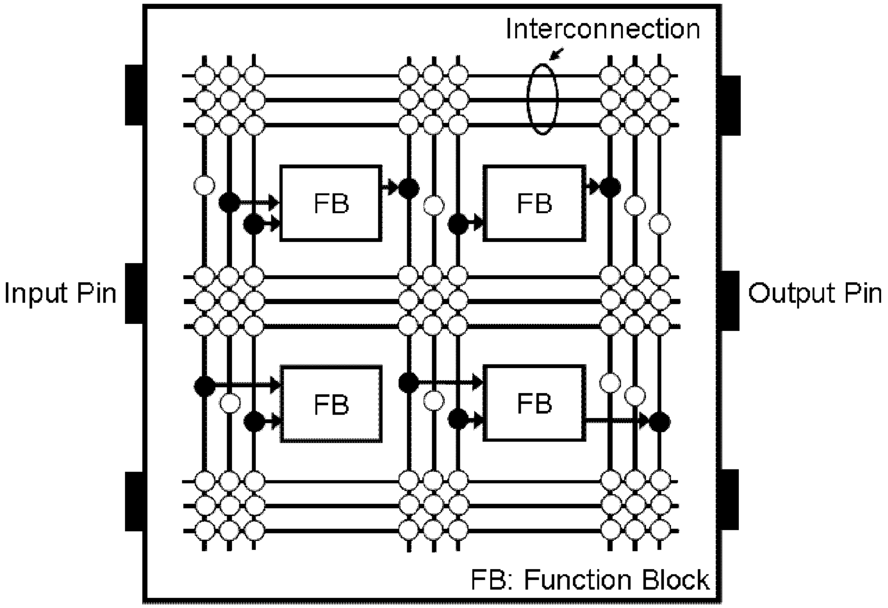


Figure 1-1. FPGA organization

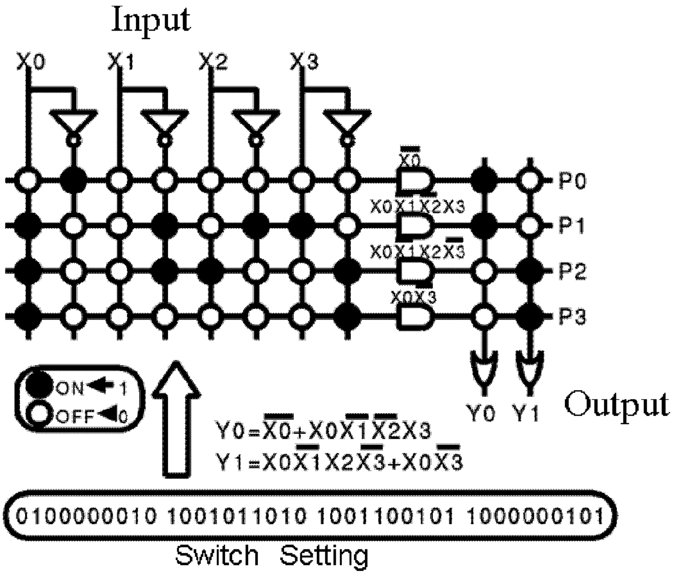

Input
X0
X1
X2
X3
$\overline{X0}$
$X0\overline{X1}\overline{X2}X3$
$X0\overline{X1}X2\overline{X3}$
$X0\overline{X3}$
P0
P1
P2
P3
ON 1
OFF 0
$Y0=\overline{X0}+X0\overline{X1}\overline{X2}X3$
$Y1=X0\overline{X1}X2\overline{X3}+X0\overline{X3}$
Y0
Y1
Output
0100000010 1001011010 1001100101 1000000101
Switch Setting

Figure 1-2. Reconfiguration of PLA

The outputs from the AND part go to the OR part. The switch settings for the OR part mean that for the second column from the right the outputs from the first and second rows of the AND part are allowed to enter. So, the output Y0 from this second column of the OR part becomes $\overline{X0}+X0\overline{X1}\overline{X2}X3$. Similarly, for the other column, the output Y1 becomes $X0\overline{X1}X2\overline{X3}+X0\overline{X3}$ with the switch settings in Figure 1-2. The switch setting is shown at the bottom of Figure 1-2. By downloading this bit string, Y0 and Y1 would be defined as explained above. If a different bit string is downloaded into the PLA, a different hardware function for the PLA would be easily realized.

2.2 Evolutionary Computation

This section briefly outlines the principles of evolutionary computation. After describing genetic algorithm in the first subsection in terms of the initial preparation stage and the search process, involving crossover, mutation, evaluation and selection procedures, the second subsection introduces genetic programming as the application of GAs to the evolution of computer programs.

2.2.1 Genetic Algorithm

Within artificial intelligence, one frequently encounters search problems where the solution cannot be identified within a finite period of time due to the combinatorial explosion of the search space. Genetic Algorithm (GA) is a general search technique that can be applied to such problems because it does not require user specific or a priori knowledge concerning the problem to be solved. First proposed by J. Holland, GA is loosely based on the notion of population genetics. Accordingly, some GA terms (e.g., chromosome, mutation) are derived from population genetics, although they do not correspond strictly to their senses within population genetics.

A search executed by a GA involves two stages: the preparation and the search stages:

(Preparation)

1. First, a set of candidate search solutions, called a population, must be prepared, because GA is a parallel search method that starts searching from this initial candidate population.
2. Each candidate is referred to as a chromosome, which is typically a binary bit string. The initial values of candidates are determined randomly.
3. An evaluation function, known as the fitness function, must also be designed for each problem. The fitness function is used to evaluate each chromosome in terms of being a good solution to the problem.

(Search)

The basic idea behind GA is to obtain a new chromosome with the optimal fitness value that can be regarded as a search solution. Until this chromosome is obtained, genetic operations, such as crossover and mutation, are repeatedly executed on the population. As shown in Figure 1-3, GA search involves repeatedly executing a cycle, referred to as a "generation," consisting of a crossover operation, a mutation operation, an evaluation, and selection.

1. Crossover operation: One of the methods of generating new chromosomes is the crossover operation. The operation randomly chooses two chromosomes as parents and exchanges parts of them, as shown in Figure 1-4.
2. Mutation operation: A particular bit is stochastically chosen and its value is flipped to generate a new chromosome.
3. Evaluation: In order to identify the candidates that may survive to the next generation, each chromosome is evaluated according to the fitness function and assigned a fitness value.

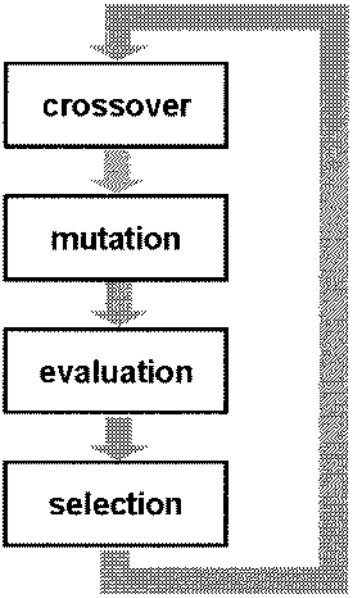


Figure 1-3. GA cycle

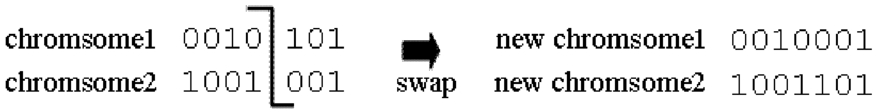


Figure 1-4. Crossover operation

4. Selection: By executing the crossover and mutation operations, new chromosomes are generated that may have higher fitness values. However, because the number of chromosomes in a population is fixed during a search, it is necessary to select from among the old and new chromosomes. While there are various policies concerning selection, the typical operation is the roulette wheel selection. With the roulette selection, chromosomes with higher fitness values are likely to survive as members of the next generation. Once selection is completed, a new population is ready for the next generation.

Let us consider the following search problem as an example of a GA search:

$$find\ x_{\max}\ where\ f(x_{\max}) = \underset{x}{Max}\ f(x). \tag{1}$$

First, the fitness function for this problem is set to be $f(x)$. Then, as shown in Figure 1-5(a), an example population of five 10-bit chromosomes is generated at random. Each chromosome can be decoded to the variable x in (1). Pairs of chromosomes are then randomly selected. These are mated and undergo genetic operations such as crossover and mutation, to yield better chromosomes in subsequent generations. After several generations of the GA search, chromosomes with lower fitness values tend to be eliminated from the population and relatively high-fitness chromosomes remain, as shown in Figure 1-5(c) (in contrast to Figure 1-5(a)).

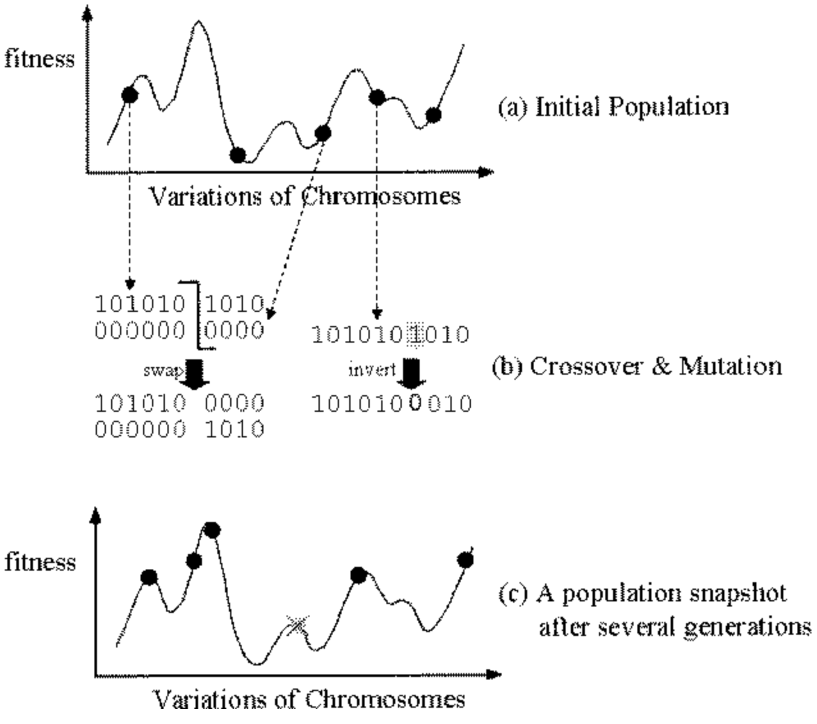


Figure 1-5. An example of GA search

2.2.2 Genetic Programming

A special subbranch of GAs is genetic programming proposed by J. Koza as the application of GAs to the evolution of computer programs. Trees (particularly, Lisp expression trees) are often used to represent individuals. Both crossover and mutation are used in genetic programming.

Genetic programming can be used to automatically create both the topology and sizing of an electrical circuit by establishing program trees for electrical circuits and defining a fitness that measures how well the behavior and characters of a given candidate circuit satisfy the design requirements. A developmental process can be employed to transform a program tree into a fully developed electrical circuit by executing component creation, topology modification, and development control functions, as well as arithmetic performance and automatic defining functions. One advantage of this kind of developmental process is in preserving the electrical validity of the circuit. Another advantage is the ability to preserve locality. Most of the functions involved in the developmental process operate on a small local area of the circuit, so subtrees within a program tree tend to operate locally. The crossover operation transplants subtrees and preserves their locality. The mutation and architecture-altering operations also both preserve locality because they only affect subtrees.

The evolutionary design offers several advantages over conventional design carried out by human designers. First, evolutionary design can explore a wider range of design alternatives in the hope of evolving novel designs. Second, evolutionary design does not assume a priori knowledge about any particular design domain, which is advantageous, where a priori knowledge is scarce or too costly to obtain. Finally, evolutionary design can cope with all kinds of constraints to satisfy the design requirements.

2.3 Integration of Genetic Algorithm and Programmable Hardware Devices

The key concept of evolvable hardware is to regard the configuration bits of programmable hardware devices as the chromosomes of GAs. By designing a fitness function to achieve a desired hardware function, the GA becomes a means of autonomous hardware reconfiguration. Figure 1-6 explains this idea. Configuration bits "evolved" by the GA are repeatedly downloaded into the programmable hardware devices until the evolved hardware performance is satisfactory in terms of fitness function values. A GA for evolvable hardware is executed either outside or inside the evolvable hardware, depending on its purpose. For example, if the speed of hardware reconfiguration is an important factor, then the GA should be inside the evolvable hardware.

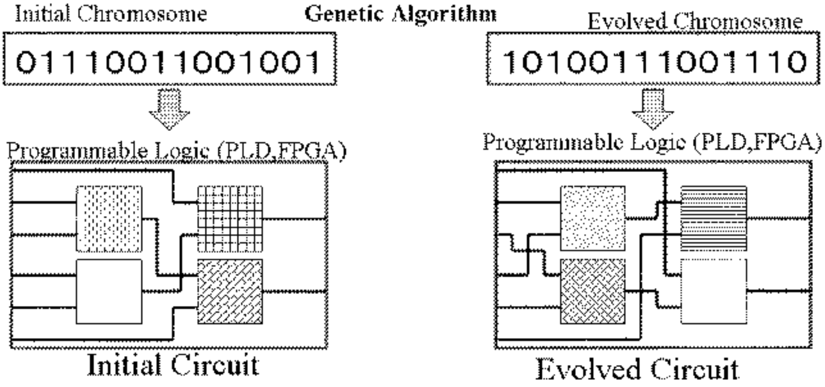


Figure 1-6. Basic idea of evolvable hardware

3. OVERVIEW OF EVOLVABLE HARDWARE RESEARCH AROUND THE WORLD

This section presents a brief overview of evolvable hardware research around the world. In this selective survey, we highlight research reported at the International Conference on Evolvable Systems (ICES). First held in 1996, ICES was the first conference to focus mainly on evolvable hardware, including all aspects of evolvable systems. As already mentioned, we are classifying evolvable hardware research under three categories: digital hardware evolution, analog hardware evolution, and mechanical hardware evolution. Some works are singled out in each category. For readers with further interest, it is recommended to read X. Yao's overview (Yao, 1999).

3.1 Digital Hardware Evolution

Digital hardware evolution is the most active category of evolvable hardware research. In this book, this category is further classified into two types: evolvable hardware based on genetic algorithms and bio-inspired machines.

3.1.1 Digital Evolvable Hardware Based on Genetic Algorithms

This category covers work in designing digital circuits using GA. Many of the works under this category have been presented at ICES.

Some new methods for the evolutionary design of digital circuits have been proposed. The first evolvable hardware chip developed for myo-electric hand control is described in Chapter 3. J. Torresen (Torresen, 2001) has proposed a new evolvable hardware architecture for pattern classification including incremental evolution. He showed that the method is applicable to prosthetic hand controllers. M. Garvie, et al. (Garvie, 2003) have shown the evolution of digital circuits capable of performing built-in self-test behavior in simulations for a one-bit adder and a two-bit multiplier. Their results suggest that evolved designs can perform a better diagnosis using less resource than hand-designed equivalents. J. Korenek, et al. (Korenek, 2005) have developed and evaluated a specialized architecture to evolve relatively large sorting networks in an ordinary FPGA.

Some new and practical applications have also been studied. For instance, a lossless data compression method for bi-level images using evolvable hardware, and a clock-timing adjusting technique are described in Chapter 2 and 4, respectively. T. Martinek, et al. (Martinek, 2005) have proposed an evolvable image filter that was completely implemented in an FPGA. The system is able to evolve an image filter in a few seconds if corrupted and original images are supplied by the user. S. L. Smith, et al. (Smith, 2003) have presented an application of GA to evolve new spatial masks for nonlinear image processing operations, which are ultimately to be implemented on evolvable hardware.

Digital evolvable hardware has also been applied to evolving robot controllers. J. G. Kim, et al. (Kim, 2001) have shown that their proposed GA guarantees satisfactory smooth and stable walking behavior in an experiment involving a real biped robot. M. M. Islam, et al. (Islam, 2001) have applied an incremental approach—a two-stage evolutionary system—to develop the control system of an autonomous robot for a complex task. Harding, et al. (Harding, 2005) have discussed the stability and reconfigurability of a real-time robot controller evolved in liquid crystal. They envisage these issues will be important when programming or evolving in other physical systems.

3.1.2 Bio-inspired Machines

Under this category, many ideas for applying the biological process to evolutionary systems have been studied. This topic has been discussed since the very beginning of evolvable hardware research.

Embryonic electronics (embryonics) (Mange, 1998) is a research project that draws inspiration from the biological process of ontogeny in seeking to implement novel digital computing machines with better levels of fault tolerance. H. F. Restrepo, et al. (Restrepo, 2001) have proposed a multicellular universal Turing machine implementation that is endowed with self-replication and self-repair capabilities. L. Prodan, et al. (Prodan, 2001) have proposed artificial cells driven by artificial DNA to implement their embryonic machines. A. Stauffer, et al. (Stauffer, 2001) have fabricated a self-repairing and self-healing electronic watch: the BioWatch, with a system based on an array of small processors. Their recent work is described in Chapter 5.

As described in Chapter 6, a new research project has also been proposed on "reconfigurable POEtic tissue." The project goal is to develop a hardware platform capable of implementing with digital hardware systems that are inspired by all three major bio-inspiration axes (phylogenesis, ontogenesis, and epigenesis). While W. Barker, et al. (Barker, 2005) have presented results of hardware fault-tolerance within the POEtic system, J. M. Moreno, et al. (Moreno, 2005) have conceived an architecture for POEtic devices, internally organizing the main constituent elements.

Some other works in this category have also been reported. For example, J. Greensted, et al. (Greensted, 2003) have proposed a software model for multiprocessor system design that uses an interprocessor communication system similar to the endocrine system. The system is able to perform arbitrary dataflow processing. P. C. Haddow, et al. (Haddow, 2001) presented the first case study using the mathematical formalism called L-systems and have applied their principles to the development of digital circuits. D. W. Bradley, et al. (Bradley, 2001) have analyzed the body's approach to fault tolerance based on immune system and have demonstrated how such techniques can be applied to hardware fault tolerance.

3.2 Analog Hardware Evolution

Analog hardware evolution is a relatively newer category compared to digital hardware evolution. The two subcategories involved here are analog evolvable hardware based on genetic algorithms and analog circuit design with evolutionary computation.

3.2.1 Analog Evolvable Hardware Based on Genetic Algorithms

This category covers work on designing analog circuits using GA. While there have been fewer reports on analog hardware compared to digital hardware, there has been some interesting work in this category.

An FPTA (Field Programmable Transistor Array) is an implementation of an evolution-oriented reconfigurable architecture for evolving analog circuits. A number of interesting reported works involve FPTAs. For instance, R. S. Zebulum, et al. (Zebulum, 2003) have presented a hardware evolution of analog circuits to perform signal separation tasks using their system called Stand-Alone Board-Level Evolvable System (SABLES). SABLES integrates an FPTA-2 chip and a digital signal processor to implement the evolutionary platform. Results demonstrate that SABLES is sufficiently flexible to adapt to different input signals without human intervention. L. Sekanina, et al.

(Sekanina, 2005) have reported that simple one-bit and two-bit controllable oscillators were intrinsically evolved using only four cells of FPTA-2. These oscillators can produce different oscillations for different settings of control signals. Trefzer, et al. (Trefzer, 2005) have tackled the problem of synthesizing transferable and reusable operational amplifiers on an FPTA. A multiobjective evolutionary algorithm has been developed, in order to be able to include various specifications of an operational amplifier into the process of circuit synthesis. Some recent works on FPTAs and SABLES for extreme environments are described in Chapters 8 and 9.

Some effective applications for analog hardware have also been studied. An analog intermediate filter LSI used in commercial cellular phones is described in Chapter 7. Y. Kasai, et al. (Kasai, 2005) have proposed adaptive waveform control in a data transceiver and demonstrated an adaptive transceiver LSI with the waveform controller. Utilizing a GA, the method has achieved a transmission speed that is four times faster than the current standards for IEEE1394. J. D. Lohn, et al. (Lohn, 1998) have proposed a method of evolving analog electronic circuits using a linear representation and a simple unfolding technique. Using a parallel GA, they have presented initial results of applying their system to two analog filter design problems.

3.2.2 Analog Circuit Design with Evolutionary Computation

Within this category, evolutionary computation is used to evolve analog circuits. J. R. Koza, et al. (Koza, 1996) have proposed utilizing genetic programming for evolving analog circuits. Genetic programming is a systematic method to get computers to automatically solve problems. Genetic programming is an extension of the GA concept into the arena of computer programming. J. R. Koza, et al. have successfully evolved a design for a two-band crossover filter using genetic programming. They have also succeeded in evolving an op amp with good frequency generalization (Bennett, 1996). Their recent works are described in Chapter 10.

3.3 Mechanical Hardware Evolution

This category includes works using evolutionary algorithms to adjust machine parts. Although this is the newest category, some interesting applications have already been reported. While an evolvable femtosecond laser system is described in Chapter 11, a tuning method for MEMS gyroscopes based on evolutionary computation is explained in Chapter 12. J. D. Lohn, et al. (Lohn, 2001) have proposed a GA-based automated antenna optimization system that uses a fixed Yagi-Uda topology and a byte-encoded antenna representation. They have also succeeded in evolving new antenna designs for NASA's Space Technology 5 mission (Lohn, 2005).

4. PERSPECTIVES FOR EVOLVABLE HARDWARE RESEARCH

There are two promising application areas for future research on evolvable hardware. One is semiconductor engineering and the other is mechanical engineering, including MEMS. This section briefly reviews important works related to these areas.

4.1 Research Direction Towards Semiconductor Engineering

Although early research interests within evolvable hardware were mainly centered on artificial intelligence and artificial life, more recent evolvable hardware research is addressing important topics for semiconductor engineering. These include post-fabrication LSI adjustment, tolerance for temperature changes, waveform control for high-speed data transmission, and human-competitive analog design, as well as self-test/self-repair LSI, which is discussed briefly below.

In current LSI manufacturing, degradation in the operational yield rate is a very serious problem because a poor LSI yield rate results in increased LSI costs. One of the main reasons for a poor yield rate is the variations in the LSI manufacturing processes. For example, some transistors in the LSI may not achieve the required performance (e.g., threshold voltage) due to inaccuracies in the manufacturing process. Due to such variations, it is not possible to guarantee the final performance of LSI products simply by making more elaborate LSI designs. However, one practical solution to achieving acceptable operational yield rates is post-fabrication LSI adjustment with GA (see Chapter 4). Adjustment circuitries are inserted in advance wherever yield rates might be degraded. Then, the parameters of the adjustment circuitries are determined by the GA after LSI fabrication. If the GA adjustment time and the adjustment circuitry space are sufficiently small, this approach represents a very important remedy for improving yield rates.

The capability of evolvable hardware to adapt to a changing environment is also very important for semiconductor engineering. For example, performance degradation due to temperature fluctuations can be controlled for by the GA approach. Stoica's work (NASA) and Zebulum's work (NASA) in this direction (see Chapter 8 and Chapter 9, respectively), which is based on Stoica's FPTA (see Chapter 8), is very important in this respect. While this book does not touch on the issue directly, performance for high-speed data transmission (over Giga-Hertz) is heavily influenced by noise through cable transmission. With the GA approach, however, circuitry to control the waveforms can be adjusted in order to satisfy the requirement for actual cable installations.

While digital hardware design has made rapid progress due to advances in EDA software tools, analog hardware design is still highly reliant on the experience and maturity of analog hardware designers. Koza's work on analog hardware design (see Chapter 10) suggests that genetic programming can generate human-competitive analog design.

Recent LSI testing has shifted away from using LSI testers to BIST (Built-In Self Test) and BISR (Built-In Self-Repair) for the following reasons. Recent LSIs have high clock frequency over Giga-Hertz and complicated functions. While they also require a number of test pins, there are severe restrictions for these test pins, which creates problems for developing customized LSI testers. Instead of testing an LSI with dedicated LSI testers, one feasible solution that has emerged is to incorporate BIST/BISR functions with the LSI. Research on bio-inspired machines such as Stauffer's work (see Chapter 5) and Tyrrell's work (see Chapter 6) are pioneering works in this direction.

4.2 Research Direction Towards Mechanical Engineering

While not covered in this book, NASA's evolvable antenna is a very important work which demonstrates the potentiality of evolvable hardware. The shape of NASA's antenna is beyond the human imagination, while maintaining its required performance.

The evolvable femtosecond laser, described in Chapter 11, is another example illustrating how the evolvable hardware approach is effective for mechanical evolution. The approach allows for very precise setting of physical laser component devices. In general, high performance mechanical systems tend to be vulnerable to environmental changes such as vibrations and temperature changes. The precise physical setting of the components in terms of position and angle is key to attaining high performance, but there may be limitations with using human engineers when such systems are used in extreme environments (e.g., extreme high/low temperatures and radiation). Autonomous positioning with the GA approach can be very effective in such circumstances.

Moreover, D. Keymulen's work (NASA) is the first paper showing that the evolvable hardware approach is effective for MEMS tuning. As with manufacturing LSIs with fine patterns (i.e. submicron), there are also unavoidable fabrication inaccuracies involved with MEMS manufacturing. Following his work (see Chapter 12), further explorations of other MEMS applications utilizing the GA approach are expected.

References

Barker, W. and A. M. Tyrrell. 2005. Hardware fault-tolerance within the POEtic system. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 25-36. Springer.

Bennett III, F. H., et al. 1996. Evolution of a 60 decibel op amp using genetic programming. In *Evolvable Systems: From Biology to Hardware, LNCS 1259*, 455-469. Springer.

Bradley, D. W. and A. M. Tyrrell. 2001. Multi-layered defence mechanisms: architecture, implementation and demonstration of a hardware immune system. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 140-150. Springer.

Garvie, M. and A. Thompson. 2003. Evolution of self-diagnosing hardware. In *Evolvable Systems: From Biology to Hardware, LNCS 2606*, 238-248. Springer.

Greensted, J. and A. M. Tyrrell. 2003. Fault tolerance via endocrinologic based communication for multiprocessor systems. In *Evolvable Systems: From Biology to Hardware, LNCS 2606*, 24-34. Springer.

Haddow, P. C., G. Tufte and P. van Remortel. 2001. Shrinking the genotype: L-systems for EHW? In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 128-139. Springer.

Harding, S. and J. F. Miller. 2005. Evolution in materio: investigating the stability of robot controllers evolved in liquid crystal. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 155-164. Springer.

Higuchi, T., et al. 1993. Evolvable hardware with genetic learning. In *Proc. of Simulated Adaptive behavior*, 417-424. MIT Press.

Islam, M. M., S. Terao and K. Murase. 2001. Effect of fitness for the evolution of autonomous robots in an open-environment. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 171-181. Springer.

Kasai, Y., et al. 2005. Adaptive waveform control in a data transceiver for multi-speed IEEE1394 and USB communication. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 198-204. Springer.

Kim, J. G., K.-G. Noh and K. Park. 2001. Human-like dynamic walking for a biped robot using genetic algorithm. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 159-170. Springer.

Korenek, J. and L. Sekanina. 2005. Intrinsic evolution of sorting networks: a novel complete hardware implementation for FPGAs. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 46-55. Springer.

Koza, J. R., et al. 1996. Reuse, parameterized reuse, and hierarchical reuse of substructures in evolving electrical circuits using genetic programming. In *Evolvable Systems: From Biology to Hardware, LNCS 1259*, 312-326. Springer.

Lohn, J. D. and S. P. Colombano. 1998. Automated analog circuit synthesis using a linear representation. In. *Evolvable Systems: From Biology to Hardware, LNCS 1478*, 125-133. Springer.

Lohn, J. D., et al. 2001. Evolutionary optimization of Yagi-Uda antennas. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 236-243. Springer.

Lohn, J. D., G. S. Hornby and D. S. Linden. 2005. Evolution, re-evolution, and prototype of an X-band antenna for NASA's space technology 5 mission. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 205-214. Springer.

Mange, D. 1993. Wetware as a bridge between computer engineering and biology. In *Preliminary Proceedings, 2nd European Conference on Artificial Life*, 658-667.

Mange, D. 1993. Life in Silico. In *11th European Conference on Circuit Theory and Design (ECCTD '93)*, 145-149.

Mange, D., A. Stauffer and G. Tempesti. 1998. Embryonics: a macroscopic view of the cellular architecture. In *Evolvable Systems: From Biology to Hardware, LNCS 1478*, 174-184 Springer.

Martinek, T. and L. Sekanina. 2005. An evolvable image filter: experimental evaluation of a complete hardware implementation in FPGA. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 76-85. Springer.

Moreno, J. M., Y. Thoma and E. Sanchez. 2005. POEtic: a prototyping platform for bio-inspired hardware. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 177-187. Springer.

Prodan, L., et al. 2001. Embryonics: artificial cells driven by artificial DNA. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 100-111. Springer.

Restrepo, H. F. and D. Mange. 2001. An embryonics implementation of a self-replicating universal turing machine. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 74-87. Springer.

Sekanina, L. and R. S. Zebulum. 2005. Intrinsic evolution of controllable oscillators in FPTA-2. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 98-107. Springer.

Smith, S. L., D. P. Crouch and A. M. Tyrrell. 2003. Evolving image processing Operations for an evolvable hardware environment. In *Evolvable Systems: From Biology to Hardware, LNCS 2606*, 332-343. Springer.

Stauffer, A., et al. 2001. A self-repairing and self-healing electronic watch: the BioWatch. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 112-127. Springer.

Torresen, J. 2001. Two-step incremental evolution of a prosthetic hand controller based on digital logic gates. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 1-13. Springer.

Trefzer, M., et al. 2005. Operational amplifiers: an example for multi-objective optimization on an analog evolvable hardware platform. In *Evolvable Systems: From Biology to Hardware, LNCS 3637*, 86-97. Springer.

Yao, X. and T. Higuchi. 1999. "Promises and challenges of evolvable hardware", *IEEE Trans. Systems, Man, and Cybernetics, C* **29**(1), 87-97.

Zebulum, R. S., et al. 2003. Automatic evolution of signal separators using reconfigurable hardware. In *Evolvable Systems: From Biology to Hardware, LNCS 2606*, 286-295. Springer.

Chapter 2

EHW APPLIED TO IMAGE DATA COMPRESSION

Hidenori Sakanashi, Masaya Iwata, and Tetsuya Higuchi
MIRAI Project, Advanced Semiconductor Research Center (ASRC), National Institute of Advanced Industrial Science and Technology (AIST)
{h.sakanashi, m.iwata, t-higuchi}@aist.go.jp

Abstract: In this chapter, EHW is applied to the lossless image compression, and it is implemented in a chip. The current international standard for bi-level image coding, JBIG2-AMD2, is modified by the proposed method to achieve high compression ratios, where compression parameters are optimized by the enhanced genetic algorithm (GA). The results of computer simulations show a 171% improvement in compression ratios with the proposed method compared to JBIG2 without optimization. The experiment shows that when the method is implemented by hardware with an evolvable hardware chip, the processing speed is dramatically faster than execution with software. This chapter also describes activities concerning ISO standardization to adopt part of the technology used in this method to the JBIG2 standard.

Key words: data compression, JBIG2 (Joint Bi-level Image experts Group, 2), ISO/IEC standard, EHW chip.

1. INTRODUCTION

Since the emergence of Desk Top Publication (DTP) in the graphic (imaging, printing and publishing) industry, digital image data has been handled in many ways, such as with digital printers, on-demand printing/publishing (ODP), and so on. On the other hand, the large costs for storage and transfer of an enormous amount of huge images have become a serious problem. For example, the electrophotographic printer must have the large storage and the broad data-bus to process many high-resolution images quickly. In the case of the on-demand publishing/printing, the large costs involved in transmission and storage pose serious problems to their practical use because the size of the data becomes extremely large for commercial printed matter with high resolutions.

To overcome these problems, image data must be compressed as much as possible and must be restored to its original state very quickly. Unfortunately, traditional data compression methods cannot satisfy these requirements, because the image data used in the printing/publishing industry have distinctive characteristics.

In this chapter, Evolvable Hardware (EHW) is applied to a data compression system. The proposed method has the following 4 features; (1) adoption of JBIG2 (ISO/IEC Int. Stand. 14492, 2001), the latest international standard, as basis, (2) introduction of a simplified initialization procedure for effective analysis in the genetic algorithm (GA) (Holland, 1975), (3) simplification of the evaluation procedure, and (4) enhanced crossover operations. The results of computer simulations show that the proposed method has a compression ratio that is 171% better than that of JBIG2 without optimization.

We have developed an EHW chip for the proposed method. The result of an experiment using the chip has demonstrated that the compression ratio is higher than for conventional data compression chips, and, moreover, that the speed is dramatically faster than with software execution.

2. LOSSLESS COMPRESSION OF HIGH-RESOLUTION GRAPHIC ART IMAGES

2.1 Image Data for Graphic Arts

In general graphic arts technology, color images are basically transformed into four high-resolution bi-level images before going to press, because press machines can only represent two levels (inked or not-inked). These bi-level images correspond to four colors (cyan (C), magenta (M), yellow (Y) and black (K)). Differences in brightness are represented by varying the density and size of the ink dots, known as halftone dots, which are composed from bi-level pixels located on a fixed rectangular grid.

Thus, the graphic arts images have the following features:

- Sets of bi-level images,
- Very high resolution, and
- Large frequency with which the pixel values switch is high in both the horizontal and vertical directions of a raster scan.

Traditionally, because the image data must be compressed in lossless fashion (reversibly) to avoid distortion or degradation in press quality, MH (Modified Haffman), MR (Modified Read) (CCITT Recommendation T.4,

1998) and MMR (Modified Modified Read) (CCITT Recommendation T.6, 1998) methods, which are well-known international standards for facsimile, have been used for encoding. These methods are based on run-length coding. MH uses a one-dimensional model, while a two-dimensional model is adopted in MR and MMR (the principle in MMR is the same as that in MR, but some error correction mechanisms are eliminated to achieve higher compression efficiency). These methods provide fairly good compression for line-art or text images. However, these methods are unsuitable for data where the switching frequency for pixel values is high, like halftone images, because they code the positions where pixel values are switched for each line.

In contrast, JBIG, a template-based arithmetic coding method (Sayood, 2000), was a general-purpose compression method for bi-level images, and JBIG2 (ISO/IEC Int. Stand. 14492, 2001) was standardized as its successor in 2000. JBIG2 was designed to upgrade the lossless JBIG encoding method and to add a lossy compression mode based on pattern matching. As lossy encoding is not relevant to the compression of graphic art images, this chapter focuses only on the lossless encoding mode in JBIG2.

2.2 Lossless Image Compression with JBIG2

The template-based arithmetic coding method is based on the hypothesis that the probability of a given "pixel to be coded" having a specific value depends on the values of a limited number of preceding pixels. In JBIG2, the MQ-Coder is adopted as an arithmetic coder; we shall limit our discussion here to this template-based coding method because a detailed explanation of the MQ-Coder would be beyond the scope of this chapter (the principle of arithmetic encoding and the procedure for the MQ-Coder are detailed in (Sayood, 2000) and (ISO/IEC Int. Stand. 14492, 2001), respectively).

In JBIG2 and its enhanced version called JBIG2-AMD2 (ISO/IEC Int. Stand. 14492/Amd.2, 2003), 16 pixels preceding the pixel to be coded are observed in calculating the probability that it takes a specific value $\{0, 1\}$. This probability is used to predict the values of the pixels to be coded, and the accuracy of prediction strongly influences the compression efficiency. Here, these observed pixels are called "reference pixels," and the configuration of their positions is called a "template." Figure 2-1 is a diagram of the template consisting of 16 reference pixels used by JBIG2-AMD2. The question mark in the figure represents the pixel to be coded and is not part of the template. While the positions for the 4 reference pixels are fixed, as marked with a #, there are also 12 special reference pixels called the adaptive template (AT) pixels, indicated in the figure as A_i $\{i = 1, \ldots, 12\}$. AT pixels can arbitrarily change positions within the range marked by the dotted line to achieve higher prediction accuracy and, in turn, higher compression efficiency.

However, it is a very difficult problem to optimize AT pixels according to the characteristics of images to be compressed. Therefore, the next section proposes an extended GA to optimize the configuration of the template of JBIG2-AMD2 quickly and efficiently.

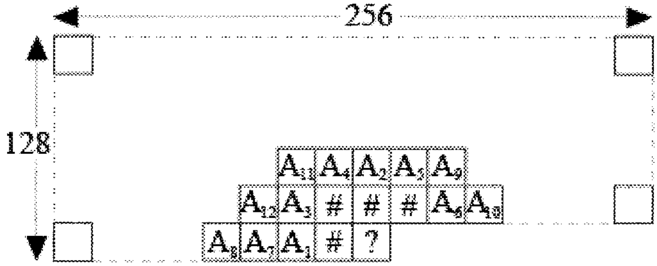

256
128
A_{11} A_4 A_2 A_5 A_9
A_{12} A_3 # # # A_6 A_{10}
A_8 A_7 A_1 # ?

Figure 2-1. Default configuration of JBIG2-AMD2 template

3. EXTENDED GA FOR TEMPLATE OPTIMIZATION

Since the basics and procedures of GA are already described in the previous chapter, this chapter shows some of the modifications and enhancements for the problem of optimizing the JBIG2 templates.

3.1 Coding of Chromosomes and Initialization of a Population

As each AT pixel exists within an area of 256 × 128, as shown in Figure 2-1, its location can be specified by 15 bits. If the number of AT pixels is M, then the total length of a chromosome is $15M$ bits. The enhanced template proposed in this chapter has 12 AT pixels, so the length of the chromosome is 180 (=15 × 12) bits, as shown in Figure 2-2. In the computational simulation described later, the population consists of 30 chromosomes.

An initial population is usually generated at random, although some GAs adopt a kind of biased initialization, based on certain information related to the problem. For example, in our previous method, the initial population was generated from a seed template by a mutation operation that was derived from a multiple regression analysis (Sakanashi, 2001). However, the computational complexity of the multiple regression analysis was enormous. Accordingly, in this chapter, the seed template is determined by assigning reference pixels one by one based on their degree of correlation to the pixel to be coded.

In order to calculate the degree of correlation between the pixel to be coded and each candidate AT pixel, it is necessary to scan the entire image to check whether the pixels are of the same value. Defining image size as XY, because the number of AT pixel candidates is approximately 256 × 128, the number of observations and comparisons required will be XY × 256 × 128. When the image size is small, the number of calculations would not pose a major problem. However, because graphic art images have very high resolutions, this number becomes extremely large. For example, as an A4 image with a resolution of 2400 dpi consists of roughly 20000 × 28000 pixels, approximately 1.8×10^{13} observations and comparisons would be needed to calculate the correlations.

Thus, to reduce the number of comparisons between pixel values, the degree of correlation is only checked for pixels to be coded, which are stochastically selected at a probability P_{scan}, and the respective candidate AT pixels. Investigating this P_{scan} probability in a second preliminary experiment, we found that a template of sufficient quality to serve as the seed template in initializing the population can be obtained with $P_{\text{scan}} = 5000/XY$, which is the P_{scan} value used in this chapter.

In the proposed method, one chromosome in the initial population is the bit string representing this seed template, with the remaining chromosomes being created by mutating this chromosome at twice the mutation rate to be explained later.

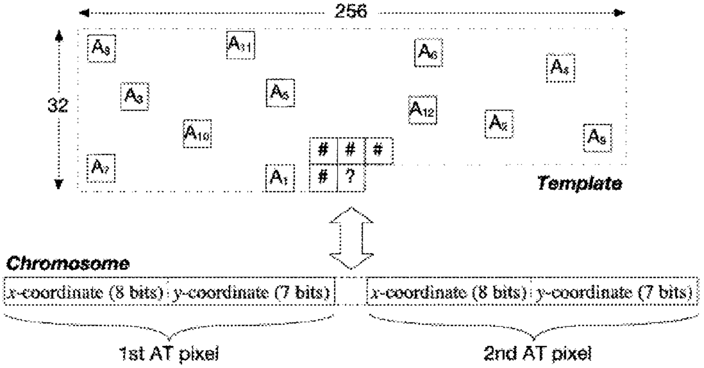


Figure 2-2. Coding of chromosomes

3.2 Random Partial Evaluation

As the objective function to calculate the fitness value of a chromosome, the proposed method uses the inverse of the compressed data size achieved by the template represented by the chromosome. Thus, a chromosome representing a good template that compresses the image data well will have a higher fitness value.

Incidentally, although it would be possible from an information theory perspective to use entropy as the fitness value rather than the compressed data size, there are two reasons, examined in preliminary experiments, for not doing so: (1) there are no significant differences between the costs for executing the MQ-Coder and for calculating entropy. (2) As the MQ-Coder can dynamically learn the statistics of the given data sequence, the template with the lowest entropy calculated in a static way does not always yield the maximum compression ratio.

Employing the compressed data size for evaluation, however, means that the image data must be repeatedly compressed to obtain the fitness value for each chromosome. The number of times the image data must be compressed will be $N \times G$ times, where N is the population size and G is the number of generations until the termination. Although the easiest way to reduce the evaluation costs is to only use a small part of the image rather than the entire image, evaluation accuracy and the reliability of the fitness value would deteriorate, making it impossible to discover the best template yielding the greatest compression efficiency.

Accordingly, this chapter proposes a procedure to solve this problem, consisting of the following two steps:

- Evaluation of the chromosomes is carried out using a small area of the entire image. The location of this area is changed at random, if a new best chromosome fails to emerge in the population within a given generation interval, G_{interval}. (If the location of the area is changed at every generation, the GA will fail because it cannot cope with such drastic fluctuations in the evaluation criteria.)
- When a new best chromosome does emerge, a hillclimb search is executed with this chromosome as the starting point. To avoid over-fit to the small area of the image, in the hillclimb search the pixels for evaluation are chosen from the entire image at a probability $P_{\text{SamplePixel}}$.

In this chapter, this procedure is referred to as "random partial evaluation," and the computational simulation described later uses the following parameters: $G_{\text{interval}} = 20$, $P_{\text{SamplePixel}} = 0.005$, and an area size of 1024×1024 pixels.

3.3 Genetic Operations with Template Crossover

This section proposes a new crossover operator suitable for template optimization. For other genetic operators, this chapter adopts existing methods such as tournament selection (Goldberg, 1991) and bit-wise point mutation. In the computational simulation in the next section, 80% of the chromosomes are chosen by tournament selection with a tournament size of 2, and the mutation ratio is 1/180.

In the template representation, there is no notion of order for the reference pixels. That is, if the AT pixels A1 and A2 in Figure 2-2 were exchanged, we would obtain the same compressed data. However, the pixels must be arranged in a certain order within the chromosome representation and this causes a serious problem.

If a simple 1-point crossover method is used, chromosomes with common reference pixels are frequently generated. For example, Figure 2-3 illustrates a case where the chromosomes C1 and C2 are generated by the 1-point crossovers of P1 = A|B|C|D and P2 = C|D|E|F. These chromosomes represent the templates PT1 and PT2, consisting of the set of reference pixels {A, B, C, D} and {C, D, E, F}, respectively. With the crossover point between the second and third pixels, one child, C2 = C|D|C|D, would unfortunately contain only two unique reference pixels, as shown as CT2 in the figure. Because the compression ratio generally tends to be higher when the number of pixels is larger, a chromosome representing a template with overlapping reference pixels will have a poorer evaluation.

Thus, in this chapter, we propose a special crossover procedure called template crossover, as follows:

1. A pair of chromosomes, P1 and P2, is compared to identify any common reference pixels, P_{common}.
2. If present, common reference pixels are removed from P1 and P2, respectively, to produce P1′ and P2′.
3. If the lengths of P1′ and P2′ are 0, P2 is mutated and the process is terminated.
4. Otherwise, reference pixels in P1′ and P2′ are exchanged in a fashion similar to bit-wise uniform crossover, with the results being defined as P1″ and P2″.
5. Finally, P_{common} is concatenated at the ends of both P1″ and P2″, with the results overwriting the original P1 and P2, respectively.

The check in step 3 for the lengths of P1′ and P2′ ensures that identical templates never appear in the population. Moreover, this elimination of redundancy efficiently reduces the search space and so contributes to improve GA search efficiency. Because a template with m reference pixels can be represented in (${}_{m}P_{m} = m!$) different ways as chromosomes, the number of redundant evaluations is greatly reduced by removing identical chromosomes.

The parameters of the proposed method mentioned above are summarized in Table 2-1.

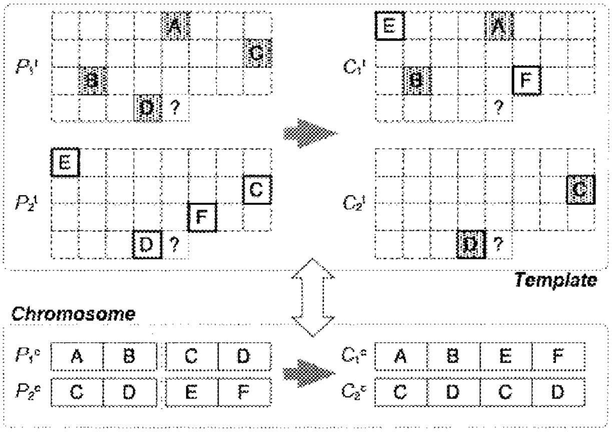


Figure 2-3. Example of a template with overlapping AT pixels being generated by a one-point crossover operation

Table 2-1. Parameter settings

Population size	30
Length of chromosome	180
Max generation	10000
Selection method	Tournament selection
Crossover method	Template crossover
Crossover ratio	0.8
Mutation method	Bit-wise point mutation
Mutation rate	1/180
P_{scan}	5000/[Image size]
Sample area size for evaluation	1024 × 1024
$\mathrm{G}_{\mathrm{interval}}$	20
$\mathrm{P}_{\mathrm{SamplePixel}}$	0.005

4. COMPUTATIONAL SIMULATIONS

This section presents the results of the computational simulations executed to examine the performance of the proposed method. This experiment used a set of test images containing the cyan and magenta images of N5, N6 and N8 in SCID (ISO/IEC Int. Stand. 12640, 1997), which were processed by raster image processor (RIP) to decompose the color image into the four bi-level images and to increase the resolution to 2400 dpi. They were chosen as a test image because they have the medium, smallest and greatest entropy levels of the eight images in SCID. Similarly, the cyan and magenta image has a larger level of entropy than the yellow and black images.

Firstly, to verify the effect of extending the GA for template optimization, an experiment was carried out with the 4 conditions shown in Table 2-2, using only the cyan N8 image.

Figure 2-4 shows a graph plotting the mean best fitness values achieved in 3 runs for each condition. As the evaluation areas changed at random periods, the fitness values fluctuated sharply, making it difficult to differentiate the performances across the 4 conditions.

The graph in Figure 2-5 plots the compression ratios rather than the fitness values, and Figure 2-6 is a close-up of Figure 2-5. These show that compression ratios improved as the GA search progressed, which is almost saturated within 1000 evaluations in every condition, and one of our future tasks would be to develop the appropriate termination criteria.

Looking at the contrast between the pairs of conditions [i]+[iii] and [ii]+[iv], which differ in terms of whether initialization was based on correlation analysis, the fact that the results for [iii] and [iv] are respectively better than [i] and [ii] indicates that the proposed population-initialization method effectively boosts the search performance of the GA.

Moreover, we can observe that the results for [ii] and [iv] are better than [i] and [iii] even though the initial populations were the same for the pairs of conditions ([i]+[ii] and [iii]+[iv]). This fact demonstrates the efficiency of the template crossover operator, which was adopted in the conditions [ii] and [iv] but not in conditions [i] and [iii].

Table 2-3 provides the results of the simulations executed to compare the performances of (1) JBIG2-AMD2 with the default template, (2) our method only with the initialization, and (3) the complete proposed method with the enhanced GA (condition [iv]). This table shows that the proposed method can achieve much better compression ratios than the default state of JBIG2-AMD2.

Additionally, the compression ratio achieved by the initialization-only method is about 16.6% ~ 99.7% better than JBIG2-AMD2 with the default template, although there is little difference between them in terms of computational costs. We can say that, from the viewpoint of practical use, it is a very reasonable performance in terms of compression efficiency and processing speed.

Table 2-2. Simulation conditions as a function of the proposed genetic operators

		Template crossover	
		OFF	ON
Initialization with correlation analysis	OFF	[i]	[ii]
	ON	[iii]	[iv]

Table 2-3. Comparison of compression ratios

		JBIG2-AMD2	Proposed method	
		(Default template)	(Initialization-only)	(Condition [iv])
N5	C	6.58	9.47 (+44.0%)	10.82 (+64.5%)
	M	6.42	8.88 (+38.4%)	9.90 (+54.3%)
N6	C	6.10	12.18 (+99.7%)	16.58 (+171.9%)
	M	5.56	10.30 (+85.3%)	13.34 (+140%)
N8	C	4.99	5.90 (+18.3%)	6.51 (+30.5%)
	M	5.15	6.00 (+16.6%)	6.49 (+26.1%)
Process time		33 sec.	45 sec.	1.2 hours

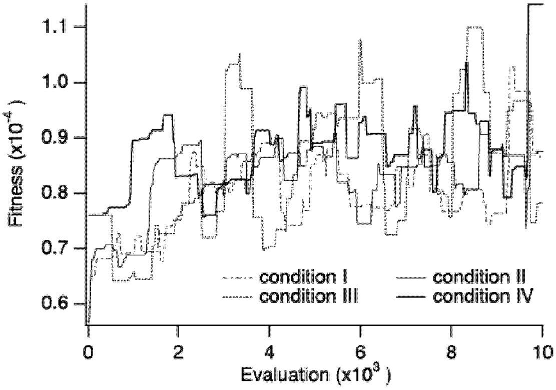


Figure 2-4. Learning curve

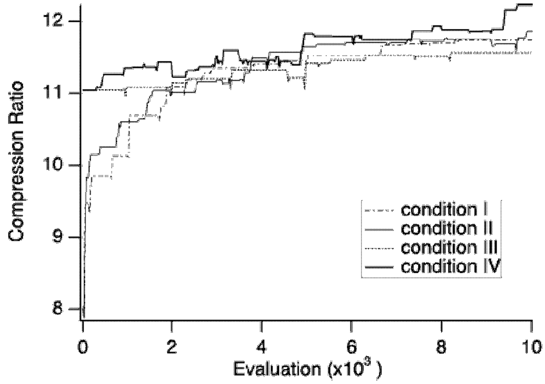


Figure 2-5. Improvements in compression ratios

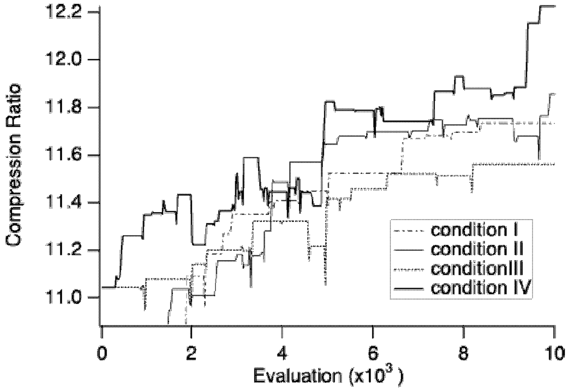


Figure 2-6. Improvements in compression ratios (close-up)

5. IMPLEMENTATION OF THE EVOLVABLE HARDWARE

As described in the previous section, our proposed method provides very high compression efficiency. However, from the practical perspective of use in the graphic industry, it is also necessary to implement high-speed compression/decompression because the data for the graphic art images handled by this method is huge. In this section, we explain the implementation of evolvable hardware (EHW) to speed up this method.

5.1 Architecture

The data compression EHW chip consists of a template optimizer, a data compressor, and an evaluator, as shown in Figure 2-7. In this implementation, both template optimization and evaluation are executed on the host PC. We adopted this kind of implementation because the systems that would employ our proposed compression method, e.g. electrographic printers, are equipped with the latest CPUs that can execute these procedures sufficiently and quickly for practical use. Moreover, processing with high flexibility becomes possible by executing optimization and evaluation on a PC. In addition, when our compression technique is in practical use on electrographic printers, only fast compression and decompression are required. In most cases, optimization and evaluation are executed in advance before the equipment is manufactured, and then the results are embedded in the equipment. For these reasons, this chip is optimized for practical use.

In the trial chip, which we discuss in this section, the area of the template is limited to 32 × 8 pixels with 10 reference pixels (Figure 2-8). In this case, 8 bits are required to specify the location of each reference pixel, so a template can be represented by 80 bits.

A block diagram of the chip is shown in Figure 2-9. The chip mainly consists of an MQ-Coder (ISO/IEC Int. Stand. 14492, 2001), which is the encoder/decoder, an image data memory that stores the input data, a reference buffer, and a context generator (shown as the hatched areas in Figure 2-9).

In order to execute the encoding/decoding procedures in parallel, and thus speed up processing, a feature of this architecture is the incorporation of two MQ-Coders.

The specifications of the sample chip are shown in Table 2-4, and a layout image of the chip is shown in Figure 2-10.

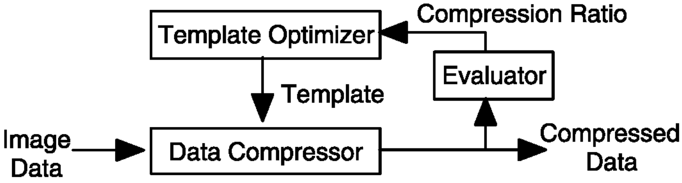


Figure 2-7. Configuration of the data compression system for our method

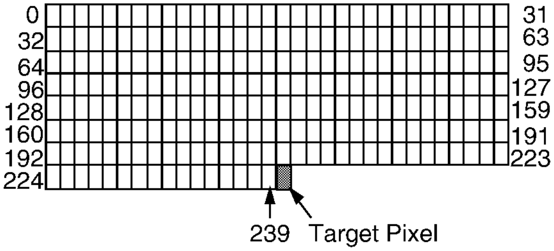


Figure 2-8. The area of the template in the chip

5.2 Elements of the Chip

5.2.1 Image Data Memory

The image data memory stores each line of the image data so that the reference buffers can extract reference areas efficiently from the image data. The size of this memory is 320 words × 32 bits × 9 lines. Each line in the memory holds one line of image data. If a line of image data is longer than a line of memory, then the image data line is divided into memory-line length units at preprocessing. The PC accesses the memory in 32-bits groups. The lines are updated whenever a process is completed. Accordingly, this memory always holds the areas for extraction by the reference buffers. The memory is divided into two groups at the middle of each line, which are used by reference buffer 1 and 2, respectively.

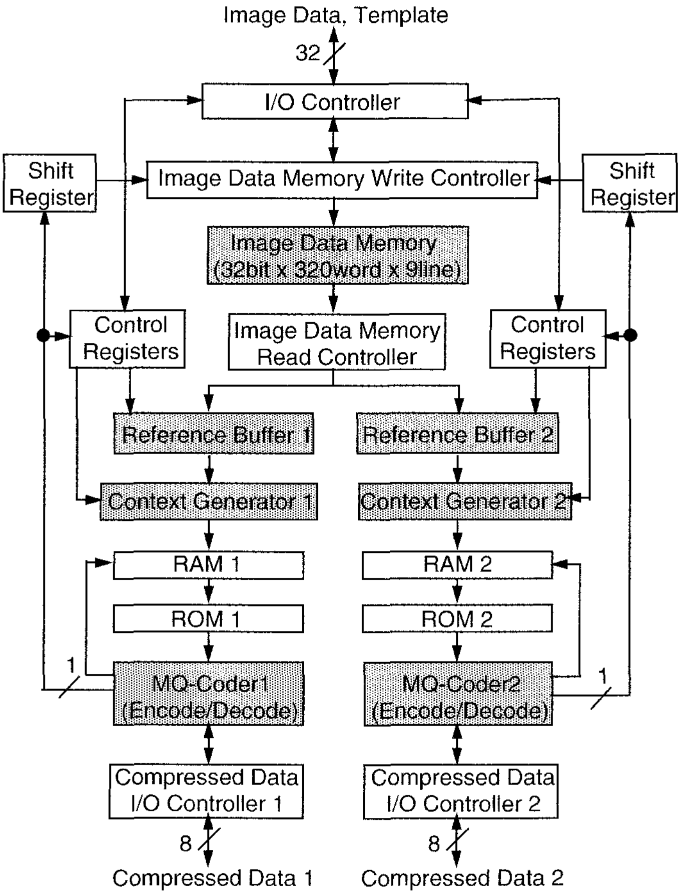


Figure 2-9. Block diagram of the chip

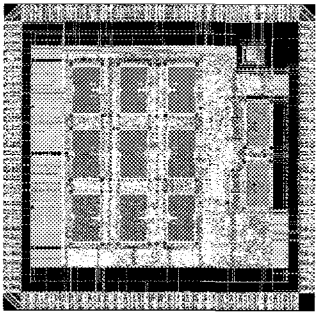


Figure 2-10. Layout image of the chip

Table 2-4. Sample chip specifications

Technology	0.18μm CMOS
Package	160 pin QFP plastic
Die Size	5.14 mm × 5.19 mm
Gate Count	about 56,000
Clock Frequency	133 MHz (maximum)
Supply Voltage	1.8 V (internal), 3.3 V (I/O)
Acceptable Image Size	10,240 × 65,536 (maximum)

5.2.2 Reference Buffer

The reference buffers are buffers for the reference areas of the templates. This chip has two buffers corresponding to the two MQ-Coders. Each reference buffer has two buffers of 64 bits × 8 lines, as shown in Figure 2-11. The data stored in the buffer represents the reference area of a template and is extracted from the image data memory (Figure 2-12). In the extraction procedure, the data is extracted by shifting the reference area in one-word (32 bits) increments, with each data set being stored into buffers A and B in turn. At the edge of an image, the data is clipped so that one-word of data protrudes from the edge, as shown in Figure 2-12. Data outside the image border is set with a pixel value of 0. The data assigning each column, as shown at the top of Figure 2-11, is used to match the data into the columns in Figure 2-12. The data is clipped in this way in order to efficiently process the data across each word. For example, a reference area crossing the border of the image would be processed with the data in the buffer A in Figure 2-11, and an area covering word0 and word1 would be processed with the data in the buffer B. The buffers are updated after the processes using the buffers are complete.

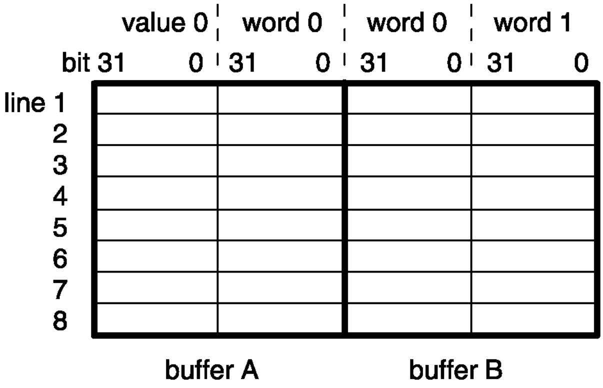


Figure 2-11. A reference buffer

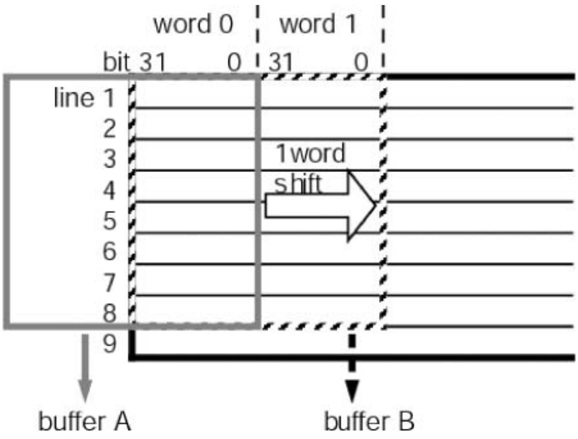


Figure 2-12. Extraction of reference data from the image data memory

5.2.3 Context Generator

The context generator generates a context (10 bits) from a template stored in the register and the data for a reference area in the reference buffer. A context, which is used by MQ-Coder, is the values of the 10 reference pixels for a template. The circuit mainly consists of 10 multiplexers of 240-inputs and 1-output. The multiplexer extracts the pixels specified by the template from the 240 pixels in the reference area of the template (Figure 2-8).

In this method, because the circuit in the context optimizer is optimized for each image, a template is selected in the learning mode as being optimal for that image. Thus, in this system the context optimizer has the same role as the reconfigurable device in EHW.

5.2.4 Encoder/Decoder (MQ-Coder)

The encoder/decoder used here is the same as that in the international standard for bi-level image encoding, JBIG2. It executes arithmetic encoding using pairs of context (10 bits) and a pixel value (1 bit). The circuit is based on the specification of JBIG2 (ISO/IEC Int. Stand. 14492, 2001) and JBIG2-AMD2 (ISO/IEC Int. Stand. 14492/Amd.2, 2003). The chip has two MQ-Coders executed in parallel. The RAM and the ROM placed in front of each MQ-Coder are also defined in the JBIG2 specifications.

5.2.5 Other Components

There is an I/O controller for the data I/O control between the chip and the PC. A shift register is used in decompression to send the decompressed data immediately into the image data memory and the reference buffers.

5.3 Execution Procedure

5.3.1 Compression

The procedure for compressing one image is as follows:

(1) Set a template in the control register.
(2) Set image data in the image data memory.
(3) Compress the data for one line. (The subprocedure is shown in (a) – (e).)
  (a) Clip data of 64 bits × 8 lines from the beginning of the image data memory and store in the reference buffer. Assign a pixel value of 0 for out of image areas (Figure 2-12).
  (b) Extract 10 pixels as directed by the template from the reference data in the reference buffer using the context optimizer.
  (c) The context (10 bits) and the image data (1 bit) are sent to the MQ-Coder (encoder part) via the RAM and the ROM. The encoded data are sent to the I/O controller.
  (d) Iterate (b) – (c) for the data in one buffer of the reference buffer. In this procedure, the reference area is shifted by 1 bit from the beginning to the end of the buffer, as shown in Figure 2-13.
  (e) Iterate (a) – (d) for the data of one line.
(4) Iterate (2) – (3) and send the compressed data into the I/O controller. The data is picked up from the external PC.

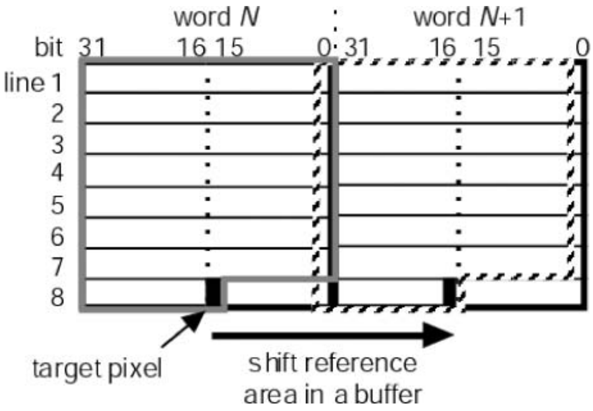


Figure 2-13. Shift of reference area in a reference buffer

5.3.2 Decompression

The procedure for decompressing one image is as follows:

(1) Set a template in the control register. Set all data in the image data memory to 0.

(2) Set the compressed data in the compressed data I/O controller.

(3) Decompress the data for one line. (The subprocedure is shown in (a) – (e).)

(a) Clip data of 64 bits × 8 lines from the beginning of the image data memory and store in the reference buffer. A pixel value of 0 represents either an out of image area or a pixel that is not decompressed yet.

(b) Extract 10 pixels directed by the template from the reference data in the reference buffer using the context optimizer.

(c) The context (10 bits) and the compressed data (8 bit) are sent to the MQ-Coder (decoder part) via the RAM and the ROM. The decoded data is sent to the reference buffer and the image data memory via the shift register.

(d) Iterate (b) – (c) for the data in a buffer in the reference buffer. In this procedure, the reference area is shifted by 1 bit from the beginning to the end of the buffer, as shown in Figure 2-13.

(e) Iterate (a) – (d) for the data of one line.

(4) Iterate (2) – (3) and send the decompressed data into the image data memory. The data are picked up from the external PC.

5.4 Performance Evaluation

5.4.1 Evaluation System

We have conducted an experiment to evaluate compression and decompression performance. The architecture of the evaluation system is shown in Figure 2-14. The system consists of a PC, an interface board, and the chip.

The PC has a hard disk and software. The hard disk stores the input and output data. The original data before compression is stored in TIFF format. The software mainly consists of I/O preprocessing and postprocessing, the interface between the board and the PC, the template optimizer, and the evaluator. The I/O preprocessing includes data reading and format conversion. The postprocessing executes header-making and combining compressed data in compression, as well as combining expanded data and conversion into TIFF format in decompression, and writing data into a file. The interface board executes the interface processing between the chip and the PC.

5.4.2 Speed

We evaluated the performance of the chip in terms of the speed of compression and decompression. The clock frequency was 133 MHz. First, we evaluated the processing speed of the chip when using only one MQ-Coder. The compression speed was 3 – 5 clocks/bit, 26.6 – 44.3 Mbit/s. The decompression speed was 5 – 7 clocks/bit, 19.0 – 26.6 Mbit/s. The speeds of this chip are the same as those of an MQ-Coder, indicating that the MQ-Coder represents a bottleneck to the processing speed of the chip. The changes in speed were due to the type of data processed.

Taking these results as a baseline, we next evaluated processing speeds when the two MQ-Coders are used in parallel. If the two MQ-Coders have the same processing speeds, then the total processing speed should be doubled when these are executed in parallel. The maximum speed for compression was 88.6 and the speed for decompression was 53.2 Mbit/s. Although the processing is executed in parallel, because the MQ-Coders process exclusively different data, the actual total speed is determined by the slower MQ-Coder. Thus, the total speed of the chip with parallel processing is twice the speed of the slower MQ-Coder.

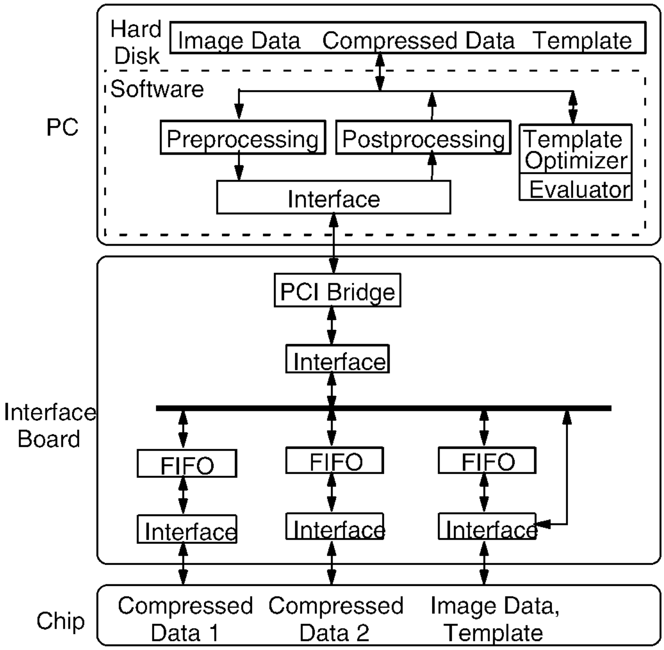


Figure 2-14. Block diagram of the evaluation system for the chip

Next, we evaluated the processing speed of the evaluation system including the PC (Figure 2-14). The speed was 5.1 Mbit/s for compression and decompression. The reason for the slower speeds is that there are too many processing threads executed in the PC. This results in slower PC processing, which is a bottleneck for the total system. Methods of improving this speed are discussed in the next section.

We compared the processing speed of the chip with the software. The compression and decompression speeds of the software using a PC (Pentium 4, 2 GHz) were 0.73 Mbit/s. This shows that the speed executed on the evaluation system is about 7 times faster than when executed on an external PC, and that the speed of the chip is two orders of magnitude faster than the speed of the software.

For reference, we present the processing speed of the JBIG chip (ISO/IEC Int. Stand. 11544, 1993), which is a standard compression method. Although JBIG is a prior version of JBIG2, we have selected this because no chip for JBIG2 currently exists. The JBIG chip is commercialized with the specifications of 1 clock/bit and 115 MHz clock speed. The speeds of the evaluation system and our chip are slower than the JBIG chip. However, it should be noted that the compression ratio obtained by our chip is about 50% better than that of JBIG (Sakanashi, 2001). The JBIG chip is faster because it compresses and decompresses data in 1 clock/bit. The possibility of 1 clock/bit processing for our chip is discussed in the next section.

As another reference, we briefly show the estimated processing speed of a JBIG2 chip, which has only been designed and evaluated for performance in simulations (Denecker, et al., 2002). The compression speed was 340 Mbit/s, and the decompression speed was 170 Mbit/s. Though the speed is faster than our chip, as we discuss in the next section, if our chip can compress data in 1 clock/bit with 200 MHz clock, then it would also execute compression and decompression at similar speeds to the simulated JBIG2 chip. However, the JBIG2 chip involves a larger template than our chip because its context has 12 pixels, and the reference area is 63 × 33 pixels. Because our chip is a prototype, we limited the size of the template due to restrictions over die size. We are currently designing new compression circuits using a technique to reduce the circuit size while still implementing a sufficient context size and reference area for practical use.

5.5 Discussion

In this section, we discuss how to improve the processing speed of the evaluation system and the chip.

To exploit the full potential of the chip, we need PC software that controls the chip without making the chip wait. However, the software is at present a bottleneck to the speed of the chip. The main reason for this is that there are too many processing threads (see Section 5.5.1) in the PC with the two MQ-Coders working in parallel, leading to slower PC processing speeds. Consequently, the speed of data transfer to and from the PC cannot match the processing speed of the chip, and much time is required for preprocessing and postprocessing, such as format conversion or file access, creating a bottleneck in the system.

Possible methods for accelerating these processes include using DMA for data transfer and using micro controllers for preprocessing and postprocessing, either on the chip or on the board. The processes will also be accelerated when the speeds of CPUs, PCI buses, and disc access on a PC become faster in the future. Moreover, a speed up in data transfer is expected by fabricating the circuits for the interface board in the chip. A PC with multiple CPUs executing many threads faster would also be effective.

Next, we discuss how to increase the processing speed of the chip. The chip has a write-back procedure into the RAM from the MQ-Coder (Figure 2-9). The MQ-Coder experiences some waiting time because of this procedure. By embedding timing adjustment circuits into the chip, we estimate that a compression speed of 1 clock/bit, 133 MHz per 1 MQ-Coder will be possible. However, the decompression process will still take more than

1 clock/bit, because the process has the write-back procedure into the reference buffer to generate a context. The problem of the additional time required for the decompression process will be a focus for our future research.

6. CONCLUSION

In order to achieve good compression efficiency for very high-resolution images, this chapter has proposed a new lossless compression method with a mechanism to optimize its own parameters using a genetic algorithm (GA). The proposed method has been developed on the basis of JBIG2, the current international standard for bi-level image encoding, and has an improved coding mechanism with an enhanced template of 12 reference pixels (AT pixels). An extended GA is also used in this method, which has the following features: (a) fast and effective mechanisms for initializing the population based on correlation analysis and random sampling of pixels, (b) a random partial evaluation procedure, and (c) template crossover.

The following results of computational simulations prove its advantageous performance: (1) a compression efficiency that was about 23% better than JBIG2, albeit after a long learning period (Section 2.3), (2) a compression efficiency approximately twice that of JBIG2 with default parameter settings with only a slight increase in computational cost, and (3) the new proposed method was able to complete learning about 96 times faster than our previous method using GA.

We have outlined the architecture of an EHW chip and reported the results of experiments to evaluate performance. The comparison of processing speeds with software execution showed that the evaluation system is about 7 times faster, and that the chip is two orders of magnitude faster.

Because of its advantageous performance in image compression, the ISO/IEC JTC 1/SC 29/WG 1 committee determined to adopt part of the technology used in this proposed method as an amendment to the JBIG2 standard. As the next step, we will propose to the ISO TC 130/WG 2 committee that TIFF/IT, the international standard of the file format for the prepress data exchange, adopts our proposed method as a compression method.

Acknowledgment

This work was supported by NEDO and AIST. We thank Dr. Nobuyuki Otsu, AIST fellow, and Dr. Masataka Hirose, the project leader of the advanced semiconductor research center, AIST.

References

CCITT Recommendation T.4. 1998. Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus.

CCITT Recommendation T.6. 1988. Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus.

Denecker, K., et al. 2002. Design of an Improved Lossless Halftone Image Compression Code, Signal Processing. In *Image Communication, vol. 17*, 277-292, Elsevier.

Goldberg, D. E. and K. Deb. 1991. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In *Foundations of Genetic Algorithms*, 69-93, Morgan Kaufmann.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

ISO/IEC International Standard 11544 | ITU-T Recommendation T.82. 1993. Coded Representation of Picture and Audio Information – Progressive Bi-level Image Compression.

ISO/IEC International Standard 12640. 1997. Graphic technology – Prepress digital data exchange – CMYK standard colour image data (CMYK/SCID).

ISO/IEC International Standard 14492 | ITU-T Recommendation T.88. 2001. Information Technology – Lossy/lossless coding of bi-level images.

ISO/IEC International Standard 14492/Amd.2 | ITU-T Recommendation T.88/Amd.2. 2003. Information Technology – Lossy/lossless coding of bi-level images, AMENDMENT 2: Extension of adaptive templates for halftone coding.

Sakanashi, H., M. Iwata and T. Higuchi. 2001. A Lossless Compression Method for Halftone Images using Evolvable Hardware. In *Evolvable Systems: From Biology to Hardware, LNCS 2210*, 314-326, Springer.

Sayood, K. 2000. *Introduction to Data Compression (2nd edition)*. Morgan Kaufmann.

Chapter 3

A GA HARDWARE ENGINE AND ITS APPLICATIONS

Isamu Kajitani, Masaya Iwata, and Tetsuya Higuchi
National Institute of Advanced Industrial Science and Technology, isamu.kajitani@aist.go.jp

Abstract: This chapter describes a GA (Genetic Algorithm) hardware engine and its applications to a controller for a hand-prosthesis and a LSI (Large Scale Integration) fabrication process. The GA hardware engine is a hardware implementation of GA operations. Therefore this enables high-speed execution of the GA search as well as their compact implementation.

Key words: genetic algorithm, LSI, FPGA, prosthesis, EMG.

1. INTRODUCTION

While a great deal of research has been carried out on GA (Goldberg, 1989) hardware, much of it has focused on accelerating the GA operators. That is, most studies have simply incorporated the GA operations onto a Field Programmable Gate Array (FPGA) in the hardware, and have evaluated the effects on the speed of the genetic operations (Kajitani, 2003).

Furthermore, because these studies have only been concerned with implementing a simple GA, most have not addressed any of the particular problems relating to the effective implementation of GA hardware, such as the following three problems (Kajitani, 2003):

1. Size of the memory for chromosomes. The size of the memory needed for chromosomes increases in proportion to the number of individuals. While the number of individuals should, ideally, be restricted when it is implemented on a fixed-size LSI or FPGA, a larger population is better for effective genetic searching because diversity of chromosomes can be easily preserved in the population.

2. Selection strategies. The generational-selection strategy (Goldberg, 1989) used in a simple GA requires extra memory to temporarily hold selected individuals. However, a steady-state selection strategy (Goldberg, 1989), which does not require such extra memory, is more appropriate for GA hardware implementation.
3. Random number generation for crossover points. In crossover operations, the crossover point (or points, in the case multipoint operations) is determined by random numbers, so their generator must be capable of providing random numbers equal to the length of the chromosome. However, because a random number generator implemented on the hardware can only generate random bit strings in the form 2^N (where N is the length of the random bit string), extra hardware for normalization is required to fit the random numbers to the length of the chromosome.

Because of these problems, clearly, it is necessary to satisfy the following three requirements in order to implement genetic operators on hardware (Kajitani, 2003):

1. An effective search method with a small population
2. A steady-state GA
3. Crossover operations that do not require extra hardware for normalization of random numbers.

Based on these design considerations, we proposed using Elitist Recombination (ER) (Thierens, 1997) and uniform crossover (UC) (Syswerda, 1989), as an ideal combination, which is capable of meeting these criteria.

There are three advantages in using this method. Firstly, this method is basically an elite-strategy in which the fittest individual always survives. However, as selection only occurs within a family (i.e., the two parents and two children), diversity is easily retained in the population. The first advantage is therefore that this method can search effectively, even with a small population and, thus, meets the first criteria. Secondly, because there are no separate selection and recombination (crossover) phases (Thierens, 1997), the crossover rate can be specified to be 1. Thirdly, this method is basically a steady-state strategy, which does not require extra memory or registers to temporary preserve selected individuals; therefore, it also satisfies the second requirement.

The workflow for Elitist Recombination is as follows:

- Firstly, initial chromosomes are generated and evaluated.
- Secondly, two individual parents are randomly selected.
- Genetic operations, such as the crossover and the mutation, are performed to two parents to generate two new chromosomes (two children).
- The fitness values are compared among the two parents and two children.

- The two fittest chromosomes from amongst the two parents and two children are selected.

The uniform crossover was proposed by Gilbert Syswerda (1989). Unlike one- or multipoint crossovers, this method is not necessary to select the crossover points. In this method, each allele will exchange its information with a fixed probability of 0.5. This method, therefore, meets the final criteria. By combining these two genetic operators it is possible to meet all three criteria.

This chapter describes a GA hardware engine and its two applications. The GA hardware engine is applied to a controller for a hand-prosthesis (Kajitani, 2003) and a LSI fabrication process (Kajitani, 2003). In the hand controller application, the GA hardware engine is utilized to enable adaptable synthesis of controller circuit. In the case of LSI fabrication process, the GA hardware engine is applied to a post-fabrication clock-timing adjustment (Kajitani, 2003).

Section 2 shows an evolvable hardware LSI (Large Scale Integration) chip technology, used in the hand controller. The hand controller is explained in Section 3. In Section 4, the GA based post-fabrication clock-timing adjustment and the GA hardware engine for this application are described, before conclusion in Section 5.

2. AN EVOLVABLE HARDWARE (EHW) CHIP

This section introduces the evolvable hardware (EHW) chip technology, which is used in the hand controller. Evolvable hardware is based on the idea of combining a reconfigurable hardware device with GAs (Higuchi, 1999) to execute reconfiguration autonomously. This section briefly describes reconfigurable hardware before explaining how these are combined to realize evolvable hardware.

2.1 Reconfigurable Hardware

The circuit structure of reconfigurable hardware devices can be changed by downloading to them a bit string called the configuration bit string.

A PLA (Programmable Logic Array, Fig. 3-1) is one of the most fundamental reconfigurable hardware devices, consisting of an AND-array and an OR-array. In Fig. 3-1, the black and white circles indicate switches, which determine the interconnections between the inputs and outputs (black circles indicate the connections). The rows of the AND-array form logical products for connected inputs, and the columns of the OR-array form logical sums for the connected row in the AND-array. We can specify these switch settings with a configuration bit string, as shown in Fig. 3-1.

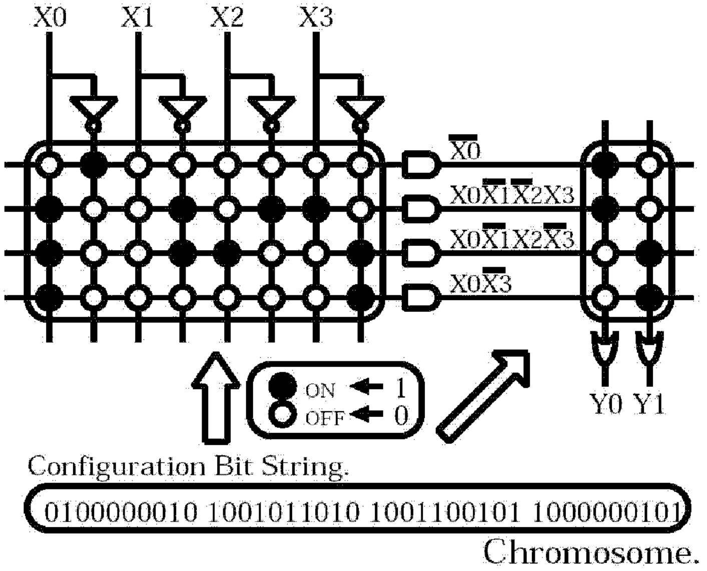


Figure 3-1. The basic structure of the PLA

2.2 Workflow

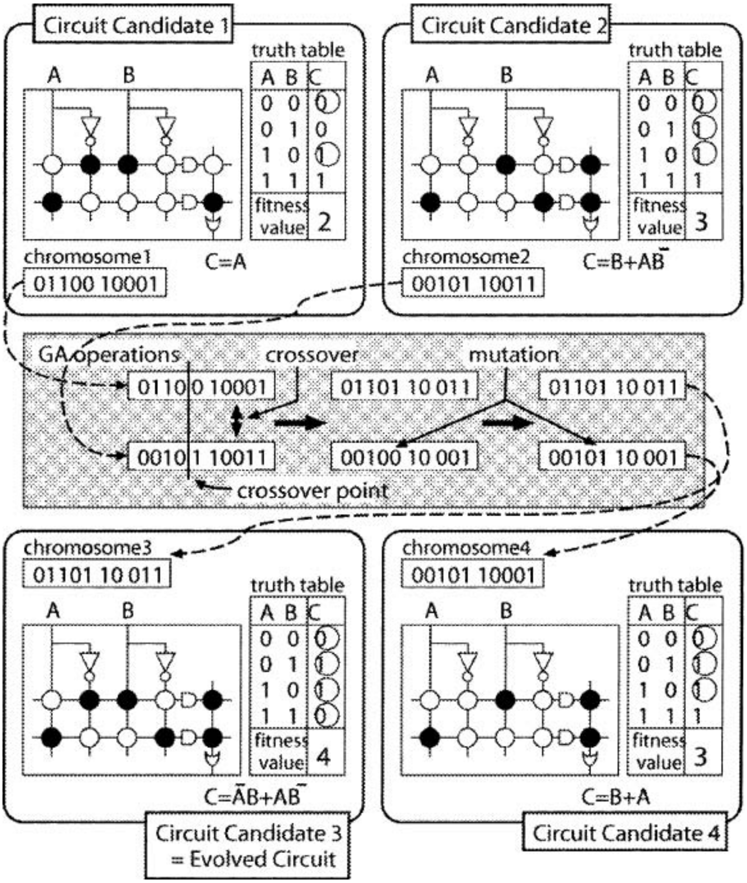
The basic concept behind the combination of reconfigurable hardware devices and GAs in EHW is to regard the configuration bits for the reconfigurable hardware devices as chromosomes for the genetic algorithms. If a fitness function is properly designed for a task, then the genetic algorithms can autonomously find the best hardware configuration in terms of the chromosomes (i.e. configuration bits).

Usually, a training sample of input-output patterns is used to evaluate the chromosome (Higuchi, 1999). In this particular application, the fitness value for a chromosome (i.e. circuit candidate) is the output pattern rate, that is, the rate at which the actual output corresponds to the expected output pattern for a given training input pattern.

In order to simplify explanations in this chapter, we adopt the simple example of synthesizing an exclusive-or (XOR) circuit. This circuit has an input width of two-bits and an output width of one-bit. In Fig. 3-2, two circuit candidates ("Circuit Candidate 1" and "Circuit Candidate 2") are shown, whose configuration bit strings are "0110010001" (Chromosome 1) and "0010110011" (Chromosome 2), respectively. The fitness value for "Circuit Candidate 1" is 2, because the output signal from this circuit for the two input patterns of "00" and "10" match the output signal from the target XOR circuit. Similarly, the fitness value for "Circuit Candidate 2" is 3.

Genetic operations are applied to these chromosomes. The one-point crossover operation, where the crossover point is between the 4th and 5th bit of chromosomes, is followed by the mutation operation. In this case, the 5th bit of the lower chromosome is changed from 0 to 1.

By applying these operations, two new chromosomes ("Chromosome 3" and "Chromosome 4") are generated, which correspond to the "Circuit Candidate 3" and the "Circuit Candidate 4", respectively. The fitness value for "Circuit Candidate 3" is 4, which indicates that all of the output signals from "Circuit Candidate 3" are the same as the signals from the target XOR circuit.



truth table

A	B	C
0	0	0
0	1	0
1	0	1
1	1	1
fitness value		2

truth table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1
fitness value		3

truth table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0
fitness value		4

truth table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1
fitness value		3

Figure 3-2. An example of circuit synthesis with genetic algorithms

2.3 Hardware Implementation of the EHW (The EHW Chip)

For most EHW research, genetic operations are executed by software running on either a PC (personal computer) or a WS (workstation). This makes it difficult to utilize EHW in situations that need circuits to be as small and light as possible, such as a prosthetic hand controller. One solution to this is to implement the GA on the hardware and to incorporate it in a single LSI chip together with the reconfigurable logic.

The first version of the EHW chip, designed by us in 1998 (Kajitani, 2003), consisted of (1) a GA hardware block, (2) a reconfigurable hardware block, (3) SRAM for chromosomes, (4) SRAM for training patterns and (5) a 16-bit CPU core block (NEC V30). Subsequent versions of the chip realized improvements to the GA hardware in terms of increased execution speeds and reduced size.

Figure 3-3 and Fig. 3-4, respectively, show a mask micrograph of the latest (fourth) version of the EHW chip and a block diagram. This EHW chip consists of (1) an EHW core block, (2) a CPU (V30) core block, (3) a control logic block, (4) an A/D converter core block and (5) an FIFO block. The EHW core block consists of a GA hardware block, as well as a random number generator, a reconfigurable hardware block (PLA), and a control register block.

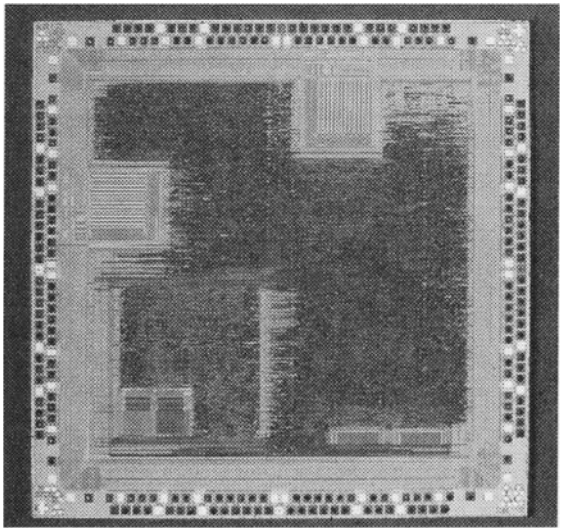


Figure 3-3. A mask micrograph of the fourth EHW chip

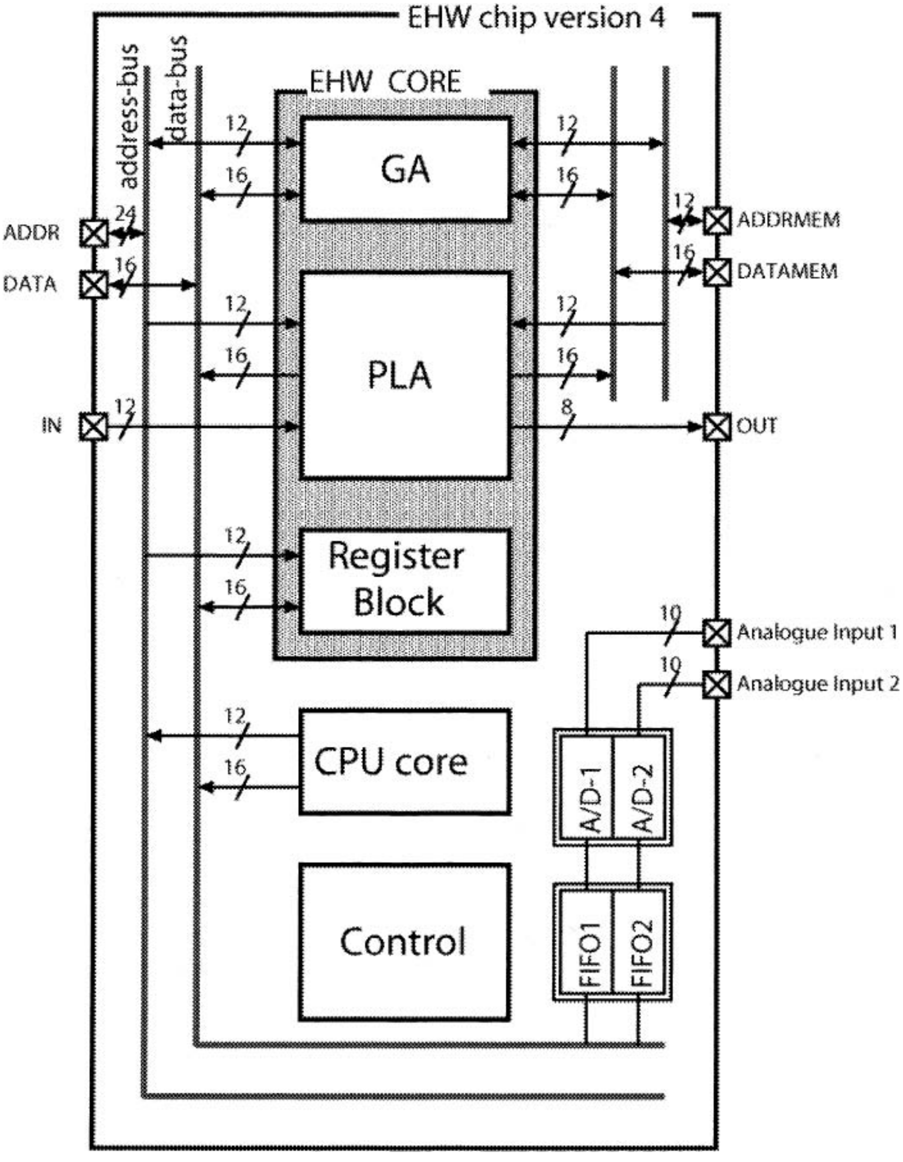


Figure 3-4. The fourth version of the EHW chip

3. THE MYOELECTRIC HAND CONTROLLER

3.1 Background

The myoelectric controller interprets control intentions from the operator by recognizing myoelectric signals. This kind of controller has typically been applied to control electric-powered prostheses. The most notable advantage of using the myoelectric controller is its capacity to utilize the residual muscular functions of physically impaired persons. For example, in the case of a hand prosthesis, the myoelectric controller enables the amputee to utilize the residual functions of remnant muscles at their stump.

Although the myoelectric control of electric-powered prosthetic hands has been researched since the early 1960s (Horn, 1963), and some prosthetic hands are already commercially available, most are single-function systems—only capable of performing a single function such as open-close—and are thus of limited usefulness in daily life. Accordingly, there have been calls to develop multifunction systems, capable of carrying out more than one function (Atkins, 1996).

In response to this demand for greater prosthetic hand functionality, recently, much research has been conducted on multifunction forearm prosthesis (Englehart, 2000; Hudgins, 1993; Kajitani, 2003), applying pattern classification methods, such as neural networks or logic circuits, in order to determine hand actions.

While neural network controllers are capable of highly accurate pattern classification, size is a major obstacle to compact implementation, which is required in an application such as a prosthetic hand controller, where the prosthetic hand has to be implemented to be both smaller and lighter than a human hand. We (Kajitani, 2003) have, therefore, proposed utilizing a logic circuit for myoelectric-pattern classification in order to realize a compact multifunction prosthetic hand.

However, because myoelectric signals vary both among individuals and over time for the same individual, it is not possible to determine in advance the exact specifications of the classification circuit. Accordingly, the evolvable hardware chip (EHW chip), described in this chapter, was adopted for myoelectric pattern classification.

Photographs in Fig. 3-5 – 3-8 show four cases of clinical evaluations to our developed hand prostheses.

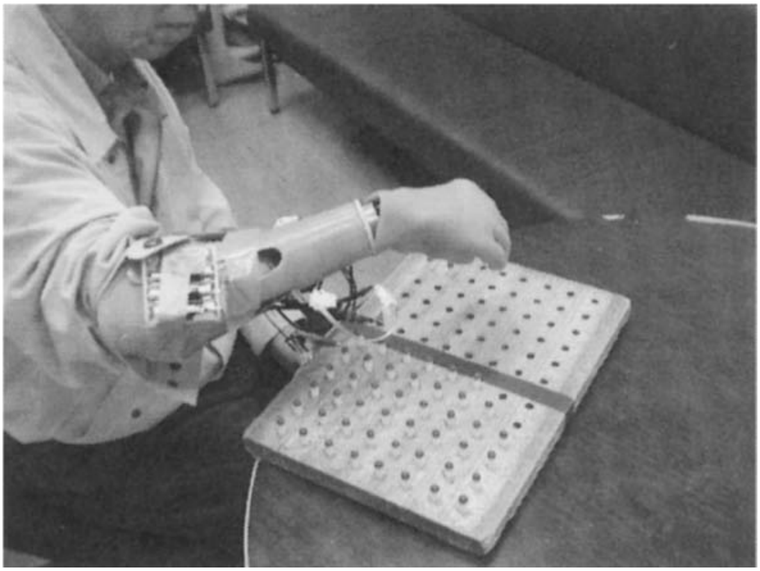


Figure 3-5. Clinical evaluation of the developed hand: case 1

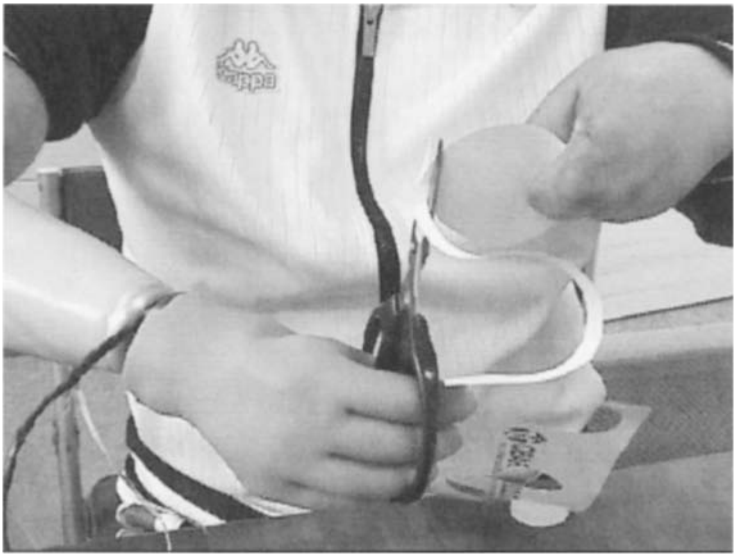


Figure 3-6. Clinical evaluation of the developed hand: case 2

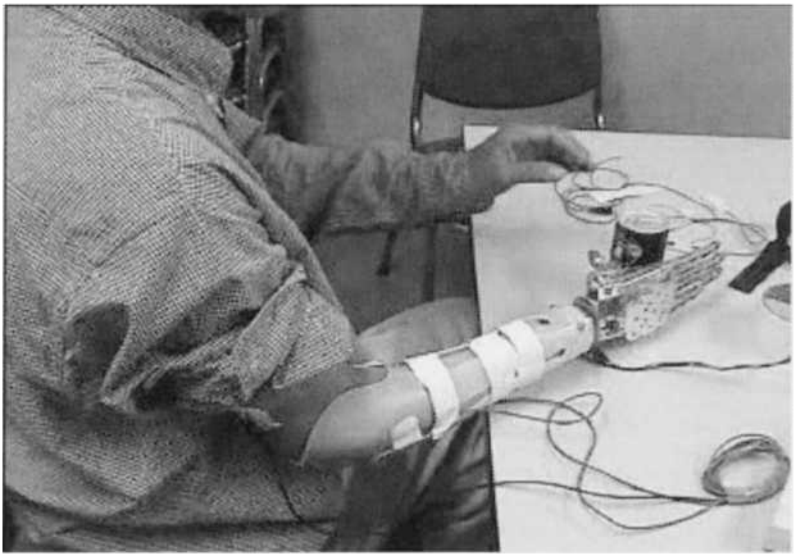


Figure 3-7. Clinical evaluation of the developed hand: case 3

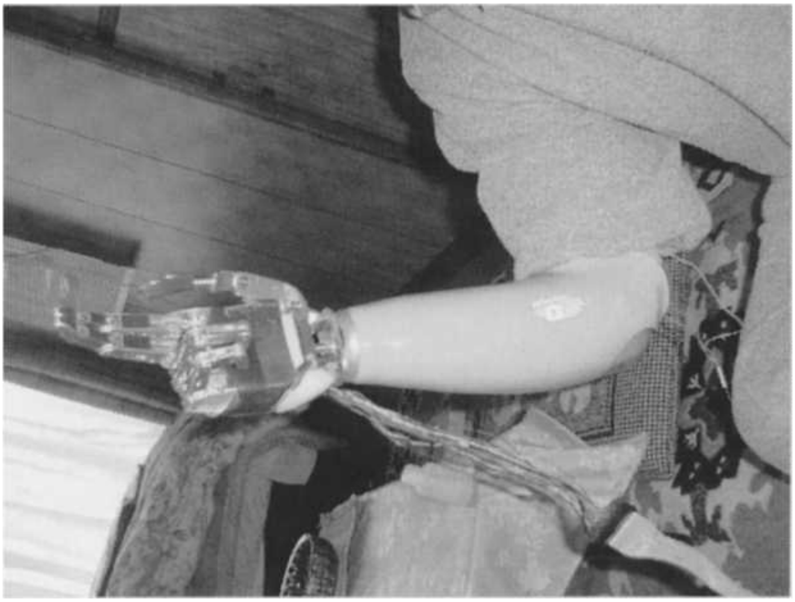


Figure 3-8. Clinical evaluation of the developed hand: case 4

3.2 A Myoelectric Pattern Classifier with the EHW Chip

The basic concept underlying our approach to myoelectric hand prosthesis is to control a multifunctional mechanical hand with the EHW chip for myoelectric pattern classification. The myoelectric hand system consists of a controller, a mechanical hand, myoelectric electrodes, and a battery. In our latest system, two commercially available myoelectric electrodes (OttoBock; 13E125=50) provide two myoelectric signal channels.

3.3 Myoelectric Pattern Quantization

Myoelectric signals were sampled during execution of the following six muscle contractions: (1) forearm supination, (2) forearm pronation, (3) wrist volar-flexion, (4) wrist dorsal-flexion, (5) hand closing and (6) hand opening, using surface electrodes attached to the forearm of the subjects. The sampled myoelectric signals were normalized to the values, which range from 0 to 7, before being quantized to discrete number, with linear quantization.

Figure 3-9 shows an example of myoelectric pattern distribution, whose x-axis means amplitude of signals detected from one electrode, while the y-axis means that from the other electrode. In the worst case of quantizing for the patterns in this figure, three different actions–forearm supination (19 patterns), forearm pronation (20 patterns) and hand opening (2 patterns)–would all be quantized as (0,0). This would lead to all the distinct patterns being classified as the same pattern, i.e., as being generated from the same action, and this kind of quantization error would hinder high-precision pattern classification.

This kind of quantization error is caused by the bias in the distribution of myoelectric patterns, which is not uniform over the distribution range, but is rather biased towards the low-value region.

3.3.1 Previous Works for Myoelectric Quantization

In order to remove the biases in the distributions of myoelectric patterns, we have proposed employing a method of logarithm quantization (Kajitani, 2001), which provides transformations with high precision for the low-value regions, but with low precision for the high-value regions. By applying logarithm quantization, the correct pattern classification rate increased by 10.5% (maximum) from 81.8% (without logarithm transformation) to 92.3% (Kajitani, 2001).

However, this method sometimes failed to increase the classification rate, and, consequently, the averaged rate only increased by 3.1% for ten subjects

(Kajitani, 2001). This is because the logarithm quantization employs a fixed algorithm, and is not, therefore, effective for all myoelectric signal pattern distributions. In order to overcome this problem, we proposed utilizing μ-LAW quantization (Kajitani, 2003; Smith, 1957).

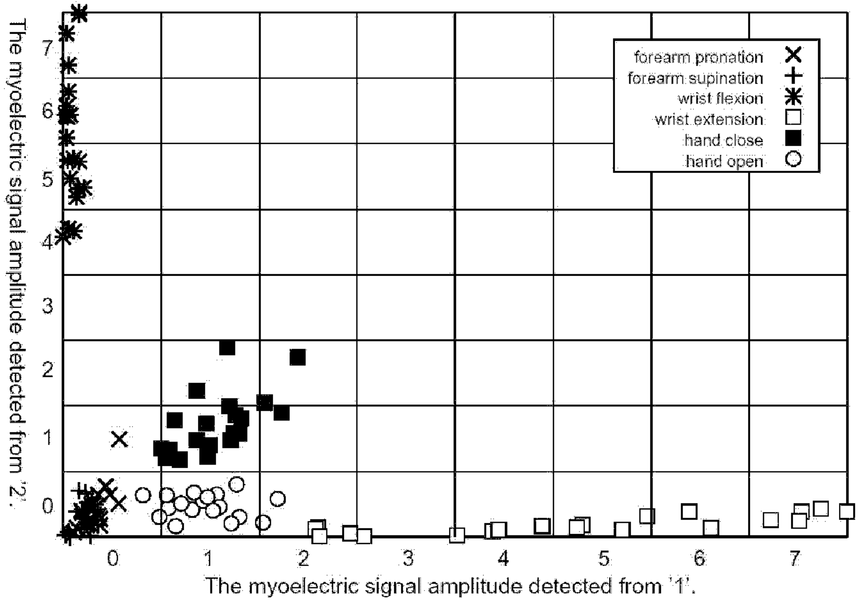


Figure 3-9. An example of the myoelectric pattern distribution and their quantization with linear-quantization

3.3.2 μ-LAW Quantization

The μ-LAW quantization is performed by applying the following transformation, before quantizing to discrete numbers.

$$v = \frac{V \log(1 + (\mu e / V))}{\log(1 + \mu)}, \quad for\ 0 \leq e \leq V, \tag{1}$$

where *e* is an input signal, *v* is an output signal, *V* is the maximum value of input signals and μ is a transformation rate, which defines the transformation characteristics (Fig. 3-10). In this method, the transformation characteristic can be adapted to the myoelectric distribution in terms of μ-value; therefore it can be applied to any exponentially-distributed myoelectric signal pattern.

Figure 3-11 shows myoelectric signal patterns, which are transformed from the patterns in Fig. 3-9 by applying equation (1). In this distribution, the worst case of quantizing for the patterns with μ-LAW quantization is (2,3), where forearm supination (8 patterns) and forearm pronation (5 patterns) are both quantized to these values. This represents a large reduction in quantization error and, so, yields high-precision pattern classification. In this example, the μ-value is set to 100.

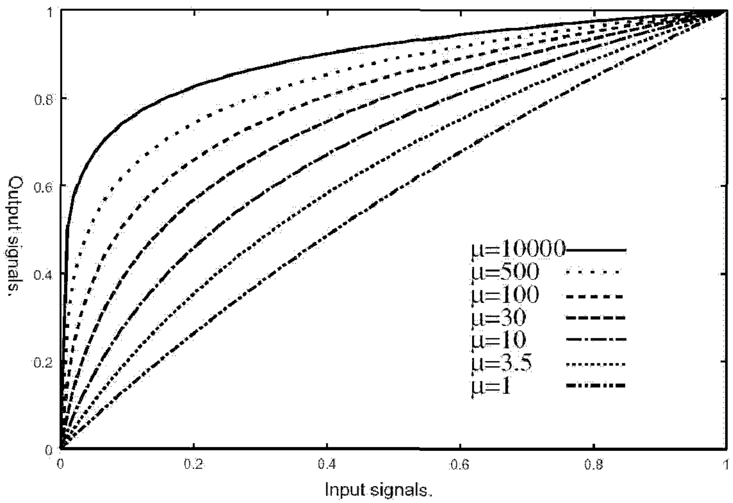


Figure 3-10. The μ-LAW transformation characteristic

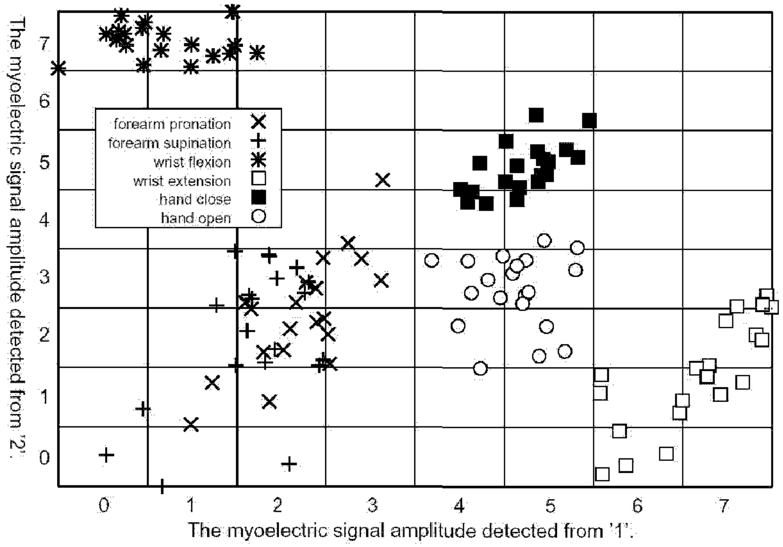


Figure 3-11. An example of the myoelectric pattern distribution and their quantization with μ-LAW-quantization

3.4 Pattern Classification Results

The effectiveness of μ-LAW quantization has been confirmed in a pattern classification experiment using samples from five subjects, including one experienced person ('skilled' in Table 3-1) who has repeatedly participated in our experiments, and four new subjects ('beginners' in Table 3-1) joining our experiments for the first time.

In these experiments, the parameters for GA operations were specified as follows:

- Number of populations: 32
- Crossover rate: 1.0
- Mutation rate: 2/256

Table 3-1 shows pattern classification rates with linear-quantization and with μ-LAW quantization. Applying μ-LAW quantization, the pattern classification rate increased by 11.1% (averaged for five subjects) and by 15.5% (maximum; subject 1). Furthermore, the classification rate for the experienced person increased to 97.8% (averaged over ten trials), clearly demonstrating that experienced persons can operate a multifunctional myoelectric hand with high accuracy.

Table 3-1. Pattern classification results

		linear	μ-LAW	
	Subject ID	rate (%)	rate (%)	μ-value
Skilled	1	82.3	97.8	100
Beginner	2	78.7	84.7	500
	3	67.8	74.3	3.5
	4	69.0	83.4	10000
	5	75.5	88.8	10000

4. A GA HARDWARE ENGINE FOR A POST-FABRICATION CLOCK-TIMING ADJUSTMENT

This section describes the development of the GA hardware engine, which is used in the fabrication of high-performance LSI chips, such as processors with clock speeds of 3 GHz or higher. In designing LSI chips, the transmission timing of the high-speed clock is a crucial issue. In order to solve the problems associated with clock timing, we have proposed a GA-based clock-timing adjustment method, presenting simulation results in 1999 (Takahashi) and evaluation data for two developed test chips in 2003 (Takahashi).

4.1 GA-Based Clock-Timing Adjustment

The GA-based clock-timing adjustment method is realized through the combination of GA adaptation and dedicated circuitry.

The core technology for the circuitry is programmable delay circuits, which are inserted into the clock lines to adjust clock transfer timing. Each delay circuit consists of a delay generation circuit and a register to store delay specification values. The delay generation circuits are capable of generating precise delays shorter than 30ps (Takahashi, 1999). Delay timings can be changed by specifying the binary bit patterns, which are treated as the chromosome for GA search.

In the LSI fabrication process, multiple programmable delay circuits are inserted into the clock lines, then the GA determines the delay value for each circuit based on error counts to function tests, which are used for the fitness value of the GA search.

This GA-based method has three important advantages: (1) enhanced clock frequencies leading to improved operating yields, (2) lower power supply voltages while maintaining operating yield, and (3) reductions in design times. We have demonstrated these advantages with a clock frequency enhancement of 25% (maximum), a power supply voltage reduction of 33%, and 21% shorter design times using GA software running on a PC (Takahashi, 2003).

The GA adjustment of clock-timing is applied during the testing phase in the LSI fabrication process. However, with the recent growth in LSI technology, demands to reduce testing costs have become especially pressing. Accordingly, methods that utilize hardware to facilitate testing are becoming extremely important in reducing testing costs, such as the BIST (Built-in Self-test) and the BOST (Built-out Self-test).

Given the need to reduce testing costs, clearly, the hardware implementation of GA adjustment is a primal technology in the realization of GA-based clock-timing adjustment.

4.2 Design Considerations

We have conducted three simulation studies to specify the crossover and mutation operations and to decide population size, with a view to compact implementation. The target circuits are the memory test pattern generator (circuit1) and the multiplier (circuit2) (Takahashi, 2003).

4.2.1 Crossover and Mutation

The first simulation was carried out to select a crossover operation. One of simplest crossover operations, called one-point crossover, was applied in the previous report (Takahashi, 1999), however, given the nonlinear characteristics of the present application, uniform crossover was considered to be more suitable. Therefore, one-point crossover and uniform crossover were compared in this simulation study.

Figure 3-12 presents the simulation results for the circuit operating yield rates. This shows that the operating yield with uniform crossover constantly exceeded the rate with one-point crossover.

The second simulation was carried out to select a mutation operator and to specify the mutation rate. The candidate mutation operators compared were the *Gaussian* random mutation used in the previous report (Takahashi, 1999), and a uniform random mutation, which is the most generally used method in GA search.

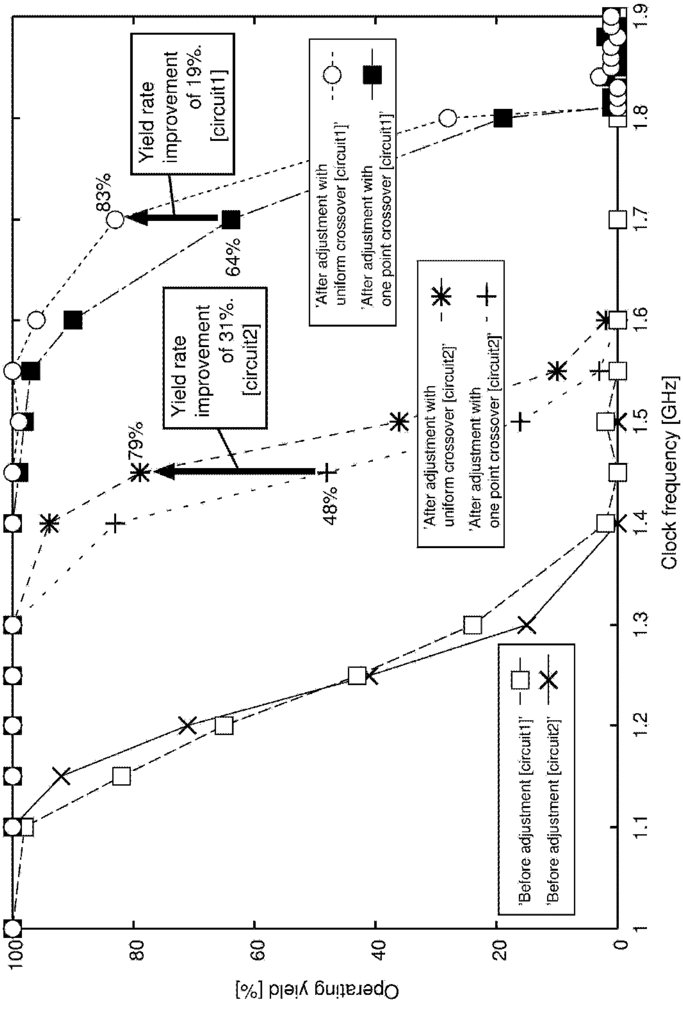


Figure 3-12. Operating yield rates for one-point crossover and uniform crossover

Figure 3-13 plots operating yields as a function of mutation operator and mutation rate, with the rate denominator along the x-axis. This graph shows that the best operating yield rate was obtained with the *Gaussian* random mutation when the mutation rate was 1. This result clearly demonstrates that it is possible to prespecify the mutation rate without adversely affecting performance by utilizing the *Gaussian* random mutation.

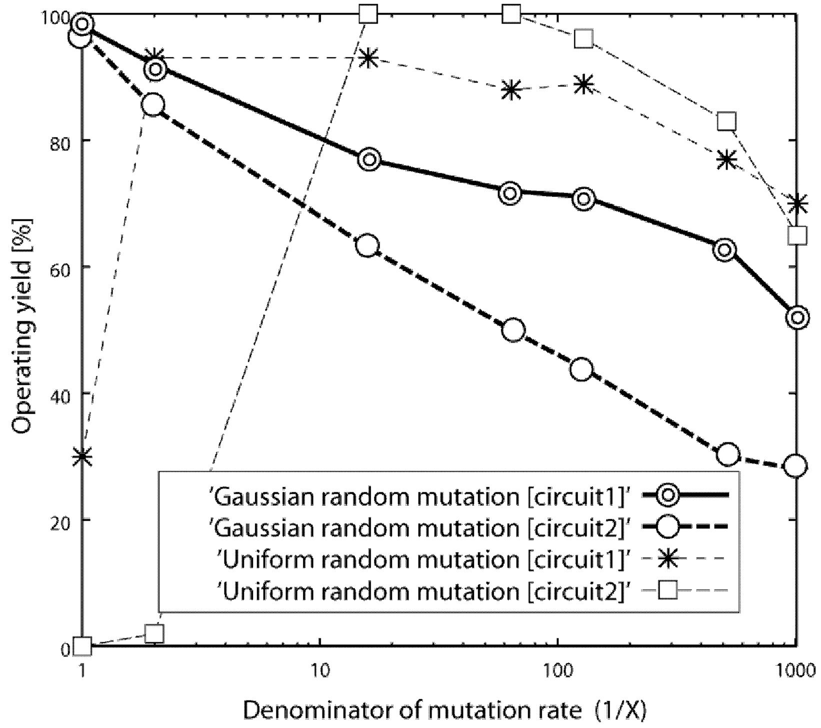


Figure 3-13. Operating yields as a function of mutation operator and mutation rate

4.2.2 Population Size

This section considers population size for GA search. A large GA population size is desirable for effective searching, however, population size is not necessarily the most important criterion for practical real-world applications.

In such applications, acceptable search durations—the time for completion of GA search—tend to be rather limited, so the evaluation iteration time is also restricted. The next two examples illustrate the relationship between population size and search duration.

Given the following specifications:

- GA search must be completed within ten seconds (10,000 ms).
- Population size is 1,000 chromosomes.
- Each evaluation requires 1 millisecond.

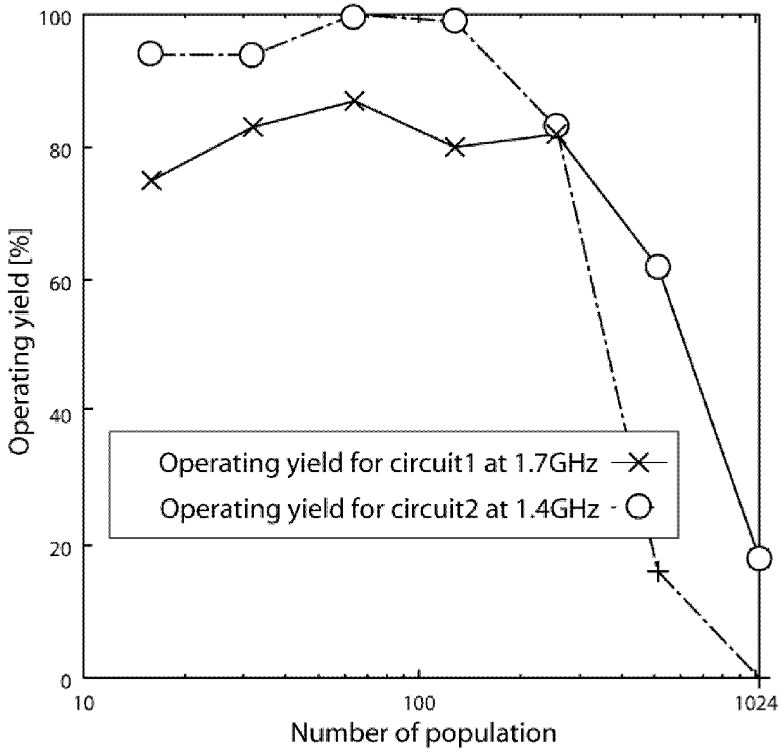


Figure 3-14. Operating yields as a function of population size

If all chromosomes undergo GA operations at the same probability, then a given chromosome would only undergo GA operations around 10 times (10,000 ms / 1,000 chromosomes / 1 ms = 10 times).

In contrast, if the population size were 10, then each chromosome would undergo GA operations around 1,000 times (10,000 ms / 10 chromosomes / 1 ms = 1,000 times). Clearly, a chromosome in a population of 10 has a much greater probability of being optimized by the GA search compared to a chromosome in a population of 1,000. However, because low populations tend to yield insufficient optimization, population size must be determined carefully, considering both acceptable search duration and hardware size.

Figure 3-14 shows a typical example of operating yields as a function of population size. These operating yields were calculated at clock frequencies of 1.7 GHz for the test circuit1 and 1.4 GHz for the test circuit2, respectively. Figure 3-14 clearly shows that operating yield decreases as population size increases. This result indicates that population size should not be too large. By considering hardware size, we set the population size to thirty-two for the GA hardware described in the following section.

4.3 Hardware Implementation of the GA

4.3.1 Overall Architecture

Figure 3-15 (upper) shows an overall architecture which consists of hardware for both the GA operations and the evaluation function. The evaluation hardware block receives a set of chromosome data (child1, child2) from the GA hardware block. The chromosomes are evaluated to calculate fitness values (fitness1, fitness2) for these, and the fitness data set is sent back to the GA hardware block.

Figure 3-15 (lower) is a block diagram of the GA hardware block, consisting of a GA control block, a GA operation block, a 3-port SRAM, two FIFOs (First-In, First-Out registers), and a selector. The GA control block receives a set of fitness values (fitness1, fitness2) in order to generate control signals for each block.

Figure 3-16 shows a block diagram of the circuit for GA operations. Note that this circuit performs the GA operations in parallel in units of 8 bits; therefore, thirty-two cycles are required to execute GA operations on each chromosome (8 bits × 32 cycles = 256 bits).

Two chromosomes in the SRAM (parent1, parent2) are selected to undergo GA operations to generate two chromosomes (child1, child2). The chromosomes for child1 and child2 are sent to the evaluation hardware block, as well as being stored in the two FIFOs.

4.3.2 Chromosome Descriptions

In this application, each delay circuit is specified by a segment of the chromosome for the GA search. Each segment of the chromosome is a four-bit binary bit string, called a gene, which corresponds to an unsigned integer ranging from 0 to 15.

As the number of delay circuits is 44 and 52, for test circuit1 and circuit2 respectively, the maximum length (number of four-bit binary segments) for a chromosome was fixed at 64 segments (4 bits × 64 segments = 256 bits).

Each chromosomes is stored in the SRAM (8 bits × 1024 words), so thirty-two words are needed for each chromosome (8 bits × 32 words = 256 bits), as shown in Fig. 3-17.

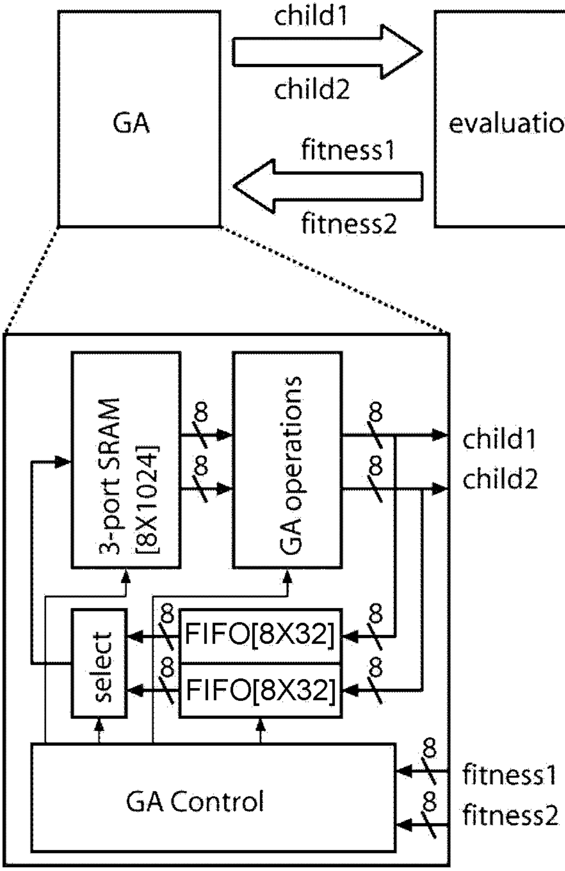


Figure 3-15. An overall architecture and a block diagram of the GA hardware engine

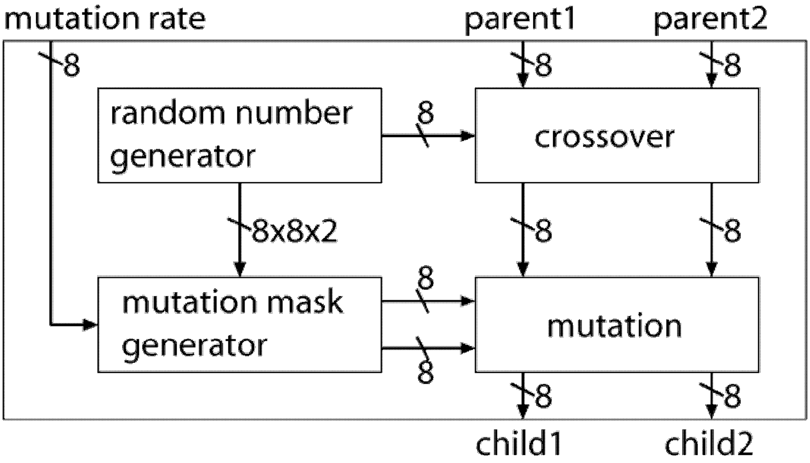


Figure 3-16. Block diagram of the circuit for GA operations

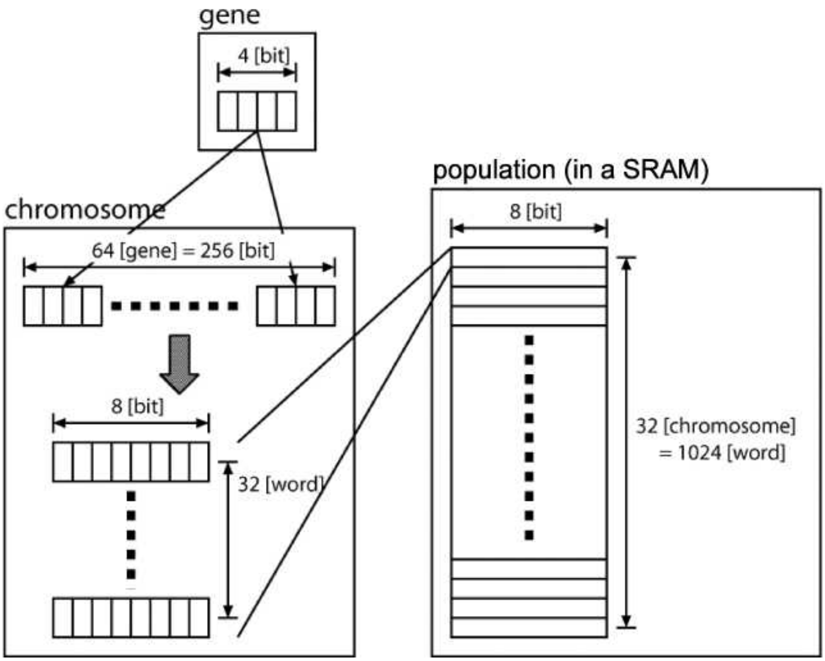


Figure 3-17. Details of genes, chromosomes and population

4.3.3 GA Operations

Based on the simulation results, uniform crossover and *Gaussian* random mutation were selected to be implemented on the hardware.

Uniform crossover is executed with a random bit string consisting of 8 bits. When a location in this random bit string has a value of '1', information is exchanged between the corresponding locations in the two chromosomes undergoing crossover. However, in this application, as each four-bit segment of a chromosome specifies a delay circuit, crossover does not split four-bit chromosome segments, as shown in Fig. 3-18.

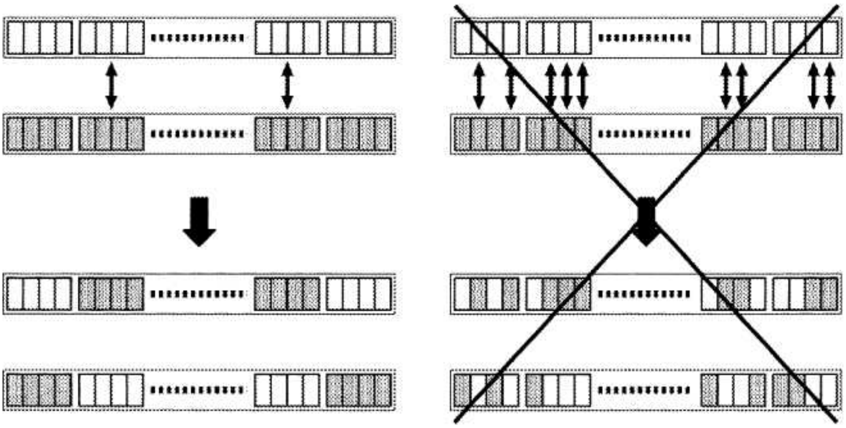


Figure 3-18. Uniform crossover that does not split four-bit chromosome segments

The circuits for *Gaussian* random mutation are divided into two blocks: a mutation mask generator (*Gaussian* random number generator) and a mutation circuit.

The mutation circuit performs the *Gaussian* mutation by adding a mutation mask (a *Gaussian* random number). In this application, the distribution range of the *Gaussian* random numbers is set from –4 to +3 (three-bit signed binary numbers), and these can be generated by the summation of eight uniformly distributed random bit-strings (fixed-point random numbers).

4.3.4 Implemented Hardware

In this chapter, the designed hardware was implemented on an FPGA, in order to validate its functionality. The hardware was designed using a Verilog-HDL simulator (Verilog-XL) and a FPGA development tool (Quartus II; Altera). The specifications of the implemented hardware are as follows.

The FPGA board used in this implementation is the "Nios Development Kit" (Altera) with the Stratix "EP1S10F780C6" device. The size of the implemented circuit is 5,082 logic elements (LE) and 69,504 memory bits. Its maximum operating clock speed is estimated to be 97.32 MHz.

5. CONCLUSION

This chapter described the GA hardware engine and its applications to a controller for a hand-prosthesis and a LSI (Large Scale Integration) fabrication process. In the hand application, the GA hardware enabled the adaptable circuit synthesis for the hand controller circuit. For the LSI fabrication process, the GA hardware engine was applied to the post-fabrication clock-timing adjustment, which lead to high yield rate of the fabricated LSI chip.

References

Atkins, D. J., D. C. Y. Heard and W. H. Donovan. 1996. "Epidemiologic Overview of Individuals with Upper-Limb Loss and Their Reported Research Priorities", *Journal of Prosthetics and Orthotics*, Vol. 8, No. 1, 2-11.

Englehart, K., B. Hudgins and P. A. Parker. 2000. Time-Frequency Based Classification of the Myoelectric Signal: Static vs. Dynamic Contractions, *Proceedings of the 22nd Annual International Conference of the IEEE Engineering in Med. and Bio. Society*.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

Higuchi, T. and N. Kajihara. 1999. "Evolvable Hardware Chips for Industrial Applications", *Communications of the ACM*, Vol. 42, No. 4, 60-66.

Horn, G. W. 1963. "Electromyographic signal produced by muscle movement controls grasp of prosthetic fingers", *Electronics*, October, 34-36.

Hudgins, B., P. Parker and R. N. Scott. 1993. "A New Strategy for Multifunction Myoelectric Control", *IEEE Transactions on Biomedical Engineering*, Vol. 40, No. 1.

Kajitani, I., et al. 2003. "An evolvable hardware chip for a prosthetic-hand controller. –New reconfigurable hardware paradigm.–", *IEICE Trans. INF.&SYST.*, Vol. E86-D, No. 5, 882-890.

Kajitani, I., et al. 2003. A GA hardware engine for post-fabrication clock-timing adjustment, *Proceedings of the Midwest Symposium on Circuits and Systems*.

Kajitani, I., et al. 2001. Improvements to the Action Decision Rate for a Multi-Function Prosthetic Hand, *The First International Symposium on Measurement, Analysis and Modeling of Human Functions (Proceedings of ISHF2001)*, 84-89.

Kajitani, I., N. Otsu and T. Higuchi. 2003. Improvements in Myoelectric Pattern Classification Rate with μ-law Quantization, *Proceedings of the XVII IMEKO World Congress*, Vol. 2, 2032-2035.

Smith, B. 1957. Instantaneous Companding of Quantized Signals, *The Bell System Technical Journal*, May, 1957.

Syswerda, G. 1989. Uniform crossover in genetic algorithms, *Proceedings of International Conference on Genetic Algorithms*, 2–9.

Takahashi, E., et al. 1999. An Evolvable-hardware-based Clock Timing Architecture towards GigaHz Digital Systems, *Proceedings of the Genetic and Evolutionary Computation Conference*.

Takahashi, E., et al. 2003. A Post-Silicon Clock Timing Adjustment Using Genetic Algorithms, *Proceedings of Symposium on VLSI Circuits*.

Thierens, D. 1997. Selection Schemes, Elitist Recombination and Selection Intensity, *Proceedings of International Conference on Genetic Algorithms*, 152-159.

Chapter 4

POST-FABRICATION CLOCK-TIMING ADJUSTMENT USING GENETIC ALGORITHMS

Eiichi Takahashi, Yuji Kasai, Masahiro Murakawa, and Tetsuya Higuchi
Advanced Semiconductor Research Center, National Institute of Advanced Industrial Science and Technology, Email: e.takahashi@aist.go.jp

Abstract: To solve the problem of fluctuations in clock timing (also known as "clock skew" problems), we propose an approach for the implementation of post-fabrication clock-timing adjustment utilizing genetic algorithms (GA). This approach is realized by the combination of dedicated adjustable circuitry and adjustment software, with the values for multiple programmable delay circuits inserted into the clock lines being determined by the adjustment software after fabrication. The proposed approach has three advantages: (1) enhancement in clock frequencies leading to improved operational yields, (2) lower power supply voltages, while maintaining operational yield, and (3) reductions in design times. Two different LSIs have been developed; the first is a programmable delay circuit, developed as an element of the clock-timing adjustment, while the second is a medium-scale circuit, developed to evaluate these advantages in a real chip. Experiments with these two LSIs, as well as a design experiment, have demonstrated these advantages with an enhancement in clock frequency of 25% (max), a reduction in the power-supply voltage of 33%, and a 21% shorter design time. Furthermore, we also propose an adjustment algorithm that takes into account the ensured timing margins to cope with fluctuations in power supply voltage and clock frequency.

Key words: clock-timing adjustment, post-fabrication adjustment, genetic algorithm, improved operational yield, clock enhancement, lower power-supply voltage, reduced design times, reduced power dissipation.

1. INTRODUCTION

LSI devices are increasingly implemented with finer patterns (below 100 nm) and operating at fast GHz clocks, which makes the problem of fluctuations in clock timing (also known as the "clock skew" problem (Rabaey, 2003)) even more crucial. In order to solve the problems associated with clock timing, we proposed an approach to Genetic Algorithm (GA)-based clock adjustment in 1999 (Takahashi, 1999). Although that report was based on simulation results, this chapter presents evaluation data for test chips. In 2002, the application of the same basic idea was also reported in the design of the 3 GHz Pentium (R) 4 micro processor (Deleganes, 2002).

Many approaches to the problem of clock timing tend to focus on the elimination of all timing variations, known as clock skew, at the physical design stage (Dike, 2003; Geannopoulos, 1998, Kurd, 2001; Roth, 2003; Tam, 2000). For example, a common approach is to use an H-tree clock network to balance all the clock paths from the clock source to all the flip-flops (Rabaey, 2003). Additional circuits are also included to reduce clock skew by making comparisons between distributed clock and reference clock and align these using a DLL (Delay Locked Loop). In contrast, the GA-based clock adjustment approach recognizes that such variations are an inevitable result of the fabrication process and, rather than attempting to remove them, actually manipulates the variations in order to enhance clock frequency and achieve improvements in operational yield (Higuchi, 2003; Takahashi, 2003).

This GA-based clock adjustment method is achieved through the combination of dedicated *circuitry* and *software*. Multiple programmable delay circuits are inserted into the clock lines and, after fabrication, GA software implemented on an LSI tester determines the delay value for each circuit, based on error counts in function tests, in order to adjust the clock timing, as shown in Fig. 4-1 and as follows:

1. Programmable delay circuits are inserted into a standard LSI, designed by normal methods.
2. LSI chips are fabricated according to standard procedures.
3. When the chips are tested, the values of the programmable delay circuits are determined by the GA-based adjustment software, which is executed on the LSI tester.
4. Shipped after adjustment, not only is the operational yield increased, but the chips operate with lower levels of power dissipation and at fast clock speeds that exceed design specifications.

This approach is just one example of our concept of "post-fabrication adjustment"; another example applied to an IF analog filter has been described (Murakawa, 2003).

Figure 4-2 depicts the hierarchical application of the GA-based clock-timing adjustment. The adjustment method handles both INTER-DOMAIN clock skew and INTRA-DOMAIN clock skew. For example, multiple programmable delay circuits (solid marks and hexagons in Fig. 4-2) are inserted at the clock inputs for each clock domain and into the critical paths within domains where clock timing is extremely severe. This hierarchical

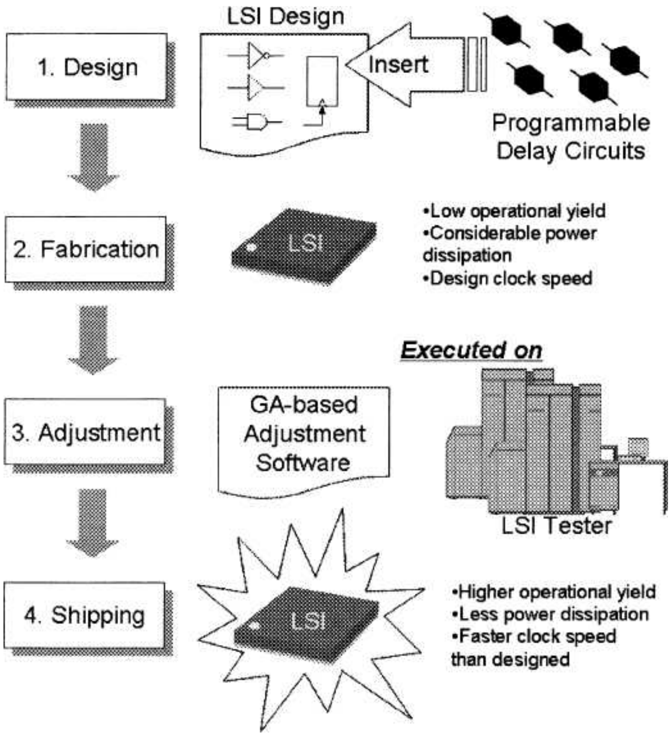


Figure 4-1. Adjustment with LSI testers after fabrication

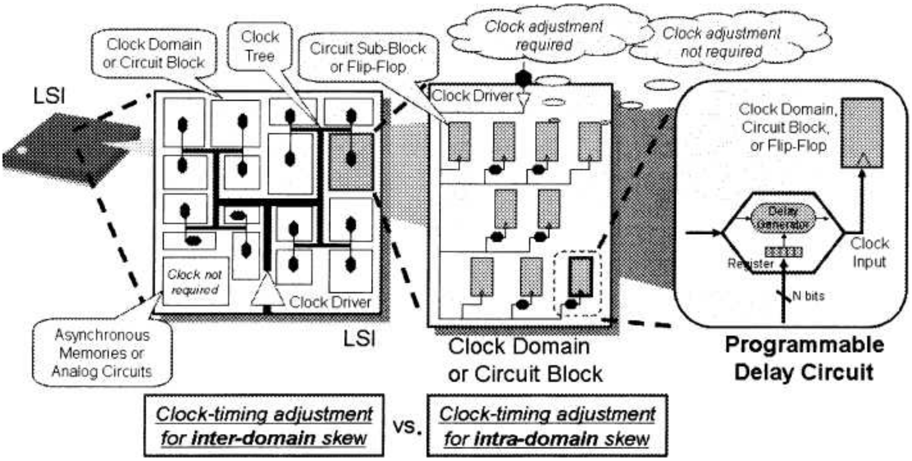

Clock Domain or Circuit Block
Clock Tree
Circuit Sub-Block or Flip-Flop
Clock adjustment required
Clock adjustment not required
LSI
Clock Driver
Clock Domain, Circuit Block, or Flip-Flop
Delay Generator
Register
Clock Input
N bits
Clock not required
Clock Driver
Asynchronous Memories or Analog Circuits
LSI
Clock Domain or Circuit Block
Programmable Delay Circuit
Clock-timing adjustment for inter-domain skew
vs.
Clock-timing adjustment for intra-domain skew

Figure 4-2. Clock-timing adjustment for both inter- and intra-domains

approach differs from the application of the adjustment method reported for the Pentium (R) 4 (Deleganes, 2002), which appears to be limited only to inter-domains.

Although each LSI chip has numerous delay circuits, the GA is able to determine the optimal delay values for all circuits. GAs are robust optimization algorithms in artificial intelligence, which can efficiently and quickly find optimal solutions from vast numbers of candidates, according to an evaluation function tailored to a particular application (Goldberg, 1989; Higuchi, 1999; Holland, 1975). An important advantage of the GA is that, unlike some search algorithms such as the steepest descent method, it does not require derivatives for the evaluation function, making it especially useful in applications like the clock timing adjustment method where derivatives are not easily available. In this work, the evaluation function is designed to find the delay settings for correct operation with higher clock frequencies and lower power-supply voltages. Parameters in a GA are represented as binary bit strings called chromosomes. In the test chip, the chromosomes are divided and held in separate registers corresponding to individual delay circuits. These chromosomes are repeatedly updated by the GA until the optimal settings for all the parameters are obtained, so that the chip correctly operates at a sufficiently fast clock and low power-supply voltage. This chapter reports a 25% (max) enhancement in clock frequency with this approach in the experiments detailed below. In addition to clock-frequency enhancements, this chapter highlights two further advantages of our approach; namely, lower power-supply voltages (V_{dd}) while maintaining operational yields and reduced design times, which are demonstrated in three LSI designs.

The achievement of a lower V_{dd} through our GA-based clock adjustment technique is illustrated in Fig. 4-3. Although a lower V_{dd} would normally impose serious timing constraints, the GA-based clock adjustment method can handle these constraints and thus maintain operational yields:

1. Before adjustment, the chips operate at the power-supply voltage specified at design.
2. If the power-supply voltage is decreased, the ratio of operational chips would fall.
3. The GA-based adjustment maintains the ratio of operational chips even at a lower power-supply.

In the experiments detailed below, the approach reduced the V_{dd} by 33%, resulting in a 54% power saving.

Moreover, our GA-based clock adjustment technique can reduce design times. Applying this approach to the design of a DDR-SDRAM controller, it was possible to achieve a 21% reduction in design time.

In the rest of this chapter, Section 2 outlines the two test chips and the experimental system. Sections 3 considers the three advantages of the GA-based clock adjustment with experimental results. Section 4 proposes the adjustment algorithm for ensured timing margins and the simulation results of the algorithm. Finally, Section 5 summarizes the results of the experiments and also discusses future directions for this concept.

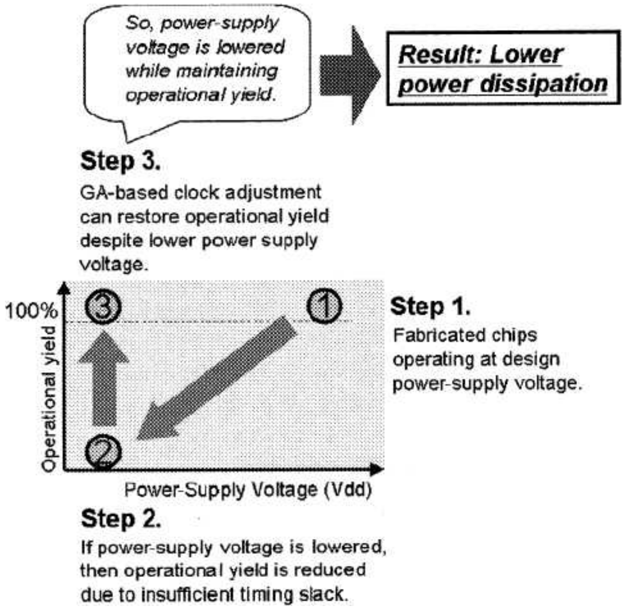
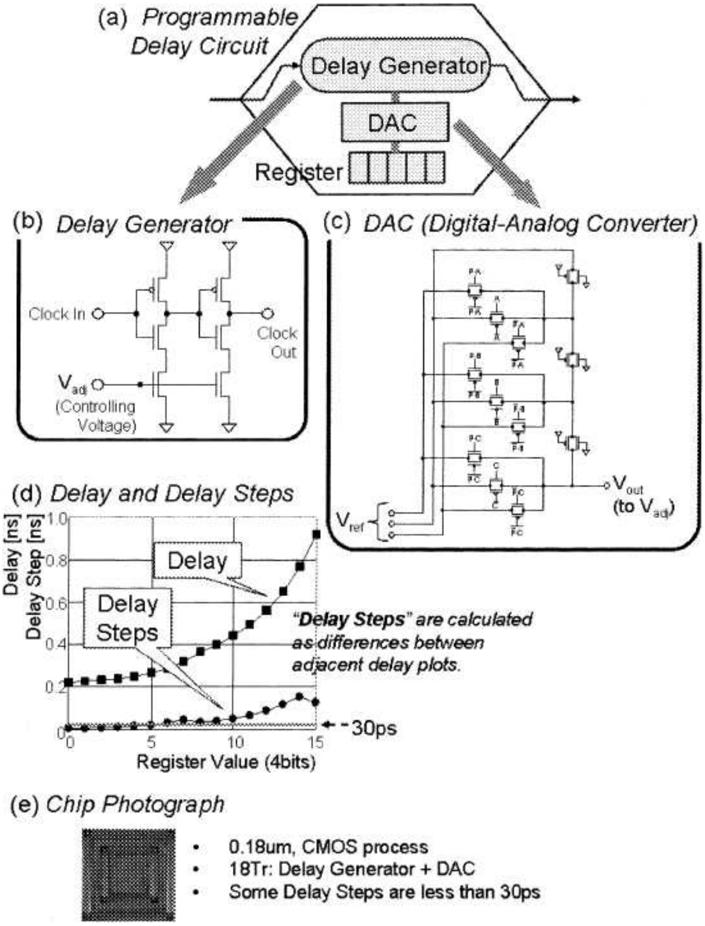

So, power-supply voltage is lowered while maintaining operational yield.
Result: Lower power dissipation
Step 3.
GA-based clock adjustment can restore operational yield despite lower power supply voltage.
100%
③
①
Step 1.
Fabricated chips operating at design power-supply voltage.
Operational yield
②
Power-Supply Voltage (Vdd)
Step 2.
If power-supply voltage is lowered, then operational yield is reduced due to insufficient timing slack.

Figure 4-3. Achievement of lower V_{dd} s

2. TWO TEST CHIPS AND EXPERIMENTAL SYSTEM

In this section, we outline the two test chips and the experimental system. We have developed two different LSI chips. The first is the programmable delay circuit, which constitutes a basic circuit component in our GA-based adjustment approach. The second LSI chip is a medium-scale circuit, consisting of two types of circuits, i.e. a multiplier and memory-test-pattern generator, which has been developed to evaluate the GA-based adjustment approach.

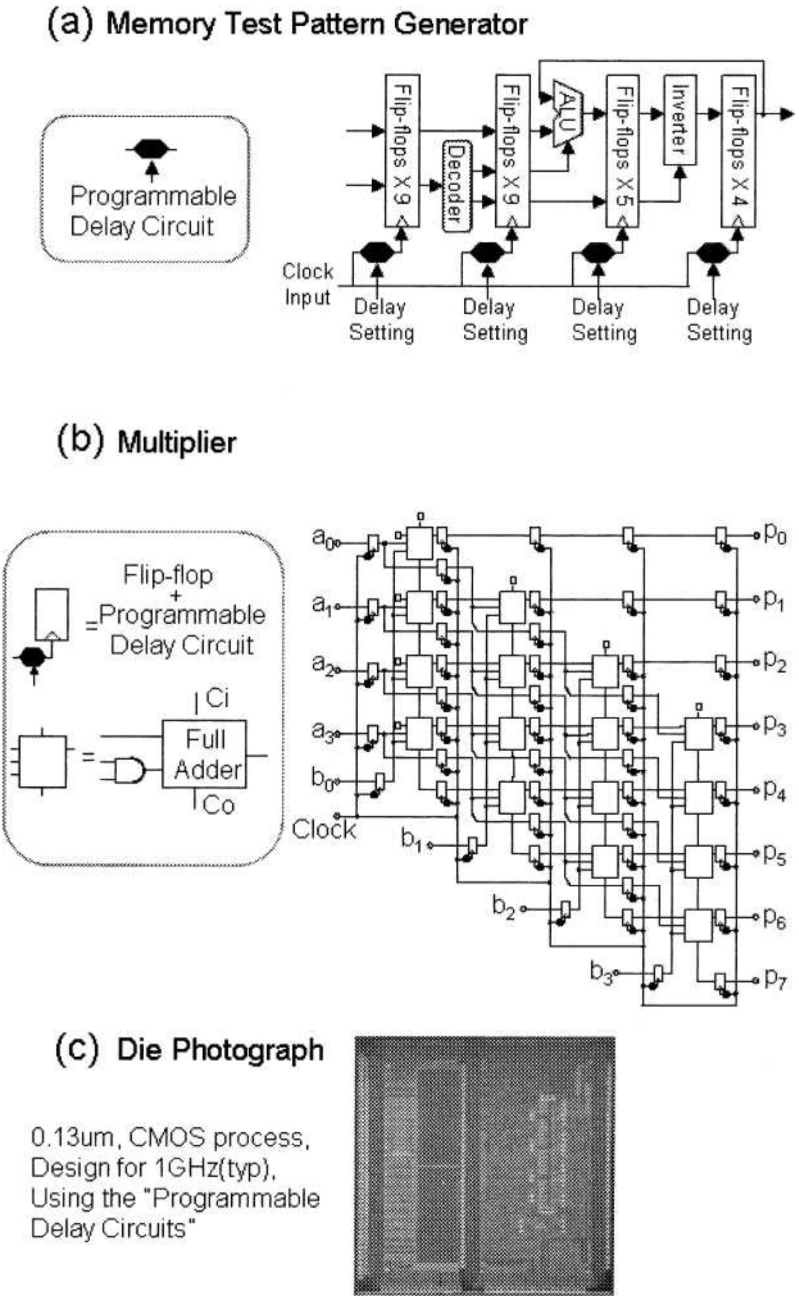
First, we describe the programmable delay circuit developed for GA-based clock adjustment, as shown in Fig. 4-4. The programmable delay circuit consists of a delay generator and a register to store delay values, although the present study focuses on the delay generator, which is small and capable of generating fine delay steps (Fig. 4-4(a)). The delay generator is a voltage-controlled delay circuit that utilizes the channel resistance (Fig. 4-4(b)). The delay generator and digital-analog converter (Fig. 4-4(c)) consist of 30 transistors, which are one-eighth in size compared to the logic gate-based implementation we developed for a previous chip (Takahashi, 1999). Making them applicable to high-speed chips operating at speeds over 1 GHz, the programmable delay circuits are also capable of generating precise delay increments shorter than 30ps (Fig. 4-4(d)). As one cycle of a 1 GHz clock is 1000ps, a delay of 30ps represents 3% of a cycle. As our adjustment method has 16 delay steps, it is capable of adjusting for clock variation of up to 48%, which is sufficient in the vast majority of cases. Fig. 4-4(e) is a photograph of the chip.



"Delay Steps" are calculated as differences between adjacent delay plots.

- 0.18um, CMOS process
- 18Tr: Delay Generator + DAC
- Some Delay Steps are less than 30ps

Figure 4-4. Test chip 1 (programmable delay circuit)


(a) Memory Test Pattern Generator

(b) Multiplier

(c) Die Photograph

Figure 4-5. Test chip 2 (multipliers and memory-test-pattern generators)

Next, we describe the second chip of memory-test-pattern generators and multipliers, as shown in Fig. 4-5(a) and (b), which is implemented with the programmable delay circuits. Note that this figure represents a specific implementation of the conceptual schematic presented in Fig. 4-2. The programmable delay circuits are inserted into the clock inputs of all the flip-flops in the multipliers and memory-test-pattern generators.

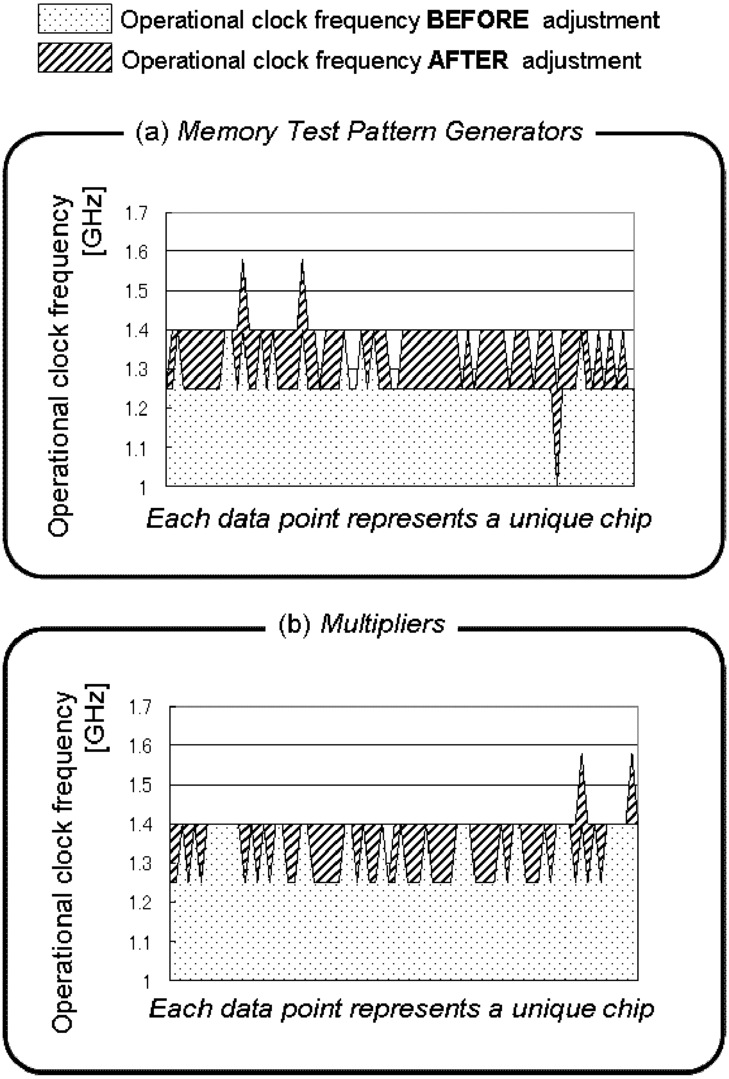
The chip used in our adjustment experiment consisted of four multipliers and four memory-test-pattern generators. Both circuits have a pipeline-structure, with the programmable delay circuits being inserted into the clock inputs of the flip-flops forming interstage registers. Fifty-two programmable delay circuits are inserted into each multiplier and 40 circuits are inserted into each memory-test-pattern generator, although the additional area required for the effective delay circuits was only 3% and 5% for the respective test circuits. Increases in power dissipation due to the incorporation of the programmable delay circuits are estimated to be similar, at approximately 3% and 5% respectively, to the increases in area. In our experiments, the test chip was connected to a PC, which executed the GA software and collected the evaluation data. Twenty chips from 2 wafers were used in the experiments; therefore, eighty multipliers and eighty memory-test-pattern generators were tested. Figure 4-5(c) is a photograph of the test chip.

3. EXPERIMENTAL RESULTS

This section describes the three advantages of our approach: clock frequency enhancements, power-supply voltage reductions, and design time reductions.

3.1 Clock Frequency Enhancements

This subsection describes the first of the three advantages: clock frequency enhancement. Figure 4-6, consisting of two graphs and a table, presents the results of the clock frequency enhancement experiment for the memory-test-pattern generators (Fig. 4-6(a)) and multipliers (Fig. 4-6(b)) chip, showing the improvement ratios for both types of circuits. In the graphs, each data point on the *X*-axis represents a unique chip with the *Y*-axis representing clock frequency. The light shaded area in the graphs shows clock frequencies at which the chips operate correctly before adjustment, while the dark striped area shows the increase in operational clock frequency after adjustment.



(a) *Memory Test Pattern Generators*

(b) *Multipliers*

Figure 4-6. Clock frequency enhancements for the two types of circuits

The experiment with the memory-test-pattern generators shows an 11% average enhancement and a maximum enhancement of 25% (from 1.0GHz to 1.25GHz).

Turning next to look at the improvements in operational yield, if $n(f)$ is taken as the number of chips that operate correctly at a given clock frequency f, and N as the total number of corresponding chips, then the operational yield $y(f)$ of chips at the clock frequency f can be defined as:

$$y(f) = \frac{n(f)}{N}. \tag{1}$$

Figure 4-7 presents the results of this experiment in a different way. In the Fig. 4-7 graphs, the *X*-axis represents clock frequency and the *Y*-axis represents operational yield. Although the operational ratio was only 15% for the memory-test-pattern generators at a frequency of 1.4 GHz before adjustment, the ratio rose to 90% after adjustment. Moreover, while no chips could operate at a frequency of 1.58 GHz before adjustment, a few could after adjustment. Our GA-based adjustment technique is able to achieve this kind of clock enhancement through the automatic incorporation of useful skew and slack borrowing at all flip-flops where the programmable delay circuits are inserted (Takahashi, 1999).

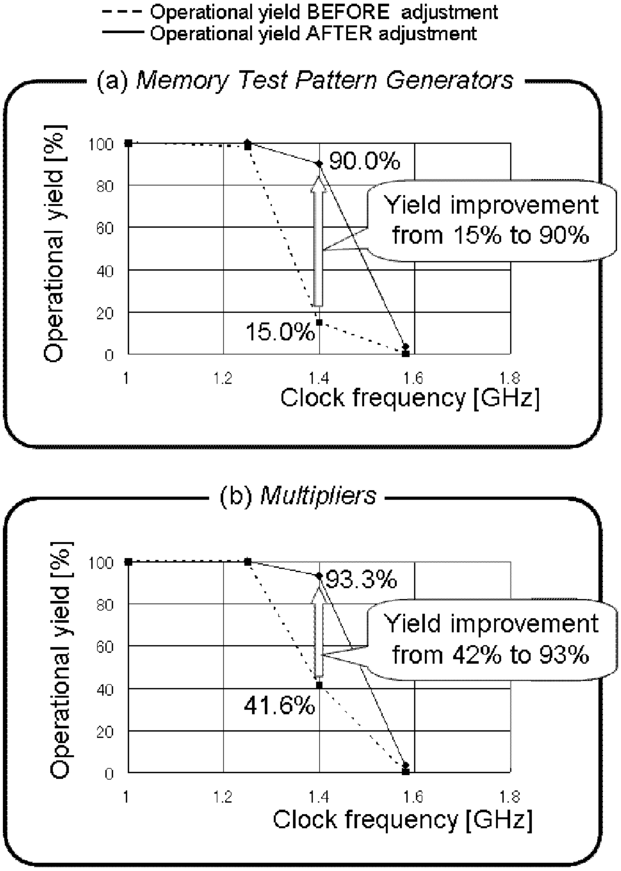
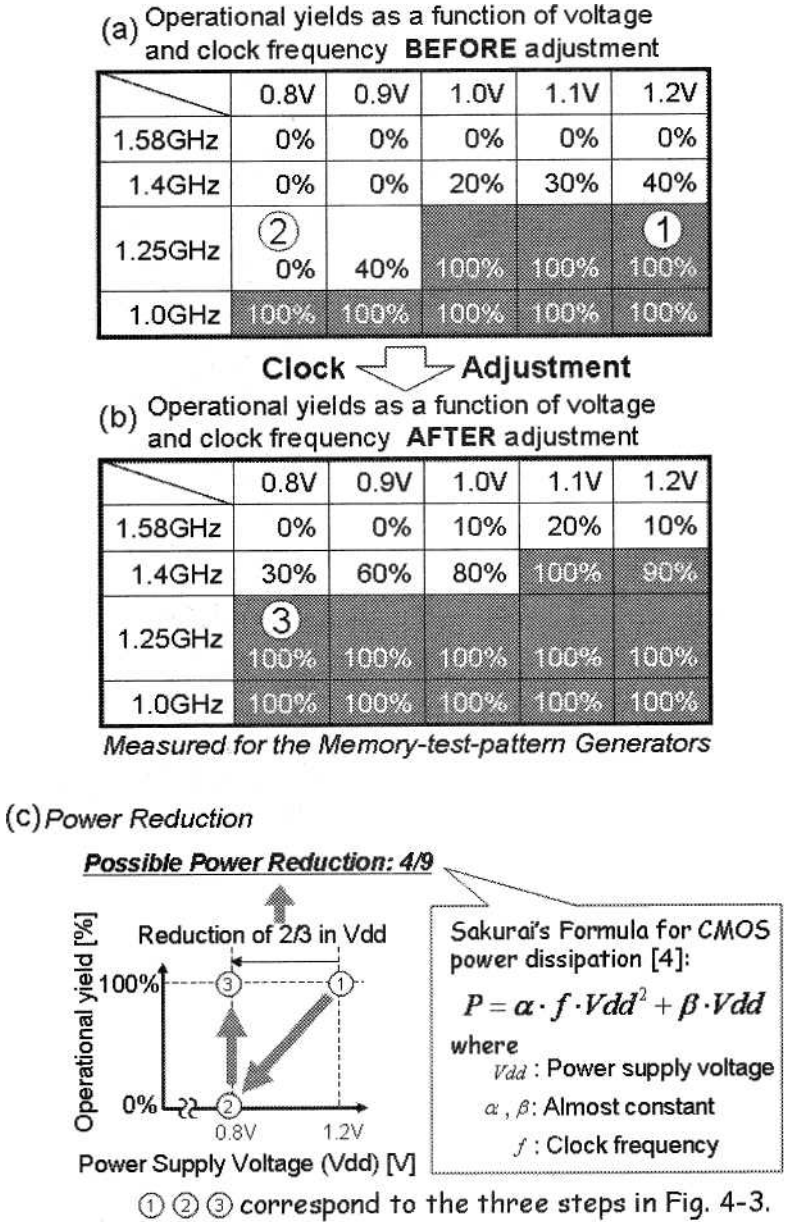


Figure 4-7. Operational yield improvements for the two types of circuits



(a) Operational yields as a function of voltage and clock frequency **BEFORE** adjustment

	0.8V	0.9V	1.0V	1.1V	1.2V
1.58GHz	0%	0%	0%	0%	0%
1.4GHz	0%	0%	20%	30%	40%
1.25GHz	② 0%	40%	100%	100%	① 100%
1.0GHz	100%	100%	100%	100%	100%

Clock Adjustment

(b) Operational yields as a function of voltage and clock frequency **AFTER** adjustment

	0.8V	0.9V	1.0V	1.1V	1.2V
1.58GHz	0%	0%	10%	20%	10%
1.4GHz	30%	60%	80%	100%	90%
1.25GHz	③ 100%	100%	100%	100%	100%
1.0GHz	100%	100%	100%	100%	100%

Measured for the Memory-test-pattern Generators

(c) *Power Reduction*

Possible Power Reduction: 4/9

Sakurai's Formula for CMOS power dissipation [4]:

$$P = \alpha \cdot f \cdot Vdd^2 + \beta \cdot Vdd$$

where

Vdd : Power supply voltage

α , β: Almost constant

f : Clock frequency

① ② ③ correspond to the three steps in Fig. 4-3.

Figure 4-8. Estimation of clock-timing adjustment execution time

In these experiments, adjustment for each chip took 9 minutes, but a careful analysis of this shows that almost all of the total time was required for communication between the PC and the interface board, which is not necessary with LSI testers. Figure 4-8(a) shows the estimated time for each operation, after excluding time for communications between PC and the interface board, and indicates that adjustment on LSI testers can be completed in less than 1 second.

Figure 4-8(b) presents the behavior of the GA search for a circuit. In the circuit, the delay settings for correct operation were found after fourteen iterations. Fig. 4-8(c) presents details of the GA.

3.2 Maintaining Operational Yield with a Lower V_{dd}

The second advantage is explained in this subsection. Our GA-based clock adjustment technique also represents an extremely efficient means of lowering the power-supply voltage (V_{dd}) while still maintaining the operational yield. Generally, the operational yield drops when the power supply voltage is lowered, because the propagation delay of logic gates increases (Rabaey, 2003) and timing violations occur at lower V_{dd} s. However, the GA-based method can handle these problems to make the chip operational at a lower voltage, and consequently avoids the drops in operational yield normally associated with lower V_{dd} s.

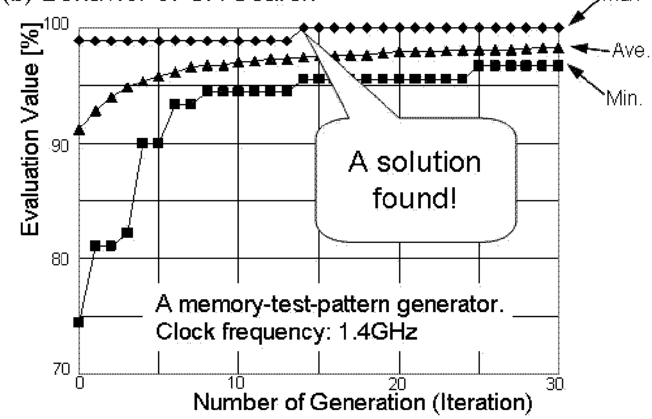
Figure 4-9 presents the results of operational yield as a function of the power-supply voltage for the memory-test-pattern circuits. Figure 4-9 includes two tables of experimental results: the upper table (Fig. 4-9(a)) shows operational yields before adjustment while the lower table (Fig. 4-9(b)) shows yields after adjustment. In both tables, the columns are for power-supply voltages and the rows are for clock frequency. As indicated by the shaded cells and the numbers 1 to 3, although the chips with a clock frequency of 1.25 GHz could all be operated at 1.2V, the operational yield fell to 0% when the V_{dd} was lowered to 0.8V. However, after timing adjustment, the operational yield for the chips was again at 100%. This represents a significant reduction in power dissipation, as shown in Fig. 4-9(c). Sakurai's formula for CMOS power dissipation P [13] is also shown in Fig. 4-9, where α and β are almost constant circuit dependants. In this formula, the term $\alpha \cdot f \cdot V_{dd}^2$ dominates the power dissipation in the Giga-Hertz domain ($f \geq 1$ GHz). Applying this formula to the data, there is a 54% reduction in power dissipation, which suggests that this approach can potentially double the duration of batteries in mobile PCs.

Clock adjustment at lower V_{dd} s also required 9 minutes, but as already explained, when executed on LSI testers the time should be less than 1 second.

(a) Adjustment Time

Operations	Time (s)
Generation of Delay Setting and Test Data for Function Test	0.55
Write Data to Chip	0.02
Execution of Test	0.01
Read Results of Test from Chip	0.03
Calculation of Fitness	0.33
Total Time	**0.94**

(b) Behavior of GA Search



(c) GA Details

Population	50
Termination	20th Generation
Selection	Tournament Selection
Crossover	One-Point Crossover
Crossover Rate	1.0
Mutation	Gaussian Mutation (σ=1.0)
Mutation Rate	1.0

Figure 4-9. Maintaining operational yield at a lower power-supply voltage

3.3 Comparison of Design Times

In this subsection, the impact of our approach on design is explained. Our GA-based clock adjustment method is also effective in reducing total design times. Specifically, the design time required for timing considerations can be dramatically reduced, because the GA-based method can adjust for timing after fabrication leading to better operational yields. In order to verify this advantage, we conducted a design experiment in which a DDR-SDRAM controller (DDR266 compliant, 5,400 gates) was designed by both a traditional (margin-based) approach and the GA-based approach, so that total design times could be compared. Both approaches used commercially available EDA tools, such as Verilog-XL(R), Design Compiler(R), Prime Time(R), Apollo(R), and Virtuoso(R).

Table 4-1 shows the results, with the times for the total design flow analyzed into the four main design steps, and indicates that the GA-based approach could reduce the overall time by 21% (23.0 day × person). Looking in more detail at the separate design stages, the GA-based approach is able to achieve design time reductions in the following areas: at the function design stage, the modeling of timing constraints is simplified; at the floor-planning stage, the time to assign pins is reduced; and at the layout design stage and verification stages, the iterations required to satisfy timing constraints are reduced.

In the case of more complex LSIs, the improvements obtainable with the GA-based approach will be even more apparent, because timing design aspects become more complicated and time-consuming.

Table 4-1. Comparison of design times with a traditional approach and the GA-based approach

Design Stage	Traditional*1	GA-based*1
Function Design	12.0	**1.5**
Logic Design	30.0	30.0
Floor Planning	7.0	**2.0**
Verification (1)	5.0	5.0
Layout Design	7.0	**1.5**
Verification (2)	6.0	**4.0**
Library Design	42.0	42.0
Total	109.0	86.0

*1 day × person

4. ADJUSTMENT METHOD FOR ENSURED TIMING MARGINS

While the GA could successfully adjust the clock timing of the developed chips, some of the adjusted chips were found to operate at lower levels of accuracy. This is because the clock timings were adjusted to the very margins of feasible timings to pass all function tests. However, if the environmental temperature or power-supply voltage (V_{dd}) were to fluctuate even slightly, then the clocks would deviate from adjusted timings. To overcome this difficulty, we propose raising the operational frequencies in the GA adjustment step. Figure 4-10 shows a conceptual schema of the proposed adjustment method:

1. Programmable delay circuits are inserted into a standard LSI, designed for the operational frequency f_0.
2. LSI chips are fabricated according to standard procedures.
3. The values of the programmable delay circuits are determined by the GA-based adjustment software, which is executed on the LSI tester. In this adjustment step, the operational frequency is raised to $f_2(> f_0)$ to impose more severe timing constraints on the chip.
4. The adjusted chip undergoes function tests at the operational frequency of $f_1(f_0 \leq f_1 < f_2)$.
5. If the chip passes the tests, then the chip can be shipped. If not, the chip is discarded. Shipped chips can operate at the frequency of f_1, while still having sufficient margins to operate at the frequency of f_2.

Through these adjustment steps, the operational frequency can be raised, ensuring that the clock timing is sufficiently robust to cope with fluctuations in the LSI environment.

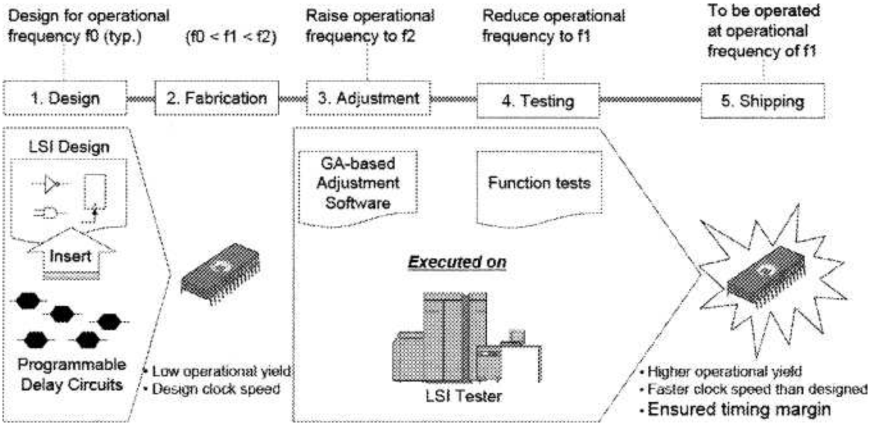

Design for operational frequency f0 (typ.)
(f0 < f1 < f2)
Raise operational frequency to f2
Reduce operational frequency to f1
To be operated at operational frequency of f1
1. Design
2. Fabrication
3. Adjustment
4. Testing
5. Shipping
LSI Design
Insert
Programmable Delay Circuits
• Low operational yield
• Design clock speed
GA-based Adjustment Software
Function tests
Executed on
LSI Tester
• Higher operational yield
• Faster clock speed than designed
• Ensured timing margin

Figure 4-10. Clock-timing adjustment ensuring timing margins

4.1 Experimental Results

Two adjustment experiment studies were conducted using simulators for the developed memory-test-pattern generator and multiplier. The first simulation was carried out to determine the timing margins present with chips adjusted by our previous adjustment method. The second simulation was carried out to demonstrate that the improved method is capable of increasing operational yields while maintaining sufficient timing margins. In the adjustment experiments, the number of chip samples was 1000, the population size was set to 100, and the termination generation was set to 100. The chromosome lengths were 176 (=4bits×44 delay circuits) for the memory-test-pattern generator and 208 (=4bits×52 delay circuits) for the multiplier, respectively.

Tables 4-2 and 4-3 present simulation results showing the timing margins of chips successfully adjusted with the previous adjustment method. Chips which worked correctly without adjustment do not appear in the tables.

Although we believe that chips to be shipped must have timing margins at more than 10% of the operational frequency in order to cope with fluctuations in the chip environment, as can be seen from Table 4-3, none of the multipliers adjusted by the previous method had timing margins at this level. Also, most of the adjusted memory-test-pattern generators had insufficient timing margins as can be seen from Table 4-2. Accordingly, the level of precision for most of the adjusted chips would fall if the environmental temperature or the Vdd were to fluctuate, even slightly.

Table 4-2. Timing margins of memory-test-pattern generators successfully adjusted with the previous adjustment method

Freq.	Unsuccessful	Successful adjustment with timing margins		
	Adjustment	0 – 5%	5 – 10%	10% –
1.2 GHz	0	216	111	42
1.4 GHz	1	861	126	12
1.6 GHz	150	804	46	0
1.8 GHz	961	39	0	0
2.0 GHz	1000	0	0	0

Table 4-3. Timing margins of multipliers successfully adjusted with the previous adjustment method

Freq.	Unsuccessful	Successful adjustment with timing margins		
	Adjustment	0 – 3%	3 – 6%	6% –
1.2 GHz	0	120	15	0
1.3 GHz	2	829	82	0
1.4 GHz	461	509	30	0
1.5 GHz	991	8	1	0
1.6 GHz	1000	0	0	0

4.2 Operational Yield Enhancements with Timing Margins

Figures 4-11 and 4-12 present simulation results showing the operational yield rates for the memory-test-pattern generator and multiplier, respectively. In the graphs, the X-axis represents clock frequency f_1 and the Y-axis represents the operational yield (the ratio of chips that operate correctly and have more than 10% timing margins.)

Although the operational ratio was 0% for the memory-test-pattern generator at a frequency of 1.2 GHz before adjustment, the ratio rose to 97% after adjustment. Moreover, the adjusted chips have timing margins of 10%, and so can cope with environmental fluctuations. Although no chip for the memory-test-pattern generator could operate correctly with 10% timing margins at a frequency of 1.6 GHz before adjustment, a few chips could after adjustment.

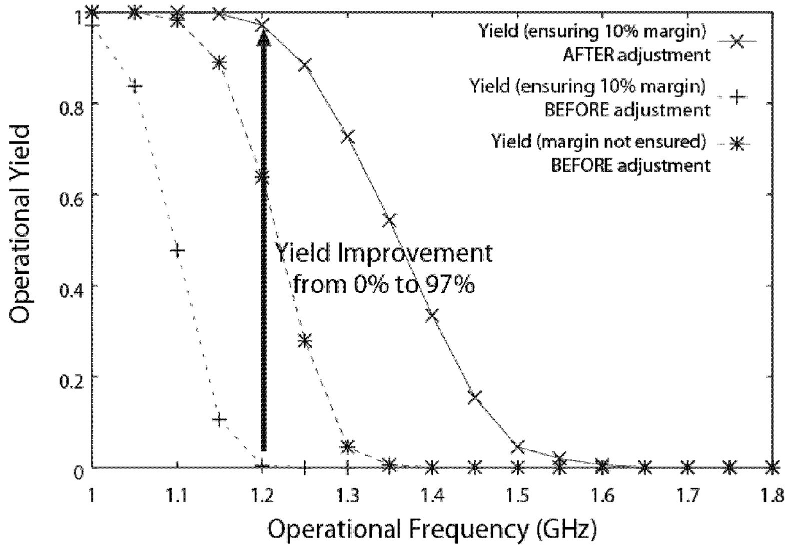


Figure 4-11. Operational yield improvements for the memory-test-pattern generator

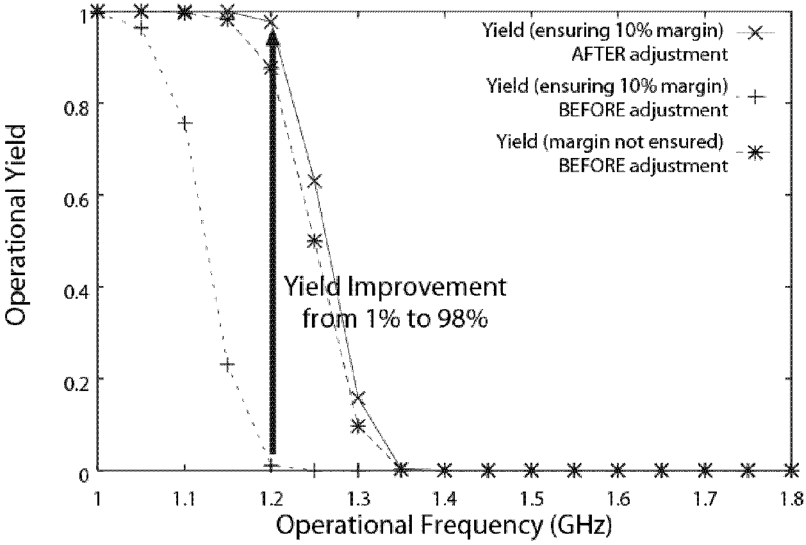


Figure 4-12. Operational yield improvements for the multiplier

5. CONCLUSION

A GA-based clock adjustment architecture has been proposed and tested. The GA-based clock adjustment method has three advantages, namely, enhanced clock frequency (11% average improvement), lower V_{dd} s (2/3 reduction in voltage) while maintaining operational yields, and shorter design times (21% reduction), which we have demonstrated in the experiments reported in this chapter. In particular, it should be noted that both higher clock frequencies and lower power dissipation are realized simultaneously by this approach. All three advantages represent extremely effective means of solving the clock timing problems associated with the increasing utilization of components with finer submicron patterns and with faster and larger chips.

Some may question the practical aspects of this approach, due to concerns about the size of the programmable delay circuits and the adjustment times. However, this study has clearly demonstrated that the developed circuits are sufficiently small and that adjustment times are sufficiently short. Accordingly, these concerns do not constitute real obstacles to the application of this approach for mass production. Although the current programmable delay circuits cannot compensate for temperature and power-supply voltage fluctuations, the next version of the circuits, currently under development, will be able to handle such fluctuations. We plan to detail these modifications and report on conducted experiments at the earliest opportunity.

The GA-based clock adjustment method is one example of the post-fabrication adjustment concept. As our method is not a pure design technique or a pure circuit technique (being rather independent in nature), it can be combined with some of these techniques. For instance, a "soft-edge flip-flop" (Nedovic, 2003) is a circuit technique to compensate for clock skew, which could be more effective in combination with our method.

We have also proposed a clock-timing adjustment method that ensures sufficient timing margins. Simulation results show that the improved GA-based adjustment can enhance yields while maintaining adequate timing margins. Chips adjusted with our proposed method are sufficiently robust to cope with fluctuations in the chip environment.

The only tasks remaining are (1) to incorporate the GA-based clock adjustment technique into EDA tools, such as clock tree synthesis (CTS), (2) to build IPs for the programmable delay circuits for specialized EDA tools, which are both already under development, (3) to improve the adjustment algorithms for ensured timing with developed chips, and (4) to develop embedded GA-based adjustment circuitry to reduce testing costs on the LSI tester. By using EDA tools capable of handling our GA-based technique, together with programmable delay circuits, users will be able to benefit from the three advantages described in this chapter.

Finally, as a practical application of the GA-based clock adjustment method, two LSIs are being developed that include programmable delay circuitry in commercial LSI designs. The purpose of the development is to confirm the effect of our method in power dissipation reduction in commercial LSIs.

Acknowledgments

This work was supported by NEDO (New Energy and Industrial Development Organization), Japan. The authors would like to thank Hitachi ULSI Systems CO., Ltd for the design data in the graphs and tables, circuit design, and the experiments. The authors also thank Prof. Sakurai at the University of Tokyo, and Dr. Hirose and Dr. Masuhara of the MIRAI Project for their critical reading of the manuscript.

References

Deleganes, D., et al. 2002. Designing a 3GHz, 130nm, Pentium (R) 4 Processor. In *Proc. of 2002 Symposium on VLSI Circuits*, 130-133.

Dike, C., et al. 2003. A Design for Digital, Dynamic Clock Deskew. In *Symp. VLSI Circuits Dig.*, 21-24.

Geannopoulos, G. and X. Dai. 1998. An Adaptive Digital Deskewing Circuit for Clock Distribution Networks. In *ISSCC Dig. Tech. Papers*, Feb. 1998, 400-401.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison Wesley.

Higuchi, T., et al. 1999. "Real-World Applications of Analog and Digital Evolvable Hardware," *IEEE Trans. Evolutionary Computation*, vol. 3, 220-235.

Higuchi, T., et al. 2003. "Designers to Raise Yields, Manufacturers to Raise Design Efficiency: Part 2 Circuit Technology," *Nikkei Microdevices*, No. 219, 38-45, Sep. 2003 (in Japanese).

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: Univ. of Michigan Press, 1975.

Kurd, N., et al. 2001. "A Multigigahertz Clocking Scheme for the Pentium 4 Microprocesor," *IEEE J. Solid-State Circuits*, vol. 36, 1647-1653, Nov. 2001.

Murakawa, M., et al. 2003. "An AI-Calibrated IF Filter: A Yield Enhancement Method With Area and Power Dissipation Reductions," *IEEE J. Solid-State Circuits*, vol. 38, 495-502, Mar. 2003.

Nedovic, N., V. Oklobdzija and W. Walker. 2003. A Clock Skew Absorbing Flip-Flop. In *ISSCC Dig. Tech. Papers*, Feb. 2003, 342-343.

Rabaey, J., A. Chandrakasan and B. Nikolic. 2003. *Digital Integrated Circuits 2nd Ed.* New Jersey: Prentice Hall, 2003.

Roth, E., et al. 2003. A Delay-line Based DCO for Multimedia Applications Using Digital Standard Cells Only. In *ISSCC Dig. Tech. Papers*, Feb. 2003, 442-443.

Sakurai, T. 1999. "LSI design toward 2010 low-power technology," *Int. Conf. VLSI & CAD '99*, 325-334.

Takahashi, E., et al. 1999. An Evolvable-Hardware-based Clock Architecture towards GigaHz Digital Systems. In *Proc. of AAAI GECCO'99 (Genetic Algorithm and Evolutionary Computation Conference)*, 1204-1210, July 1999.

Takahashi, E., et al. 2003. A Post-Silicon Clock Timing Adjustment Using Genetic Algorithms. In *Symp. VLSI Circuits Dig.*, 2003, 13-16.

Tam, S., et al. 2000. "Clock Generation and Distribution for the First IA-64 Microprocessor," *IEEE J. Solid-State Circuits*, vol. 35, 1545-1552, Nov. 2000.

Chapter 5

BIO-INSPIRED COMPUTING MACHINES WITH ARTIFICIAL DIVISION AND DIFFERENTIATION

Daniel Mange, André Stauffer, Gianluca Tempesti, Fabien Vannel, and André Badertscher

Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

{daniel.mange,andre.stauffer,gianluca.tempesti,fabien.vannel,andre.badertscher}@epfl.ch

Abstract: In order to design computing machines able to self-repair and self-replicate, we have borrowed from nature two major mechanisms which are embedded in silicon: cell division and cell differentiation. Based on the so-called Tom Thumb algorithm, cellular division leads to a novel self-replicating loop endowed with universal construction. The self-replication of the totipotent cell of the "LSL" acronym serves as an artificial cell division example of the loop and results in the growth and differentiation of a multicellular organism.

Key words: embryonics, self-replication, cell division, cell differentiation.

1. INTRODUCTION

Referring to a standard model for describing bio-inspired computing, the POE model (for Phylogeny-Ontogeny-Epigenesis) (Sipper, 1997), one may observe that countless results have been obtained along the Phylogeny and the Epigenesis axes: evolutionary algorithms and genetic programming on one hand, artificial neural and immune systems on the other hand. The third axis of the POE model, the Ontogeny axis, has been strangely neglected for a long period, perhaps due to its close relationship to hardware, less popular and more difficult to master than software.

The relatively recent emergence of a new kind of computing hardware, the so-called field-programmable gate array (FPGA) (Villasenor, 1997), announced perhaps the dawn of a new era of computing science, when the hardware, until now untouchable, is itself becoming programmable or "configurable". The configuration information is a string of bits, aimed at specializing a piece of FPGA in order to implement a given specification; this configuration is very similar to the DNA string, the genetic information needed by a living organism to grow, live, self-repair and self-reproduce. It was therefore quite natural to try to embed in an FPGA-based architecture the basic mechanisms of the ontogeny of living organisms, i.e. cellular division and cellular differentiation.

The final goal of our Embryonics (embryonic electronics) project (Mange, 2000) is to obtain artificial organisms in silicon able to grow, self-repair and self-replicate; these properties are highly desirable for "extreme" applications of computer engineering (space exploration, radioactive environments, avionics, etc.) and, more importantly, are indispensable for the design of the future nanoscale electronic components whose characteristics will be very close to those of living organisms (Heath, 1998). In conclusion, the challenge to be met is to make perfect systems out of imperfect components.

Starting from a very simple artificial organism with only three cells, the "LSL" acronym (for Logic Systems Laboratory), we will show how it is possible to grow such an artifact in silicon through cellular division and cellular differentiation. The LSL artifact will eventually be implemented in a custom computing platform, the BioWall, a giant electronic tissue made up of two thousand FPGAs. It will be shown that this artificial organism is endowed with self-replicating properties and meets all the criteria outlined by John von Neumann when he was proposing his historical self-replicating cellular automaton, 50 years ago (von Neumann, 1966).

2. ARTIFICIAL CELLULAR DIVISION: THE TOM THUMB ALGORITHM

2.1 Cell Division in Biology

Before describing our new algorithm for the division of an artificial cell, let us remember the roles that cellular division plays in the existence of living organisms (Campbell, 1999). When a unicellular organism divides to form duplicate offspring, the division of a cell reproduces an entire organism. But cell division also enables multicellular organisms, including humans, to grow and develop from a single cell, the zygote. Even after the organism is fully grown, cell division continues to function in renewal and repair, replacing cells that die from normal wear or accidents. The reproduction of an ensemble as complex as a cell cannot occur by mere splitting in two. Cell division involves the distribution of identical genetic material (DNA) to two daughter cells. A dividing cell duplicates its DNA, allocates the two copies to opposite ends of the cell, and only then splits into two daughter cells. In conclusion, we can summarize the two key roles of cell division:

- The construction of two daughter cells in order to grow a new organism or to repair an already existing one (genome "translation").
- The distribution of an identical set of chromosomes in order to create a copy of the genome from the mother cell aimed at programming the daughter cells (genome "transcription").

Switching to the world of silicon, we will propose a new algorithm, the "Tom Thumb algorithm", which, starting with a minimal cell made up of four artificial molecules, the *Annulus elegans*, constructs both the daughter cells and the associated genomes. A tissue of such molecules will in the end be able to constitute a multicellular organism endowed with cellular differentiation.

2.2 Constructing an Artificial Cell

The minimal cell is organized as a square of 2 rows × 2 columns = 4 molecules (Figure 5-1). Each molecule is able to store in its four memory positions four hexadecimal characters of the artificial genome, and the whole cell thus embeds 16 such characters (Mange, 2004). The original genome for the minimal cell is organized as a string of eight hexadecimal characters, i.e. half the number of characters in the cell, moving counterclockwise by one character at each time step ($t = 0, 1, 2, ...$). The 15 hexadecimal characters composing the alphabet of our artificial genome are detailed in Figure 5-2a. They are either "empty data" (0), "message data" (from 1 to 7) or "flag data" (from 8 to E). Message data will be used for configuring our final artificial organism, while flag data are indispensable for constructing the skeleton of the cell. Furthermore, each character is given a status and will eventually be "mobile data", indefinitely moving around the cell, or "fixed data", definitely trapped in a memory position of a molecule (Figure 5-2b).

At each time step, a character of the original genome is shifted from right to left and simultaneously stored in the lower leftmost molecule (Figures 5-1

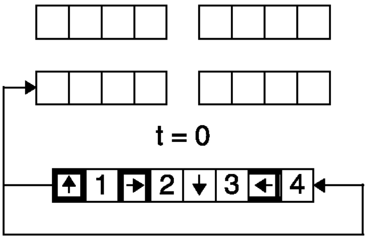
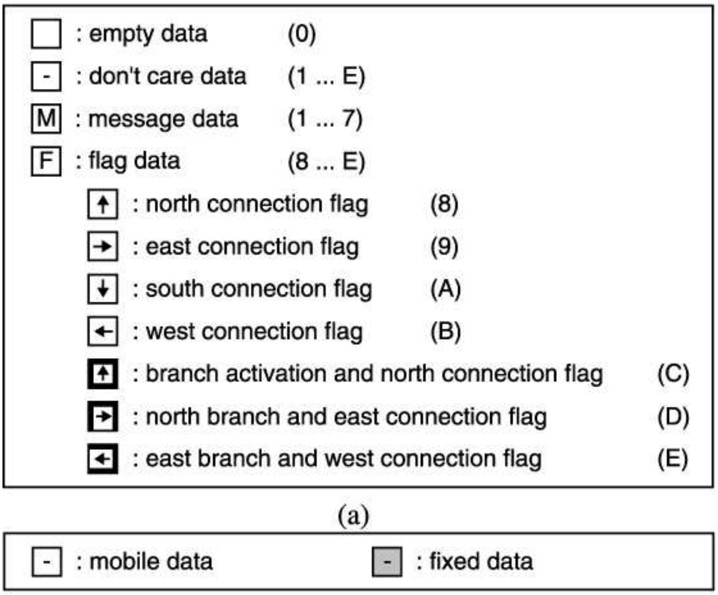


Figure 5-1. The minimal cell (2 × 2 molecules) with its genome at the start ($t = 0$)


: empty data (0)
- : don't care data (1 ... E)
M : message data (1 ... 7)
F : flag data (8 ... E)
: north connection flag (8)
: east connection flag (9)
: south connection flag (A)
: west connection flag (B)
: branch activation and north connection flag (C)
: north branch and east connection flag (D)
: east branch and west connection flag (E)
- : mobile data
- : fixed data

(a)

(b)

Figure 5-2. The 15 characters forming the alphabet of an artificial genome. (a) Graphical and hexadecimal representations of the 15 characters. (b) Graphical representation of the status of each character

and 5-3). The construction of the cell, i.e. storing the fixed data and defining the paths for mobile data, depends on two major patterns (Figure 5-3):

- If the four, three or two rightmost memory positions of a molecule are empty (blank squares), the characters are shifted by one position to the right ($t = 0, 1, 2$).
- If the rightmost memory position is empty ($t = 3$), the characters are shifted by one position to the right; in this situation, the two rightmost characters are trapped in the molecule (fixed data), and a new connection is established from the second leftmost position toward the northern, eastern, southern or western molecule, depending on the fixed flag information F (for $t = 3$: $F = \mathrm{C}$, and the connection is toward the northern molecule).

At time $t = 16$, 16 characters, i.e. twice the contents of the original genome, have been stored in the 16 memory positions of the cell (Figure 5-3). Eight characters are fixed data, forming the phenotype of the final cell, and the eight remaining ones are mobile data, composing a copy of the original genome, i.e. the genotype. Both "translation" (i.e. construction of the cell) and "transcription" (i.e. copy of the genetic information) have therefore been achieved. The fixed data trapped in the rightmost memory position of each molecule reminds us of the pebbles left by Tom Thumb for memorizing his way.

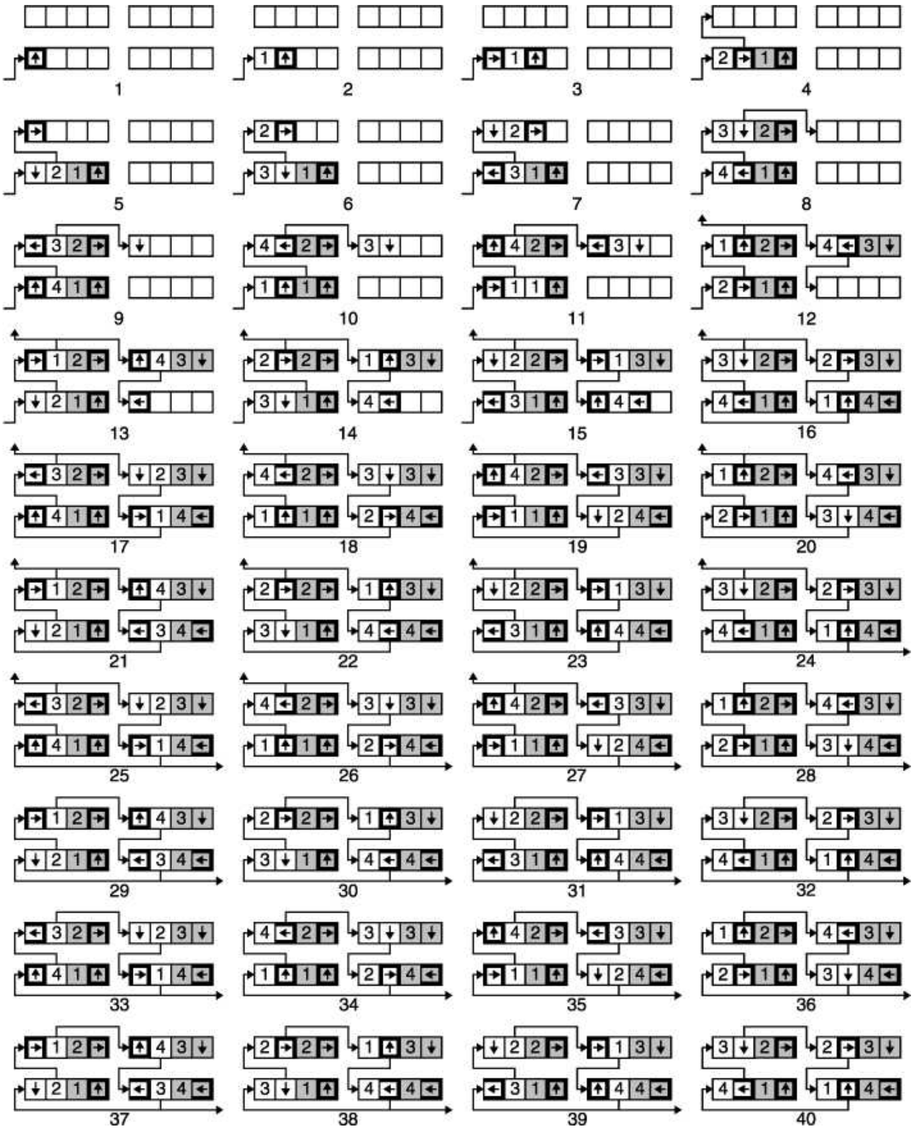


Figure 5-3. Constructing the minimal cell ($t = 4$: north path, $t = 8$: east path, $t = 12$: south path and north branch, $t = 16$: west path and loop completion, $t = 24$: east branch, $t = 28$: north branch cancellation, $t = 40$: east branch cancellation)

2.3 Growing a Multicellular Organism

In order to grow an artificial organism in both horizontal and vertical directions, the mother cell should be able to trigger the construction of two daughter cells, northward and eastward. At time $t = 11$ (Figure 5-3), we observe

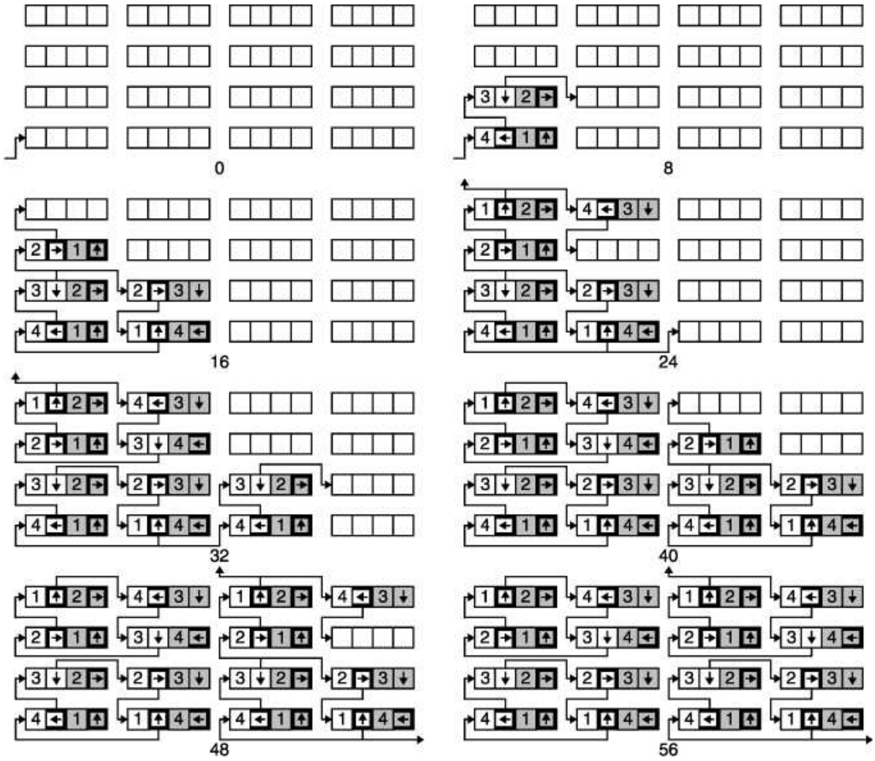


Figure 5-4. Analyzing a multicellular organism made up of 2 × 2 minimal cells (t = 16: closing of the mother cell (lower leftmost cell), t = 32: closing of the upper leftmost cell, t = 40: closing of the lower rightmost cell, t = 56: closing of the upper rightmost cell)

a pattern of characters which is able to start the construction of the northward daughter cell; the upper leftmost molecule is characterized by two specific flags, i.e. a fixed flag indicating a north branch (F = D) and the branch activation flag (F = C). The new path to the northward daughter cell will start from the second leftmost memory position at time t = 12. At time t = 23, another particular pattern of characters will start the construction of the eastward daughter cell; the lower rightmost molecule is characterized by two specific flags, i.e. a fixed flag indicating an east branch (F = E), and the branch activation flag (F = C). The new path to the eastward daughter cell will start from the second leftmost memory position at time t = 24.

In order to analyze the growth of a multicellular artificial organism, we are led to carefully observe the interactions of the different paths created inside and outside of each individual cell. Figure 5-4 shows the growth of a colony of cells. When two or more paths are simultaneously activated, a clear priority should be established. We have therefore chosen three growth patterns:

- For cells in the lower row, a collision can occur between the closing loop and the path entering the lower leftmost molecule; the inner loop, i.e. the westward path, will have the priority over the eastward path.
- With the exception of the mother cell, for cells in the leftmost column, the inner loop, i.e. the westward path, will have priority over the northward path.
- For all other cells, two types of collisions may occur: between the northward and eastward paths (2-signal collision) or between these two paths and a third one, the closing loop (3-signal collision); in this case, the northward path will have priority over the eastward path (2-signal collision), and the westward path will have priority over the two other ones (3-signal collision).

The results of such a choice are as follows: a closing loop has priority over all other outer paths, which makes the completed loop entirely independent of its neighbors, and the organism will grow by developing bottom-up vertical branches. This choice is quite arbitrary and may be changed according to other specifications.

The detailed architecture of the final molecule can be implemented in the form of a new paradigm for the design of generalized cellular automaton, the "data and signals cellular automaton" (DSCA) (Stauffer, 2004). This implementation is described elsewhere (Mange, 2004).

3. ARTIFICIAL CELLULAR DIFFERENTIATION

3.1 Cell Differentiation in Biology

The nematode *Caenorhabditis elegans* is a small worm that has come to occupy a large place in the molecular biology of development. Thanks to its very particular characteristics (notably, the worm is transparent), it has been possible to reconstruct the entire anatomical history of each cell as the fertilized egg develops into a multicellular adult (Watson, 1987). Two amazing conclusions emerged when the information gathered from detailed anatomical studies of living worms was combined with more classical anatomical studies obtained by electron microscopy of serial thin sections of the worm at different developmental stages. First, each cell in the adult worm is derived from the "zygote", the first mother cell of the organism, by a virtually invariant series of cell divisions called a "cell lineage". Second, as a direct consequence of the invariant cell lineages, individual nematodes are anatomically invariant carbon copies of each other. The mature adult hermaphrodite always consists of exactly 945 cells.

Cell differentiation can proceed in at least two very different styles—"mosaic" and "regulatory"—which presumably reflect profoundly different molecular mechanisms. In mosaic development, or temporal development (so called because the organism is assembled from independent parts), the differentiation of a cell does not depend on the behavior or even the existence of neighboring cells. Apparently, internal events within each cell determine its actions; such events could be triggered by cell division itself or by the ticking of an internal biological clock that is set in motion by fertilization. In regulatory development, or spatial development, differentiation is partially or completely dependent on the interaction between neighboring cells. Mosaic development governed by strict cell lineages is the overwhelming rule in *C. elegans*, and regulative development is the exception. In most other organisms (*Homo sapiens* included), the reverse is almost certainly true. The price of such mosaicism may be very small size (perhaps only a limited number of cell divisions can be so rigidly programmed) and only modest complexity (cell-cell interactions may be required to construct a more elaborate anatomy).

3.2 Artificial Cell Differentiation

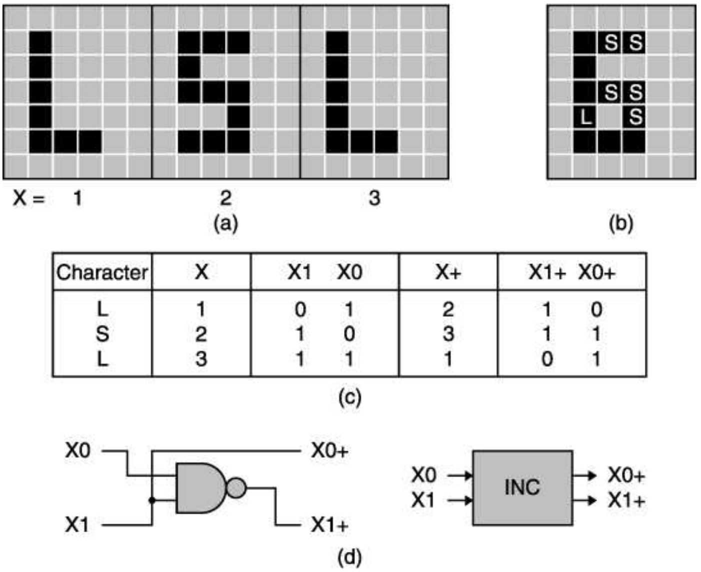
As we are dealing with the construction of rather complex computing machines, we are led to choose the model of regulatory development, or spatial development, in the framework of the Embryonics project. Even if our final goal is the development of complex machines, in order to illustrate the basic mechanisms of our artificial ontogeny we shall use an extremely simplified example, the display of the acronym "LSL", for Logic Systems Laboratory (Mange, 2004).

The machine that displays the acronym can be seen as a one-dimensional artificial organism, *Acronymus elegans*, composed of three cells (Figure 5-5a). Each cell is identified by a X coordinate, ranging from 1 to 3 in decimal or from 01 to 11 in binary. For coordinate values $X = 1$ and $X = 3$, the cell should implement the L character, while for $X = 2$, it should implement the S character. A "totipotent cell" (in this example, a cell capable of displaying either the S or the L character) comprises $6 \times 7 = 42$ molecules (Figure 5-5b), 36 of which are invariant, five display the S character, and one displays the L character. An incrementer—an adder of one modulo 3—is embedded in the final organism; this incrementer implements the truth table of Figure 5-5c and is represented by the logic diagram and symbol of Figure 5-5d. According to the table, the value of the binary variable $X0$ is sufficient to distinguish the display of character L ($X0 = 1$) from the display of character S ($X0 = 0$ or $X0' = 1$).

These specifications are sufficient for designing the final architecture of the totipotent cell (Figure 5-6), whose 42 molecules can be divided into seven categories (from 1 to 7) depending on their functionality:

1: Two busses for the horizontal transfer of the X coordinate.

2: Modulo 3 incrementer.



Character	X	X1	X0	X+	X1+	X0+
L	1	0	1	2	1	0
S	2	1	0	3	1	1
L	3	1	1	1	0	1

Figure 5-5. Specifications of the *Acronymus elegans*. (a) Three cell artificial organism. (b) The totipotent cell. (c) Truth table of the coordinate incrementer. (d) Logic diagram and symbol of the incrementer

3: One bus for the vertical distribution of the $X0$ logic variable.
4: Permanent display of characters L and S.
5: Display of S character only ($X0 = 0$ or $X0' = 1$).
6: Display of L character only ($X0 = 1$).
7: Neutral molecule, no display, no functionality.

The existence of an incrementer adding modulo 3 will allow for the organism self-replication, clearly visible in Figure 5-7c.

4. TOM THUMB ALGORITHM DESIGN METHODOLOGY AND GENERALIZATION

In 1995, Tempesti has shown how to embed the acronym “LSL” (for Logic Systems Laboratory) into a self-replicating loop implemented on a classical cellular automaton. Thanks to a “cut-and-try” methodology and a powerful simulator, he was able to carry out the painful derivation of over 10,000 rules for the basic cell. Unlike his heuristic method, we will show that the same example can be designed in a straightforward and systematic way, thanks to the use of our new data and signals cellular automaton (DSCA) associated to the Tom Thumb algorithm.

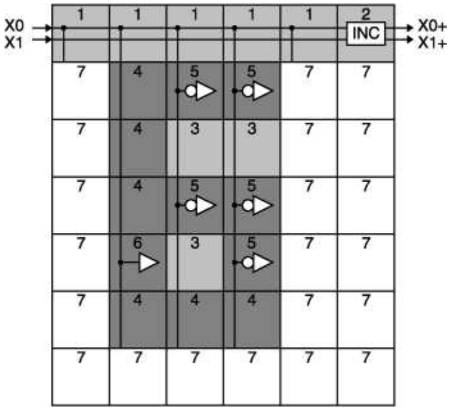
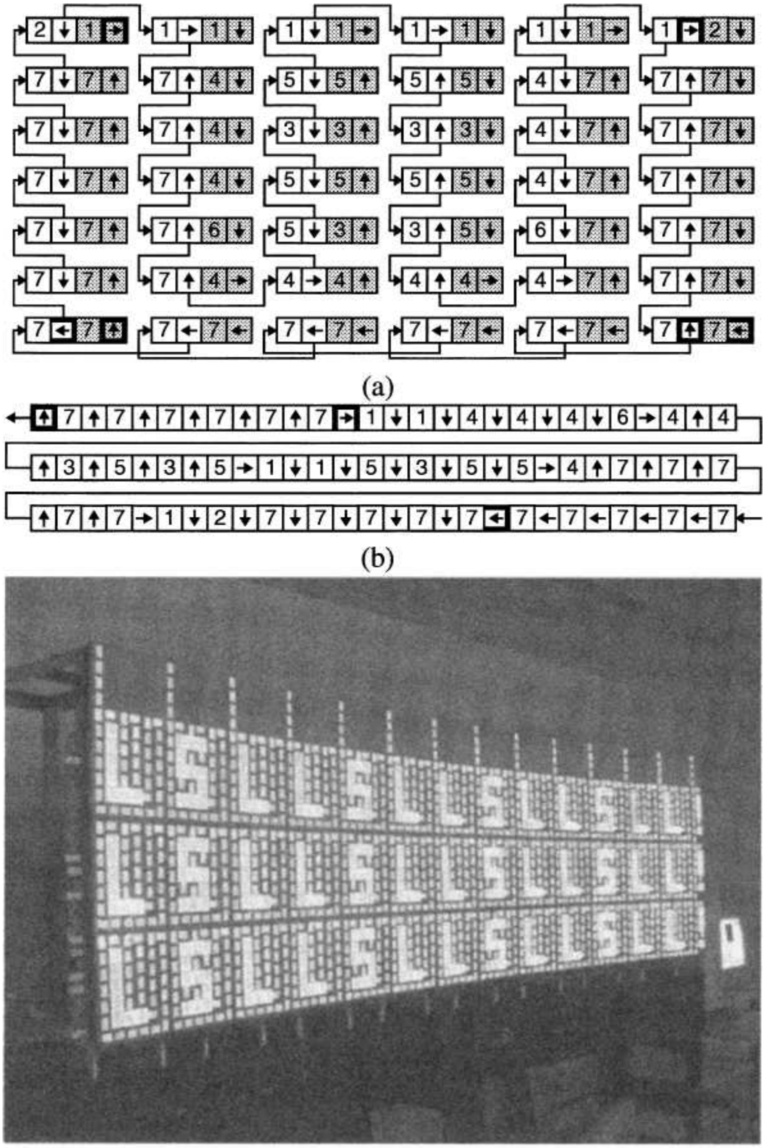


Figure 5-6. The totipotent *Acronymus elegans* cell: detailed functionalities of the 42 molecules

The totipotent cell of the "LSL" acronym example is first represented in a rectangular array of 6 columns × 7 rows (Figure 5-6). While the number of rows is indifferent, the number of columns should be even in order to properly close the loop. The totipotent cell is therefore made up of $6 \times 7 = 42$ molecules connected according to the pattern in Figure 5-7a: bottom-up in the odd columns, and top-down in the even columns, with the lower row reserved for closing the loop. It is then possible to define all the flags in the rightmost memory position of each molecule (grey characters in Figure 5-7a) without forgetting the branch activation and north connection flag in the lower molecule of the first column, the north branch and east connection flag in the upper molecule of the first column, and the east branch and west connection flag in the lower molecule of the last column. The 42 molecules are then given message data (from 1 to 7) according to the detailed architecture of the totipotent cell (Figure 5-6), i.e. the decimal value of each molecule. The detailed information of the final genome, i.e. $42 \times 2 = 84$ hexadecimal characters (Figure 5-7b), is derived by reading clockwise the fixed characters (black and grey characters in Figure 5-7a) of the whole loop, starting with the lower molecule of the first column.

The LSL acronym design example can be easily generalized to produce the following algorithm:

- Divide the given problem in a rectangular array of *C* columns by *R* rows. While the number of rows *R* is indifferent, the number of columns *C* should be even in order to properly close the loop.
- Define all the flags in the rightmost memory position of each molecule according to the following patterns: bottom-up in the odd columns and top-down in the even columns, with the lower row reserved for closing the loop.



(a)

(b)

(c)

Figure 5-7. Realization of the *Acronymus elegans.* (a) The 6 × 7 = 42 molecules of the totipotent cell. (b) Genome. (c) BioWall implementation (Photograph by E. Petraglio)

- Complete the path by adding the branch activation and north connection flag (C) in the rightmost memory position of the lower molecule of the first column, the north branch and east connection flag (D) in the rightmost memory position of the upper molecule of the first column, and the east branch and west connection flag (E) in the rightmost memory position of the lower molecule of the last column, in order to trigger the two daughter loops northwards and eastwards respectively.

- According to the original specifications, complete all the message data in the second rightmost memory position of each molecule. These message data constitute the phenotypic information of the artificial cell.
- The detailed information of the final genome, i.e. the genotypic information of the artificial cell, is derived by reading clockwise along the original path the fixed characters of the whole loop, i.e. the two rightmost characters of each molecule, starting with the lower molecule of the first column. The genotypic information, or artificial genome, is used as the configuration string of the artificial cell and will eventually take place in the two leftmost memory positions of each molecule.

By embedding the basic molecule of our new data and signals automaton (DSCA) in each of the 2000 field-programmable gate arrays (FPGAs) of the BioWall (Tempesti, 2003), it is possible to show the artificial cellular division of the original totipotent cell, followed by the cellular differentiation and by the self-replication of the whole "LSL" organism in both the vertical and horizontal directions (Figure 5-7c).

5. CONCLUSION

5.1 Present and Future Applications

Several years before the publication of the historical paper by Crick and Watson (1953) revealing the existence and the detailed architecture of the DNA double helix, von Neumann was already able to point out that a self-replicating machine required the existence of a one-dimensional description, the genome, and a universal constructor able to both interpret (translation process) and copy (transcription process) the genome in order to produce a valid daughter organism. Self-replication allows not only to divide a mother cell (artificial or living) into two daughter cells, but also to grow and repair a complete organism and is now considered as a central mechanism indispensable for circuits that will be implemented through the nascent field of nanoelectronics (Drexler, 1992), particularly when the fault-tolerant properties associated with our developmental approaches are taken into consideration.

A first field of application of our new self-replicating loop with universal construction is quite naturally classical self-replicating automata, such as three-dimensional reversible automata (Imai, 2002) or asynchronous cellular automata (Nehaniv, 2002).

A second, and possibly more important field of application is Embryonics, where artificial multicellular organisms are based on the growth of a cluster of cells, themselves produced by cellular division (Macias, 2002; Mange, 2000). It is within this context that cellular differentiation will become a key aspect of our growth mechanism, as each newly-created cell identifies its position and its designated role within the complete organism.

Finally, other possible open avenues concern the evolution of such loops and/or their capability to carry out massive parallel computation (Chou, 1998).

5.2 Emergence and Complexity

If we assume the existence of a silicon substrate organized as a homogeneous matrix of basic elements or "molecules", we observe that the injection of a finite string of discrete symbols, the genome, successively produces the emergence of cells (by cellular division), of multicellular organisms (by cellular differentiation) and, finally, of a population of identical organisms, i.e. clones (by cyclic repetition of the coordinates). This emergence is of course not "magic", and follows from a determinist use of logic symbols considered as configuration of the molecules, themselves implemented by the field-programmable gate arrays (FPGAs).

Emergence is then the automatic result of a chain of several mechanisms (cellular division and differentiation, repetition of coordinates). This appearance of a complex system (a population of *Acronymus elegans* in our example) from rather simple molecules and a short genome, simply creates the illusion of an internal growth of complexity, meeting, in our opinion, the loosely-specified criteria that define emergence.

Acknowledgments

This work was supported in part by the Swiss National Science Foundation under grant 20-100049.1, by the Leenaards Foundation, Lausanne, Switzerland, and by the Villa Reuge, Ste-Croix, Switzerland.

References

Campbell, N. A., J. B. Reece and L. G. Mitchell. 1999. *Biology, 5th edition.* Menlo Park, CA: Benjamin/Cummings.

Chou, H.-H. and J. A. Reggia. 1998. "Problem solving during artificial selection of self-replicating loops", *Physica D*, Vol. 115, No. 3–4, 1998, 293–312.

Drexler, K. E. 1992. *Nanosystems: Molecular Machinery, Manufacturing, and Computation.* New York, NY: John Wiley.

Heath, J. R., et al. 1998. "A defect-tolerant computer architecture: opportunities for nanotechnology", *Science*, Vol. 280, No. 5370, June 1998, 1716–1721.

Imai, K., T. Hori and K. Morita. 2002. "Self-Reproduction in Three-Dimensional Reversible Cellular Space", *Artificial Life*, Vol. 8, No. 2, 2002, 155–174.

Macias, N. J. and L. J. K. Durbeck. 2002. "Self-reproduction in asynchronous cellular automaton", Proceedings of the 2002 NASA/DOD Workshop Conference on Evolvable Hardware, *IEEE Computer Society Press*, 2002, 46–55.

Mange, D., et al. 2000. "Toward Robust Integrated Circuits: The Embryonics Approach". In *Proceedings of the IEEE*, Vol. 88, No. 4, April 2000, 516–541.

Mange, D., et al. 2004. "Embryonics Machines that Divide and Differentiate". In *Proceedings of the First International Workshop Bio-ADIT 2004, Lecture Notes in Computer Science*, Springer-Verlag 2004, 201–216.

Mange, D., et al. 2004. "Self-replicating loop with universal construction", *Physica D*, Vol. 191, No. 1–2, April 2004, 178–192.

Nehaniv, C. L. 2002. "Self-reproduction in asynchronous cellular automaton". In *Proceedings of the 2002 NASA/DOD Workshop Conference on Evolvable Hardware, IEEE Computer Society Press*, 2002, 201–209.

Sipper, M., et al. 1997. "A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, 1997, 83–97.

Stauffer, A. and M. Sipper. 2004. "The data-and-signals cellular automaton and its application to growing structures", *Artificial Life*, Vol. 10, No. 4, 2004, 463–477.

Tempesti, G. 1995. "A New Self-Reproducing Cellular Automaton Capable of Construction and Computation". In *Proceedings of the Third European Conference on Artificial Life ECAL'95, Lecture Notes in Computer Science*, Springer-Verlag, 555–563.

Tempesti, G. and C. Teuscher. 2003. "Biology Goes Digital", *Xcell Journal*, No. 47, Fall 2003, 40–45.

Villasenor, J. and W. H. Mangione-Smith. 1997. "Configurable Computing", *Scientific American*, Vol. 276, No. 6, June 1997, 54–59.

von Neumann, J. 1966. *Theory of Self-Reproducing Automata*. Illinois, MI: University of Illinois Press.

Watson, J. D. and F. H. C. Crick. 1953. "A structure for desoxyribose nucleid acid", *Nature*, Vol. 171, 1953, 737–738.

Watson, J. D., et al. 1987. *Molecular Biology of the Gene*. Menlo Park, CA: Benjamin/Cummings Publishing Company.

Chapter 6

THE POETIC HARDWARE DEVICE: ASSISTANCE FOR EVOLUTION, DEVELOPMENT AND LEARNING

Andy M. Tyrrell and Will Barker
Department of Electronics, Heslington, University of York, York YO10 5DD, UK.
amt@ohm.york.ac.uk

Abstract: The developments presented in this chapter are results of a new research project, "Reconfigurable POEtic Tissue". The goal of the project was the development of a hardware platform capable of implementing bio-inspired systems in digital hardware. In particular, the final hardware device, while similar to other FPGAs, was designed with a number of novel features which facilitate evolution, development and learning. These include dynamic reconfiguration and on-chip reprogramming. This chapter gives some details of the architecture of the device, followed by a simple example. The example considers these features available on POEtic in the context of fault-tolerant system design and shows how an ensemble of different, but often complementary, techniques might be produced using these novel device features. It is argued that these features are generic for many evolutionary-type applications.

Key words: evolvable hardware, dynamic reconfiguration, fault tolerance, novel hardware device.

1. INTRODUCTION

The majority of evolvable hardware experiments and applications in the digital domain have typically been performed on "standard" digital devices, usually FPGAs. There are notable exceptions to this, probably the best known being the devices produced by Professor Higuchi and his team (Kajitani, 1999). Standard FPGAs are not designed with bio-inspired techniques in mind. This means that many of the things we would like to do, such as partial reconfiguration (either via the environment or by other cells), growth (either real or artificial), free evolution (i.e., not worrying about creating configurations that might destroy the device), or non-local cell interactions, are either protracted at best or impossible in many cases. Similar comments might be addressed at hardware implementations of learning and development.

What we describe here is a FPGA which, for the first time, has been designed and implemented with bio-inspired techniques at the fore of the process.

The aim of this chapter is to present some of the ideas developed in the framework of a new research project called "Reconfigurable POEtic Tissue" (or "POEtic" for short) (Tyrrell, 2003), recently completed under the aegis of the European Community. After a short introduction to the POEtic project for the design of bio-inspired hardware, the chapter will present some of the novel features of the POEtic device built during the project. To consider the usefulness of the device, a simple but illustrative example is chosen which shows how the main features designed into the device aid fault-tolerant system design; experimental results are given to illustrate the efficacy of the device and these special features.

2. THE POETIC PROJECT

The POEtic tissue (for tissue read device or devices) is a multicellular, self-contained, flexible, and physical substrate designed to develop and dynamically adapt its functionality through the processes of evolution, development, and learning. It should interact with a dynamic and partially unpredictable environment to effect these biologically inspired ideas, and to self-repair parts damaged by aging or environmental factors in order to remain viable and perform similar functionalities.

To summarize, the goal of the POEtic project was the:

"*... development of a flexible computational substrate inspired by the evolutionary, developmental and learning phases in biological systems.*"

Following the three models of bio-inspiration, the POEtic tissue was designed logically as a three-layer structure (Figure 6-1 gives an abstract view of this relating to hardware):

- The phylogenetic model acts on the genetic material of a cell. Each cell can contain the entire genome of the tissue. Typically, in the architecture defined, it could be used to find and select the genes of the cells for the *genotype layer*.
- The ontogenetic model concerns the development of the individual. It acts mostly on the *mapping or configuration layer* of the cell, implementing cellular differentiation and growth. In addition, ontogenesis will have

an impact on the overall architecture of the cells where self-repair (healing) is concerned.

- The epigenetic model modifies the behavior of the organism during its operation, and is therefore best applied to the *phenotype layer*.

Defining separate layers for each model has a number of advantages, as it allows the user to decide whether to implement any or all models for a given problem, and allows the structure of each layer to be adapted to the model. This adaptability is achieved by implementing the cells on a *molecular substrate*, in practice, a surface of programmable logic.

The final hardware design (VLSI device) has a number of specific novel features built into its fabric to assist with bio-inspired designs. It is shown here that these features can also be used effectively for the design of fault-tolerant systems.

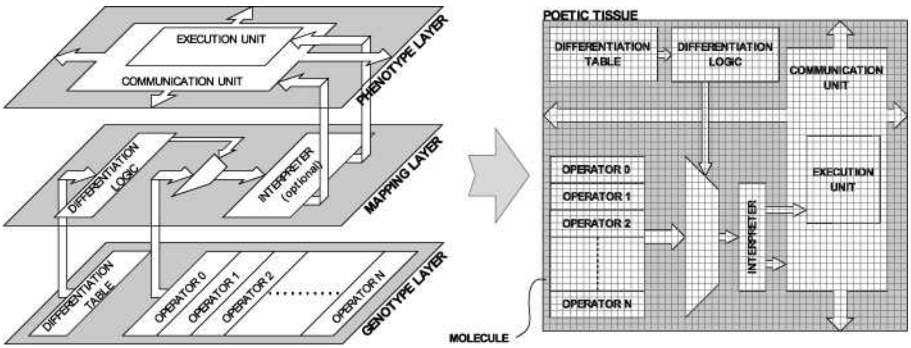


Figure 6-1. The three organisational layers of the POEtic project

2.1 Operational Mode

Following the logical structure presented in Figure 6-1, the actual implementation of the POEtic chip had to allow the implementation of artificial organisms, consisting of cells (or in this case molecules) capable of growth, self-repair, and learning, as well as the capacity of letting a population of such organisms evolve. The architecture of the device was divided in two sections: a microprocessor and a reconfigurable array of simple elements (molecules). We will concentrate here on the molecules, as the microprocessor is much less interesting! Among the main novel features, which set this device apart from conventional programmable devices, are:

- A *distributed routing algorithm* that allows the automatic creation of connections between parts of the chip (or of many chips connected together) by dynamically finding and exploiting data paths.

- The possibility for the array to *self-reconfigure* parts of itself, a powerful feature for evolvable hardware and self-repairable systems.
- The impossibility of short-circuits, implying that no configuration bitstream can damage the chip (again, very useful for evolutionary approaches).
- The presence of a dedicated microprocessor that allows a rapid reconfiguration of the array via a parallel access.

The implementation of a bio-inspired system as described in the preceding section implies that the organisms (i.e., the arrays of cellular processors) are loaded into the hardware reprogrammable part of the device, in our case called the "organic subsystem". The subsystem is logically divided into two separate planes: the *functional plane*, hosting the reconfigurable logic, and the *routing plane*, which has the task of creating data paths for intercellular communication at runtime. The first of these is composed of basic elements that play the role of *molecules*, and is general enough to implement any digital circuit, while the second implements a distributed dynamic routing algorithm. The main specificities of the organic subsystem are its capabilities of changing *at runtime* the function realized by the molecules and the connections between different parts of the chip. In the next subsections we will describe in some detail the two planes, defining first the architecture and the modes of operation of the molecules and then the hardware and algorithm that control the dynamic paths creation (more details in (Moreno, 2004)).

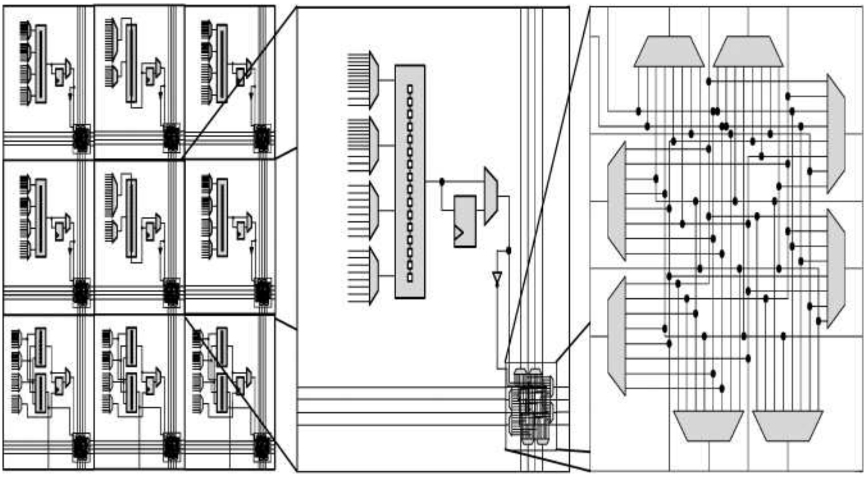


Figure 6-2. The molecules

2.1.1 Molecules

A molecule (Figure 6-2) contains a 4-input LUT and a flip-flop; the output can thus be combinational or sequential. This architecture is similar to many commercial FPGAs, but a number of novel features have been added to support bio-inspired applications. A molecule can be configured into one of 8 different operational modes (note that some of these modes relate to the routing mechanism described in the next section):

- *4-LUT*: the functionality corresponds to a 4-input LUT.
- *3-LUT*: the LUT is split into two 3-input LUTs.
- *Shift memory*: the LUT is used as a 16-bit shift register.
- *Comm*: the LUT is split into an 8-bit shift register and a 3-input LUT.
- *Input*: the molecule is an input from the routing plane and the identifier of its source is stored in the 16-bit shift register.
- *Output*: the molecule is an output to the routing plane, with its identifier stored in the 16-bit shift register.
- *Trigger*: this mode is used by the routing algorithm to synchronize the identifiers decoding process. Furthermore, two inputs are responsible for resetting the routing plane and to disable some molecules.
- *Configure*: a molecule in this mode can modify the configuration bits of other molecules in the chip, therefore allowing self-reconfiguration of parts of a cell, for instance.

As in all programmable logic, connections are a fundamental part of the circuit. Besides the dynamic communication network that exploits the routing plane described next (used for intercellular communication between processors), intermolecular communication (used for the conventional interconnection between the programmable logic elements) goes through switchboxes in the molecules. Each molecule contains a switchbox composed of eight 8-input multiplexers, two for each direction (right diagram in Figure 6-2). Contrary to the dynamic routing, the configuration bits of the switchbox are loaded and fixed at configuration time, and can be changed only by a full or a partial reconfiguration of the circuit.

2.1.2 Partial Reconfiguration

Partial reconfiguration is one of the main innovations in the molecular substrate. This feature not only allows any molecule to change the configuration of another molecule, but also allows the user to decide specifically which parts of the configuration should be modifiable.

For this purpose, the 76 bits that represent the configuration of a molecule are split into five blocks: *LUT*, *LUT input multiplexers*, *switchbox*, *operational mode*, and *other bits*. For each block, a special bit indicates if the block should be reconfigured or bypassed during any partial reconfiguration process. Thus, only part of a molecule can be modified; for instance, it is possible to change only the contents of the LUT, and hence the function realized by the molecule, while keeping all of its connections to the rest of the array. Figure 6-3 shows these 5 blocks, with their corresponding special bit, and the chain of partial reconfiguration passing through the LUT inputs and the mode. We can observe that the molecule needs two configuration bits to indicate the origin of an accepted partial reconfiguration, and a bit to enable its propagation to its neighbours. These 3 bits, with 5 additional special bits, cannot be modified by the molecules themselves, but only by the microprocessor through a direct reconfiguration. It is also interesting to notice that a molecule is able to transmit the configuration bits through the connection network, allowing for long distance reconfiguration (i.e., not only of the directly connected neighbours).

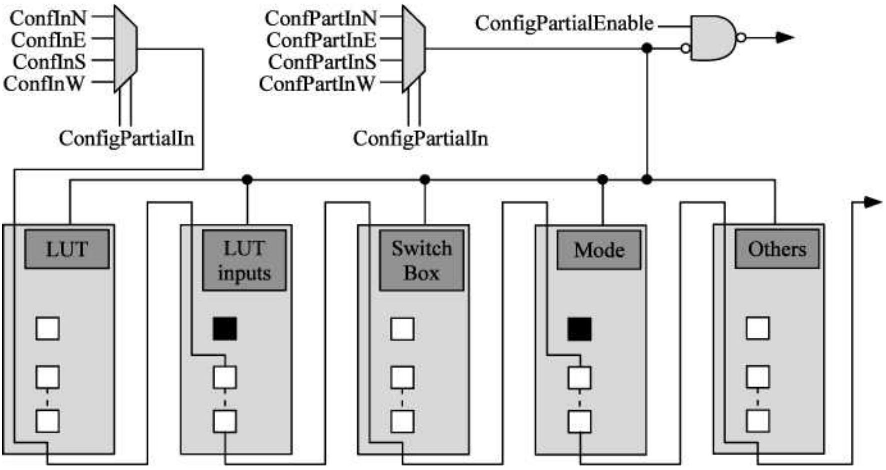


Figure 6-3. Configuration bits of a molecule split into 5 blocks, two of them being reconfigured

The partial reconfiguration mechanism can obviously serve to modify the behaviour of the molecules. This feature is very useful for evolvable hardware processes and necessary for self-repair, as a spare cell would have to execute the task of a dead one. Through this mechanism, it is possible to entirely create the functional part of the new cell by copying part of an existing cell – a way forward for hardware growth. Furthermore, the three first blocks can serve to store information, up to 54 bits per molecule. This kind of memory, accessed serially, can be very useful to efficiently store a genome.

2.1.3 Molecular Enable

Another useful feature of the molecules is the *global molecular enable.* Trigger molecules control this enable, and every molecule owns a configuration bit that indicates if it has to be sensitive or not to this signal. For example, for self-repair, a cell can be divided in two parts: a functional part and a part responsible for the self-repair. The first part can then be sensitive to the global enable, and so be disabled while a self-repair process occurs, in order to guarantee the correct behaviour of the whole system.

2.1.4 Routing Plane

The routing plane, logically superposed on the molecular plane, allows the device to dynamically configure connections that join separate molecules across one or multiple chips. In a sense, this plane is the POEtic equivalent of long-distance connection busses in a conventional FPGA, with some crucial differences.

In practice, the routing plane consists of a set of regularly-spaced *routing units* that use multiplexers to direct the flow of data across the chip. One routing unit corresponds to a set of four molecules in the subjacent plane[1] and is itself connected to its four cardinal neighbouring routing units (Figure 6-4).

The routing units implement a distributed routing algorithm, and configure multiplexers that are then used to transmit data from output molecules to input molecules. The fundamental novelty of this mechanism is the absence of configuration bits; the communication paths are found and established dynamically at runtime and the user need not define the connection network at design time. All details regarding this algorithm, that implements a parallel breadth-first search algorithm to find the shortest path between two points, can be found in (Thoma, 2003). We will focus here on the features useful for self-repair.

A routing unit (Figure 6-5) is composed of a controller (a finite state machine) and five 4-input multiplexers selecting the value to send in each direction and to the molecules underneath. Each multiplexer is controlled by two bits that are modified by the finite state machine during the routing process, and an additional bit indicates if the multiplexer is currently in use, in order to avoid collisions.

[1] This choice of one routing unit corresponding to a set of four molecules represents a compromise between area and efficiency. The routing tools must ensure that only one of the four subjacent molecules has access to the routing unit.

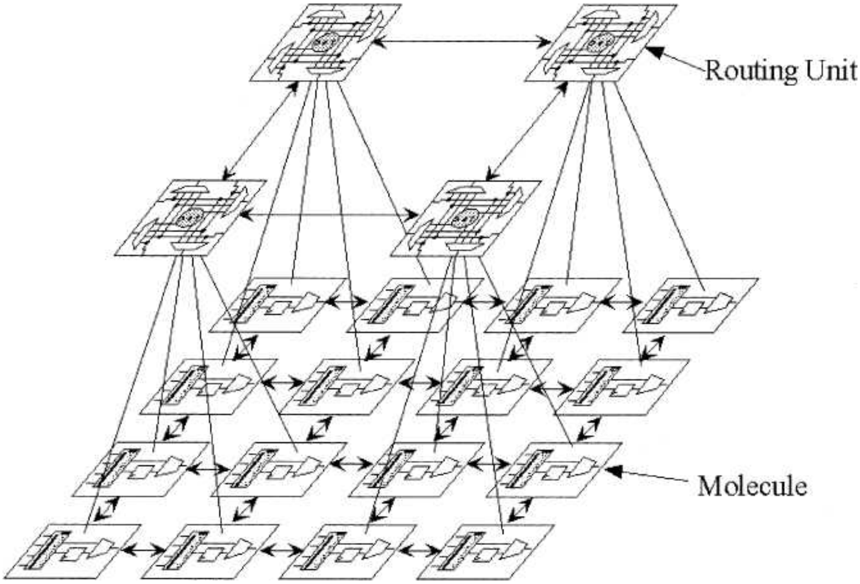


Figure 6-4. Molecules and routing units

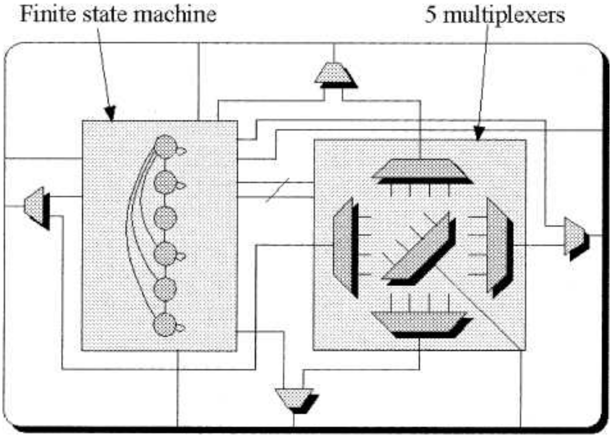


Figure 6-5. Routing unit

The algorithm is based on 16, 8, 4, 2 or 1 bits long IDs stored in the shift register of molecules in input or output mode; during the routing process, molecules connect with other molecules having the same ID. The molecules in input or output mode manage the creation of paths by controlling a *routing enable signal* sent to the routing units. When a molecule enables the signal, a routing process is launched until the molecule is connected to the other molecules with the same ID; if the signal is disabled, the molecule is ready to accept a connection. This approach is very well adapted for the implementation of cellular self-repair or growth mechanisms, since it allows *spare cells* to be placed on the array simply by giving them all the same ID; a cell that wants to connect to a spare cell will then simply create a path with the nearest cell, without needing to know its actual physical position.

Another interesting feature of the algorithm is that it is totally distributed, and that every routing unit can try to start a routing process at the same time, avoiding the need for explicit synchronization mechanisms; a hardware-based algorithm grants priority to the most southwest molecule and serializes the routing process by allowing a single path at a time to be established.

Finally, the molecules can force a reset of the routing plane after which every connection needed by the molecules will be recreated. A self-repair mechanism can therefore reinitiate new connections when a cell is replaced by a new one.

In a routing process, every routing unit (RU) that needs a connection propagates a signal through the chip and priority is given to the most south-west unit, the *master*, who then sends its ID to all other RUs serially. At the end of this phase every RU knows if it is involved in the current routing. The master then disables all molecules in the same mode (input or output), in order to avoid contentions. The next phase is a parallel implementation of Lee's algorithm (Lee, 1961), and is depicted in Figure 6-6, where we can see the creation of the shortest path between the source S1 and the target T1, taking into account the existing paths.

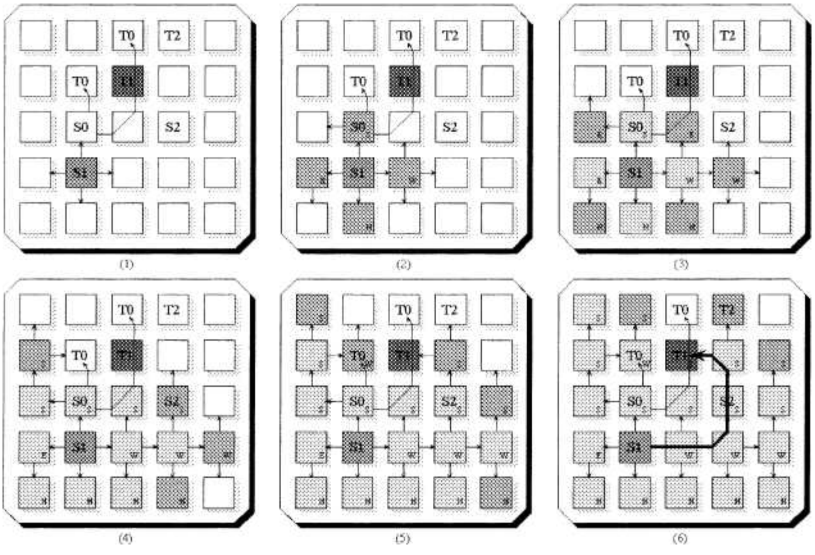


Figure 6-6. Creation of a path during a routing process

3. RECONFIGURATION AND FAULT-TOLERANCE

The POEtic project provides a unique platform for investigating mechanisms at work in biological systems, for example ones which exhibit fault-tolerant behaviours. It is the intention of the rest of this chapter to demonstrate this through the development of a cellular ontogenetic fault-tolerant mechanism on the POEtic tissue based upon growth.

Self-repair, or healing, is a critical mechanism within an organism's response to damage involving the growth of new resources, in the form of cells, and their integration into the organism replacing damaged ones. An electronic system cannot grow new silicon resources in response to device faults in the same way and so growth in silicon is generally emulated by having redundant resources which the system can use. While this mechanism is used here to aid self-repair, consider the mechanisms used for evolutionary processes, or for types of learning. These are a generic set of mechanisms provided on POEtic that can be used for all of these processes and more. The POEtic architecture provides novel features which are particularly useful for implementing models of growth in digital hardware including the underlying molecular architecture, dynamic routing and self-configuration of the tissue.

3.1 Growth

The work reported here is inspired by two important features of growth: cell division and cellular differentiation. Cell division is the process of self-replication through which cells produce copies of themselves. Cellular differentiation is the process through which cells organise themselves by taking on specific functional types depending upon their positions (and possibly other characteristics such as chemical gradients) within an organism.

Prompted by these distinct modes of growth, a novel cell design has been implemented on the POEtic tissue and here illustrated in a test application. We describe an implementation of an embryonic array (similar in style, but very different in implementation to those reported in (Mange, 2000)) emulating the processes of cellular differentiation.

3.2 Test Application

In order to investigate issues regarding the implementation of cellular fault-tolerant mechanisms (including growth and partial reconfiguration) on the POEtic tissue, a test application has been defined based upon the audio application presented in (Cooper, 2004). The test application consists of a cell constructed from nine one-dimensional waveguide mesh elements (for those not familiar with waveguide meshes, each mesh element can be considered as a particular type of processing element. The Ps in Figure 6-7 represent values passing between neighbouring mesh (processing) elements) – this application requires real-time (audio) processing. Cells can be chained together to form a one-dimensional waveguide with length an arbitrary multiple of nine, as shown in Figures 6-7 & 6-8. While this is a specific application executed within the POEtic project, the work reported here is generic and appropriate for any application on one or more devices.

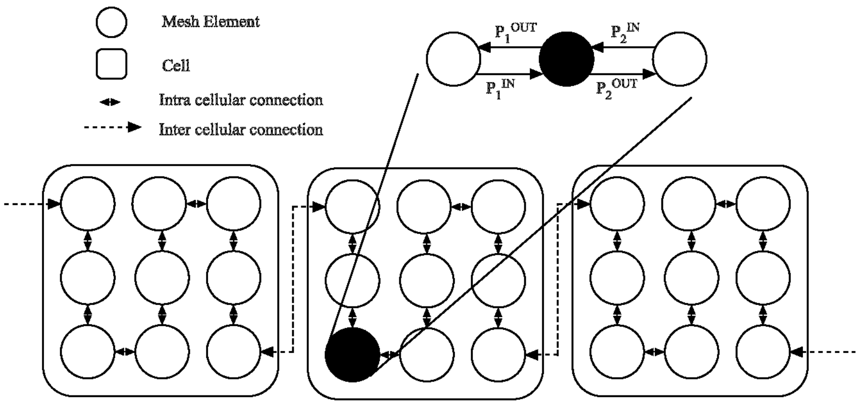


Figure 6-7. Nine mesh element one-dimensional waveguide cells with left and right input streams

3.3 Fault Detection

Biological mechanisms for fault detection, in themselves, provide a rich field of research to which the POEtic platform is applicable (Canham, 2002). However, as the aim of the cell designs here is to investigate growth mechanisms, a standard hardware redundancy technique has been chosen to provide fault detection in the designs. It is based upon the assumptions that faults will occur discretely in time and that a fault is only of significance if it causes the cell function at its outputs to deviate from correct behaviour. Based upon these assumptions we can duplicate and compare two systems; a fault will cause the values at the outputs of cell function copies to differ. This discrepancy is detected by XOR logic functions comparing the cell function outputs and combined into a fault flag by an OR logic function. The advantages of this method for fault detection are that it is simple, acts at the resolution of a single clock cycle, operates online, and is applicable to any cell function. Of course, what is really of interest to us here is what happens then?

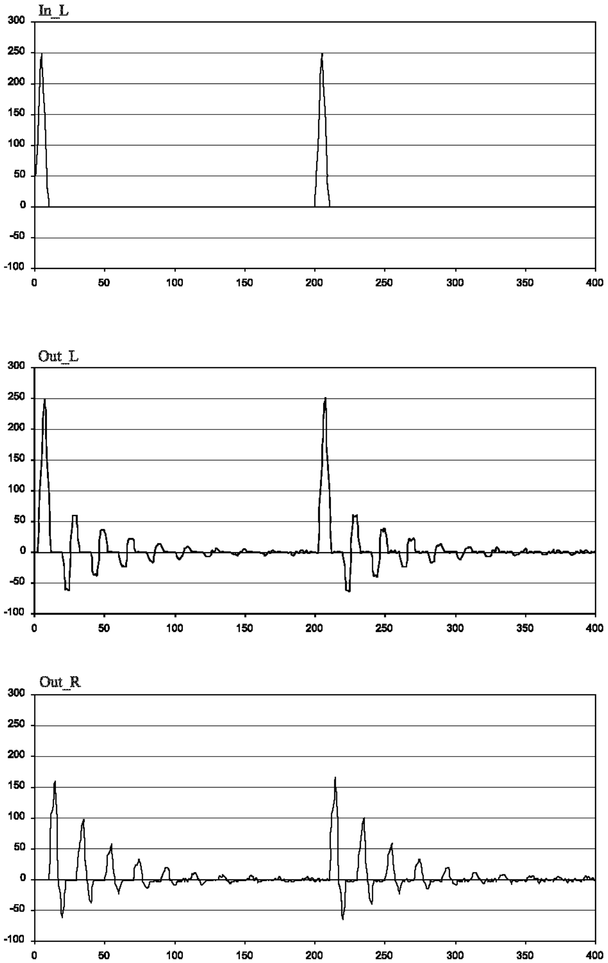


Figure 6-8. Cell output in response to pulses applied to one input

4. EMBRYONIC IMPLEMENTATION ON THE POETIC TISSUE

An embryonic array consists of an array of cells implemented in reconfigurable logic, each of which contains a set of configuration strings describing every cell function within the system to be implemented. A particular cell will take on just one of these functions at any given time (but may change through configuration during operation if required). This set of configuration strings is analogous to the biological genome contained within every living cell. Each configuration string is analogous to a gene and can be directly translated into the cell function it describes in the hardware of the cell. Development of the system is achieved through differentiation during which each cell identifies its configuration string with respect to its location within the array and uses it to configure its function (Moreno, 2004), detailed in Figure 6-9.

4.1 Cell Function Areas

Cell function areas are the areas of the cell where the cell application functionality is performed. Duplication of the cell function enables fault detection by the method described. The areas are initially blank and require configuration from a stored gene. They consist of molecules which have the partial configuration inputs from their neighbours chained together as illustrated in Figure 6-10 and all configuration registers enabled for configuration. This allows an arbitrary cell function to be configured within them. The stored gene therefore consists of the contents of the configuration registers for each molecule in the function listed in the order in which they appear in the chain from the head to the tail.

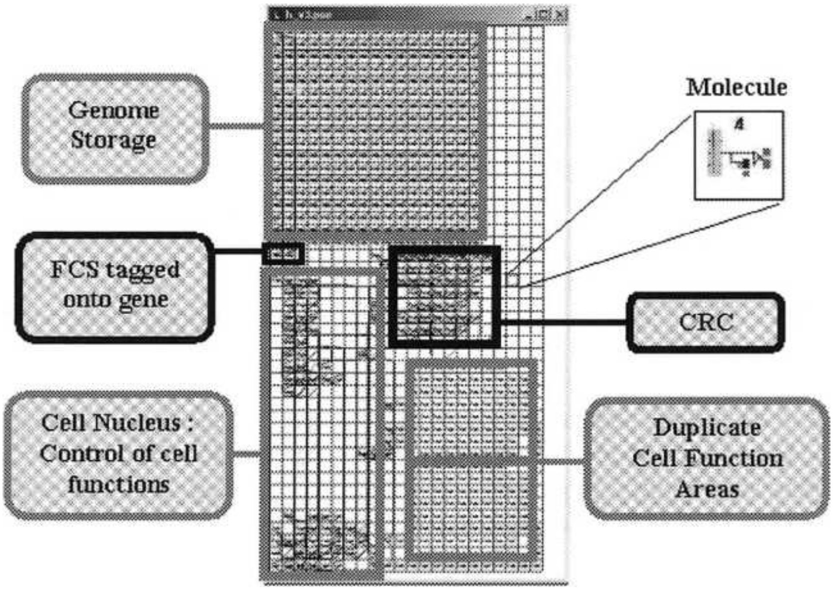

Genome Storage
FCS tagged onto gene
Cell Nucleus : Control of cell functions
Molecule
CRC
Duplicate Cell Function Areas

Figure 6-9. Embryonic cell design on the POEtic tissue (cell has a single gene in its genome for illustrative purposes)

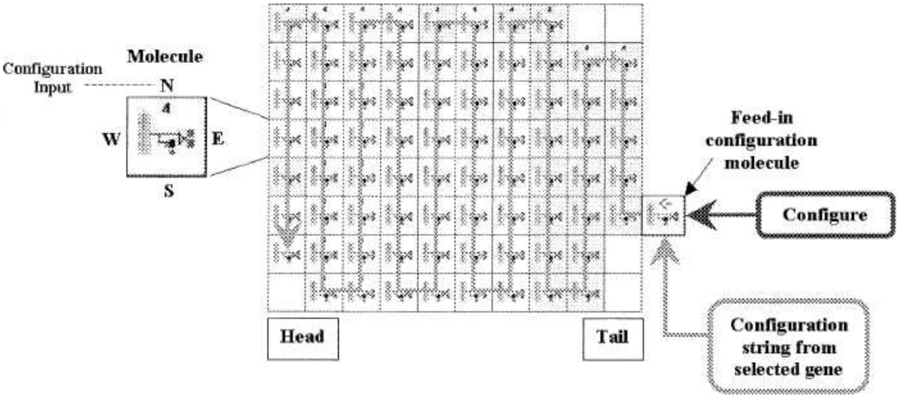

Configuration Input
Molecule
N
W
E
S
Feed-in configuration molecule
Configure
Configuration string from selected gene
Head
Tail

Figure 6-10. Cell function area configuration chain

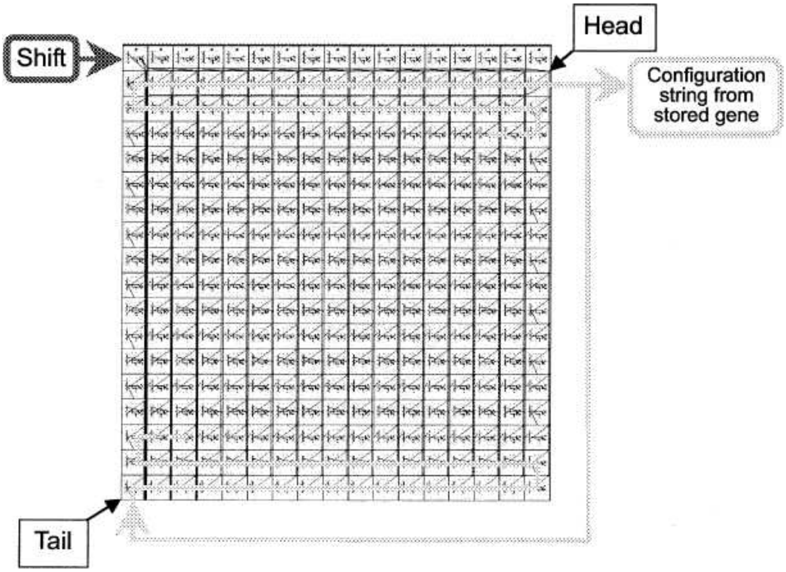


Figure 6-11. Gene block

4.2 Genome Storage

The stored genome consists of individual gene blocks, each of which can be selected by the differentiation system to be the source for configuration of the function areas within the cell. The genes consist of shift memory molecules which store the configuration string in their look up tables (LUTs). The inputs and outputs of the memory molecules are chained together in the same way as the molecules in the function configuration areas. During configuration of the function areas, every gene in the stored genome shifts its contents out from its head with the string from the gene selected by the differentiation process being channelled into the cell function areas.

The head of each gene block is looped back into the tail by connecting the memory molecule output of the head molecule to the input of the tail molecule so that the contents of the gene block are retained, illustrated in Figures 6-11 & 6-12.

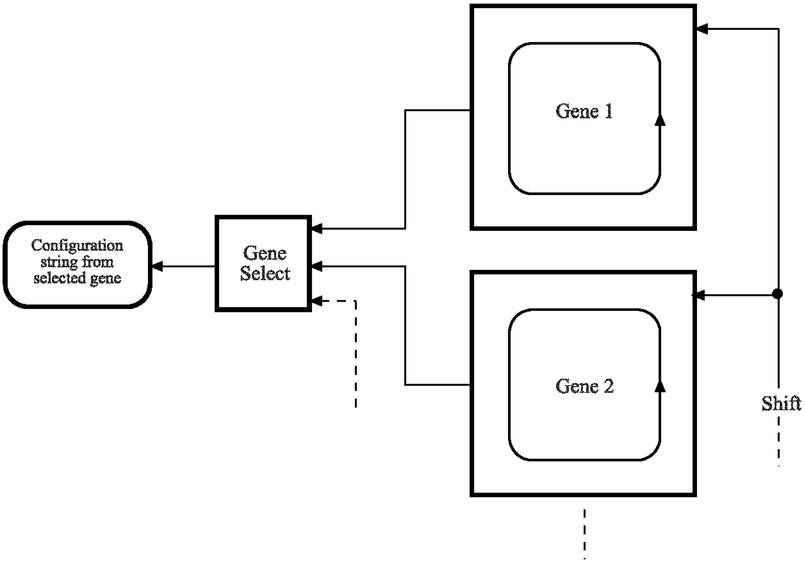


Figure 6-12. Stored genome consisting of selectable gene blocks

4.3 Fault Detection in the Stored Genome: Cyclic Redundancy Check

Approximately four times as many molecules are required to store each gene than are used in the function block that it describes, and an embryonic cell will require as many genes as there are different cells in the system. As both function copies are configured from the same stored gene, a fault in the gene will go undetected by the redundancy fault-detection system as both copies will be producing the same erroneous outputs.

A second fault detection system has therefore been implemented in the embryonic cell design in the form of a cyclic redundancy code (CRC) which can detect faults in the gene being used to configure the cell function by means of a frame check sequence (FCS), which is tagged onto the end of every gene (Costello, 2004).

On configuration of the cell function areas, the configuration string for the selected gene (including the FCS) is passed through the CRC register as the cell function areas are configured. If the stored gene is incorrupt then the output of all of the CRC register elements will be zero at the end of this process. Otherwise, at least one of the outputs of the register elements will be high, indicating a fault in the gene.

4.4 Cell Nucleus

The cell nucleus is responsible for controlling the five main processes of the embryonic cell. These are cellular differentiation, cell function configuration, fault detection, apoptosis and routing.

4.4.1 Cellular Differentiation

Each of the embryonic cells in the array has a differentiation input and output molecule. These inputs and outputs are linked in a chain across the tissue. On receiving a zero at its input each cell asynchronously sets its output to zero. The first cell in the chain has its input connected to a source external to the tissue into which the differentiation signal is driven. This signal consists of a series of ones equal in length to the number of cells in the organism being developed. The first cell in the chain therefore receives a series of ones equal to the number of cells in the application. At this point the output of cell one is asynchronously reset to zero, causing a chain reaction through which all cell differentiation outputs down the chain asynchronously reset to zero. This terminates the differentiation process in the cells, each of which will have received one less series of ones at its differentiation input than the previous cell in the chain. The cells then select their allocated genes (and hence their function) depending upon the number of ones received at their differentiation inputs. Cells which receive no ones at their inputs blank their function areas and are left as unused spare cells. This process can be instigated at any point by simply driving the differentiation signal into the differentiation chain.

4.4.2 Cell Function Configuration

Completion of the differentiation process triggers a molecule configuration counter. The counter enables the shift input to each of the gene blocks in the stored genome and enables a configuration molecule which feeds the output of the selected gene into the configuration input of the tail of the function area configuration chain.

When the counter indicates that the number of molecules in a cell function area have been configured, the enable to the configuration molecule is disabled and the counter resets. At this point the function areas of the cell have been programmed with the selected gene and are ready for integration into the system. Before this can occur however, the FCS must be shifted through the CRC register to check that the gene is correct. This is controlled by a second counter which is triggered by the overflow of the first and shifts the gene blocks by a further 32 bits, driving the FCS out of the gene block through the CRC register.

4.4.3 Fault Detection

In the cell nucleus the values at the outputs of the two cell function copies are compared and a fault is flagged in response to a discrepancy. The integrity of the configuration of these function copies is tested by the CRC register on configuration, and if corrupt a fault is flagged. The cell nucleus combines these two fault flags into a single signal which triggers cell apoptosis and differentiation of the system.

Differentiation in response to a fault is triggered by the faulty cell setting the '*mol_enable_out*' signal on its trigger molecule low. This is detected by the external system controlling the differentiation signal input to the tissue which drives the signal into the differentiation chain in response.

4.4.4 Apoptosis

Apoptosis in a faulty embryonic cell is achieved by selecting the blank gene, simply a source of zeros, and bypassing the delay which the cell would otherwise introduce into the differentiation chain. This shifts the differentiation values received downstream of the faulty cell one cell down the chain, and causes the cell to blank its function areas removing any molecules (such as input and output molecules) which may interfere with the operation of the system.

4.4.5 Routing

Having completed the processes of cellular differentiation and configuration, the final step in producing the functioning embryonic system is to route together any input and output molecules which have been configured in the function areas of the cells.

The cell which receives the differentiation value equal to the number of cells in the system, i.e. the first healthy cell in the differentiation chain, is assigned the task of triggering the routing process. Every cell contains a trigger molecule capable of this. On completing the configuration of their function areas every cell sends a pulse on the '*start_routing_enable*' signal to its input, entering them into the routing process. This pulse is also sent to the '*reset_routing*' input on the cell's trigger molecule via a gate. The output of this gate is enabled if it is the first working cell in the chain, thereby triggering the routing process. This system is required, as firing multiple trigger molecules on the tissue will result in multiple routing processes being instigated, wasting clockcycles (Thoma, 2004).

4.5 Simulations and Results

The cell design described above has been simulated in the presence of randomly generated faults using the POEtic design tool POEticmol (Thoma, 2004). As POEticmol simulates the behaviour of the POEtic device using the VHDL description from which the device is fabricated, accurate simulations of the effects of faults on system behaviour can be made.

A number of signal elements are randomly chosen and are forced into a fault condition during the simulation. Each signal element is randomly allocated a clock cycle number from the prespecified duration of the simulation upon which to become faulty. The fault model used is the 'stuck-at' fault, or single hard error. The number of faults forced in the simulation is set at a value high enough to guarantee a satisfactory yield of terminal cell faults, rather than at a value which is a realistic representation of fault rates for the real device with respect to the test application.

Some example results showing the behaviour of the embryonic cell design in response to randomly generated faults can be seen in Figures 6-13 to 6-15. The figures each show the input and output data for two cells, a working cell and a spare cell (both working, and spare cells are exposed to faults during the simulations), simulated over a number of runs of a fixed number of output words. Each run has a different randomly generated set of faults which are to be forced into the tissue. The design is reset after the end of each run and any faults forced into the tissue are removed. Each cell design is simulated under three conditions. In the first simulation, no faults are forced into the tissue (Figure 6-13). This generates the target output which the fault-tolerant system is aiming to achieve. In the second simulation, faults are forced into the tissue but the fault-detection and growth mechanisms are disabled (Figure 6-14). In the third simulation, the same faults are forced into the tissue with the fault-detection and growth mechanisms enabled (Figure 6-15).

In Figure 6-14, it can be seen that unprotected embryonic cell sustains a terminal fault in run 2. In run 2 of Figure 6-15 it can be seen that with the fault-tolerant systems enabled, the system has detected the induced fault and instigated apoptosis of the faulty cell and regrowth of the system. Data loss in the embryonic system is illustrated by the zero output between points a) and b) produced by the newly grown system. At point a) the system is repaired and fully functional, but its response to the input pulse previous to the fault being detected has been wiped by regrowth. By point b) the correct system response to this input has become negligible and a new input pulse stimulates the repaired embryonic system, producing incorrupt output data.

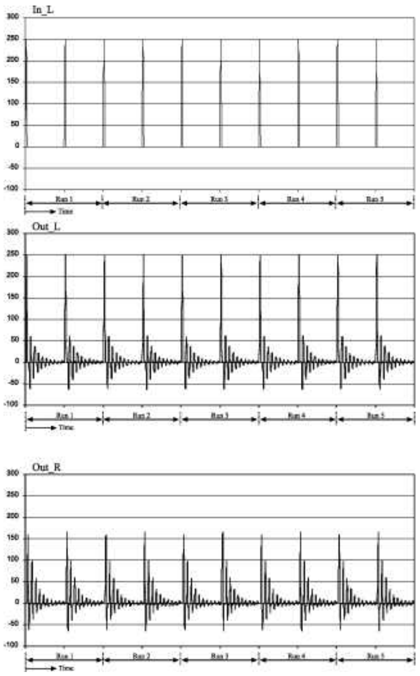


Figure 6-13. Fault-free cell I/O

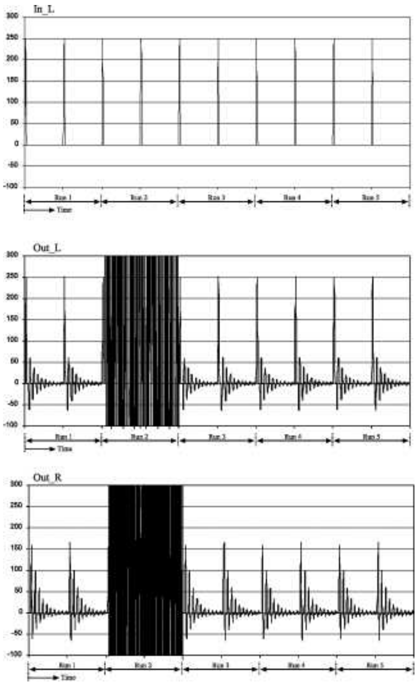


Figure 6-14. I/O with faults – fault-tolerance OFF

5. CONCLUSIONS

An embryonic cellular fault-tolerant mechanism has been successfully implemented in simulation on the POEtic tissue. The transparency of the process of mapping an embryonic design onto the POEtic architecture has also been demonstrated. Unlike embryonic implementations on generic FPGA architectures, which require complex stages of synthesis and careful tailoring of the embryonic architecture for the target device, compact embryonic designs can be built directly on the POEtic tissue at the molecular level. Results of preliminary simulations in the presence of randomly introduced faults show that the cell design is capable of successfully detecting, repairing and recovering from terminal faults in cell function.

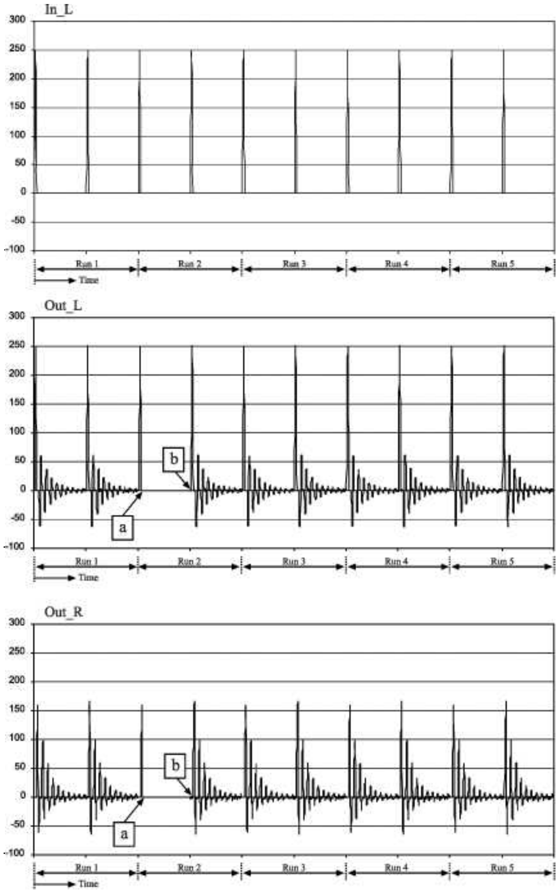


Figure 6-15. I/O with faults – fault-tolerance ON

However, the specific results shown here have much greater potential for the generic research field of evolutionary computation. The mechanisms designed into the POEtic devices, such as partial reconfiguration, dynamic routing and open evolution, will allow researchers in the future to investigate their evolutionary, developmental and learning algorithms and mechanisms on a hardware platform that will assist their work, not hinder it. This should allow greater advance to be made in these areas of research.

Acknowledgements

The authors would like to acknowledge the contributions of all the other members of the POEtic project (in particular, Yann Thoma, Gianluca Tempesti and Manuel Moreno) funded by the Future and Emerging Technologies programme (IST-FET) of the European Community, under grant IST-2000-28027 (POETIC). The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. The Swiss participants to this project are supported under grant 00.0529-1, by the Swiss government.

References

Canham, R. O., and A. M. Tyrrell. 2002. "A Multilayered Immune System for Hardware Fault Tolerance within an Embryonic Array", 1st International Conference on Artificial Immune Systems, Canterbury, September 2002.

Cooper, C. H. V., D. M. Howard, and A. M. Tyrrell. 2004. "Using GAs to Create a Waveguide Model of the Oral Vocal Tract", 6th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing, Coimbra, Portugal, 2880-288.

Costello, D. J., Jnr, and S. Lin. 2004. *Error Control Coding, 2nd Ed.* Prentice Hall, 2004.

Kajitani, I., et al. 1999. An Evolvable Hardware Chip and its Application as a Multi-function Prosthetic hand controller. In *Proceedings of 16th National Conference on Artificial Intelligence (AAAI-99).* American Association of Artificial Intelligence, 182-187. Menlo Park, CA:IEEE Press, 1999.

Lee, C. Y. 1961. "An Algorithm for Path Connections and Its Applications", *IRE Transactions on Electronic Computers*, Vol. EC-10, No. 3, September 1961, 346-365.

Mange, D., et al. 2000. "Towards Robust Integrated Circuits: The Embryonics Approach", *Proceedings of the IEEE*, vol. 88, no. 4, April 2000, 516-541.

Moreno, J.-M., et al. 2004. Hardware Realization of a Bio-inspired POEtic tissue. In *Proc. 2004 NASA/DoD Conference on Evolvable Hardware*, 237-244. IEEE Computer Society.

Ortega, C., et al. 2000. "Embryonics: A Bio-Inspired Cellular Architecture with Fault-Tolerant Properties", *Journal of Genetic Programming and Evolvable Machines*, Vol. 1, No. 3, July 2000, 187-215.

Thoma, Y., et al. 2003. A Dynamic Routing Algorithm for a Bio-Inspired Reconfigurable Circuit. In *Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FP'03), LNCS*, Vol. 2778, 681-690. Springer-Verlag.

Thoma, Y., et al. 2004. "POEtic: An Electronic Tissue for Bio-Inspired Cellular Applications", *BioSystems*, 74: 1-3, 2004, 191-200.

Thoma Y., et al. 2004. "Prototyping with a bio-inspired reconfigurable chip", 15th International Workshop on Rapid System Prototyping (RSP 2004), Geneva, Switzerland, June 2004.

Tyrrell, A. M., et al. 2003. "POEtic Tissue: An Integrated Architecture for Bio-Inspired Hardware", Proceedings of 5th International Conference on Evolvable Systems, Trondheim, 129-140, March 2003.

Chapter 7

EVOLVABLE ANALOG LSI

a yield enhancement method with area and power dissipation reductions

Masahiro Murakawa, Yuji Kasai, Hidenori Sakanashi,
and Tetsuya Higuchi
Advanced Semiconductor Research Center, National Institute of Advanced Industrial Science and Technology, Email: m.murakawa@aist.go.jp

Abstract: We propose a concept of evolvable analog LSIs and apply it to (1) Intermediate Frequency (IF) filters and (2) Image-Rejection Mixers (IRM). The developed LSI for IF filters have attained (1) a 63% reduction in filter area, (2) a 26% reduction in power dissipation, compared to existing commercial products using both the same process technology and filter topology, and (3) a yield rate of 97%. The developed chip is calibrated within a few seconds by the genetic algorithm (GA). The developed microwave circuit for IRM is also calibrated by GA to outperform a circuit adjusted by an experienced engineer. Our calibration method, which can be applied to a wide variety of analog circuits, leads to cost reductions and the efficient implementation of analog LSIs.

Key words: analog LSI, yield, power dissipation, die area, process variation, IF filter, Gm–C filter, image-rejection mixer, microwave circuit.

1. INTRODUCTION

An inherent problem in implementing analog LSIs or ICs is that the values of manufactured analog circuit components often differ from the precise design specifications. Such discrepancies cause poor yield rates, especially for high-end analog circuits. For example, in intermediate frequency (IF) filters, which are widely used in cellular phones, even a 1% discrepancy from the center frequency is unacceptable. It is therefore necessary to carefully examine the analog LSIs, and to discard any which do not meet the specifications. In this chapter, a GA-based calibration method, which can correct these variations in the analog circuits values, is introduced. Using this method provides us with three advantages:

1. *Enhanced yield rates.* If an analog LSI is found not to satisfy specifications, then the GA is executed at the wafer-sort testing to alter the defective analog circuit components in line with specifications. Once calibrated, the circuit values are fixed by the laser trimming.
2. *Smaller circuits and less power consumption.* The conventional approach to overcome discrepancies of the component values in analog LSIs has been to use large components. However, this requires larger spaces, and, in turn, means higher manufacturing costs and greater power consumption. In contrast, with our approach, the area size of the analog circuits can be made smaller, because chip performance can be calibrated after production with the GA software.
3. *Integration of peripheral circuits to LSIs.* As process technology allows for smaller and smaller implementations, it is possible to integrate more peripheral analog circuits in addition to digital signal processing circuits within a single LSI. However, because it is extremely difficult to integrate analog circuits with high-end specifications due to process variations, these analog circuits are usually made as separate components such as ceramic filters. In contrast, with the proposed method it is possible to integrate these analog circuits on a single LSI, resulting in cost reduction and circuit area reduction.

We have applied our method to an integrated Gm–C IF filter (Murakawa, et al., 1998), where performance can be calibrated by GAs to enhance yield rates. In the developed chip, there are 39 Gm components, which may differ from target values by as much as 10%. However, the GAs effectively absorb the variations in the Gm values through calibration circuits. Although it has been difficult to reduce the size of Gm–C IF filters with existing CMOS technology, while maintaining acceptable yield rates due to process-dependent variations in device parameters, with our method we have successfully achieved both a 63% reduction in filter area and a 97% yield rate. In calibration experiments, we were able to successfully correct 29 out of 30 test chips to satisfy IF filter specifications, although all failed to meet these specifications prior to GA calibration. The 63% reduction in filter area has also led to a 26% reduction in power dissipation compared to existing commercial products. Existing commercial products and the developed chips have the same filter topology, i.e. an 18th order linear filter consisting of three cascaded 6th order cells, and are fabricated with the same technology, i.e. 0.6μm 2-poly 2-metal CMOS process. However, because the products adopt a conventional master-slave-based tuning system, the analog components in the chips must be large to achieve acceptable yield rates (over 90%).

Also, we have applied our method to Image-Rejection Mixers (IRM) (Kasai, et al., 2000). IRM is a microwave circuit of which design and production are plagued by the serious problems posed by the behavior of distributed-constant circuits in the microwave range. Parasitic capacitances and inductances in these circuits, which are extremely difficult to predict at the design stage, cause variations in the performance levels of the circuits. However, the GAs automatically calibrate the circuits to optimize the performance. Although experienced analog engineers can generally improve the rejection ratio to 55 dB for a circuit by manual adjustments, the developed circuit has achieved a rejection ratio of more than 69 dB.

This chapter is organized as follows: in Section 2, calibration method using GAs is described. Section 3 introduces the developed LSI for IF filters, and describes the results of calibration experiments. In Section 4, we outline the architecture of our image-rejection mixer circuit, and detail an adjustment experiment for the image-rejection mixer circuit, which compared the adjustment performances for the GA, a hill-climbing method and manual adjustment. The future work and applications are also discussed in Section 4 before the final summary Section 5.

2. CALIBRATION METHOD USING GENETIC ALGORITHMS FOR ANALOG LSIS

In analog LSIs, the characteristics of analog circuits such as resistors and capacitors vary widely due to process variation and environmental influence. As the process technique allows for finer implementations, these variations pose an even greater problem. This is especially problematic for conventional design methods, as it is impractical to incorporate sufficiently large performance margins. Therefore, to improve the yield rate, designers need to have a detailed understanding of circuit behavior and devise dedicated circuits to correct for the variations. However, theoretical design procedures for such correction have yet to be established. As a result, designers depend heavily on accumulated know-how.

In order to compensate for these variations, some LSIs are manually adjusted after product, with a method called trimming (Gray and Meyer, 1984). In laser trimming, some resistors in the analog LSIs are partially burnt off using laser beams to adjust their values. However, such manual adjustment cannot be applied to mass production if the number of adjustment points is large. Moreover, this kind of manual adjustment method requires the skills of analog experts, who are in scarce supply due to recent advances in digital circuits.

To overcome these difficulties, we propose a calibration method using GAs for analog LSIs. We acknowledge that the values of analog components will

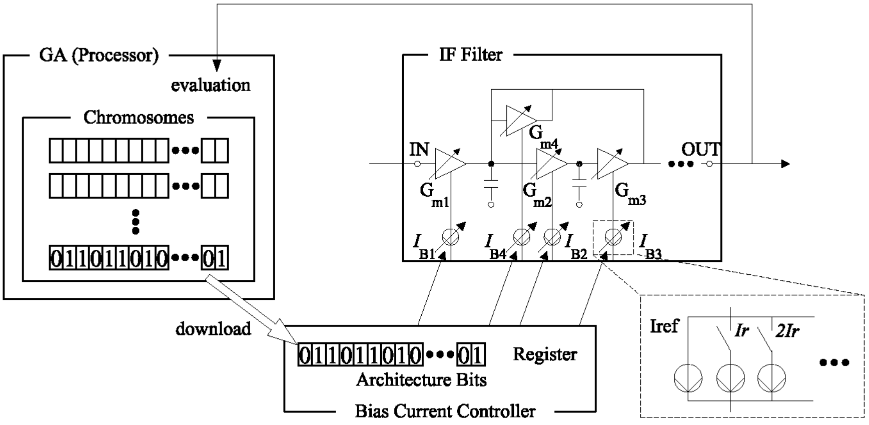


Figure 7-1. Calibration method using genetic algorithms

deviate from specifications and instead of raising operational margins, we include a number of adjustable circuit parameters which are adjusted by genetic algorithms in order to calibrate the LSI to specifications. Two applications of our method have been developed on integrated Gm–C IF filters and a microwave circuit for IRM.

Figure 7-1 illustrates the GA calibration method for the developed Gm–C IF filter, consisting of 39 Gm amplifiers connected in a cascade fashion. Each Gm amplifier was designed so that its transconductance can be varied by altering a bias current fed to the Gm amplifier. Each bias current can be finely adjusted according to a binary bit string written to a register for the Gm amplifier. Collectively, the 39 binary bit strings that determine the 39 bias currents are referred to as the architecture bits. The optimal architecture bits for a chip is the set of binary bit strings that will minimize the effects of process-dependent variations in the circuit parameters and provide the target filter response.

However, because of the sheer size of the search space representing all possible permutations of the variables, it is impossible to manually determine the optimal architecture bits for each chip. For example, when the bias current control for each Gm amplifier consists of 6 bits, the search space will represent $2^{39\times6} \approx 10^{70}$ permutations. GAs, which can be executed on either an LSI tester or PC, are extremely efficient at solving this kind of difficult optimization problem.

For the calibration of analog LSIs, the key is in treating the architecture bits as GA chromosomes. The GA software seeks the optimal architecture bits for each chip after production. Figure 7-2 shows a flowchart of the GA calibration for the IF filter chip. Every chromosome is downloaded into the control regis-

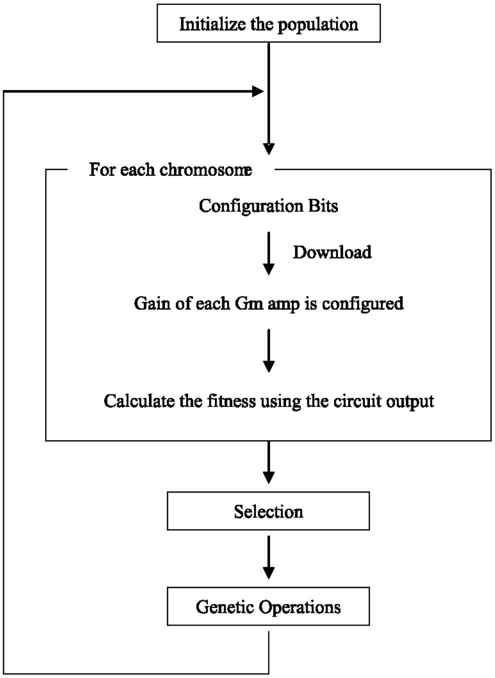

Initialize the population
For each chromosome
Configuration Bits
Download
Gain of each Gm amp is configured
Calculate the fitness using the circuit output
Selection
Genetic Operations

Figure 7-2. A flowchart of the GA calibration

ter in the filter LSI. A fitness function is calculated by giving test inputs to the filter and observing the outputs. Transconductance values are varied to simultaneously optimize both gain (frequency response) and group delay. This kind of simultaneous multiobjective optimization, which is an important feature of GA, is difficult with other approaches.

3. IF FILTER LSI FOR GA CALIBRATION

This section explains the developed IF filters in detail, before description for results of calibration experiments. The ideal frequency response (Proakis, 1988) and typical specifications for an PDC IF filter are shown in Figures 7-3 and 7-4. The center frequency is 455kHz with a bandwidth of 21kHz. The frequency response should be attenuated to 30dB at 25kHz and to 60dB at 50kHz from the center. The most severe requirement is that the level of attenuation must be 3dB at the frequencies of 444.5 ± 1.5 kHz and 465.5 ± 1.5 kHz. This requirement cannot be satisfied if there is so much as a 1% shift from the center frequency. However, by using the GA, it is possible to calibrate the chip to conform to the target specifications and thus enhance yield rates. This is achieved by adjusting for the shift in the center frequency due to process variations in order to meet this 3dB attenuation requirement.

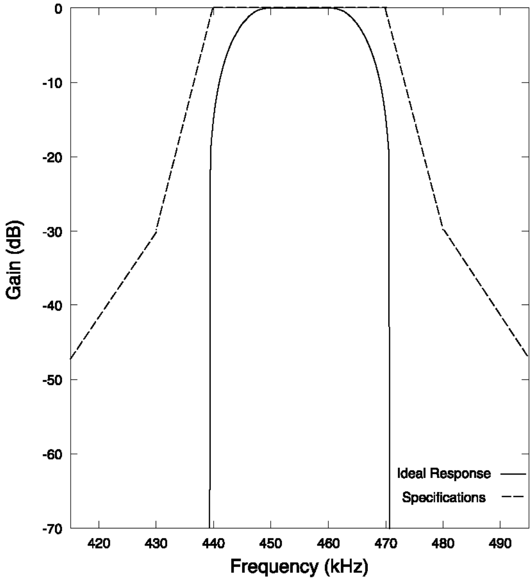


Figure 7-3. Ideal frequency response and specifications of an IF filter

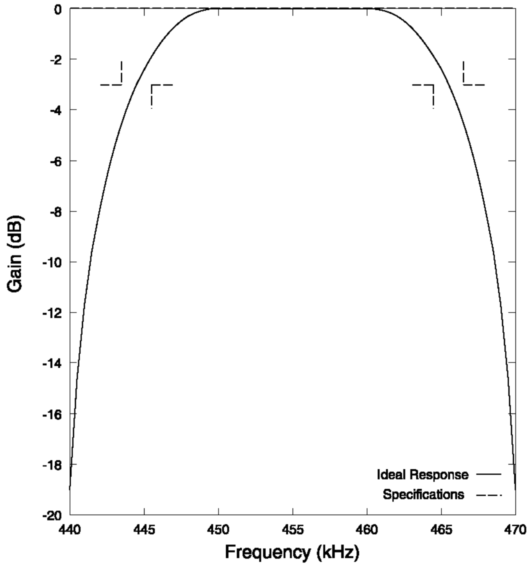


Figure 7-4. Magnification of Figure 7-3

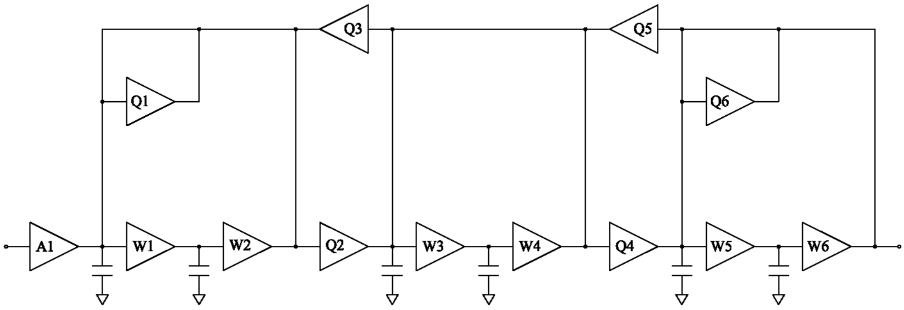


Figure 7-5. 6th order band pass filter section

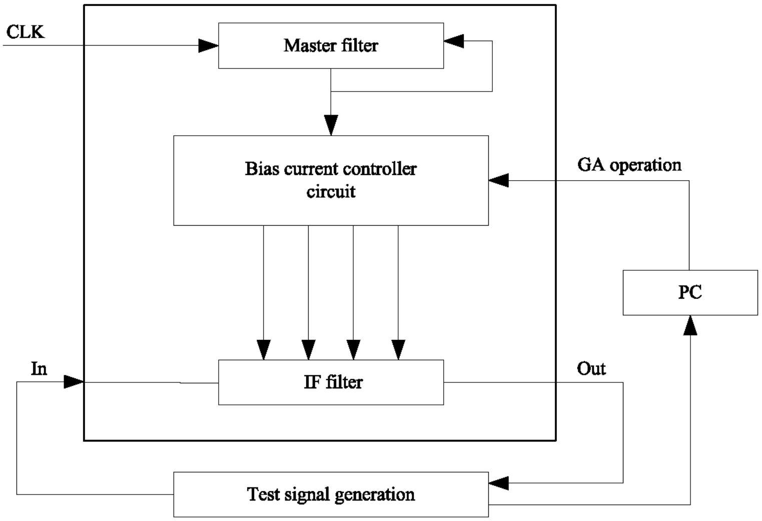


Figure 7-6. System architecture for the developed IF filter

3.1 Filter Architecture

Although many filter structures could provide a 455kHz center frequency and 21kHz bandwidth, we have adopted an 18th order linear filter organization consisting of three cascaded structures (Adachi, et al., 1994). Each of the three filter blocks has an all pole 6th order Gm–C leapfrog structure shown in Figure 7-5. The architecture for the filter is depicted in Figure 7-6. As in most Gm–C filters, a master-slave approach is used to tune the Gm values, with the master circuit being the filter in a PLL. This IF filter has 39 parameters

Table 7-1. Ideal parameter values

Parameter	*Value*
$w_1, \ldots, w_{18}$	$2\pi f_0$
$Q_1, \ldots, Q_{18}$	$2\pi f_0/22$
a_1, a_2, a_3	$2\pi f_0/11 \quad (f_0 = 455 \times 10^3)$

in total, of which 18 are related to the center frequency ($w_1, \ldots, w_{18}$), 18 for bandwidth ($Q_1, \ldots, Q_{18}$), and 3 for filter gain (a_1, a_2, a_3). Table 7-1 shows the target values for these parameters. In the integrated filter circuit, these parameters correspond to the transconductance values of the Gm amplifiers. As the chip was designed to be smaller, the transconductance values can deviate by as much as to 10% from the target values.

Although master-slave controllers can be used to correct for variations such as those due to temperature, where all the transconductance values deviate equally from the target values, they are not, however, capable of adjusting for variations in order to satisfy design specifications, where the variations in transconductance values due to the manufacturing process will be random. Moreover, these controllers are unable to correct for group delay, which is in conflict with gain optimization. To overcome these difficulties, the developed chip utilizes GAs to compensate for these variations. While GAs can execute parameter optimization even when the parameters are interdependent, optimization is faster when the parameters are independent. In order to speed up the calibration times for the IF filter, two different kinds of Gm amplifiers, shown in Figure 7-7, are used. The designs for Gm amplifiers for bandwidth (Figure 7-7(c)) and those for center frequency (Figure 7-7(d)) differ with the common-source pair circuits replaced with source-coupled pair circuits. If common-source pair circuits were also used in the Gm amplifiers for bandwidth, it would not be possible to control bandwidth Gm values independently of the center frequency Gm values, because the input voltages for bandwidth Gm amplifiers would be common to those for the center frequency Gm amplifiers. Such interdependence would lead to longer calibration times.

3.2 GA Calibration for the Filter

The GA chromosome for the IF filter is 39 × 6 bits, with each 6-bit string determining the transconductance value of a Gm amplifier. The 6 bits correspond to the switches in a DAC (Figure 7-8). The first bit determines whether the other 5 bits correspond to either Sw1–Sw5 or Sw6–Sw10 in Figure 7-8. These switches subtly vary the bias current (bias in Figure 7-7) fed to a Gm amplifier. Current sources are provided for I_r, $2I_r$, $4I_r$, $8I_r$ and $16I_r$ in Figure 7-8 (i.e.

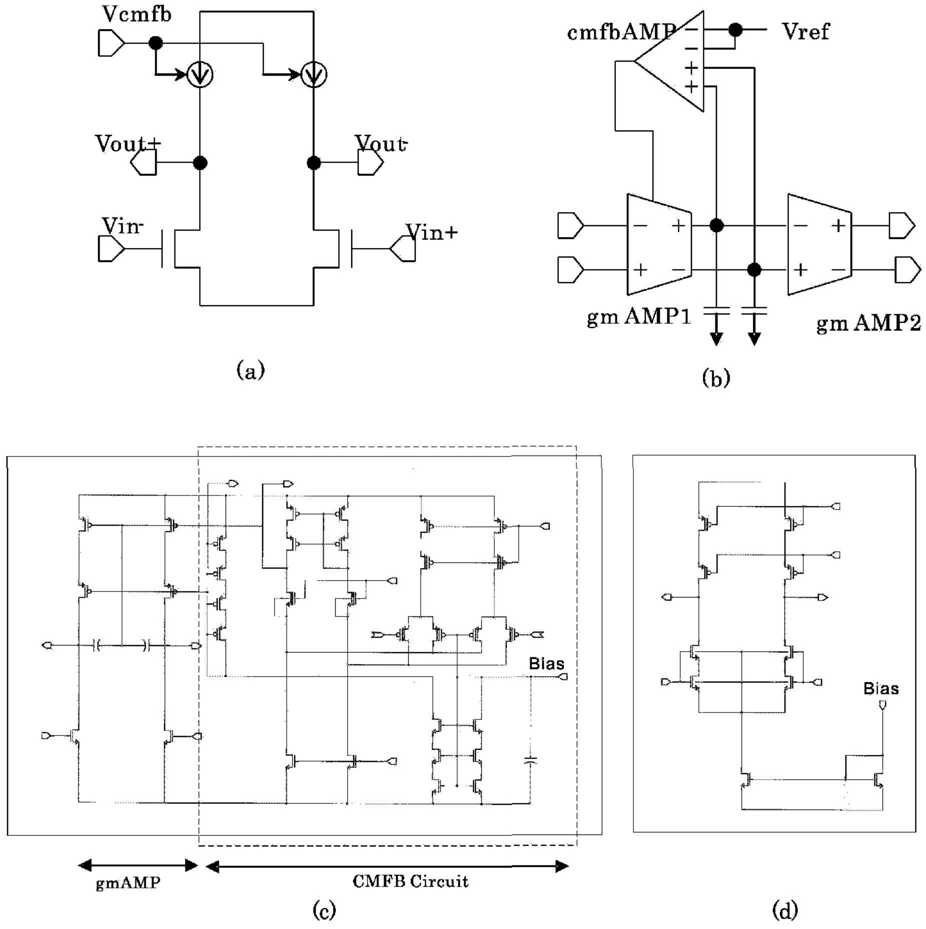


Figure 7-7. Gm AMPs (a) Prototype Gm AMP (b) Gm/cmfb (common feedback) AMP (c) Gm AMP for center frequency (d) Gm AMP for bandwidth

small currents for calibration that are proportional to the I_{ref} generated at the PLL circuit). The I_r is set to $0.0039 I_{ref}$. The chromosome 111011, for example, would set Sw1, Sw2, Sw4, and Sw5 to ON, so that the bias current is raised to be $I_{ref} + 16Ir + 8Ir + 2Ir + Ir$, i.e. $(1 + 27 \times 0.0039)I_{ref} = 1.1053 I_{ref}$. Taking the chromosome 000111 as a further example, however, Sw8, Sw9, and Sw10 would be ON with the bias current reduced to be $I_{ref} - 4Ir - 2Ir - Ir$, i.e. $(1 - 7 \times 0.0039)I_{ref} = 0.9727 I_{ref}$.

The fitness function for the GA is defined as follows:

$$\text{fitness} = \sum_{i=1}^{6} w_i |O(f_i) - S(f_i)| + w_g |G_{max} - G_{min}| \tag{7.1}$$

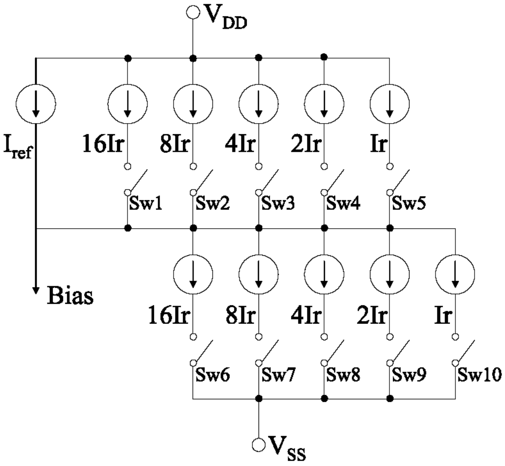


Figure 7-8. DAC for the bias current controlled by GAs

$$G_{max} = \operatorname*{Max}_{i=1}^{5} G_i, \ G_{min} = \operatorname*{Min}_{i=1}^{5} G_i, \ G_i = (\Phi(f_{i+1}) - \Phi(f_i))/(f_{i+1} - f_i) \tag{7.2}$$

This fitness function consists of a weighted sum of gain deviations and group delay differences measured around the center frequency. Here, $S(f_i)$ is the ideal gain. $O(f_i)$ and $\Phi(f_i)$ are the gain and the phase obtained by the filter chip at frequency f_i. G_{max} and G_{min} are the maximum and the minimum of the group delay in the passband. w_i and w_g are the weighing coefficients for the gain and the group delay.

While the group delay requirement for the filters is 20 microseconds, the design estimation for our filter using an 18th order filter is 24 microseconds. While this indicates that further calibration is required to meet this specification, it should be noted that even though it is impossible with linear filter theory to obtain optimal solutions for both group delay and frequency response (due to their trade-off relation), GA calibration using the fitness function is capable of satisfying both of these specifications.

3.3 Developed IF Filter Chip

The chips were fabricated in a 0.6μm 2-poly 2-metal CMOS process. The supply voltage is 5.0V and the die area is 3.0 × 3.2mm^2. Figure 7-9 shows a microphotograph of the die. In terms of circuit design, there are a number of important consequences of utilizing of GAs within this IF filter chip. The first is the very compact implementation of the Gm amplifier which GA calibration makes possible. We can reduce current dissipation and the transconductance

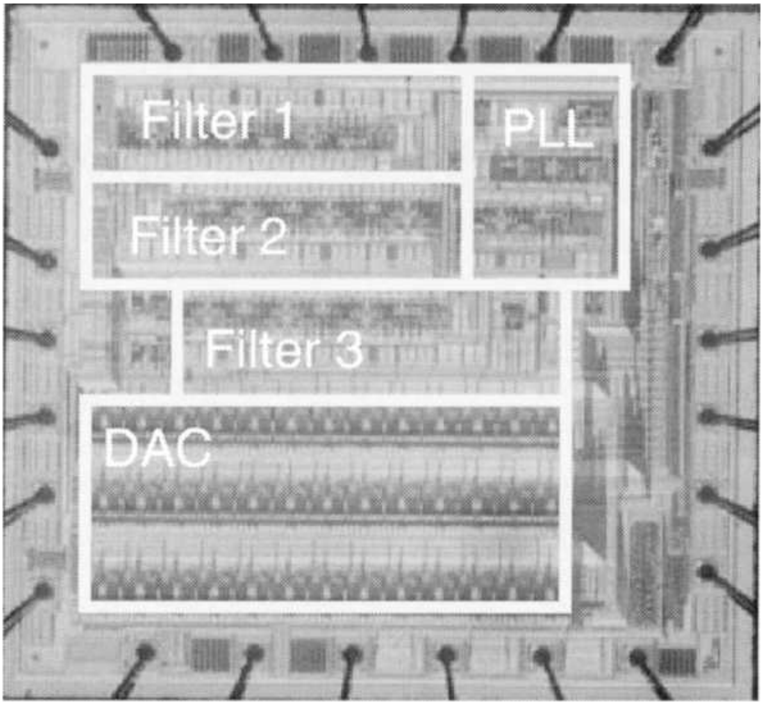


Figure 7-9. Die microphotograph

of the Gm amplifier because the transistor channel width can be shortened. The second related point is that because the center frequency is determined by the ratio of capacitance to Gm value, capacitance is small with smaller Gm amplifiers. The third important point is that because the GA calibration is accomplished using a number of Gm amplifiers, it is possible to achieve satisfactory levels of precision for the current sources with relatively simple bias current circuits.

As a result of this new design method employing GA calibration, filter area was reduced by 63% compared to existing commercial products (from 3.36mm^2 to 1.26mm^2), resulting in a 26% reduction in power dissipation (from 3.4mA to 2.5mA, 38% reduction for the Gm amplifiers; 58μA $\rightarrow$ 36μA).

Because an area of 1.81mm^2 was initially devoted for the calibration circuits (due to excessive provision of 39×6-bit registers and 39 DACs for calibration) in the test chips referred to above, the total chip area was only reduced to 91% (from 3.36mm^2 to 3.07mm^2). However, we have subsequently achieved a 100% yield rate with GA calibration experiments involving 18 parameters (3 bits each). As a result, a commercial chip has been designed with adjustment circuitry for 18×3-bit registers and 18 DACs within an area of 0.41mm^2. Thus, even including the area required for calibration, the total chip area is reduced

Table 7-2. Comparison of the circuit area

	Filter circuit area (a)	Calibration circuit area (b)	(a)+(b)
Existing Products	3.36mm^2	—	3.36mm^2
Developed Chip	1.26mm^2	1.81mm^2	3.07mm^2
Commercial Version	1.26mm^2	0.41mm^2	1.67mm^2

to 49% compared to conventional filter designs. This result is summarized in Table 7-2.

3.4 Calibration Experiments

We used a population of 20 chromosomes, of which chromosome length was 234. The fitness in the GA was defined using the gain obtained by the chip at frequencies 444kHz, 446kHz, 453kHz, 455kHz, 464kHz, and 466kHz. The target response is obtained by root-Niquist conditions; the ideal responses at the 6 frequencies are −3.7dB, −1.4dB, 0dB, 0dB, −1.4dB, and −3.7dB (see Figure 7-4).

We conducted calibration experiments for 30 real chips. On average, the GA was able to find an optimal architecture bit and terminate after 100 measurement iterations within a few seconds on an LSI tester. Clearly GA calibration does not represent an obstacle to mass-production. Figure 7-10 shows the frequency responses and group delays for the best architecture bits obtained for a chip. In the calibration experiments, 29 out of the 30 tests chips could be calibrated to satisfy specifications, although none of these chips conformed prior to calibration (i.e. 97% yield rate), and the remaining chip was successfully calibrated with additional iterations. This result is consistent with statistical simulations for GA calibration, in which 952 out of 1000 virtual chips were successfully calibrated.

For comparison, we conducted calibration experiments with a conventional hill-climbing method instead of the GA. A run terminated after fitness was evaluated 100 times. The result of this was that only 17 chips (57%) could meet the specifications. These results suggest the effectiveness of the GA in avoiding the local minimums in the evaluation (fitness) function.

4. IMAGE-REJECTION MIXER FOR GA CALIBRATION

After a brief description of image-rejection mixers (Archer, et al., 1981; Minakawa and Tokumitsu, 1993; Baumberger, 1994; Rudell, et al., 1997), the

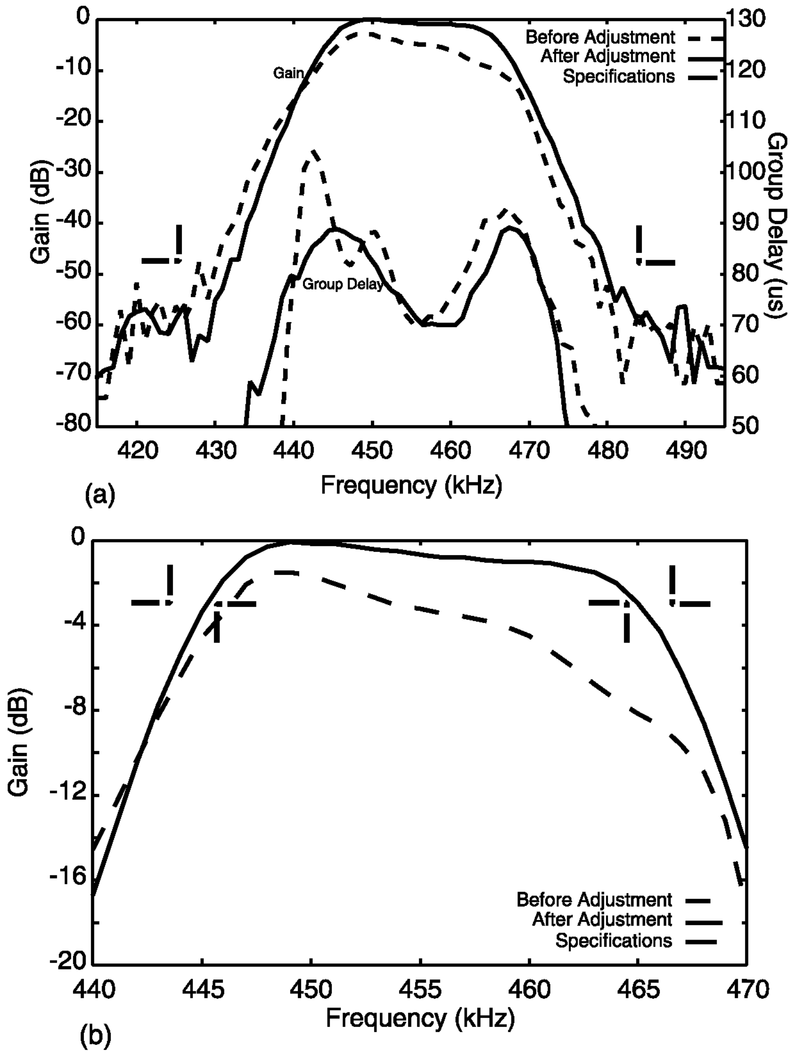


Figure 7-10. (a) Measured gain/delay response for the developed chip and (b) Magnification of (a)

developed image-rejection mixer circuit and its experimental results are described in this section.

Broadcast signals received by an antenna are processed by image-rejection mixers, which select the frequency for a particular TV channel and convert it to an intermediate frequency. For example, with US communication satellites, TV programs are broadcast at frequencies between 3.7 GHz to 4.2 GHz. Image-rejection mixers process incoming signals by selecting the desired signal frequency f_{si} for a TV channel with respect to a variable local oscillator

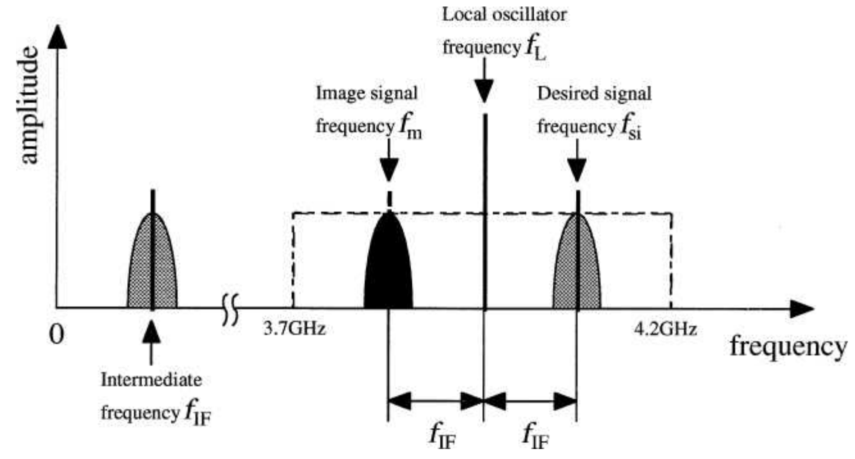


Figure 7-11. Relation between desired signal frequency f_{si}, local oscillator frequency f_L, intermediate frequency f_{IF}, and image signal frequency f_m

(LO) frequency f_L, and converting the f_{si} to an intermediate frequency f_{IF} (see Figure 7-11).

However, because the intermediate frequency f_{IF} is the difference between the desired signal frequency f_{si} and the local oscillator frequency f_L, the conversion will also include a mirror image frequency f_m, at an equal distance from the local oscillator frequency in the opposite direction. To provide a high quality intermediate frequency signal, the mixer has to suppress this image frequency as much as possible (that is, maintain a high image rejection ratio).

However, such mixers are very difficult to design because of the nature of distributed-constant circuits in the microwave range. These circuits are also difficult to adjust for optimum performance, even for experienced engineers. These related problems make development costs of microwave circuits very high. In order to overcome these problems, we propose an evolvable microwave circuit where performance adjustment is carried out automatically by a GA.

4.1 Image-rejection Mixer for GA Calibration

Figure 7-12 shows the organization of the developed image-rejection mixer system. It consists of two mixers, a divider, a divider/phase-shifter, and a phase-shifter/combiner. If this circuit were set for a desired frequency fsi of 2.441 GHz (assumed signal bandwidth 200 kHz), with a local oscillator frequency f_L of 2.440 GHz, then the task for the image-rejection mixer would be to convert the 2.441 GHz signal to an intermediate frequency (IF) of 1 MHz,

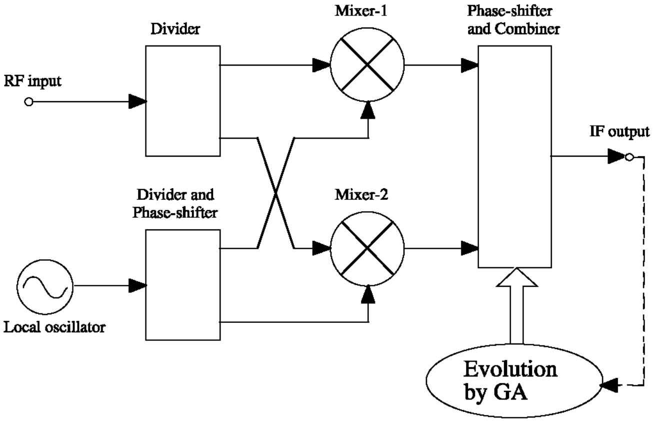


Figure 7-12. Block diagram of the developed image-rejection mixer

while rejecting the image frequency signal fm of 2.439 GHz. In order to improve the image-rejection ratio, the phase-shifter/combiner circuit would be calibrated by GA.

Although the absolute performance levels of the developed image-rejection mixer are not in themselves particularly remarkable, and it is undoubtedly potential to improve on the design of the mixer which was developed using off-the-shelf discrete devices, the primary purpose of this development is to present experimental data to show that the performance of microwave circuits can, indeed, be calibrated autonomously by GA.

The operation of the image-rejection mixer is as follows. The divider receives the microwave signals (RF signal (Radio Frequency) in Figure 7-12), which it feeds to the two mixers in phase. The divider/phase-shifter distributes the local oscillator signal to the mixers with a phase shift of 90 degrees. Based on the RF signal and the LO signal, the two mixers produce two IF signals which are input into the phase-shifter/combiner, which then composes an output signal with a phase shift of 90 degrees. If the values for the phase shift and the amplitude in these signal transformations match the design specification exactly, then the image frequency is canceled completely. However, due to variations in the analog devices and parasitic capacitances and inductances, it is impossible to control these phase-shift and amplitude values completely, and as a result signal quality is degraded (Archer, et al., 1981; Rudell, et al., 1997).

The proposed image-rejection mixer, therefore, employs a GA to execute adjustments to circuitry to control the parameters of the device and to match the phase and amplitude to the exact values of a target signal. Figure 7-13 is a detailed circuit diagram of the image-rejection mixer. The divider and

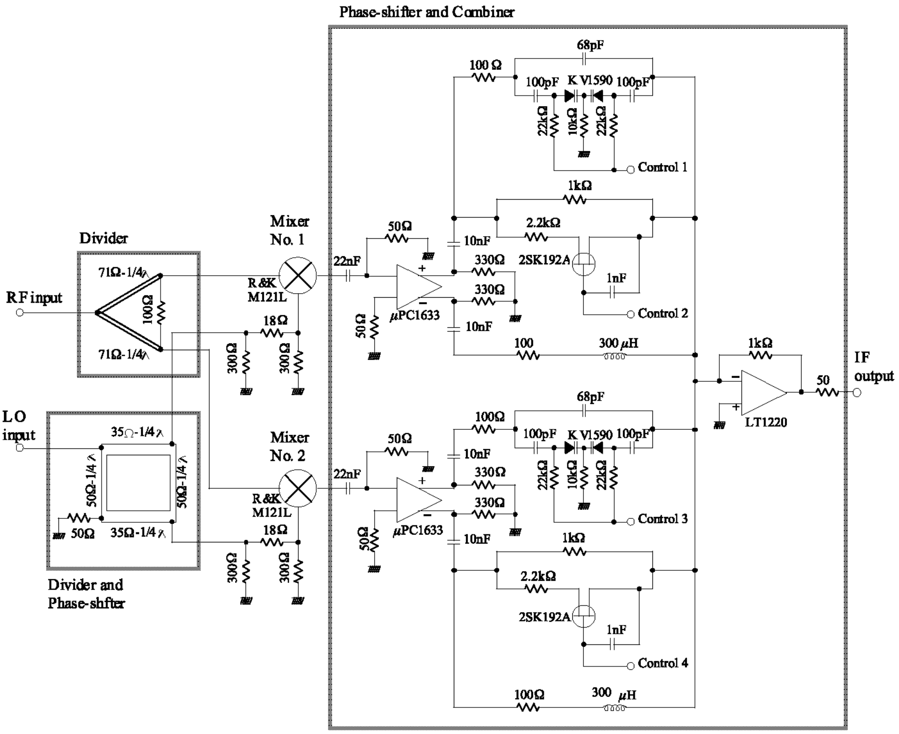


Figure 7-13. Circuit diagram of the developed image-rejection mixer

the divider/phase-shifter consist of microstrip lines (a Wilkinson coupler and a branched hybrid circuit, respectively). The two mixers (R&K and M121L) both produce intermediate frequency signals. The phase-shifter/combiner is composed of a number of discrete elements.

4.2 GA Calibration for the Mixer

A GA is executed to adjust the device parameters for the phase-shifter/combiner circuit. Figure 7-14 is a schematic representation of the phase-shifter/combiner. Two pairs of capacitors, resistors, and inductors shift the phase of the IF signals. By varying C1, C2, R1, and R2, the optimum phase difference and amplitude ratio between two input signals can be determined when the values of L1 and L2 are fixed.

Figure 7-15 shows the circuit for the capacitors C1 and C2 utilizing variable-capacitance diodes. In order to change the capacitance, a binary bit string (i.e., a chromosome for the GA) is input to a digital-to-analog converter (DAC). This represents a control voltage that is used as a reverse bias in the variable-capacitance diode. Similarly, the resistance values for the resistor circuits R1

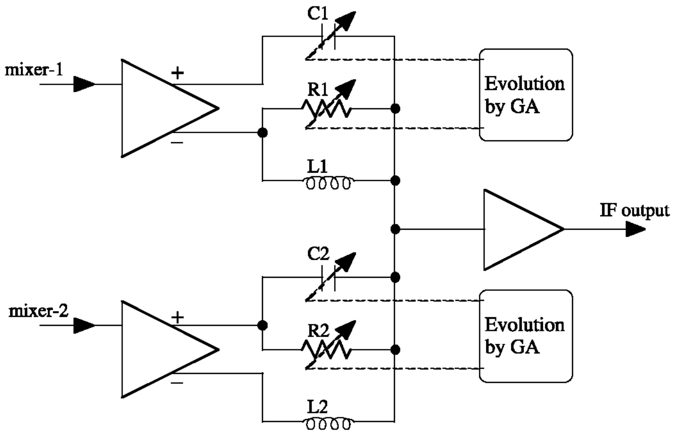


Figure 7-14. Schematic representation of the adjustable phase-shifter/combiner in the image-rejection mixer

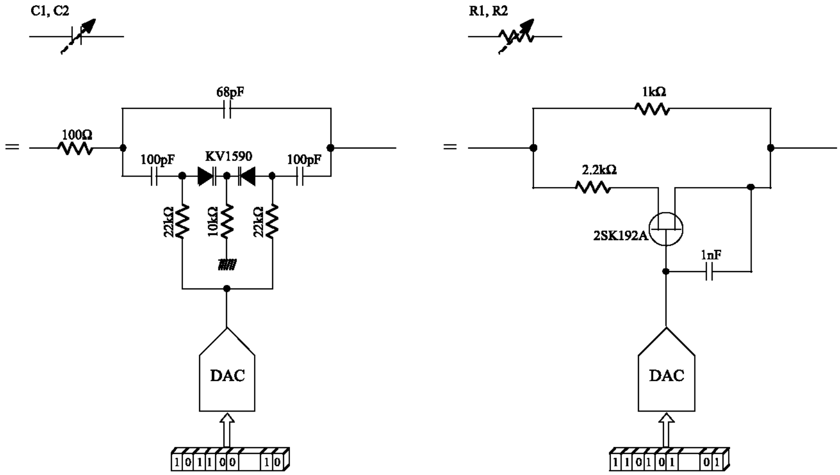


Figure 7-15. Actual circuit for the adjustable capacitors C1, C2, R1, and R2

and R2 can be controlled by the GA (Figure 7-15). The output voltage of the DAC varies the resistance of the field effect transistor (FET). The lengths of the chromosomes for C1, C2, R1 and R2 are all 12 bits.

Adjustments to C1, C2, R1, and R2 by the GA can compensate for undesirable phase shifts caused by errors in microstrip lengths and width variations, and, thus, image frequency signals can be eliminated.

The GA operators utilized in this application include a uniform-crossover, with a crossover rate of 0.9, a bit-flip mutation, with a mutation rate of 0.05 per

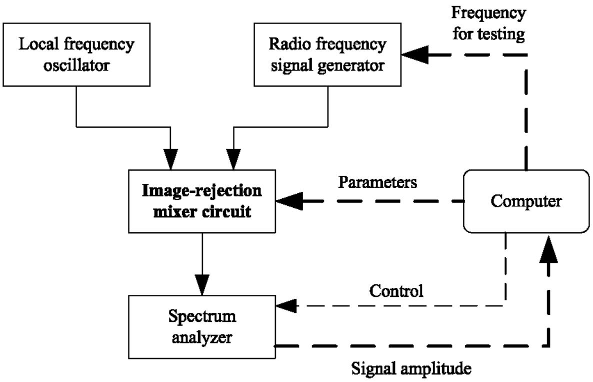

Local frequency oscillator
Radio frequency signal generator
Frequency for testing
Image-rejection mixer circuit
Parameters
Computer
Control
Spectrum analyzer
Signal amplitude

Figure 7-16. Block diagram of the setup for calibrating the image-rejection mixer

bit, and a tournament selection, with a tournament size of 2 and a population size of 30. The fitness function for the GA is the summation of the signal intensities at three frequencies, namely, at the center and two edges of the bandwidth for a given intermediate frequency. When this summation is greater, the image rejection ratio is higher. For the calibration experiments, which we discuss in the next subsection, this fitness function was the summation of the intensities of three frequencies (0.9 MHz, 1.0 MHz, and 1.1 MHz) based on the intermediate frequency of 1.0 MHz.

4.3 Calibration Experiments

As shown in Figure 7-16, the experimental setup consisted of the image-rejection mixer circuit to be calibrated, a local oscillator (LO), a microwave signal generator (HP 33120A and a frequency multiplier), a spectrum analyzer (HP 8591E with Keithley 2002), a DAC interface, and a PC. The local oscillator and the microwave signal generator were connected to the mixer input. The microwave signal generator provided the test signal (1mW) with a specified frequency according to instructions from the computer. The output signal from the mixer was monitored by the spectrum analyzer, which measured the amplitude of the output signal and transferred the values to the computer. The computer calculated the image-rejection ratio and ran the GA to determine new parameters. The new evolved parameters were sent as a control voltage for the mixer circuit by the DAC.

We conducted an adjustment experiment to compare three approaches on improving the performance level of an image-rejection mixer: GA, IHC or manual adjustments by an experienced engineer. The results, which are summarized in Table 7-3, indicate that the performance after adjustment was better for the GA than for IHC and the manual adjustments.

Table 7-3. Comparison of calibration results

Search Method	Image-Rejection Ratio (dB)	
	Band Center	Band Edge
GA	69.32	38.61
IHC	41.85	35.18
Human Engineer	55	35
No Adjustment	23.40	23.38

Table 7-4. Experimental results for the IHC

	Image-Rejection Ratio (dB)	
	Band Center	Band Edge
Average	41.85	35.18
Maximum	60.05	39.79
Minimum	27.49	25.43
σ^2	71.33	12.96

IHC repeatedly makes hill-climbing searches from different starting points selected randomly. We executed the IHC 21 times. Table 7-4 presents the results of the trials. The average image-rejection ratio was 41.85 dB at the center of the IF band and was 35.18 dB at the edge of the IF band. The adjustments by the human engineer were made on the same circuit. The engineer adjusted DAC values, while monitoring the spectrum analyzer display and substituting the test frequencies.

Figure 7-17 shows the average image-rejection ratio curve for all 3 GA trials as a function of the number of generations. The image-rejection ratio at the band center was 69 dB, which exceeds the best solution by the IHC search by 9 dB. At the edge of the IF band, the image-rejection ratio obtained by GA was about 3 dB higher than the ratio for the IHC search. These results, therefore, indicate that the GA outperformed both the IHC method and the manual adjustments by an experienced engineer in improving the rejection ratio of the image-rejection mixer.

These results demonstrate that a GA is capable of successfully and autonomously adjusting the performance of image-rejection mixers. A further evaluation criterion for image-rejection mixers is the width of the frequency band where a high image rejection ratio is maintained. For example, as Figure 7-18 shows, although the image-rejection ratio is at a maximum near the center of the IF band, it decreases as the frequency approaches the edge of the

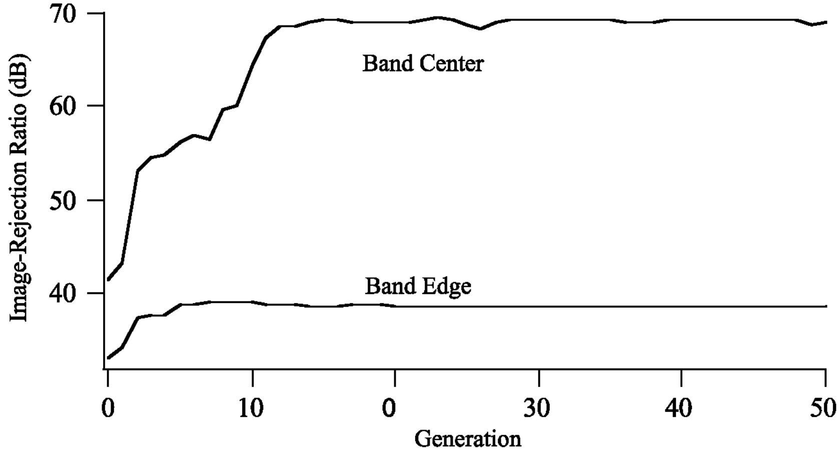


Figure 7-17. Averaged image-rejection ratio curve for the GA evolutions. The upper curve and the lower curve are the measured image-rejection ratio at the center and at the edge of the IF band, respectively

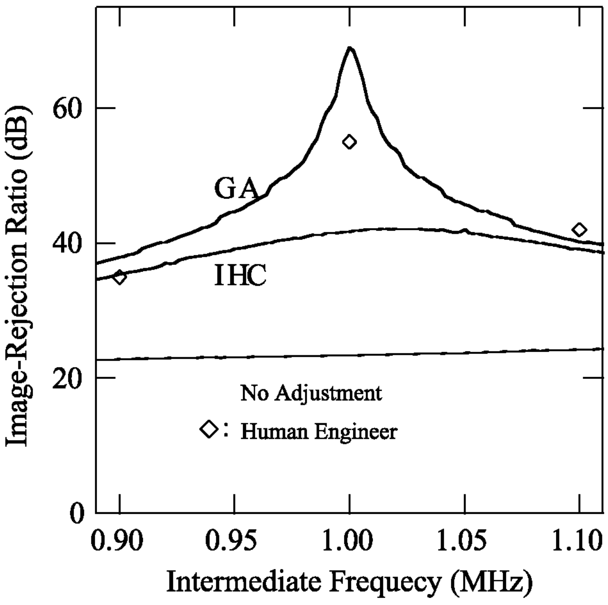


Figure 7-18. Characteristics of the image-rejection ratio as a function of intermediate frequency

IF band. However, this point was not considered in the present design, being beyond the objective of this experiment which has been to present evidence that GA can adjust the performance of microwave circuits. This is something that we shall deal with in designing our next image-rejection mixer, by increasing the signal bandwidth used in the image-rejection ratios.

4.4 Discussion

This section discusses future problems of the analog EHW. Other applications of our proposed method are also discussed.

The production cost of analog LSIs consists mainly of (1) the wafer cost, (2) the package cost, and (3) the testing cost. Because the wafer cost (1) is proportional to the mask area, the proposed method can reduce the cost (1) greatly. However, as the proposed method does involve additional calibration time, the overall costs at the testing stage are higher. It is crucial, therefore, to reduce the calibration time for the commercial use of the proposed method. We have already further finely tuned GA parameters (e.g. population size, crossover rate, mutation rate) and weight values of the fitness function. However, because optimal GA parameters are dependent on an application, systematic procedure for such tuning should be developed for a wide use of our method.

The basic idea behind the GA calibration is to regard the architecture bits of the reconfigurable analog circuits as chromosomes for GAs and to identify optimal configurations by running GAs. This approach can be applied to a wide variety of analog circuits and is not limited to the IF filter LSI and the IRM.

In the above applications, the chips are reconfigured only once or only at relatively longer intervals. Reconfiguration in these cases is usually conducted off-line by the chip manufacturers or by the chip user. However, online GA calibration could also be useful. This calibration can reconfigure chips online at regular intervals by observing the outputs from the chip. This online calibrated chip could be used in environments that vary over time (e.g., due to temperature or power supply). In such cases, GA hardware (Kajitani, et al., 1998) should be incorporated into the chips, especially for analog LSIs in embedded systems. The GA hardware can handle self-reconfiguration without host-machine control.

5. CONCLUSIONS

We have proposed the evolvable analog LSI and applied it to IF filters and image-rejection mixers which can correct discrepancies in analog LSI implementations. Even if some analog circuit elements have different values from the design specifications which prevent the analog LSI from performing at required levels, the LSI can change the values of variant analog circuit elements by GAs.

The developed IF filter LSI includes current-adjustment circuits that employ the GA to correct for deviations from the target frequency response due to variations in the values of analog devices. Using our method, 97% of the chips tested are successfully calibrated to conform to the specifications, although none of them satisfied them prior to calibration. Moreover, the filter area is reduced by 63%, resulting in a 27% reduction in power dissipation. The commercial version of the developed chip has been mass-produced.

The calibration experiments for the developed IRM circuit demonstrates that the circuit automatically adjusted is capable of outperforming a circuit adjusted by an experienced engineer. Following this successful initial evaluation of the approach, we plan to develop the image-rejection mixer further, by reducing its size and by employing a wider signal band for the image-rejection ratio.

Our method has great potential, particularly for SoC (system-on-a-chip) implementations, where one major obstacle has been the poor yield rates associated with analog circuit integration.

Acknowledgments

This work was partly supported by METI RWCP and NEDO. The authors would like to thank Asahi-Kasei Microsystems Co., Ltd., for circuit design and the experiments. We also thank Dr. Hirose in ASRC and Prof. Sakurai in University of Tokyo for their suggestions.

References

Adachi, T., et al. 1994. A low noise integrated AMPS IF filter. In *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, 159–162.

Archer, J. W., J. Granlund, and R. E. Mauzy, 1981. “A broad-band UHF mixer exhibiting high image rejection multidicade baseband frequency range”, *IEEE Journal of Solid-State Circuits*, 16:385–392.

Baumberger, W. 1994. “A single-chip image rejecting receiver for the 2.44 GHz band using commercial GaAs-MOSFET-technology”, *IEEE Journal of Solid-State Circuits*, 29:1244–1249.

Gray, P. R. and R. G. Meyer, 1984. *Analysis and design of analog integrated circuits*. John Wiley & Sons, Inc.

Kajitani, I., et al. 1998. A gate-level EHW chip; a implementation of hardware for GA operations and reconfiguable hardware on a single LSI. In *Proceedings of the Second International Conference on Evolvable Systems*.

Kasai, Y., et al. 2000. Initial evaluation of an evolvable microwave circuit. In *Proceedings of the Third International Conference on Evolvable Systems*, 103–112. Springer Verlag.

Minakawa, A. and T. Tokumitsu, 1993. “A 3–7 GHz wide-band monolithic image-rejection mixer on a single-chip”, *IEICE Transactions on Electron*, E-76-C:955–960.

Murakawa, M., et al. 1998. Analogue EHW chip for intermediate frequency filter. In *Proceedings of the Second International Conference on Evolvable Systems*, 134–143. Springer Verlag.

Proakis, J. 1988. *Digital Communications*. Prentice Hall, Inc.

Rudell, J. C., et al. 1997. "A 1.9 GHz wide-band IF double conversion CMOS receiver for cordless applications", *IEEE Journal of Solid-State Circuits*, 32:2071–2088.

Chapter 8

RECONFIGURABLE ELECTRONICS FOR EXTREME ENVIRONMENTS

Adrian Stoica[1], Didier Keymeulen[1], Ricardo S. Zebulum[1], and Xin Guo[2]
[1]Jet Propulsion Lab, 4800 Oak Grove Dr., Pasadena, CA 91109 USA; [2]Chromatech, Alameda, CA USA, Email: adrian.stoica@jpl.nasa.gov

Abstract: This chapter argues in favor of adaptive reconfiguration as a technique to expand the operational envelope of analog electronics for extreme environments (EE). In addition to hardening-by-process and hardening-by-design, "hardening-by-reconfiguration", when applicable, could be used to mitigate drifts and functional deviations coming from degradation or complete damage on electronic devices (integrated circuits) in EE, by using reconfigurable devices and an adaptive self-reconfiguration of their circuit topology. Conventional circuit design exploits device characteristics within a certain temperature/radiation range; when that is exceeded, the circuit function deviates from its set-point. On a reconfigurable device, although component parameters change in EE, a new circuit design suitable for new parameter values may be mapped into the reconfigurable structure to recover the initial circuit function. Partly degraded resources can still be used, while completely damaged resources may be bypassed. Designs suitable for various environmental conditions can be determined prior to reaching destination or can be determined in situ, by using adaptive reconfiguration algorithms running on built-in digital controllers. Laboratory demonstrations of this hardening-by-reconfiguration technique were performed by JPL in several independent experiments in which bulk CMOS reconfigurable devices were exposed to, and functionally altered by, low temperatures (~ −196°C), high temperatures (~280°C) or radiation (300kRad TID), and then recovered by adaptive reconfiguration using evolutionary search algorithms.

Key words: evolvable hardware, extreme environments, hardening-by-reconfiguration, adaptive reconfiguration.

1. INTRODUCTION

Future NASA missions to the Moon, Mars and beyond (Terry, et al., 2004) will face EE, including environments with large temperature swings, such as between –180°C and 120°C at the initial landing sites on the Moon, low temperatures of –220°C to –230°C during the polar/crater Moon missions, –180°C for Titan in situ mission, –145°C and high radiation levels for Jupiter's moons, 5MRad Total Ionizing Dose (TID) for Europa Surface and Subsurface mission, high temperature of 460°C for Venus Surface Exploration and Sample Return mission, etc. EE induce degradation and damage to electronic devices manifested in drifts and deviations of their electronic characteristics.

The current approach for space electronics designs is to use commercial/military range electronics protected through passive (insulation) or active thermal control, and high weight shielding for radiation reduction. This adds to sizable weight and volume, compounded by power loss and additional cost for the mission. More importantly, as missions will target operations with smaller instruments/rovers and operations in areas without solar exposure, these approaches become infeasible. In many cases the electronics must be collocated with the sensor or actuator in the extreme environment, without the option of being insulated or shielded properly. Therefore, developing EE-electronics would have several advantages including lower costs, less power, and offering in some cases, the only reasonable solution.

Conventional approaches to EE-electronics include *hardening-by-process* (HBP) (Chen, et al., 1991), i.e. fabricating devices using materials and device designs with higher tolerance to EE, (e.g. using special materials like silicon carbide for high temperatures, or silicon-on insulator for radiation). Another promising approach is *hardening-by-design* (HBD) (Anelli, 2000), i.e. use of special design/compensation schemes. For example, circuit techniques such as auto-zero correction are used to alleviate the problem of the (temperature dependent) offset voltages in Operational Transconductance Amplifiers (OTA) operated at low temperatures (Terry, et al., 2004). Both of these hardening approaches are limited, in particular for analog electronics, by the fact that current designs are fixed and, as components are affected by EE, these drifts alter functionality.

A recent approach pioneered by JPL is to mitigate drifts, degradation, or damage on electronic devices in EE by using reconfigurable devices and an adaptive self-reconfiguration of circuit topology. This new approach is referred to here as hardening-by-reconfiguration (HBR). In HBR, although device parameters change in EE, a new circuit design, suitable for new parameter values, is mapped into the reconfigurable system to recover the initial circuit functionality. Partly degraded resources are still used in the circuit, while completely damaged resources are bypassed. The new designs, suitable for various environmental conditions, can be determined prior to operation or determined in situ by reconfiguration algorithms running on a built-in digital controller.

HBR can also be seen as a related technique to HBD – with the characteristic that it uses an in situ (re)design. HBR would benefit from HBD for the resources available on the reconfigurable chips. It would also work well in conjunction with HBP, contributing, as an extra layer of protection, to the expansion of the limits of operation in EE; HBP would provide inherent survivability to keep devices operational at higher EE limits, while HBR would provide the adaptation to changes in device characteristics needed for precise functions, especially needed for analog circuits. A somehow degenerated form of HBR would be the on-chip use of several fixed circuits, each optimally designed for a temperature range which can be multiplexed/switched in/out depending on the temperature at that moment (this is different than mapping optimal designs on a reconfigurable array). A simple form of HBR would use a reconfigurable circuit, but circuit configuration for various extreme conditions would be predetermined and memorized for access when needed, as opposed to being determined in situ, which is the harder but potentially more powerful aspect of this technique.

This chapter overviews the HBR technique and shows the results of experiments of recovery under radiation and at low and high temperatures. The chapter is structured as follows: Section 2 reviews the Evolvable Hardware (EHW) approach. Section 3 describes the experiments. Section 4 outlines conclusion and future work.

2. EVOLVABLE HARDWARE

Automated reconfiguration of reconfigurable devices has been developed in the field of EHW. EHW is a set of techniques that address hardware that reconfigures under the control of evolutionary algorithms (the name is also used to refer to the hardware that evolves). Thus, EHW has two components: the Reconfigurable Hardware (RH) and the Reconfiguration Mechanisms (RM) that control reconfiguration. Not only evolutionary algorithms (EA), but other search algorithms can be used to find those circuit solutions for EE; nevertheless, EA are particularly efficient search techniques for the large search spaces to explore when programming reconfigurable chips.

An example of RH is the Field Programmable Transistor Array (FPTA) family of chips developed at JPL and used in the experiments presented later in the chapter. The FPTA is a reconfigurable architecture with programmable granularity, the lowest being at the transistor level. It can map analog, digital and mixed signal circuits. The FPTA architecture is cellular; for example, in one of the chips in the family, the FPTA-2, it consists of an 8 × 8 array of reconfigurable cells. Each cell has a transistor array as well as a set of programmable resources, including programmable resistors and static capacitors. Figure 8-1 provides a diagram of the chip architecture together with a more detailed schematic of the reconfigurable transistor array cell. The reconfigurable circuitry consists of 14 transistors connected through 44 switches. The reconfigurable circuitry is able to implement different building blocks for analog processing, such as two and three stages OpAmps, Gaussian computational circuits, etc. It includes three capacitors of 100fF, 100fF and 5pF respectively. Control signals come on the 9-bit address bus and 16-bit data bus, and access each individual cell providing the addressing mechanism for downloading the bit-string configuration of each cell. A total of ~5000 bits is used to program the whole chip. The pattern of interconnection between cells is similar to the one used in commercial FPGAs; each cell interconnects with its north, south, east and west neighbors. More details of FPTA-2 are presented in Stoica, et al. (2001).

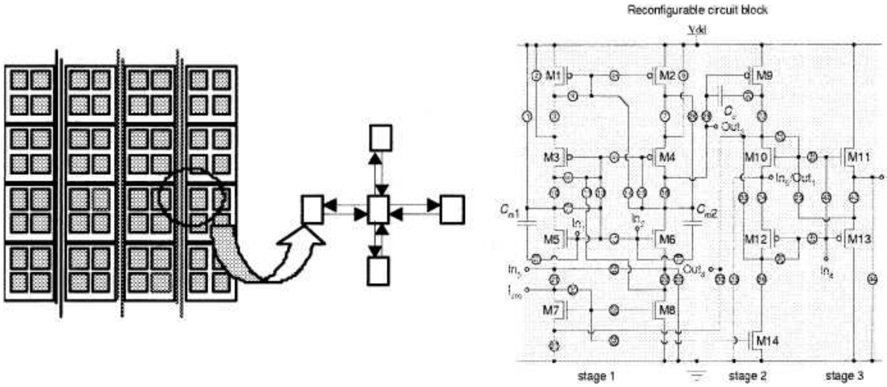


Figure 8-1. FPTA-2 architecture (left) and schematic of cell transistor array (right). The cell contains additional capacitors and programmable resistors (not shown)

For example, the RM determines the appropriate configuration by a search procedure. Thus, in evolutionary circuit synthesis/design and EHW, an evolutionary/genetic search/optimization algorithm searches the space of all possible circuits and determines solution circuits with desired functional response (here the word synthesis is used in the most general sense). The EA is tightly coupled with a coded representation of the candidate circuits. Each circuit gets associated a "genetic code" or chromosome; the simplest representation of a chromosome is a binary string, a succession of 0s and 1s that encode a circuit. Synthesis is the search in the chromosome space for the solution corresponding to a circuit with a desired functional response. The EA follows a "generate and test" strategy: a population of candidate solutions is maintained each time; the corresponding circuits are then evaluated and the best candidates are selected and reproduced in a subsequent generation, until a performance goal is reached. In this work, since very accurate device models for EE are not available, circuit evaluation is done directly in reconfigurable hardware. The steps of an EA are sketched in Figure 8-2. More details on evolutionary circuit design can be found in Stoica, et al. (2000).

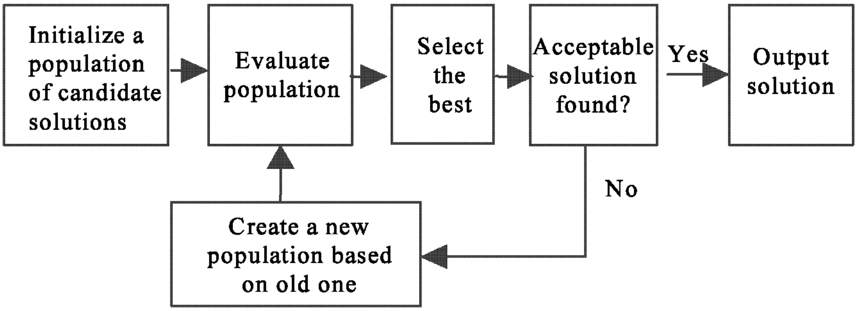

Initialize a population of candidate solutions
Evaluate population
Select the best
Acceptable solution found?
Yes
Output solution
No
Create a new population based on old one

Figure 8-2. Block diagram of a simple population-based search such as EA

To create a complete EHW system, a Stand-Alone Board-Level Evolvable System (SABLES) was built by integrating the FPTA and a DSP implementing the evolutionary recovery algorithm (Stoica, et al., 2002). The system is connected to the PC only for the purpose of receiving specifications and communicating back the result of evolution for analysis. The system fits in a box 20cm × 20cm × 8cm. Communication between DSP and FPTA is fast, with a 32-bit bus operating at 7.5MHz. The FPTA can be attached to a Zif socket attached to a metal electronics board to perform extreme temperature and radiation experiments The evaluation time depends on the tests performed on the circuit. Many of the tests attempted here require less than two milliseconds per individual, and runs of populations of 100 individuals for 200 generations require less than a minute. Figure 8-3 shows the schematic of the SABLE system.

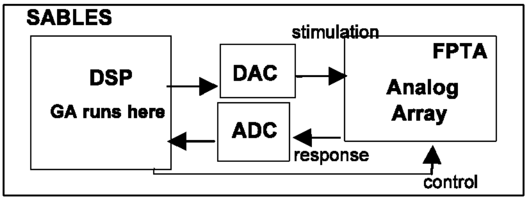


Figure 8-3. Schematic of the SABLE system. JPL implementation used two chips; the reconfigurable analog circuits being part of a FPTA chip, the other functional blocks being run in a Digital Signal Processor (DSP). An analog-to-digital converter (ADC) and a digital-to-analog converter (DAC) are used to interface the FPTA and DSP. Only the FPTA was exposed to EE

3. EXPERIMENTS IN FUNCTIONAL RECOVERY BY EVOLUTION

3.1 Generalities

Multiple experiments on recovery of various functional circuits were performed. Initially, a human designed circuit was mapped into the FPTA or a circuit was evolved at room temperature, for the function of interest. The chips were then exposed to EE and degradation was observed. At that point evolutionary recovery was started and a new circuit with the same intended function was obtained, sometimes imperfect but still providing good approximation for the problem at hand. In some cases the function cannot be recovered within the imposed precision; this may be either a limit of the hardware (no solution exists) or of the algorithm that was not able to reach a solution in the given time (suboptimal algorithm or insufficient time). All the experiments illustrated below present the recovery of functionality of a half-wave rectifier when excited with sine wave input. FPTA resources were limited to the use of only two cells. Other circuits that were recovered by evolution include logical gates, filters, amplifiers, and various analog computational circuits.

The fitness function given below does a simple sum of error between the target function and the output from the FPTA. The input was a 2 kHz excitation sine wave of 2V amplitude, while the target waveform was the rectified sine wave. The fitness function rewarded those individuals exhibiting behavior closer to target (by using a sum of differences between the response of a circuit and the target) and penalized those farther from it. The fitness function was:

$$F = \sum_{t_s=0}^{n-1} \begin{cases} |R(t_s) - S(t_s)| & \text{for } (t_s < n/2) \\ |R(t_s) - V_{\max}/2| & \text{otherwise,} \end{cases}$$

where $R(t_s)$ is the circuit output, $S(t_s)$ is the circuit stimulus, n is the number of sampled outputs, and $V_{\max}$ is 2V (the supply voltage). The output must follow the input during half-cycle, staying constant at a level of half-way between the rails (1V) in the other half.

After the evaluation of 100 individuals, these were sorted according to fitness and a 9% (elite percentage) portion was set aside, while the remaining individuals underwent crossover (70% rate) either among themselves or with an individual from the elite, and then mutation (4% rate). The entire population was then reevaluated. An oscilloscope snapshot during executions over a generation and a detail fragment are shown in Figure 8-4.

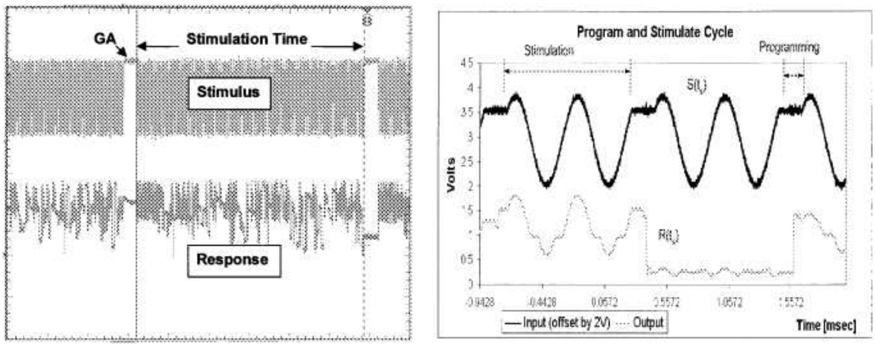


Figure 8-4. Stimulus-response waveforms during the evaluation of a population in one generation (left) and zoom-in for 2 individuals in the population (right). A full EA cycle includes stimulus/response sequence (113ms) and the generation of the next generation (6ms)

3.2 Recovery of Half-wave Rectifier Irradiated to 300krad TID

A number of papers in the literature have examined the effect of radiation on CMOS devices (e.g. Johnston and Guertin, 2000). These could be classified into those papers that studied the effect of radiation on various cells and macroblocks fabricated in silicon (Ragaie and Kayed, 2002) or that considered the design of radiation-hardened components and cell libraries (Hatano, 1992). Works on studying the impact of radiation have considered custom implementation and conventional digital FPGA platforms such as Xilinx (Guertin, et al., 1999). These works have focused on studying both total dose radiation effects, where the effect is permanent, and Single Event Upsets (SEU)s, where individual bits in memory elements flip when exposed to a certain quantity of radiation.

However, most of these papers focused on technologies which are above 0.5 micron and hence the effects could not be generalized to devices implemented in the latest Deep Submicron (DSM) technologies, where leakage currents dominate. In addition, no research has been carried out on the development of custom reconfigurable architectures implemented at transistor level, hence enabling the implementation of both analog and digital circuits.

Our research presents a framework for the development of radiation tolerant mixed analog and digital circuits on a DSM reconfigurable CMOS device. Experiments are carried out in which the device is subjected to various radiation dosages, using a Cobalt 60 source, and the performance of the device is tested by mapping a number of functional circuits. When the device fails any of the tests, an EA is used to recover the functionality of the device where possible.

We conducted the radiation experiments over the course of 60 hours (Figure 8-5). The radiation experiment conducted followed the radiation and test methodology first described in Stoica, et al. (2004a), with the difference being a larger focus on the quantitative aspects of the change in individual NMOS and PMOS transistors and how this affects behavior at the circuit level. Test transistors were provided in the layout for behavior characterization under radiation. These test transistors are created on the same process as the rest of the transistors in the FPTA-2 cells and are thus assumed to behave similarly. If the radiation beam is homogenous across the transistor, we can further assume that the radiation affects the transistors in the cells in a similar way to the test transistors. Since radiation is a cumulative effect, we were particularly interested in the Total Ionizing Dose (TID) effect of radiation on the transistor and circuit levels.

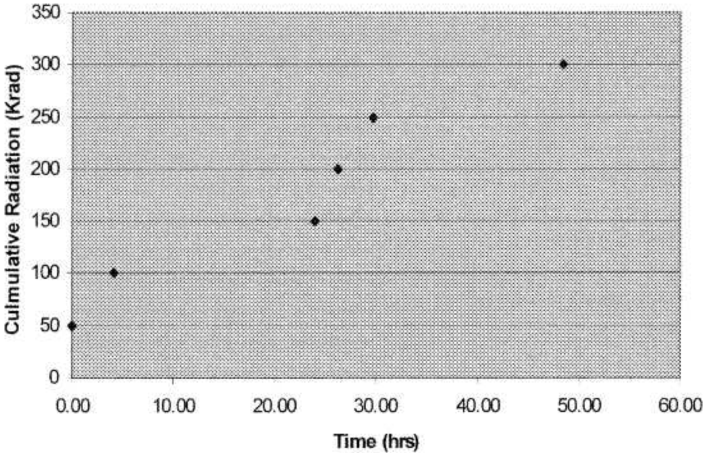


Figure 8-5. Cumulative radiation graph

The radiation setup used in the experiments presented here used Cobalt 60 sourced gamma rays instead of the electron beam used previously (Stoica, et al., 2004a). Logistically, the test involved placing a FPTA chip and its associated host board 28.5 cm away from a shielded Cobalt 60 source. The host board was included to provide easy biasing. When exposed at that distance, the Cobalt 60 source subjects the chip to 50 rad/sec of gamma ray radiation. FPTA chips were radiated under both biased (power on) and unbiased (power off) conditions. The biasing scheme promotes ionization by introducing a potential difference across the inputs and outputs, thus facilitating the trapping of charge. The biasing scheme is shown in Figure 8-6, with inputs at Vdd and outputs grounded.

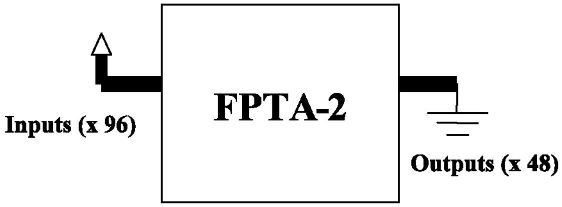


Figure 8-6. Biasing scheme of the FPTA

After irradiation, we measured the I-V, drain current versus gate voltage (Id vs. Vg) and threshold voltage (Vth) characteristics using an Agilent 4155B semiconductor parameterization device. Vth characterization was done according to the application note in Agilent Application Notes. Id vs. Vg characteristics were measured with the setup indicated in Figure 8-7.

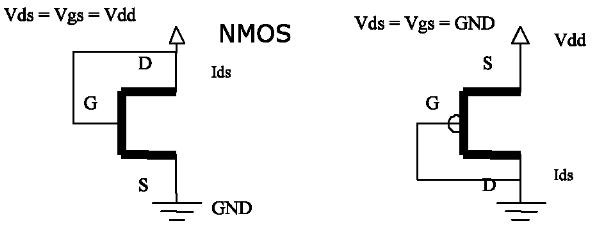


Figure 8-7. MOS transistors setup for Id × Vg characterization

Figure 8-8 displays the change in the threshold voltage for two MOS transistors: N2 (W/L = 0.54u/0.27u) and N3 (W/L = 0.72u/0.36u). Figure 8-9 shows the the I-V curve (drain current × gate voltage) for different radiation levels. PMOS transistor behavior has also been characterized in the same way.

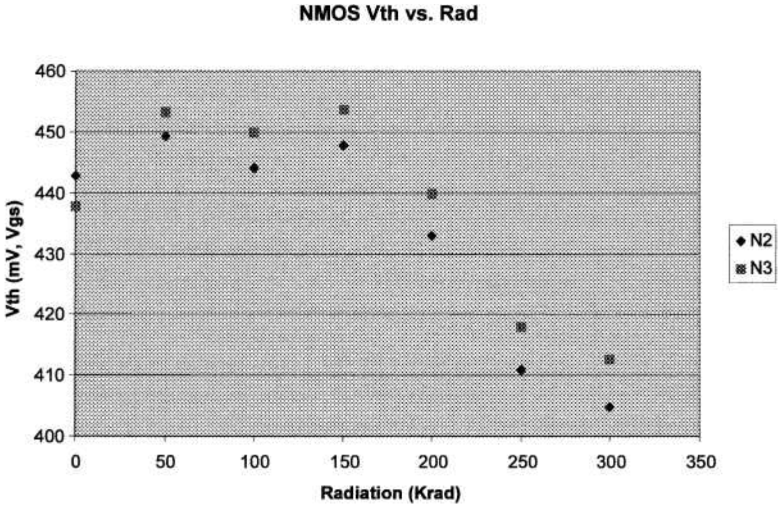


Figure 8-8. Graph of Vth with radiation on two NMOS

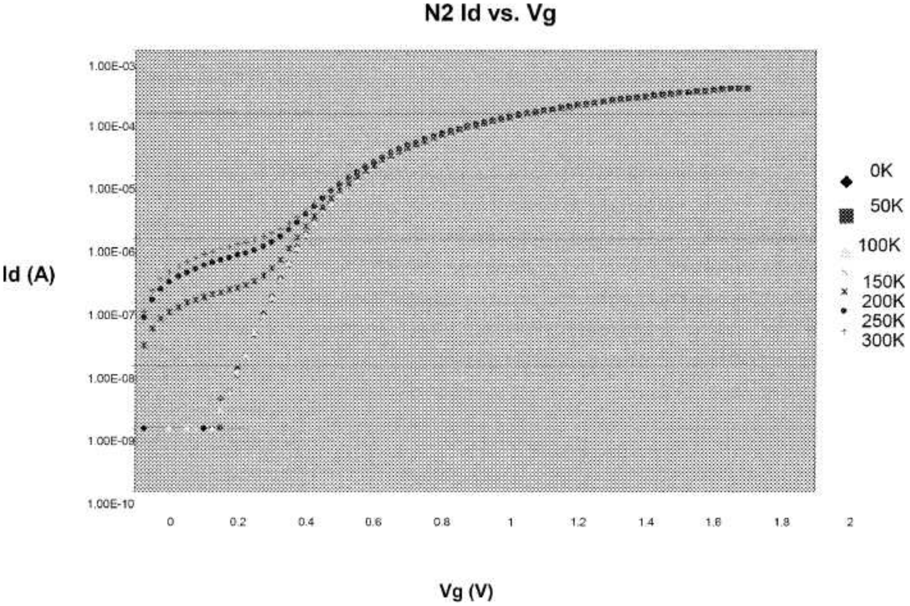


Figure 8-9. NMOS gate voltage vs. drain current on a logarithmic scale

The overall results from these experiments closely conformed to what was expected based on the results of our previous experiments with EAs. A clear link between the shifts in the NMOS threshold voltage (Vth) and Id vs.

Vg, as well as qualitative distortion in the behavior of the downloaded circuit functions can easily be seen in the biased chip.

As opposed to our previous experiments reported in Stoica, et al. (2004a), in which case degradation started below 50kRad, no change in the behavior of the circuit for TID levels up to 200Krad was observed. This was consistent with the monitored values of NMOS threshold voltage value, for which noticeable changes started at 200Krad. Figure 8-10a shows the original response without radiation. Figure 8-10b shows the slightly degraded half-wave rectifier response at 200Krad. Figure 8-11a depicts the degraded response at 300Krad, and Figure 8-11b the response of the recovered circuit obtained through evolution. The circuit recovery was performed in about a 5 minute execution of EA, sampling 200 individuals per generation, over 100 generations. More details are presented in Stoica, et al. (2005).

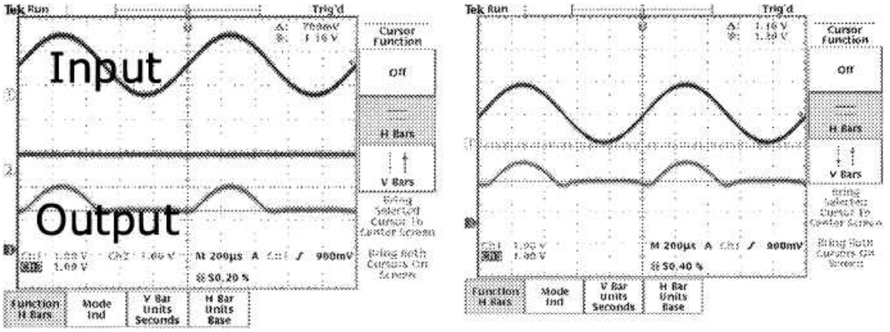


Figure 8-10. a) Response of the half-wave rectifier circuit before radiation was applied to the chip (left) b) Half-wave rectifier response at 200Krad (right)

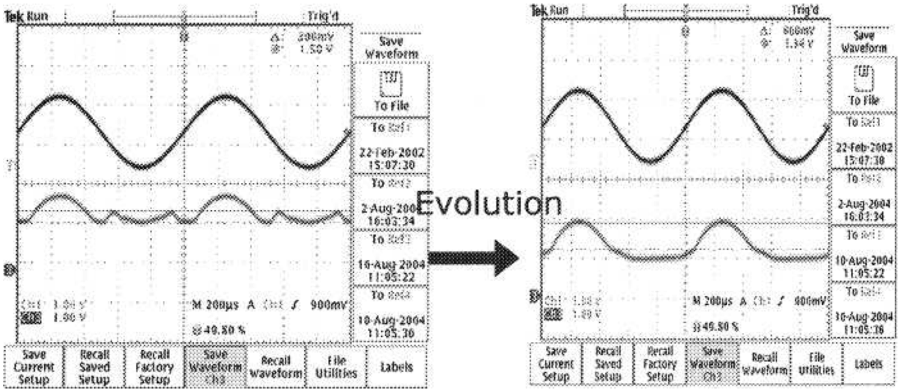


Figure 8-11. a) and b) Response of the rectifier circuit after being radiated to 300kRads resulting in deterioration (a) through loss of rectification, followed by recovery through evolution (b)

3.3 Recovery of Half-wave Rectifier at Low Temperature of –196.6°C

The low temperature testbed used liquid nitrogen, establishing a temperature of –196.6°C. In order to study the effect of low temperatures on the FPTA device only, the chip was placed on a separate board that was immersed into liquid nitrogen. This setup did not allow a control for intermediate temperatures between room ambient and liquid nitrogen. A standard ceramic package was used for the chip. A half-wave rectifier was evolved at –196.6°C, the oscilloscope caption being shown in Figure 8-12 (left). This was not a robust solution (and it was not even expected to be, since the EA was not asked, through the fitness function, to respond to an entire temperature range) and when taken out to room temperature the response deteriorated, as shown in Figure 8-12 (right).

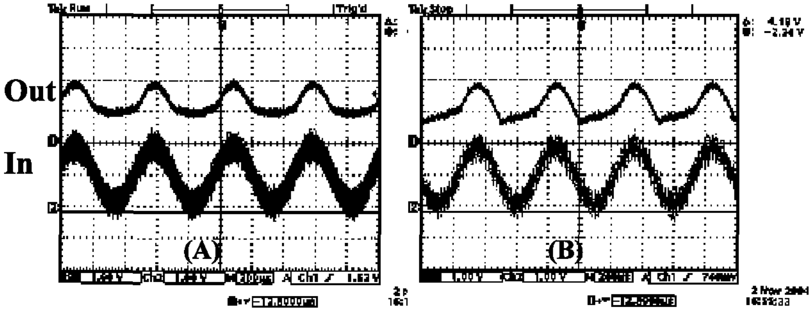


Figure 8-12. Half-wave rectifier recovered at –196C (left); solution is not robust and degrades when returned to room temperature (right)

3.4 Recovery of Half-wave Rectifier at High Temperature of 280°C

A high temperature testbed was built to achieve temperatures exceeding 350°C on the die of the FPTA-2 while staying below 280°C on the package. The testbed included an Air Torch system firing hot compressed air through a small hole of a high temperature resistance ceramic protecting the chip. Details of the setup are given in Stoica, et al. (2004b). Figures 8-13a and 8-13b depict the response of the evolved circuit at room temperature and the degraded response at high temperature. Figure 8-14 shows the response of the circuit obtained by running evolution at 280°C, where the functionality is recovered.

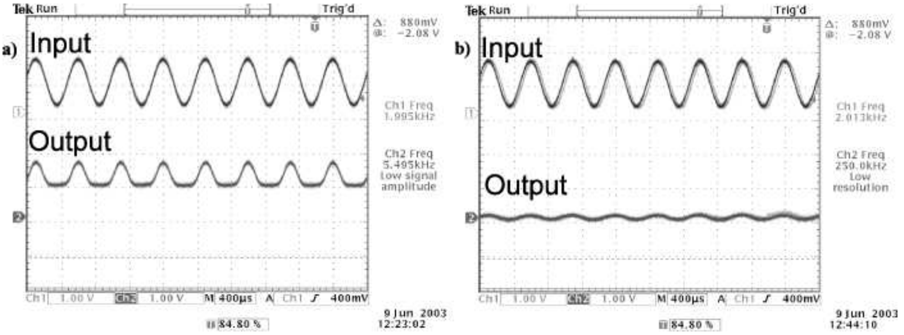


Figure 8-13. Input and output waveforms of the half-wave rectifier. a) response of the circuit evolved at 27°C. b) degraded response of the same circuit when the temperature is increased to 280°C

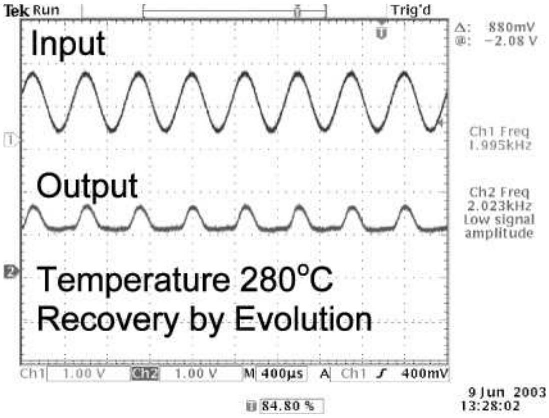


Figure 8-14. Recovered response from evolved circuit

A limiting factor to efficiency of HBR is the T-gate switch used for reconfiguration. We performed a detailed analysis of the behavior of T-gate switches of the FPTA-2 at high temperature. The switch is a key element since it modifies the topology of the circuit as a function of the signal applied to its gate and controlled by the chromosome. We have observed that the response of the device was independent of the chromosomes downloaded when the FPTA-2 was operated above 280°C.

Figure 8-15 illustrates the resistance of seven T-gates placed in series on the FPTA-2. At 125°C, the resistance of the T-gate in Off state is more than 1Gohm, while in On state it has a resistance of 31 kOhm. In comparison, when the FPTA-2 is warmed up to 300°C, the resistance of the T-gate is 60kOhm for both On and Off state. The leakage currents between the Nwell, Pwell, Vdd and Ground makes the T-gate switch behave at high temperature (over 280°C) as a low resistance independent of the signal applied to the T-gate. These current leakages increase as the technology feature size decreases. A previous device in the family, the FPTA-0, used a higher feature size (0.5μ) compared to the lower feature (0.18 μ) on the FPTA-2. As a consequence, we were able to recover functionality of circuits at higher temperatures (320°C) using FPTA-0. The current leakage of the T-gate is the limiting factor for the HBR, but it can be compensated by using HBD and HBP.

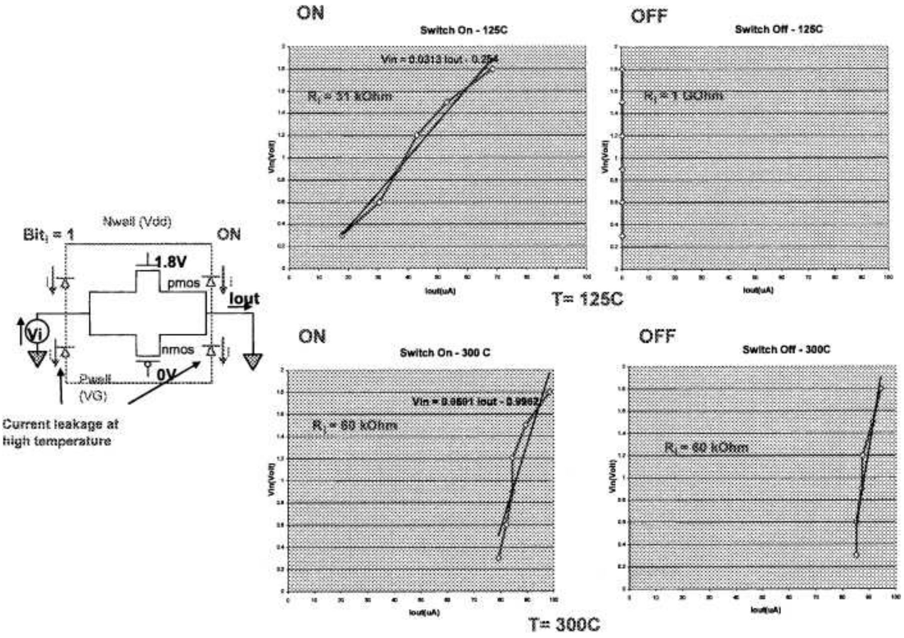


Figure 8-15. Current leakage and behavior of T-gate at 125°C and 300°C in ON and OFF state

4. CONCLUSIONS AND FUTURE WORK

This chapter presented the evolution-driven HBR approach and demonstrated it with laboratory experiments. The capability of adaptive self-configuration was demonstrated for radiation, low and high temperatures.

This demonstration proves the concept, yet has the following limitations: 1) the tests were of short duration, 2) the work did not implement temperature cycling, 3) the work did not use combined temperature/radiation testing, 4) the reconfiguration mechanism was running on a separate DSP that was not affected by temperature/radiation, and 5) the results were not demonstrated on complex analog circuits performing in an application.

Overcoming these limitations is the objective of our current ongoing work aiming to demonstrate a self-reconfigurable single chip under temperatures cycles replicating those on the Moon, for over 1000 hours duration, and in combined extreme radiation/temperature tests, performing a sensor processing function. More specifically, the overall objective is to develop and demonstrate reconfigurable analog electronics performing characteristic analog functions (filtering, amplification, etc.) for extended operations in EE with temperatures cycling in the range of –180°C and 120°C, and cumulative radiation of at least 300kRad total ionizing dose (TID). The temperature range of –180°C and 120°C covers the temperature range for both Moon and Mars environments and 300KRad TID reflects accumulative dose during very long Mars missions (100KRad for near-term missions), or missions beyond the Moon and Mars, such as to Jupiter's moons. This would validate the technology for Moon and Mars temperature and radiation environments and the even harsher radiation environments for missions beyond.

Acknowledgments

The work described in this chapter was performed at the Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the National Aeronautics and Space Administration (NASA). NASA Program Managers for the effort were Dr. Mita Desai and Dr. Chris Moore.

References

Anelli, G., 2000. IEEE, Trends in CMOS technologies and radiation tolerant design, *Nuclear Science Symposium Conference Record*, Volume: 1, 15-20 Oct. 2000, 2/2.

Chen, Panxun, et al., 1991. Total dose radiation effects on the hardened CMOS/bulk and CMOS/SOS. In *Radiation and its Effects on Devices and Systems. RADECS 91, First European Conference on*, 9-12 Sept. 1991, 249-253.

Guertin, S. M., G. M. Swift, and D. Nguyen, 1999. Single-event upset test results for the Xilinx XQ1701L PROM. In *Radiation Effects Data Workshop, 1999*, 12-16 July 1999, 35-40.

Hatano, H., 1992. Radiation hardened high performance CMOS VLSI circuit designs, Circuits, Devices and Systems. In *IEE Proceedings G*, Volume: 139 Issue: 3. 287-294.

Johnston, A. and S. Guertin, 2000. The Effects of Space Radiation on Linear Integrated Circuits. *IEEE Aerospace Conference*, Vol. 5, 363-369.

Ragaie, H., and S. Kayed, 2002. Impact of CMOS device scaling in ASICs on radiation immunity, Electronic Devices. (EWAED). *The First Egyptian Workshop on Advancements of*, 28 Sept. 2002, 18-27.

Stoica, A., et al., 2000. Evolution of analog circuits on Field Programmable Transistor Arrays. Lohn, J., et al. (eds.), *Proceedings of NASA/DoD Workshop on Evolvable Hardware (EH2000)*, July 13-15, 2000, 99-108. Palo Alto, CA, USA: IEEE Computer Society.

Stoica, A., R. S. Zebulum, and D. Keymeulen, 2001. Progress and Challenges in Building Evolvable Devices. *Third NASA/DoD Workshop on Evolvable Hardware*, 33-35. Long Beach: IEEE Computer Society.

Stoica, A., et al. 2002. Evolving Circuits in Seconds: Experiments with a Stand-Alone Board-Level Evolvable System. *2002 NASA/DoD Conf. on Evolvable Hardware*, July 15-18, 2002, 67-74. IEEE Computer Press.

Stoica, A., et al. 2004a. Evolutionary Recovery of Electronic Circuits from Radiation Induced Faults. In *The IEEE Conference on Evolutionary Computation*, CEC, Portland, OR.

Stoica, A., et al. 2004b. Circuit Self-Recovery Experiments in Extreme Environments. In *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware*, 142-145, Seattle, USA.

Stoica, A., et al. 2005. Characterization and Recovery of Deep Sub Micron (DSM) Technologies Behavior Under Radiation. In *The 2005 IEEE Aerospace Conference*, Montana.

Terry, S. C., et al. 2004. Development of Robust Analog and Mixed-Signal Electronics for Extreme Environments Applications. *IEEE Aerospace Conference*, Big Sky, MT.

Chapter 9

CHARACTERIZATION AND SYNTHESIS OF CIRCUITS AT EXTREME LOW TEMPERATURES

Ricardo S. Zebulum[1], Didier Keymeulen[1], Rajeshuni Ramesham[1],
Lukas Sekanina[2], James Mao[1], Nikhil Kumar[1], and Adrian Stoica[1]
[1]Jet Propulsion Laboratory 4800 Oak Grove Dr. 91109 Pasadena CA USA; [2]Brno University of Technology, Email: ricardo.s.zebulum@jpl.nasa.gov

Abstract: This chapter describes circuit evolutionary experiments at extreme low temperatures, including the test of all system components at this extreme environment (EE). Standard circuit designs have their target behavior limited to particular ranges of temperature, which usually does not achieve the extreme temperatures encountered in some planets of our solar system. One approach to this problem is the use of reconfigurable devices programmed by adaptive/evolutionary algorithms. Through reconfiguration, the new characteristics of the devices at EE may be fully explored and correct functionality can be recovered. This chapter demonstrates this technique for circuit characterization, evolution and recovery at liquid nitrogen temperatures (–196.6°C). In addition, preliminary tests are performed to assess the survivability limitations of the evolutionary processor at low temperatures.

Key words: evolvable hardware, extreme environment (EE), extreme low temperatures, stand-alone board level evolvable system (SABLES).

1. INTRODUCTION

Planetary exploration and long-term satellite missions require radiation and extreme-temperature hardened electronics to survive the harsh environments beyond Earth's atmosphere. Failure to take precautions could lead to disastrous consequences, potentially jeopardizing the vehicle, the mission and future funding. Thus, for long-term missions it is essential that all electronics can somehow deal with faults. The traditional approaches to preserve electronics incorporate shielding, insulation and a good deal of redundancy, at the expense of power and weight. Newer technologies are taking advantage of materials-based solutions and different circuit topologies; however, it is quite unlikely that it will operate fully at normal temperatures as well as the extreme temperatures, as mentioned above.

A novel approach to this challenge is the concept of Evolvable Hardware (EHW). EHW is based on search/adaptive algorithms that focus on a population of potential solutions. After a number of iterations and through the application of operators like selection, crossover and mutation, a circuit solution is potentially found that meets the problem requirements (Stoica, et al., 2002).

The in situ evolvable hardware system consists of two important subsystems: reconfigurable hardware (RH) comprised of functional blocks and reconfigurable fabric, and an evolutionary processor (EP) running the evolutionary algorithm (the evolutionary processor). Since the EP is generating the stimulus and evaluating the results, its behavior must be characterized in the entire range of operation of the EHW system. Ideally, the system would not change in performance across temperatures. Temperature coefficients of components cause behavioral changes in the system. Commercial vendors of both ICs and systems typically minimize these effects by integrated compensation (e.g. balancing positive temperature coefficient and negative temperature coefficient parts to eliminate temperature dependence). While these techniques improve the performance of the hardware in the designed operation ranges (i.e., 0° to 70°C for commercial grade, –40° to +85°C for industrial grade, and –55° to +125°C for military grade), outside of these ranges this compensation may actually worsen the situation by adding another order to the temperature effects.

The focus of this chapter is on the application of EHW for low temperatures. Other studies have been performed for high temperatures and radiation environments (Stoica, et al., 2004). The applications described here encompass the separate testing of the whole Evolvable Hardware system (Evolutionary Processor + Data Converters + Reconfigurable chip) at low temperatures, following the assumption that the entire system will be exposed to the space EE. In the experiments, we demonstrate the evolution and recovery of circuits at liquid nitrogen temperatures (–196.6°C) and verify the operational limitation of the evolutionary processor at low temperatures. This adds to our previous experiments where only the re-configurable chip was exposed to EE (Stoica, et al., 2004).

The Stand-Alone Board-Level Evolvable (SABLE) system (Stoica, et al., 2002) designed by JPL is used in the experiments described in this chapter. This system is constituted by a DSP working as an evolutionary processor and a reconfigurable mixed signal chip, the Field Programmable Transistor Array (FPTA). Section 2 of this chapter overviews the SABLE system. Section 3 describes the experiments and Section 4 concludes the work.

2. OVERVIEW OF SABLES

SABLES integrates an FPTA device and a DSP implementing the Evolutionary Platform (EP). Analog-to-Digital converters (ADCs) and Digital-to-Analog Converters (DACs) provide an interface between the FPTA and the DSP for analog I/O. The system is stand-alone and is connected to the PC only for the purpose of receiving specifications and communicating back the results of evolution for analysis. The reader can refer to Stoica, et al. (2002) for a more detailed description of evolvable systems.

The last version of the family of FPTA chips, the FPTA-2, has transistor level reconfigurability, consisting of an 8 × 8 array of reconfigurable cells. Each cell has a transistor array as well as a set of other programmable resources, including programmable resistors and static capacitors. Figure 9-1 provides a detailed view of the reconfigurable transistor array cell. The reconfigurable circuitry consists of 14 transistors connected through 44 switches and is able to implement different building blocks for analog processing, such as two- and three-stage OpAmps and Gaussian computational circuits. Details of the FPTA-2 can be found elsewhere (Stoica, et al., 2002).

The evolutionary algorithm is implemented in a DSP that directly controls the FPTA-2, together forming a board-level evolvable system with fast internal communication ensured by a 32-bit bus operating at 7.5MHz. Details of the EP were presented in (Ferguson, et al., 2002). Over four orders of magnitude speed up of evolution was obtained on the FPTA-2 chip compared to SPICE simulations on a Pentium processor (this performance figure was obtained for a circuit with approximately 100 transistors; the speed up advantage increases with the size of the circuit).

3. LOW TEMPERATURE EXPERIMENTS

This chapter focuses on analog/digital and mixed-signal electronics at low temperatures. The literature (Hairapetian, et al., 1989) reports studies on the change in characteristics of CMOS devices at low temperatures. The experiments cover separate tests of the whole evolvable hardware system: the evolutionary processor (the DSP in the SABLE system), data converters; and the FPTA tested at extreme low temperatures.

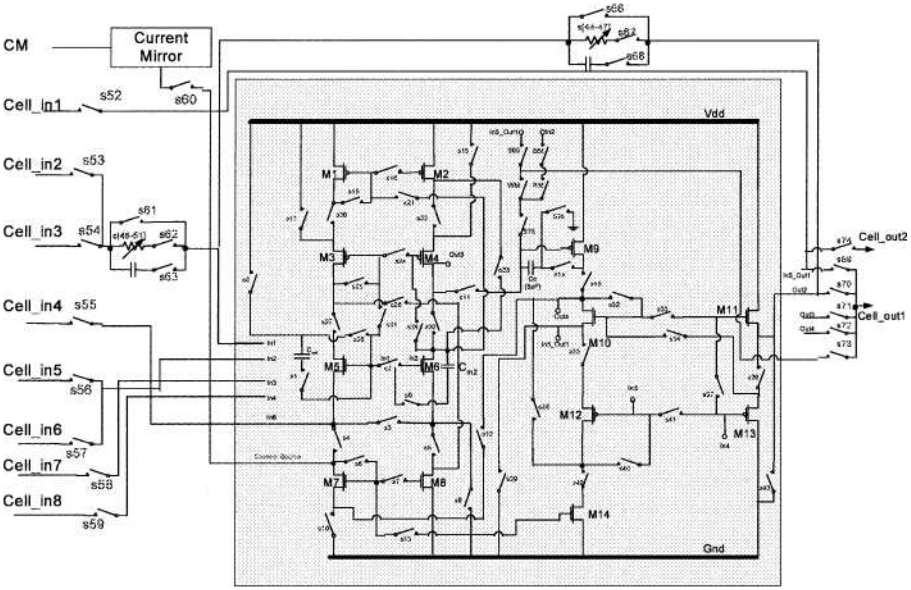


Figure 9-1. Schematic of the FPTA-2 cell

3.1 DSP Tests at Low Temperatures

Previous experiments focused exclusively on the tests of the FPTA chips at EEs. However, no tests have been reported so far on the behavior of the evolutionary processors at EEs. This particular experiment focuses on low temperature characterization of the DSP working as the EP.

A 320C6701 DSP was tested in a board fabricated by Innovative Integration (SBC62). The board communicates with a PC through a JTAG connection. During the test only the DSP board was placed on the low temperature chamber; the PC and the JTAG were outside. The FPTA chip was not used in this arrangement.

The DSP was tested by running a simple GA (Genetic Algorithm) whose target was a simple optimization problem (the maximization of the number of '1's in the chromosomes). This problem is solved in less than 1 minute, after 464 generations. The GA results are deterministic, i.e., the same for each run.

The temperature of the chamber/test article has been driven to 0°C with a scan rate of 5°C/min from room temperature. The dwell time at 0°C temperature was for 8 minutes and electrical measurements were made during this time. Later, the temperature of the chamber has been driven to –30°C, –60°C, –90°C, –120°C at a scan rate of 5°C/min and electrical measurements were made respectively during the dwell (Figure 9-2).

A failure was observed during the testing at –120°C step. Electrical measurements were made at –90°C again and the DSP regained its characteristics. This procedure was repeated again; the temperature was driven to –90°C, –100°C, –110°C and –120°C to narrow the temperature range. The dwell time at each temperature was for 5 minutes and electrical measurements were made during this time. The DSP was functioning at –90°C, –100°C, and –110°C. The failure was again observed during the testing in a temperature range of –110°C to –120°C. During the failure, the DSP did not communicate with the PC. The PC-DSP communication link was the only means to read out the DSP outputs in this experiment.

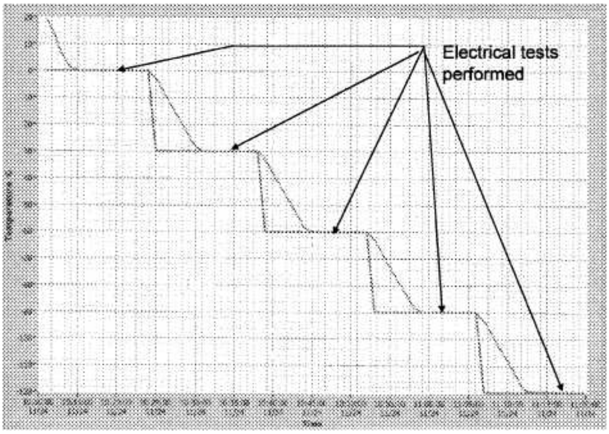


Figure 9-2. Temperature profile in the DSP test

Other evolutionary processors implementations, including Field Programmable Gate Arrays (FPGAs) and other DSP models, will be tested. The final goal of the experiments is to have an implementation operational at –180°C or below.

3.2 Data Converters

Data converters were also characterized at extreme low temperatures. Particularly, the following components are planned to be used in future evolvable systems working at low temperatures: AD9772A (Digital-to-Analog Converter, DAC) and AD6645 (Analog-to-Digital Converter, ADC).

The AD9772A is a 14-bit single-supply, oversampling digital-to-analog converter optimized for baseband or IF waveforms. Unfortunately, the waveforms used by JPL spread a broad range of frequencies which require both accuracy and speed, which industry usually do not supply in a single chip. In this case accuracy was preferred over sampling speed because several bits of precision are predicted to be lost due to noise, gain, and temperature effects when integrated to the evolutionary platform.

A ribbon cable was first created to communicate with the DAC. This was taped to the outside of the oven and was wired in, along with banana plugs for power and coaxial cables with BNC connectors for analog output. The ribbon cables' individual lines were then wire-wrapped to two breakout boxes just outside the oven. Two NI-DAQ boards (one for output, one for input) were then wired to the breakout boards with a proprietary cable. Below is a diagram of the setup.

A program was then created in National Instruments Labview that output 2.5V or 0.5V along 14 analog channels corresponding to a digital value of 1 or 0 on the NI-DAQ output board. The program output every code from 0 to 16383 and for each code read back in (on the NI-DAQ input board) 100 analog values. The program then averaged these values and output a formatted file with the digital value and the corresponding analog value.

A second Labview program controlled the temperature of the oven according to specifications, by controlling the amount of liquid nitrogen used to cool it.

Both programs were run simultaneously and the DAC underwent a test at room temperature (25°C), then subsequent tests from 0°C to −180°C in increments of 30°C. Time was provided at each interval for the stabilization of the temperature, a complete testing of digital codes, and about 5 to 6 extra minutes before continuing to the next temperature. Each test took approximately 70 minutes.

The formatted file was then read in by a C++ program that computed the offset, gain, Integral Nonlinearity (INL) and Differential Nonlinearity (DNL). The program output an INL and a DNL file with it that provided their values at each digital code. A few scripts in Matlab then processed all the data for visual display.

The differential nonlinearity metrics were used to characterize the DAC. Differential nonlinearity (DNL) is computed with the equation:

$$\mathrm{DNL} = (V_{d+1} - V_d) - (V_{\mathrm{LSB\text{-}IDEAL}}) \text{ where } 0 < d < 2^{N-2}.$$

This records the change from one step to another in the unit of Least Significant Bits (LSBs). The DAC performed very well using the DNL metric. The results show that the chip was not damaged in the process of testing and retained its DNL properties after returning to room temperature. It should be noted that slight degradation is apparent in the standard deviation below −90°C. The DAC is guaranteed to have values of +/−2.0 LSBs for DNL in the specified range of −40°C to 85°C. The experiment performed qualifies this claim, since only at –120°C there is a slight shift to the right past +2.0 LSBs (the DNL should stay between –1LSB and 1LSB). Figure 9-3 illustrates a histogram of the DNL at room temperature and at –180°C. It can be seen that the DNL clearly increases at –180°C.

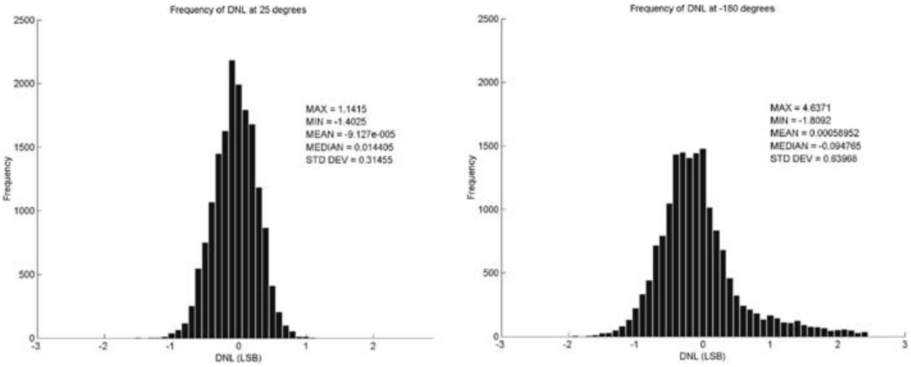


Figure 9-3. DNL for AD9772 at 25°C (left) and –180°C (right)

The Analog Devices, Inc. AD6645 is a 14-bit 80/105 MSPS analog-to-digital converter maintaining performance up to 200MHz. It has a differential analog input comprised of pins AIN and –AIN; these input pins take signals with a DC level of 2.4 V and an AC level of +/– .55 V. This allows a differential analog input signal of 2.2 Vpp. This ADC also requires an encode clock signal, which must be fed a high-quality, low phase-noise source to maximize signal to noise ratio. On the evaluation board, there is also an inductor, a capacitor and several resistors that will contribute noise and nonlinearity.

A flat ribbon cable was used to connect the ADC evaluation board to the FIFO board outside the oven. Power lines were brought outside with molded banana cables. The SMA signal jacks on the evaluation board were connected to a SMA-M to BNC-F converter and then brought out using 50 ohm BNC coaxial cable. The FIFO board was then hooked up to a PC through a USB cable.

Analog Devices included software for data acquisition with the FIFO board. The provided ADC Analyzer^{TM} was inadequate to test the ADC since it could only take 16383 samples at a time. To measure INL and DNL accurately using the histogram method, about 4 million samples are needed. For this reason, a Windows macro creator was installed called Autoit and a macro was written to automate capturing 200 sets of data.

Proper technique for testing an ADC involves slightly clipping the input wave. This amounts to several more codes at the boundaries (0 and 16383 in this case) and it allows a more accurate calculation of INL and DNL. In this case a 1.75Vpp covered the full range of the ADC.

For each 30°C increment from –180°C to +25°C, 200 files of 16383 samples were acquired. Next, a MATLAB script was written to plot a histogram and calculate the mean values. From this histogram, INL and DNL were calculated.

ADCs are often tested using the histogram method. This method relies on the fact that there is a specific probability density function for sinusoidal waves, governed by the equation:

$$p(V) = \frac{1}{\pi\sqrt{A-V}},$$

where A = amplitude of the input sine wave; and V = voltage correspondent to a particular digital code.

When plotted, this equation yields (Figure 9-4):

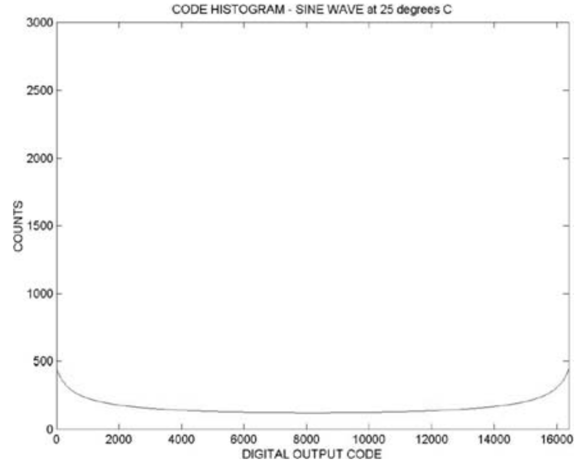


Figure 9-4. Ideal ADC histogram

If an adequate number of samples are collected, a histogram of ADC output should mimic this equation closely. Any deviation can be used to calculate Differential Nonlinearity (DNL) and Integral Nonlinearity (INL), which can be used to assess the accuracy of the ADC. A sample at 25°C shows a nice bathtub-like distribution for the AD6645 under test (Figure 9-5, left). However, by –180°C, several chunks of codes are missing (Figure 9-5, right).

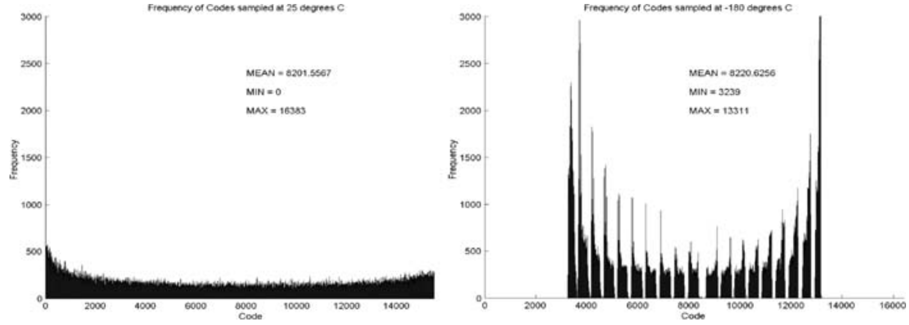


Figure 9-5. AD6645 histogram at room temperature (left) and –180°C (right)

Though performance deteriorates, with compensation the ADC-DAC chain can be effective between the temperatures of –180°C to 25°C with 8 to 9 effective numbers of bits (which is enough for our target applications).

3.3 FPTA

3.3.1 NOR Gate

A NOR gate was evolved at –196.6°C using the same method described in Section 3.2. Two FPTA-2 cells were used and the experiment processed 100 individuals along 300 generations. Figure 9-6a shows the oscilloscope picture of the evolved solution at –196.6°C. The same solution was tested at room temperature using another FPTA-2 chip, producing an almost identical behavior (Figure 9-6b) opposing to the rectifier case study.

3.3.2 Recovery of Controllable Oscillator at Low Temperatures

Four cells of the FPTA-2 were used to evolve a controllable oscillator. This circuit receives a digital input and it should oscillate when the input is at one digital level (either '0' and '1') and stay at ground for the other level. Initially, a controllable oscillator was evolved at room temperature, the circuit behavior being depicted in Figure 9-7a. The circuit outputs a 70kHz sine

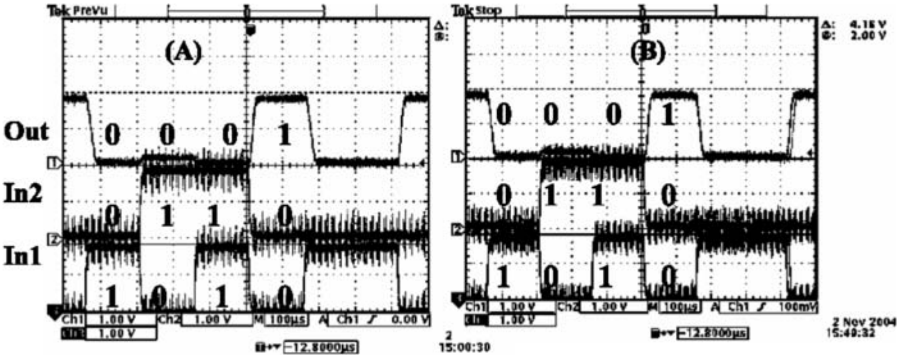


Figure 9-6. NOR circuit evolved and tested at –196.6°C; (A) the same circuit was tested successfully at room temperature (B) an environmental noise signal is also present at the circuit input

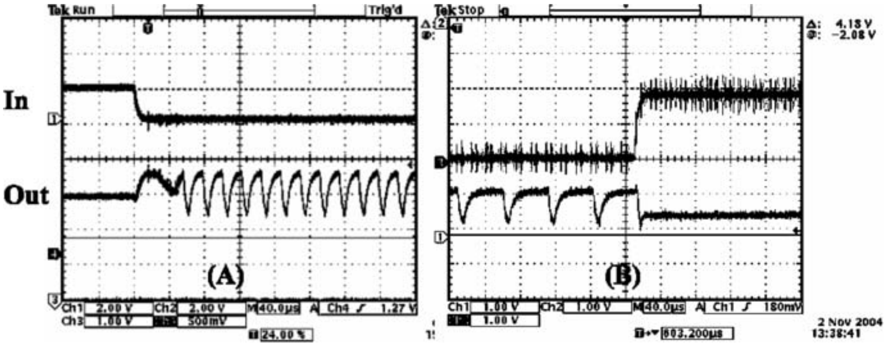


Figure 9-7. Evolved controllable oscillator at room temperature and deteriorated response at –196.6°C

wave (with a small degree of harmonic components) when the input is ‘0’. When the same circuit is tested at –196.6°C, it can be observed a distortion (increase in harmonics) at the output (Figure 9-7b).

The controllable oscillator was evolved again at –196.6°C, the response being displayed in Figure 9-8. It can be observed that the output distortion is largely removed. In addition, evolution found a circuit that oscillates for a high level input, opposing to the room temperature solution.

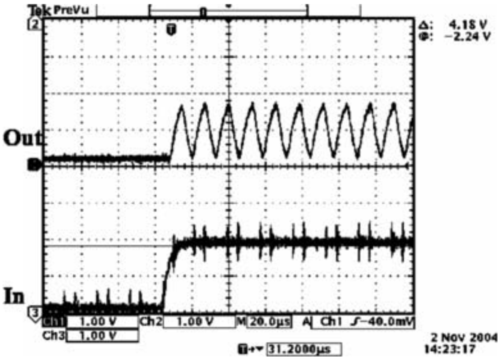


Figure 9-8. Evolved controllable oscillator at low temperature

4. CONCLUSIONS AND FUTURE WORK

The data converter characterization results at –180°C were satisfactory, since they still kept enough accuracy needed for the reconfiguration algorithm to perform function recovery. It was also demonstrated that the FPTA could be reconfigured for functionality recovery at this low temperature. However, the DSP characterization showed that the device failed at –120°C, which is not satisfactory for our program objectives.

Ongoing tests using a particular FPGA as evolutionary processor shows that this device can potentially work until –180°C. Near future work will demonstrate an overall evolutionary system (FPGA + Data Converters + reconfigurable device) working at –180°C.

Acknowledgments

The work described in this chapter was performed at the Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the National Aeronautics and Space Administration (NASA). NASA Program Managers for this effort were Dr. Mita Desai and Dr. Chris Moore.

References

Ferguson, M. I., et al. 2002. An Evolvable Hardware Platform based on DSP and FPTA. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2002)*, July 7-11, 2002, 145-152, Menlo Park, CA. AAAI Press.

Hairapetian, A., D. Gitlin, C. R. Viswanathan, 1989. "Low-Temperature Mobility Measurements on CMOS Devices", *IEEE Transactions on Electron Devices*, V. 36, N. 8, August, 1989.

Stoica, A., et al. 2002. "Evolving Circuits in Seconds: Experiments with a Stand-Alone Board-Level Evolvable System." *2002 NASA/DoD Conf. on Evolvable Hardware,* July 15-18, 2002, 67-74. IEEE Computer Press.

Stoica, A., et al. 2004. Circuit Self-Recovery Experiments in Extreme Environments. In *Proceedings of the 2004 NASA/DoD Conference on Evolvable Hardware,* 142-145, Seattle, USA.

Terry, B. J., et al. 2004. "Development of Robust Analog and Mixed-Signal Electronics for Extreme Environments Applications", *IEEE Aerospace Conference,* Big Sky, MT, USA.

Chapter 10

HUMAN-COMPETITIVE EVOLVABLE HARDWARE CREATED BY MEANS OF GENETIC PROGRAMMING

John R. Koza[1], Martin A. Keane[2], Matthew J. Streeter[3],
Sameer H. Al-Sakran[4], and Lee W. Jones[4]
[1]Stanford University, Stanford, California, Email: koza@stanford.edu; [2]Econometrics Inc., Chicago, Illinois; [3]Carnegie Mellon University, Pittsburgh, Pennsylvania; [4]Genetic Programming Inc., Mountain View, California

Abstract: Genetic programming is a systematic method for getting computers to automatically solve problems. Genetic programming is an extension of the idea of the genetic algorithm into the arena of computer programs. Genetic programming uses the Darwinian principle of natural selection and analogs of recombination (crossover), mutation, gene duplication, gene deletion, and certain mechanisms of developmental biology to progressively breed, over a series of many generations, an improved population of candidate solutions to a problem. Many human-competitive results have been produced using the genetic programming technique, including the automated reinvention of previously patented inventions. This chapter concentrates on the automatic synthesis of six 21st century patented analog electrical circuits by means of genetic programming. The automatic synthesis of analog circuits is done "from scratch"—that is, without starting from a preexisting good design and without prespecifying the circuit's topology or number or sizing of components. This chapter also briefly summarizes some examples of the automatic synthesis of other types of complex structures by means of genetic programming.

Key words: evolvable hardware, analog electrical circuits, genetic programming, automated design, developmental process, reinvention of previously patented entity, human-competitive result.

1. INTRODUCTION

Design of complex structures is a major activity of practicing engineers. Engineers are often called upon to design complex structures (e.g., electrical circuits, controllers, antennas, optical systems, and mechanical systems) that satisfy prespecified high-level design and performance goals. Some designs are sufficiently creative enough that they are considered to be inventions.

In design problems in many fields (e.g., analog electrical circuits, optical lens systems, antennas, controllers, mechanical systems, quantum computing circuits), existing mathematical methods and software tools enable a practicing engineer to *analyze* the behavior and characteristics of a particular already-designed structure. However, in these fields, there is typically no general mathematical solution to the inverse problem, namely the problem of *synthesizing* a structure from a high-level specification of the structure's desired behavior and characteristics. In fact, the problem of automatically synthesizing the design of complex structures in these fields is generally regarded as an art, rather than a science.

In designing complex structures, human engineers typically employ logic and domain knowledge about the field of interest. Conventional approaches to automated design (such as those employing artificial intelligence) are typically knowledge-intensive, logically sound, and deterministic.

Tellingly, two of the most successful approaches to design—namely the evolutionary process (occurring in nature) and the invention process (performed by creative humans)—are not logical, deterministic, or knowledge-intensive. The fact that these two highly successful approaches are so different from conventional approaches to automated design (employing artificial intelligence) suggests that there may be important lessons to be learned from the evolutionary process and the invention process.

In nature, solutions to design problems are discovered by means of evolution and natural selection. Evolution is not deterministic. It does not rely on a knowledge base and is certainly not guided by mathematical logic. Indeed, one of the most important characteristics of the evolutionary process is that it actively generates and encourages the maintenance of inconsistent and contradictory alternatives. Logically sound systems do not do that. In fact, the generation and maintenance of inconsistent and contradictory alternatives (called *genetic diversity*) is a precondition for the success of the evolutionary process.

Likewise, the invention process (performed by creative humans) is a nondeterministic process that is not governed by logic. The invention process is typically characterized by a singular moment when the prevailing thinking concerning a long-standing problem is, in a "flash of genius," overthrown and replaced by a new approach that could not have been logically deduced from what was previously known. That is, inventions are characterized by a logical discontinuity that distinguishes the creative new design from that which can be logically deduced from what was previously known.

Patent law provides some insight into the invention process. In this connection, it is noteworthy that a purported invention is not considered to be worthy of a patent if the new idea can be logically deduced from facts that are well-known in a field or by means of transformations that are well-known in the field. A new idea is patentable only if there is an "illogical step"—that is to say, a logically unjustified step. In the patent law, this legally-required illogical step is sometimes referred to as a "flash of genius" and it is the essence of inventiveness and creativity.

In short, both the invention process and the evolutionary process in nature are very different from conventional approaches to automated design (such as those employing artificial intelligence).

Genetic programming is an extension of the idea of the genetic algorithm into the arena of computer programs. Genetic programming starts from a high-level statement of what needs to be done and automatically creates a computer program to solve the problem. Genetic programming uses the Darwinian principle of natural selection and analogs of recombination (crossover), mutation, gene duplication, gene deletion, and certain mechanisms of developmental biology to progressively breed an improved population over many generations (Koza, 1990, 1992, 1994; Banzhaf, et al., 1998; Koza, et al., 1999; Langdon and Poli, 2002; Koza, et al., 2003). Recent work on genetic programming is reported in the *Genetic Programming and Evolvable Hardware* journal, the annual Genetic and Evolutionary Computation Conference (GECCO) (Deb, et al., 2004), the annual Euro-Genetic Programming conference (Keijzer, et al., 2005), the annual Genetic Programming Theory and Applications workshop (O'Reilly, et al., 2004), the annual Asia-Pacific Workshops on Genetic Programming (Cho, Nguyen, and Shan, 2003) as well as the proceedings of the many other conferences and journals (such as *Evolutionary Computation* and *IEEE Transactions on Evolutionary Computation*) in the field of genetic and evolutionary computation. For additional sources of information about genetic programming including links to software for implementing genetic programming visit `www.genetic-programming.org`.

Genetic programming has now been successfully used to automatically synthesize designs in a number of fields, including analog electrical circuits, optical lens systems, antennas, controllers, mechanical systems, quantum computing circuits, and networks of chemical reactions. In fact, many of the automatically created designs produced by genetic programming are "human-competitive" (as defined in Koza, et al., 1999 and Koza, et al., 2003), including the automatic synthesis of

- analog electrical circuits patented in the 20th century (Koza, et al., 1999) and 21st century (Koza, et al., 2003),
- an X-Band Antenna for NASA's Space Technology 5 Mission (Lohn, Hornby, and Linden, 2004),
- quantum computing circuits (Spector, 2004),

- a previously patented mechanical system (Lipson, 2004),
- previously patented controllers (Koza, et al., 2003),
- new controller designs that have been granted patents (Keane, Koza, and Streeter, 2005), and
- previously patented optical lens systems (Al-Sakran, Koza, and Jones 2005; Koza, Al-Sakran, and Jones, 2005).

This chapter concentrates on the automatic synthesis of analog electrical circuits by means of genetic programming.

In particular, Section 2 describes our method for automatically synthesizing analog electrical circuits by means of genetic programming.

Section 3 presents our results in automatically synthesizing six 21st century patented analog electrical circuits.

Section 4 briefly describes the automatic synthesis of two other kinds of complex structures by means of genetic programming, namely

- six optical lens systems that were previously patented, and
- two new controller designs that have been granted patents.

Section 5 is the conclusion.

2. METHOD FOR AUTOMATICALLY SYNTHESIZING ANALOG CIRCUITS BY MEANS OF GENETIC PROGRAMMING

The field of evolvable hardware was pioneered in the early 1990's (Higuchi, et al., 1993a, 1993b). In the mid-1990's, the genetic algorithm (Holland, 1975) was successfully used to automatically synthesize analog circuits (Kruiskamp and Leenaerts, 1995; Grimbleby, 1995) and for programming a digital field programmable gate array (FPGA) to perform analog functions (Thompson, 1996).

The design process for electrical circuits begins with a high-level description of the circuit's desired behavior and characteristics. The process entails creation of both the topology and the sizing of a satisfactory circuit.

The *topology* of a circuit comprises

- the total number of components in the circuit,
- the type of each component (e.g., resistor, capacitor, transistor) at each location in the circuit,
- a list of all the connections that exist between the leads of the circuit's components, input ports, output ports, power sources, and ground.

The *sizing* of a circuit consists of the parameter value(s) associated with each component.

Genetic programming was first used to automatically synthesize both the topology and sizing of analog electrical circuits in 1995 (Koza, et al., 1996a, 1996b). Since then, both the topology and sizing of numerous analog electrical circuits composed of transistors, capacitors, resistors, inductors, and other components have been automatically synthesized (Koza, et al., 1999 and Koza, et al., 2003).

The automatic synthesis of analog circuits is done "from scratch"—that is, without starting from a preexisting good design and without prespecifying either topology or component sizing.

Our approach to the problem of automatically creating both the topology and sizing of an electrical circuit by means of genetic programming involves

- establishing a developmental representation for electrical circuits involving program trees, and
- defining a fitness measure that quantifies how well the behavior and characteristics of a candidate circuit satisfy high-level design requirements.

During the run of genetic programming, the evaluation of the fitness of each individual in the population involves

- converting each individual program tree in the population into a netlist for an electrical circuit using the developmental process,
- ascertaining the circuit's behavior and characteristics, and
- using the circuit's behavior and characteristics to calculate a value of fitness.

Electrical circuits are ordinarily represented as labeled graphical structures with cycles (circuit diagrams). However, the program trees ordinarily used in genetic programming are *acyclic* graphs. Our approach to the automatic synthesis of circuits using genetic programming employs a developmental process to bridge this representational difference and convert a program tree into a circuit.

The developmental process used for automated synthesis of circuits by means of genetic programming transforms a program tree (an acyclic graph) into a fully developed electrical circuit (a graph structure with cycles). This approach is inspired by the principles of developmental biology, the pioneering ideas of Wilson (1987), the innovative work of Kitano (1990) on using developmental genetic algorithms to evolve neural networks, the creative work of Gruau (1992) on using developmental genetic programming (cellular encoding) to evolve neural networks, and the use of developmental processes to create complex structures (Koza, 1993).

The developmental process used for the automated synthesis of circuits by means of genetic programming entails the execution of functions in a circuit-constructing program tree. These functions include component-creating functions, topology-modifying functions, development-controlling functions, arithmetic-performing functions, and automatically defined functions.

The starting point for our developmental process consists of a simple initial circuit. The initial circuit consists of an embryo and a test fixture. The embryo contains at least one modifiable wire. All development originates from the embryo's modifiable wire(s).

An electrical circuit is developed by progressively applying the functions in a circuit-constructing program tree to the modifiable wires of the original embryo and, as the circuit grows, to the modifiable wires and modifiable components that sprout from it. The execution of the functions in the program tree transforms the initial circuit into a fully developed circuit. That is, the functions in the circuit-constructing program tree progressively side-effect the embryo and its successor structures until a fully developed circuit emerges.

A test fixture (external to the entity that is being automatically created) facilitates measurement of the performance of the fully developed circuit. The test fixture is a hard-wired structure composed of nonmodifiable wires and nonmodifiable electrical components. The test fixture feeds external inputs into the circuit that is being evaluated. It also enables the circuit's outputs to be probed. The test fixture supplies the measurements that enable the fitness measure to assign a single numerical value of fitness to the behavior and characteristics of the fully developed circuit.

The functions in the circuit-constructing program trees are divided into five categories:

- component-creating functions that insert components (e.g., resistors, capacitors, transistors) into the developing circuit,
- topology-modifying functions (e.g., series division, parallel division, cut, via to connect two distant points in a circuit, via to connect a point in the circuit to ground, via to connect a point to a power supply, via to connect a point to the incoming signal, via to connect a point to an output port) that modify the topology of the developing circuit,
- development-controlling functions that control the developmental process by which the embryo and its successor structures are converted into a fully developed circuit (e.g., the development-ending function),
- arithmetic-performing functions (e.g., addition, subtraction) that may appear in a value-setting subtree that is an argument to a component-creating function and that specifies the numerical value of the component, and
- automatically defined functions that enable certain substructures to be reused (including parameterized reuse).

The component-creating functions generally have value-setting subtree(s) that establish the value of the component (e.g., the capacitance of a capacitor).

Many of the component-creating and topology-modifying functions possess one or more construction-continuing subtrees.

The terminals in the circuit-constructing program trees may include

- constant numerical values,
- perturbable numerical values,
- symbolic values (used, for example, to create discrete alternatives for certain components),
- zero-argument functions (e.g., the development-ending function, zero-argument automatically defined functions), and
- externally supplied free variables (used, for example, when a circuit is to be parameterized by an external value).

In the usual implementation of genetic programming, all the individual program trees in the initial random population (generation 0) are syntactically valid executable programs. All the genetic operations used in genetic programming (i.e., crossover, mutation, reproduction, and the architecture-altering operations) operate so as to create syntactically valid executable programs from syntactically valid executable programs. Thus, all the individuals encountered during the run (including, in particular, the best-of-run individual produced upon completion of the run) are necessarily syntactically valid executable programs.

Each circuit-constructing program tree in a run of genetic programming for automatic circuit synthesis is created in accordance with a constrained syntactic structure (strong typing) that imposes grammatical constraints on how the available functions and terminals may be combined. For example, a value-setting subtree establishing the numerical value of a capacitor is permitted to appear only as a designated argument of the capacitor-creating function. All the individuals in generation 0 of a run of genetic programming comply with the constrained syntactic structure appropriate for automated circuit synthesis. All the genetic operations that are performed during the run operate so as to preserve the constrained syntactic structure. Thus, all the individuals encountered during the run (including the best-of-run individual) comply with the constrained syntactic structure.

Our developmental approach is more than just a mechanism for mapping an acyclic graph (the circuit-constructing program tree) into a graphical structure with cycles (the fully developed circuit). That is, the developmental approach does far more than just enable us to overcome a representation obstacle.

The developmental process has the advantage of preserving the electrical validity of the circuit. There are no unconnected leads in the initial circuit. Each component-creating, topology-modifying, and development-controlling function operates so as to preserve this connectivity at each stage of the developmental process. The result is that there are no unconnected leads in the fully developed circuit.

To summarize the foregoing points, *every* developmental step associated with *every* individual circuit in *every* generation of the population of a run of genetic programming using our approach is

- syntactically valid,
- executable,
- compliant with required constrained syntactic structure, and
- electrically valid (connected).

The developmental approach has the additional advantage of preserving locality. Most of the component-creating, topology-modifying, and development-controlling functions intentionally operate on a small local area of the circuit. Subtrees within a program tree therefore tend to operate locally. The crossover operation (the main workhorse of genetic programming and genetic algorithms) transplants subtrees. It should be remembered that the premise behind the crossover operation in genetic programming (and the genetic algorithm) is that individuals with relatively high fitness are likely to contain some local substructures which, when recombined, may (at least some of the time) create offspring with even higher fitness. In genetic programming, the conventional crossover operation recombines a subtree from one parent's program tree with the second parent's program tree. Over many generations, functions and terminals that are close together in a program tree tend to be differentially and preferentially preserved by crossover. In particular, smaller subtrees are preserved to a greater degree than larger ones. Moreover, when circuits are represented by circuit-constructing program trees containing the functions that we use, a subtree tends to represent a local area in the fully developed circuit. Thus, the crossover operation works in conjunction with the developmental process in preserving locality. The mutation operation and architecture-altering operations similarly work in conjunction with the developmental process in preserving locality (because they affect subtrees).

Useful parts of a circuit-constructing program tree can be reused. Real-world circuits are replete with reuse. Reuse eliminates the need to "reinvent the wheel" on each occasion when a particular sequence of steps may be useful. Reuse makes it possible to exploit a problem's modularities, symmetries, and regularities and thereby potentially accelerate the problem-solving process (Koza, Keane, and Streeter 2003).

The efficiency of genetic programming in the domain of automatic circuit synthesis stems from the combined effects of the

- preservation of syntactic validity,
- preservation of executability of the circuit-constructing program trees,
- preservation of constrained syntactic structure,
- preservation of electrical connectivity,
- preservation of locality during crossover (and other operations), and
- the facilitation of reuse.

3. AUTOMATIC SYNTHESIS OF SIX 21ST CENTURY PATENTED CIRCUITS BY MEANS OF GENETIC PROGRAMMING

This section presents the six instances where genetic programming automatically created both the topology (graphical structure) and sizing (numerical component values) for analog electrical circuits composed of transistors, capacitors, and resistors that were patented after January 2000 (as shown in Table 10-1).

Table 10-1. Six 21st century patents for analog electrical circuits

Invention	**Date**	**Inventor**	**Place**	**Patent**
Cubic function generator	2000	Stefano Cipriani and Anthony A. Takeshian	Conexant Systems, Inc.	U. S. 6,160,427
Mixed analog-digital variable capacitor circuit	2000	Turgut Sefket Aytur	Lucent Technologies, Inc.	U. S. 6,013,958
Voltage-current conversion circuit	2000	Akira Ikeuchi and Naoshi Tokuda	Mitsumi Electric Co., Ltd.	U. S. 6,166,529
Low-voltage balun circuit	2001	Sang Gug Lee	Information and Communications University	U. S. 6,265,908
High-current load circuit	2001	Timothy Daun-Lindberg and Michael Miller	International Business Machines Corporation	U. S. 6,211,726
Tunable integrated active filter	2001	Robert Irvine and Bernd Kolb	Infineon Technologies AG	U. S. 6,225,859

In each instance, genetic programming started from a high-level statement of a circuit's desired behavior and characteristics (e.g., its desired output given its input). In producing results, genetic programming used only *de minimus* knowledge about analog circuits. Specifically, genetic programming employed a circuit simulator (e.g., SPICE) for the *analysis* of candidate circuits, but did not use any deep knowledge or expertise about the *synthesis* of circuits.

The human user communicates the high-level statement of the problem to the genetic programming algorithm by performing certain well-defined preparatory steps.

The five major preparatory steps for the basic version of genetic programming require the human user to specify

- the set of terminals (e.g., the independent variables of the problem, zero-argument functions, and random constants) for each branch of the to-be-evolved program,
- the set of primitive functions for each branch of the to-be-evolved program,
- the fitness measure (for explicitly or implicitly measuring the fitness of individuals in the population),
- certain parameters for controlling the run, and
- the termination criterion and method for designating the result of the run.

The first two preparatory steps specify the ingredients that are available to create the computer programs. A run of genetic programming is a competitive search among a diverse population of programs composed of the available functions and terminals. The function and terminal sets for all six problems (described below) permit the construction of any circuit composed of transistors, resistors, and capacitors.

The third preparatory step concerns the fitness measure for the problem. The fitness measure specifies what needs to be done. The fitness measure is the primary mechanism for communicating the high-level statement of the problem's requirements to the genetic programming system. The main difference among the runs of genetic programming for the six problems is that we supplied a different fitness measure for each problem. Construction of a fitness measure requires translating the problem's high-level requirements into a precise computation. In each case, we read the patent document to find the performance that the invention was supposed to achieve. We then created a fitness measure quantifying the invention's performance and characteristics. The fitness measure specifies the desired time-domain output values or frequency-domain behavior. For each problem, a test fixture consisting of certain fixed components (such as a source resistor, a load resistor) is connected to the desired input ports and the desired output ports. Circuits may be simulated using a simulator such as SPICE (Quarles, et al., 1994).

We supplied models for transistors appropriate to the problem. We used the commercially common 2N3904 (*npn*) and 2N3906 (*pnp*) transistor models unless the patent document called for a different model. We used 5-Volt power supplies unless the patent specified otherwise.

The fourth and fifth preparatory steps are administrative. The fourth preparatory step entails specifying the control parameters for the run. The most important control parameter is the population size. Other control parameters include the probabilities of performing the genetic operations, the maximum size for programs, and other details of the run. The control parameters and termination criterion were the same for all six problems, except that we used different population sizes to approximately equalize each run's elapsed time per generation.

After the user has performed the preparatory steps for a problem, the run of genetic programming can be launched. Once the run is launched, a series of well-defined, problem-independent steps is executed.

Genetic programming typically starts with an initial population of randomly generated computer programs composed of the available programmatic ingredients (as provided by the human user in the first and second preparatory steps). These programs are typically created by recursively generating a rooted point-labeled program tree composed of random choices of the primitive functions and terminals. The initial individuals are usually generated subject to a preestablished maximum size (specified by the user as a minor parameter in the fourth preparatory step). In general, the programs in the population are of different size (number of functions and terminals) and of different shape (the particular graphical arrangement of functions and terminals in the program tree).

Genetic programming iteratively transforms a population of computer programs into a new generation of the population by applying analogs of naturally occurring genetic operations. These operations are applied to individual(s) selected from the population. The individuals are probabilistically selected to participate in the genetic operations based on their fitness (as measured by the fitness measure provided by the human user in the third preparatory step). The iterative transformation of the population is executed inside the main loop (called a *generation*) of a run of genetic programming.

Specifically, genetic programming breeds computer programs to solve problems by executing the following three steps:

(1) Generate an initial set (called the population) of compositions (typically random) of the functions and terminals appropriate for the problem.

(2) Iteratively perform the following group of substeps (called a generation) on the population of programs until the termination criterion has been satisfied:

  (A) Execute each program in the population and assign it a fitness value using the problem's fitness measure.

  (B) Create a new population (the next generation) of programs by applying the following operations to program(s) selected from the population with a probability based on fitness (with reselection allowed):

    (i) Reproduction: Copy the selected program to the new population.

   (ii) Crossover: Create a new offspring program for the new population by recombining randomly chosen parts of two selected programs.
   (iii) Mutation: Create one new offspring program for the new population by randomly mutating a randomly chosen part of the selected program.
   (iv) Architecture-altering operations: Create one new offspring program for the new population by applying a selected architecture-altering operation to the selected program.

(3) Designate an individual program (e.g., the individual with the best fitness) as the run's result. This result may be a solution (or approximate solution) to the problem.

Genetic programming is problem-independent in the sense that the above sequence of executional steps is not modified for each new run or each new problem. There is usually no discretionary human intervention or interaction during a run of genetic programming (although a human user often exercises judgment as to when to terminate a run).

3.1 Fitness Measures for the Six Analog Circuits

We now describe the six analog circuits. Details of these six circuits may be found in *Genetic Programming IV: Routine Human-Competitive Machine Intelligence* (Koza, et al., 2003).

3.1.1 Low-Voltage Balun Circuit

The purpose of a balun (balance/unbalance) circuit is to produce two outputs from a single input, each output having half the amplitude of the input, one output being in phase with the input while the other is 180 degrees out of phase with the input, and with both outputs having the same DC offset. The patented balun circuit (Lee, 2001) uses a power supply of only 1 Volt. The fitness measure consisted of (1) a frequency sweep analysis designed to ensure the correct magnitude and phase at the two outputs of the circuit and (2) a Fourier analysis designed to penalize harmonic distortion.

3.1.2 Mixed Analog-Digital Register-Controlled Variable Capacitor

This mixed analog-digital circuit (Aytur, 2000) has a capacitance that is controlled by the value stored in a digital register. The fitness measure for this problem employed 16 time-domain fitness cases. The 16 fitness cases ranged over all eight possible values of a 3-bit digital register for two different analog input signals.

3.1.3 Voltage-Current Conversion Circuit

The purpose of the voltage-current conversion circuit (Ikeuchi and Tokuda, 2000) is to take two voltages as input and to produce a stable current whose magnitude is proportional to the difference of the voltages. We employed four time-domain input signals (fitness cases) in the fitness measure. We included a time-varying voltage source beneath the output probe point to ensure that the output current produced by the circuit was stable with respect to any subsequent circuitry to which the output of the circuit might be attached.

3.1.4 High-Current Load Circuit

The patent for the high-current load circuit (Daun-Lindberg and Miller, 2001) covers a circuit designed to sink a time-varying amount of current in response to a control signal. The patented circuit employs a number of FET transistors arranged in parallel, each of which sinks a small amount of the desired current. The fitness measure for this problem consisted of two time-domain simulations, each representing a different control signal.

3.1.5 Low-Voltage Cubic Signal Generator

The patent for the low-voltage cubic signal generator (Cipriani and Takeshian, 2000) covers an analog computational circuit that produces the cube of an input signal as its output. The circuit is "compact" in that it contains a voltage drop across no more than two transistors. The fitness measure for this problem consisted of four time-domain fitness cases using various input signals and time scales. The compactness constraint was enforced by providing only a 2-Volt power supply.

3.1.6 Tunable Integrated Active Filter

The patent for the tunable active filter (Irvine and Kolb, 2001) covers a tunable integrated active filter that performs the function of a lowpass filter whose passband boundary is dynamically specified by a control signal. The circuit has two inputs: a to-be-filtered incoming signal and a control signal. The fitness measure for this problem consisted of a performance penalty and a parsimony penalty. The passband boundary, *f*, ranges over nine values between 441 and 4,414 Hz. The performance penalty is a weighted sum, over 61 frequencies for each of the nine values of *f*, of the absolute weighted deviation between the output of the individual candidate circuit at its probe point and the target output. The parsimony penalty is equal to the number of components in the circuit.

3.2 Results for Six 21st Century Patented Circuits

This section presents the results for the six 21st century patented circuits.

3.2.1 Low-Voltage Balun Circuit

Genetic programming automatically created the circuit shown in Figure 10-1. This best-of-run evolved circuit was produced in generation 97 and has a fitness of 0.429. The patented circuit has a fitness of 1.72. That is, the evolved circuit is roughly a fourfold improvement (less being better) over the patented circuit in terms of our fitness measure. In addition, the evolved circuit is superior to the patented circuit both in terms of its frequency response and its harmonic distortion.

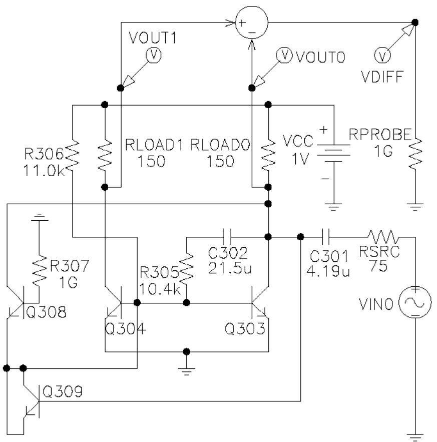


Figure 10-1. Best-of-run balun circuit

In the patent documents, Lee (2001) shows a previously known conventional (prior art) balun circuit. This previously known circuit is shown as Figure 10-2 of this chapter.

Lee's patented low-voltage balun circuit is shown in Figure 10-3 of this chapter. Lee (2001) states that the essential difference between the prior art and his invention is a coupling capacitor C_2 located between the base and the collector of the transistor Q_2. Lee explains the essence of his invention as follows:

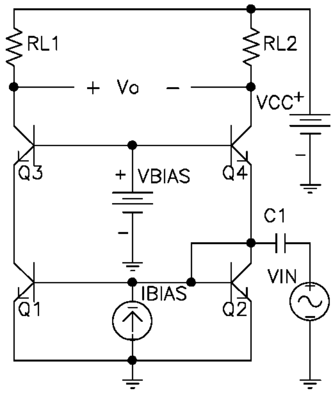


Figure 10-2. Prior art balun circuit shown in U.S. patent 6,265,908

> "The structure of the inventive balun circuit shown in [Figure 10-3 of this chapter] is identical to that of [Figure 10-2 of this chapter] except that a capacitor C_2 are further provided thereto. The capacitor C_2 is a coupling capacitor disposed between the base and the collector of the transistor Q_2 and serves to block DC components which may be fed to the base of the transistor Q_2 from the collector of the transistor Q_2."

As can be seen, the best-of-run genetically evolved circuit (Figure 10-1) possesses the very feature that Lee identifies as the essence of his invention, namely the coupling capacitor called "C_{302}" in Figure 10-1 and called "C_2" in Figure 10-3.

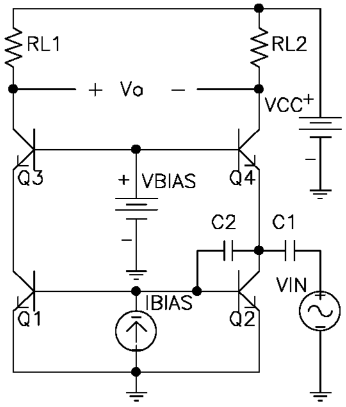


Figure 10-3. Lee's low-voltage balun circuit shown in patent 6,265,908

The genetically evolved circuit also reads on three additional elements of claim 1 of Lee's 2001 patent. However, the genetically evolved circuit achieves the circuit's remaining functionality in a manner different from that specified by the remaining elements of Lee's patent. Thus, although the evolved circuit captures the essence of Lee's invention and many of the secondary features of Lee's circuit, the evolved circuit does not infringe Lee's patent. This suggests that genetic programming can be used to create new inventions as well as to engineer around existing patents.

3.2.2 Mixed Analog-Digital Register-Controlled Variable Capacitor

Over our 16 fitness cases, the patented circuit has an average error of 0.803 millivolts. In generation 95, a circuit emerged with an average error of 0.808 millivolts, or approximately 100.6% of the average error of the patented circuit. During the course of this run, we harvested the smallest individuals produced on each processing node with a certain maximum level of error. Examination of these harvested individuals revealed a circuit from generation 98 (Figure 10-4) that approximately matches the topology of the patented circuit (without infringing). The genetically evolved circuit reads on all but one of the elements of claim 1 of the patented circuit (and hence does not infringe the patent).

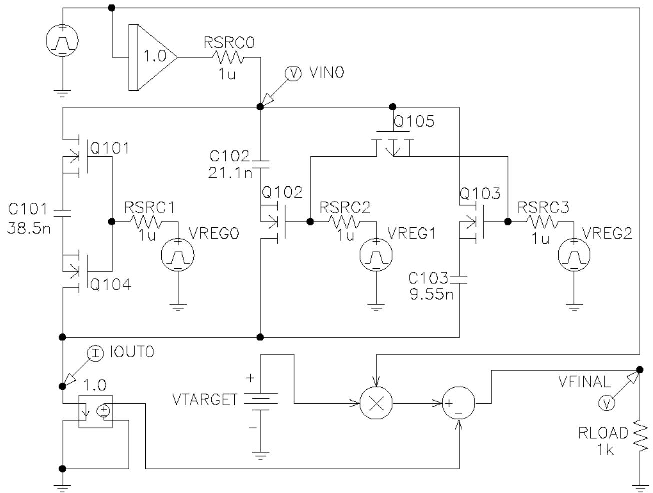


Figure 10-4. Evolved compliant register-controlled capacitor circuit

3.2.3 Voltage-Current Conversion Circuit

A circuit emerged on generation 109 of our run of this problem with a fitness of 0.619 (so that the evolved circuit has about 62% of the average weighted error of the patented circuit). The evolved circuit was subsequently tested on unseen fitness cases that were not part of the fitness measure and outperformed the patented circuit on these new fitness cases. The best-of-run circuit solves the problem in a different manner than the patented circuit.

3.2.4 High-Current Load Circuit

On generation 114, a circuit emerged that duplicated Daun-Lindberg and Miller's parallel FET transistor structure. This circuit has a fitness (weighted error) of 1.82 (so that the evolved circuit has about 182% of the weighted error of the patented circuit).

The genetically evolved circuit shares the following features found in claim 1 of U.S. patent 6,211,726:

> "A variable, high-current, low-voltage, load circuit for testing a voltage source, comprising: ..."
>
> "a plurality of high-current transistors having source-to-drain paths connected in parallel between a pair of terminals and a test load."

However, the remaining elements of claim 1 in U.S. patent 6,211,726 are very specific and the genetically evolved circuit does not read on these remaining elements. In fact, the remaining elements of the genetically evolved circuit bear hardly any resemblance to the patented circuit. In this instance, genetic programming produced a circuit that duplicates the functionality of the patented circuit using a different structure.

3.2.5 Low-Voltage Cubic Signal Generator

The best-of-run evolved circuit (Figure 10-5) was produced in generation 182 and has an average error of 4.02 millivolts. The patented circuit had an average error of 6.76 millivolts. That is, the evolved circuit has approximately 59% of the error of the patented circuit over our four fitness cases.

The claims in U.S. patent 6,160,427 amount to a very specific description of the patented circuit. The genetically evolved circuit does not read on these claims and, in fact, bears hardly any resemblance to the patented circuit. In this instance, genetic programming produced a circuit that duplicates the functionality of the patented circuit and does so using a very different structure.

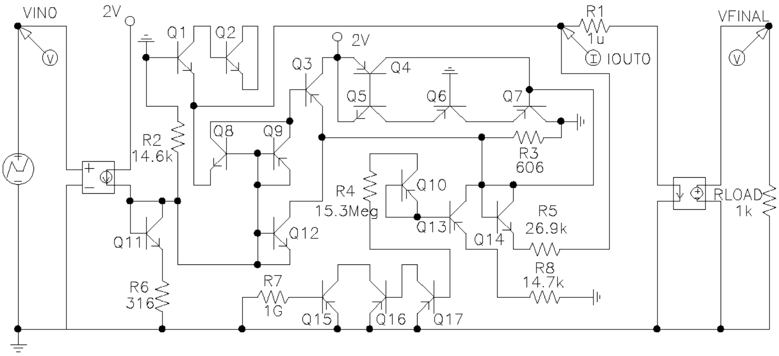


Figure 10-5. Best-of-run cubic signal generation circuit

3.2.6 Tunable Integrated Active Filter

Averaged over the nine values of frequency, the best-of-run circuit from generation 50 (Figure 10-6) has 72.7 millivolts average absolute error for frequencies in the passband and 0.39 dB average absolute error for other frequencies.

The best-of-run genetically evolved circuit reads on every element of claim 1 of U.S. patent 6,225,859 and therefore infringes the patent. That is, it infringes the patent and is essentially identical to the patented invention.

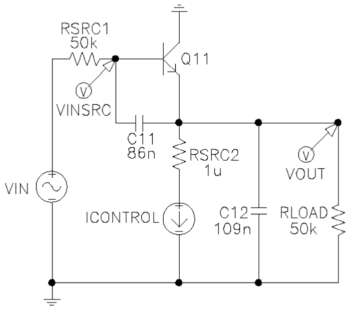


Figure 10-6. Best-of-run circuit for the tunable integrated active filter

Table 10-2 summarizes the results for the six problems involving the 21st century patented inventions in terms of their novelty.

Table 10-2. Summary of the six 21st century patented inventions

Problem	**Result**
Low-voltage balun circuit	The evolved circuit captures the essence of the previously patented circuit and many of its secondary features, but does not infringe the existing patent.
Mixed analog-digital variable capacitor	The evolved circuit reads on all but one of the elements of claim 1 of the patented circuit, but does not infringe the existing patent.
Voltage-current conversion circuit	The evolved circuit duplicates the functionality of the patented circuit, but in an entirely different way.
High-current load circuit	The evolved circuit duplicates the functionality of the patented circuit, but in an entirely different way.
Cubic function generator	The evolved circuit duplicates the functionality of the patented circuit, but in an entirely different way.
Tunable integrated active filter	The evolved circuit is an infringing duplicate of the previously patented circuit.

4. EXAMPLES OF AUTOMATIC SYNTHESIS OF OTHER COMPLEX STRUCTURES BY MEANS OF GENETIC PROGRAMMING

This section briefly describes the automatic synthesis of two other kinds of complex structures by means of genetic programming, namely

- six optical lens systems that were previously patented, and
- two new controller designs that have been granted patents.

4.1 Automatic Synthesis of Six Previously Patented Optical Lens Systems by Means of Genetic Programming

A complete design for a classical optical lens system encompasses numerous decisions, including the choice of the system's topology (that is, the number of lenses and their physical layout), choices for numerical parameters, and choices for nonnumerical parameters.

The layout decisions required to define a lens system include the sequential arrangement of lenses between the object and the image, decisions as to whether consecutive lenses touch or are separated by air, the nature of the mathematical expressions defining the curvature of each lens surface (traditionally spherical, but nowadays often aspherical), and the locations and sizes of the field and aperture stops that determine the field of view and the maximum illumination of the image, respectively.

The numerical choices include the thickness of each lens and the separation (if any) between lens surfaces, the numerical coefficients for the mathematical expressions defining the curvature of each surface (which, in turn, implies whether each is concave, convex, or flat), and the aperture (semidiameter) of each surface.

The nonnumerical choices include the type of glass (or other material) for each lens. Each type of glass has various properties of interest to optical designers, including the index of refraction, n (which varies by wavelength); the Abbe number, V; and the cost. Choices of glass are typically drawn from a standard glass catalog.

Genetic programming has recently been used as an invention machine to automatically synthesize complete designs for six optical lens systems that duplicated the functionality of previously patented lens systems (Al-Sakran, Koza, and Jones, 2005; Koza, Al-Sakran, and Jones, 2005). The six patented optical systems are shown in Table 10-3.

Table 10-3. Six patents for optical lens systems

Invention	Date	Inventor	Place	Patent
Telescope eyepiece	1940	Albert Konig	Carl Zeiss GmbH	U. S. 2,206,195
Telescope eyepiece system	1958	Robert B. Tackaberry and Robert M. Muller	American Optical Company	U. S. 2,829,560
Eyepiece for optical instruments	1953	Maximillian Ludewig	Ernst Leitz GmbH	U. S. 2,637,245
Wide angle eyepiece	1968	Wright H. Scidmore	United States Army	U. S. 3,390,935
Wide angle eyepiece	1985	Albert Nagler	No affiliation listed	U. S. 4,525,035
Telescope eyepiece	2000	Noboru Koizumi and Naomi Watanabe	Fuji Photo Optical Co., Ltd.	U. S. 6,069,750

The automatic synthesis is done “from scratch”—that is, without starting from a preexisting good design and without prespecifying the number of lenses, the physical layout of the lenses, the numerical parameters of the lenses, or the nonnumerical parameters of the lenses. One of the six genetically evolved lens systems infringed a previously issued patent; three contained many of the essential features of the patents, without infringing; and the others were noninfringing novel designs that duplicated (or improved upon) the performance specifications contained in the patents. One of the six patents was issued in the 21st century. The six designs were created in a substantially similar and routine way, suggesting that the approach used may have widespread utility.

4.2 Automatic Synthesis of Two New Controller Designs That Have Been Granted Patents

U.S. patents have recently been granted for two new inventions that were automatically created by means of genetic programming (Keane, Koza, Streeter, 2005). The first invention covers improved tuning rules for PID (proportional, integrative, and derivative) controllers. The second is for a group of general-purpose non-PID controllers, which we call Keane-Koza-Streeter (KKS) controllers. We believe these are the first patents granted for inventions that were automatically created by genetic programming. Both the genetically evolved PID tuning rules and non-PID KKS controllers outperform controllers tuned using the widely used Ziegler-Nichols tuning rules (Ziegler and Nichols, 1942) and the recently developed Åström-Hägglund tuning rules (Åström and Hägglund, 1995).

The general-purpose non-PID KKS controller (Figure 10-7) is an example of what we call a *parameterized topology*. A parameterized topology is a general (parameterized) solution to a problem in the form of a graphical structure whose nodes or edges represent components and where the parameter values of the structure's components are specified by mathematical expressions containing free variables (Koza, et al., 2003). In a parameterized topology, the genetically evolved graphical structure represents a complex structure of some kind (e.g., controller, electrical circuit). In the automated process, genetic programming determines the graph's size (its number of nodes) as well as the graph's connectivity (specifying which nodes are connected). Genetic programming also assigns, in the automated process, component types to the graph's nodes or edges. In the automated process, genetic programming also creates mathematical expressions that establish the parameter values of the components (e.g., the capacitance of a capacitor in a circuit, the amplification factor of a gain block in a controller). Some of these genetically created mathematical expressions contain free variables. The free variables confer generality on the genetically evolved solution by enabling a single genetically evolved graphical structure to represent a general (parameterized) solution to an entire category of problems. Genetic programming can do all of the above in an automated way in a single run.

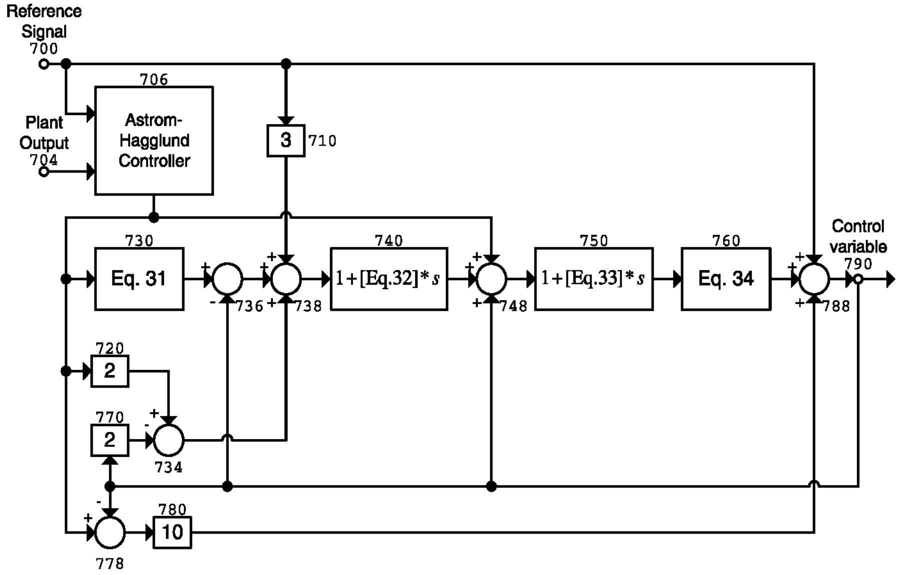


Figure 10-7. Genetically evolved general-purpose non-PID KKS controller

The general-purpose non-PID controller (Figure 10-7) has a number of blocks that are parameterized by mathematical expressions containing the problem's four free variables, namely the plant's time constant, *Tr*, ultimate period, T_u, ultimate gain, K_u, and dead time, *L*. For example, gain block 730 of Figure 10-7 has a gain that is parameterized by the following non-genetically-evolved constant mathematical expression (equation 31):

$$\left|\log\left|T_r - T_u + \log\left|\frac{\log(L^L)}{T_u + 1}\right|\right|\right|. \tag{31}$$

In addition, gain block 760, lead block 740, and lead block 750 contain genetic genetically-evolved mathematical expressions.

Additional information on this genetically evolved controller and the use, in general, of genetic programming to automatically synthesize the topology and tuning of controllers can be found in *Genetic Programming IV: Routine Human-Competitive Machine Intelligence* (Koza, et al., 2003).

5. CONCLUSIONS

This chapter describes the automatic synthesis of six 21st century patented analog electrical circuits by means of genetic programming. This chapter also briefly summarizes some examples of the automatic synthesis of other types of complex structures by means of genetic programming, including six optical lens systems that were previously patented, and two new controller designs that have been granted patents.

References

Al-Sakran, Sameer H., John R. Koza, and Lee W. Jones. 2005. Automated re-invention of a previously patented optical lens system using genetic programming. In Keijzer, M., et al. (Eds.). *Genetic Programming: 8th European Conference, EuroGP 2005, Lausanne, Switzerland, March 30-April 1, 2005, Proceedings, Lecture Notes in Computer Science,* 3447, 25-37. Heidelberg: Springer-Verlag.

Åström, Karl J. and Tore Hägglund. 1995. *PID Controllers: Theory, Design, and Tuning.* Second Edition. Research Triangle Park, NC: Instrument Society of America.

Aytur, Turgut Sefket. 2000. *Integrated Circuit with Variable Capacitor.* U.S. patent 6,013,958. Filed July 23, 1998. Issued January 11, 2000.

Banzhaf, Wolfgang, et al. 1998. *Genetic Programming – An Introduction.* San Francisco, CA: Morgan Kaufmann and Heidelberg: dpunkt.

Cho, Sung-Bae, Hoai Xuan Nguyen, and Yin Shan (Eds.). 2003. *Proceedings of the First Asian-Pacific Workshop on Genetic Programming.* www.aspgp.org.

Cipriani, Stefano and Anthony A. Takeshian. 2000. *Compact cubic function generator.* U. S. patent 6,160,427. Filed September 4, 1998. Issued December 12, 2000.

Comisky, William, Jessen Yu, and John Koza. 2000. Automatic synthesis of a wire antenna using genetic programming. *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference, Las Vegas, Nevada.* 179–186.

Daun-Lindberg, Timothy Charles, and Michael Lee Miller. 2000. *Low Voltage High-Current Electronic Load.* U. S. patent 6,211,726. Filed June 28, 1999. Issued April 3, 2001.

Deb, Kalyanmoy, et al. (Eds.). 2004. *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 2004. Proceedings, Part I, Lecture Notes in Computer Science 3102.* Berlin: Springer.

Grimbleby, J. B. 1995. Automatic analogue network synthesis using genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA).* 53-58. London: Institution of Electrical Engineers.

Gruau, Frederic. 1992. *Cellular Encoding of Genetic Neural Networks.* Technical report 92-21. Laboratoire de l'Informatique du Parallélisme. Ecole Normale Supérieure de Lyon. May 1992.

Higuchi, Tetsuya, et al. 1993a. Evolving hardware with genetic learning: A first step towards building a Darwin machine. In Meyer, Jean-Arcady, Herbert L. Roitblat and Stewart W. Wilson (Eds.). *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior,* 417-424. Cambridge, MA: The MIT Press.

Higuchi, Tetsuya, et al. 1993b. *Evolvable Hardware–Genetic-Based Generation of Electric Circuitry at Gate and Hardware Description Language (HDL) Levels.* Electrotechnical Laboratory technical report 93-4. Tsukuba, Japan: Electrotechnical Laboratory.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Ann Arbor, MI: University of Michigan Press. Second edition. Cambridge, MA: The MIT Press, 1992.

Ikeuchi, Akira and Naoshi Tokuda. 2000. *Voltage-Current Conversion Circuit*. U. S. patent 6,166,529. Filed February 24, 2000 in U. S. Issued December 26, 2000 in U. S. Filed March 10, 1999 in Japan.

Keane, Martin A., John R. Koza, and Matthew J. Streeter, 2005. *Apparatus for Improved General-Purpose PID and Non-PID Controllers*. U. S. Patent 6,847,851. Filed July 12, 2002. Issued January 25, 2005.

Keijzer, Maarten, et al. (Eds.). *Genetic Programming: 8th European Conference, EuroGP 2005, Lausanne, Switzerland, March 30-April 1, 2005, Proceedings, Lecture Notes in Computer Science 3447*. Heidelberg: Springer-Verlag.

Kitano, Hiroaki. 1990. "Designing neural networks using genetic algorithms with graph generation system." *Complex Systems,* 4 (1990), 461-476.

Koza, John R. 1990. *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems.* Stanford University Computer Science Dept. technical report STAN-CS-90-1314. June 1990.

Koza, John R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: MIT Press.

Koza, John R. 1993. Discovery of rewrite rules in Lindenmayer systems and state transition rules in cellular automata via genetic programming. Symposium on Pattern Formation (SPF-93), Claremont, California. February 13, 1993.

Koza, John R. 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: MIT Press.

Koza, John R., Sameer H. Al-Sakran and Lee W. Jones. 2005. Automated re-invention of six patented optical lens systems using genetic programming. In Beyer, H.-G., et al. (Eds.). *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2005.* 1953–1960. New York, NY: ACM Press.

Koza, John R., et al. 1996a. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In Gero, John S. and Fay Sudweeks (Eds.). *Artificial Intelligence in Design '96.* 151-170. Dordrecht: Kluwer Academic Publishers.

Koza, John R., et al. 1996b. Reuse, parameterized reuse, and hierarchical reuse of substructures in evolving electrical circuits using genetic programming. In Higuchi, Tetsuya, Masaya Iwata, and Weixin Liu (Eds.). *Proceedings of International Conference on Evolvable Systems: From Biology to Hardware (ICES-96), Lecture Notes in Computer Science, Volume 1259.* 312-326. Berlin: Springer-Verlag.

Koza, John R., et al. 1999. *Genetic Programming III: Darwinian Invention and Problem Solving.* San Francisco, CA: Morgan Kaufmann.

Koza, John R., et al. 2003. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* Kluwer Academic Publishers.

Kruiskamp, Marinum Wilhelmus and Domine Leenaerts. 1995. DARWIN: CMOS opamp synthesis by means of a genetic algorithm. In *Proceedings of the 32nd Design Automation Conference,* 433-438. New York, NY: Association for Computing Machinery.

Langdon, William B. and Riccardo Poli. 2002. *Foundations of Genetic Programming.* Springer-Verlag.

Lee, Sang Gug. 2001. *Low Voltage Balun Circuit.* U. S. patent 6,265,908. Filed December 15, 1999. Issued July 24, 2001.

Lipson, Hod. 2004. How to draw a straight line using a GP: Benchmarking evolutionary design against 19th century kinematic synthesis. In Keijzer, Maarten (Ed.). *Genetic and Evolutionary Conference 2005 Late-Breaking Papers. CD ROM.* Seattle, WA: International Society for Genetic and Evolutionary Computation.

Lohn, Jason D., Greg S. Hornby, and Derek S. Linden. 2004. An evolved antenna for deployment on NASA's Space Technology 5 Mission. In O'Reilly, Una-May, et al. (Eds.).

Genetic Programming Theory and Practice II, Chapter 18. Boston: Kluwer Academic Publishers.

O'Reilly, Una-May, et al. (Eds.). 2004. *Genetic Programming Theory and Practice II*. Boston: Kluwer Academic Publishers, 121-142.

Quarles, Thomas, et al. 1994. *SPICE 3 Version 3F5 User's Manual*. Department of Electrical Engineering and Computer Science, University of California. Berkeley, CA. March 1994.

Smith, Warren J. 2000. *Modern Optical Engineering. 3rd edition*. New York: McGraw-Hill.

Spector, Lee. 2004. *Automatic Quantum Computer Programming: A Genetic Programming Approach*. Boston: Kluwer Academic Publishers.

Thompson, Adrian. 1996. Silicon evolution. In Koza, John R., et al. (Eds.). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28–31, 1996, Stanford University*, 444-452. Cambridge, MA: MIT Press.

Wilson, Stewart. W. 1987. The genetic algorithm and biological development. In Grefenstette, John J. (Ed.). *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, 247-251. Hillsdale, NJ: Lawrence Erlbaum Associates.

Ziegler, J. G. and N. B. Nichols. 1942. "Optimum settings for automatic controllers." *Transactions of ASME*, (64), 759–768.

Chapter 11

EVOLVABLE OPTICAL SYSTEMS

Hirokazu Nosato, Masahiro Murakawa, Yuji Kasai, and Tetsuya Higuchi
National Institute of Advanced Industrial Science and Technology, Advanced Semiconductor Research Center, Email: h.nosato@aist.go.jp

Abstract: This chapter describes evolvable optical systems and their applications developed at the National Institute of Advanced Industrial Science and Technology (AIST) in Japan. Five evolvable optical systems are described: (1) an evolvable femtosecond laser system, (2) an automatic wave-front correction system, (3) a multiobjective adjustment system, (4) an automatic optical fiber alignment system, and (5) an evolvable interferometer system. As the micron-meter resolution alignment of optical components usually takes a long time, to overcome this time problem, we propose five systems that can automatically align the positioning of optical components by genetic algorithms (GA) in very short times compared to conventional systems.

Key words: genetic algorithms, optical system, laser system, automatic alignment, fiber alignment.

1. INTRODUCTION

In recent years, the market for optical technology has been growing rapidly, particularly due to the advances in optical communication systems. In general, optical systems consist of many components, such as light source, mirrors, lens, prisms, semiconductor elements, and optical fibers. In order to obtain the optimal performance from an optical system, it is necessary to align these optical components to the optimal position with micron-meter precision. Moreover, as shifts in the positioning of each component impact on the alignment of the whole system, each component must be repeatedly adjusted until the optimal alignment is achieved. As the number of experienced technicians is extremely limited, these difficulties increase the manufacturing time and the costs of optical systems.

Although algorithms for automatic alignment have been devised, they are often not suitable for systems where the parameters to be set optimally are

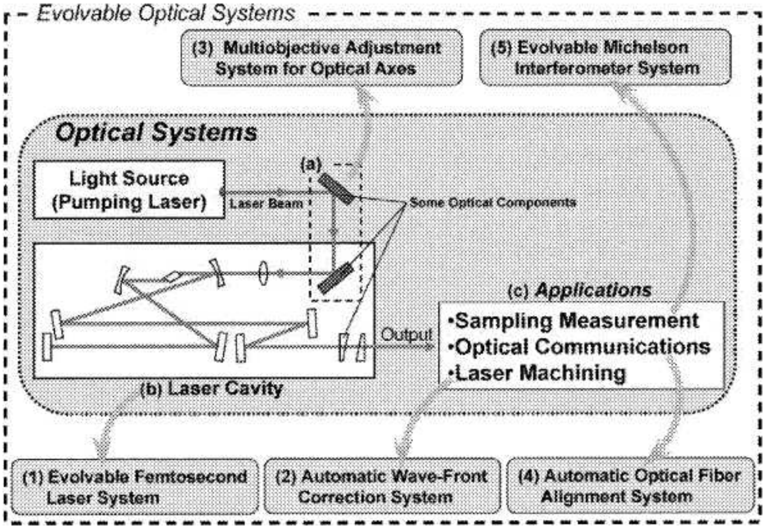

Evolvable Optical Systems
(3) Multiobjective Adjustment System for Optical Axes
(5) Evolvable Michelson Interferometer System
Optical Systems
Light Source (Pumping Laser)
Laser Beam
(a)
Some Optical Components
Output
(c) Applications
•Sampling Measurement
•Optical Communications
•Laser Machining
(b) Laser Cavity
(1) Evolvable Femtosecond Laser System
(2) Automatic Wave-Front Correction System
(4) Automatic Optical Fiber Alignment System

Figure 11-1. Overall view of optical systems and relations with our developed evolvable optical systems

numerous, where there are local optimum, and where the parameters are interdependent. In this chapter, we propose methods of automatic alignment using genetic algorithms (GA) (Goldberg, 1989) for optical systems. These methods have the following three advantages:

1. *Automatic adjustment for maintenance-free systems*
   The performance of an optical system is affected by the temperature and the stability of the environment in which it is used. However, variations in performance can be adjusted automatically by our algorithms online. This means that the optical systems are maintenance-free, making them easy for nonexperts to use.
2. *Cost reduction*
   Using less expensive mechanical parts in optical systems, unfortunately, often leads to longer adjustment times, because the precision of these components is usually inferior. However, as our algorithms provide a quick and flexible way of adjusting performance, it is possible to use less expensive mechanical parts in order to reduce the costs of the system.
3. *Compact implementation*
   With automatic adjustment methods, it is possible to reduce the spaces between the components in an optical system, which are necessary when adjustments are made by technicians. Thus, it is possible to downsize optical systems.

Our proposed methods can be applied to a variety of optical systems, such us laser beam alignment (Figure 11-1 (a)), laser cavity alignment (Figure 11-1 (b)), and laser applications (Figure 11-1 (c)). This chapter describes five evolvable optical systems using GA-based automatic alignment algorithms for optical systems:

(1) Evolvable femtosecond laser system (Murakawa, 2000; Nosato, 2003).

(2) Automatic wave-front correction system for femtosecond laser (Nosato, 2004).

(3) Multiobjective adjustment system for optical axes (Murata, 2005).

(4) Automatic optical fiber alignment system (Murakawa, 2003).

(5) Evolvable Michelson interferometer system (Nosato, 2002).

Figure 11-1 shows relations between optical systems and these evolvable optical systems.

2. EVOLVABLE FEMTOSECOND LASER SYSTEM

Laser systems, invented in the 1960's (Maiman, 1960), have been applied to various fields, such as semiconductor lithography, optical communications, sampling measurement methods, and laser machining. During the last decade, the development of femtosecond lasers, based on Kerr-lens mode-locking techniques, has progressed rapidly and many commercial products are now available. Ultra-short optical pulses in the femtosecond region ($\sim 10^{-15}$ sec.) are essential for the development of ultra-fast optical communication and measurement methods.

However, because the Kerr-lens mode-locking technique used in femtosecond lasers requires precise positioning of the focusing mirrors within the laser cavity, it is difficult to adjust femtosecond lasers, which makes them less attractive for industrial use. For example, even a 10 μm discrepancy in a focusing mirror will prevent the formation of an optical resonation with a 1.0 W pumping power. It, therefore, typically takes three or more days to manually adjust a femtosecond laser. Moreover, because nonlinear lens effects must be considered when high-peak power is involved in the laser cavity, the optimal adjustment is dependent on the pumping power level used so that the system must be readjusted if a different pumping power is used.

In order to overcome these problems, we propose an automatic adjustment method using GA for the femtosecond laser. This laser system, as shown in Figure 11-2, can adjust the positioning of the laser cavity components (e.g.

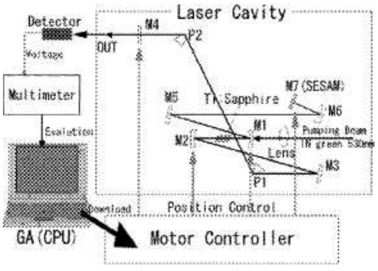


Figure 11-2. Evolvable femtosecond laser system

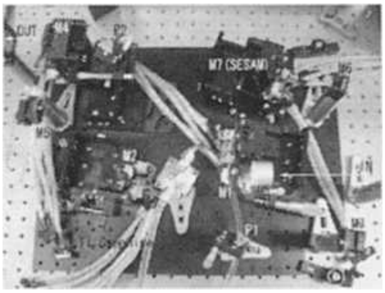


Figure 11-3. Developed evolvable femtosecond laser system

the mirrors and prisms) using GA. In our experiments with the developed laser system (Figure 11-3), automatic adjustment was realized in 15 minutes, which is less than one-hundredth of the manual adjustment time (three days and more).

3. AUTOMATIC WAVE-FRONT CORRECTION SYSTEM FOR FEMTOSECOND LASER

A characteristic of femtosecond lasers is that the high-peak power is inversely proportional to the short duration of the laser pulses, so they can generate high power levels, over one megawatt, during femtosecond pulses. Femtosecond lasers can be used for high-precision machining without abrasions due to heat. In the laser-machining field, femtosecond lasers are especially important for micromachining (Mendes, 1988), allowing for the fabrication of more complex double-figure microscopic mechanics than possible with traditional methods. Combining real micromachining with focused femtosecond laser beams, it is possible to fabricate components for nanotechnology.

However, because femtosecond laser beams fluctuate based on some aberrations (such as coma aberration, astigmatism, spherical aberration and wave-front aberration), it is difficult to focus the beams efficiently using focusing lenses or mirrors. It is, therefore, essential to reduce these aberrations. In order to achieve this, we propose an automatic correction method using GA and a deformable mirror (DM) (Grosso, 1977) to correct the wave-front of femtosecond lasers. This correction system, as shown in Figure 11-4, corrects the wave-front of the femtosecond laser beam with a DM, where the mirror membrane is controlled by the GA. As the experimental results demonstrate, this method can achieve automatic wave-front correction to increase the laser peak power by 21% with the developed correction system shown in Figure 11-5.

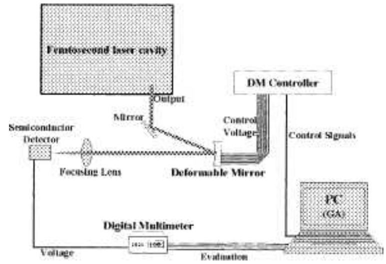


Figure 11-4. Evolvable DM femtosecond laser system

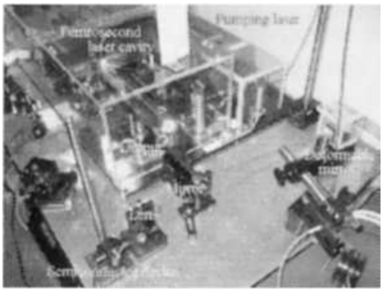


Figure 11-5. Developed evolvable DM femtosecond laser system

4. MULTIOBJECTIVE ADJUSTMENT SYSTEM FOR OPTICAL AXES

Laser systems are now essential in various industrial fields. A laser system consists of a pumping laser as the light source, and a laser cavity, where the laser beam from the pumping laser is amplified (see Figure 11-1). The laser system performance deteriorates when the optical axes deviate from their precise positioning and parallelism is lost, due to long-term use and disturbances such as vibrations. Therefore, adjustment of the optical axes of the pumping laser is crucial. However, it is very difficult to adjust the optical axes, because the adjustment requires high-precision positioning and angle settings with μm resolutions, and because the adjustment must simultaneously satisfy multiple objectives that have trade-off relations. Consequently, a major problem for conventional methods is to achieve multiobjectives with approaches that are fundamentally single-objective in nature.

In order to overcome this problem, we propose an automatic multi-objective adjustment method using GA. Specifically, there are three advantages with the proposed method: (1) It is not necessary to set up weighing coefficients, which are used by conventional methods, because the proposed method does not require weighing coefficients. (2) Readjustments are unnecessary because a solution is selected from the Pareto optimum solutions which are calculated at initial adjustment when the user preferences are changed. (3) Adjustment accuracy is improved compared to conventional methods, because the GA, which is a stochastic search method capable of simultaneously evaluating multiple solution candidates, is hardly trapped in local solutions in achieving an optimum adjustment. Moreover, we have been able to realize an easy-to-use automatic adjustment system, as shown in Figure 11-6, for optical axes. In experiments using developed adjustment system shown in Figure 11-7, simultaneous alignment for the positioning and the angle (parallelism) of optical axes, which is difficult with conventional methods, was realized within three hours.

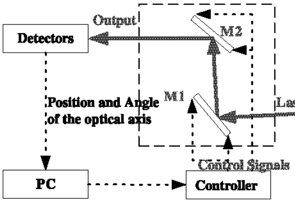


Figure 11-6. Multiobjective adjustment system

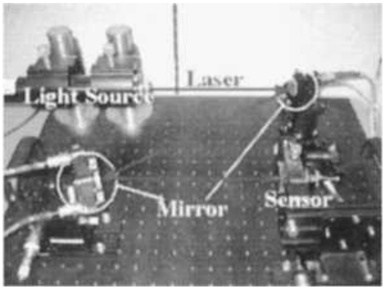


Figure 11-7. Multiobjective adjustment system

5. AUTOMATIC OPTICAL FIBER ALIGNMENT SYSTEM

With the growth in optical fiber communications, fiber alignment has become the focus of much industrial attention (Hayes, 2001). This is a key production process because its efficiency greatly influences the overall production rates for the optoelectric products used in optical fiber communications. Fiber alignment is necessary when two optical fibers are connected, when an optical fiber is connected to a photo diode (PD) or a light emission diode (LED), and when an optical fiber array is connected to an optical wave guide.

Metallic wire connection is relatively easy because an electric current will flow as long as the two wires are in contact. The connection between two optical fibers, however, requires much greater precision, in the order of submicronmeters. Therefore, experienced technicians are needed for fiber alignment, but as such technicians are in limited supply, this causes a bottleneck in the mass production of optoelectric components for optical fiber communication. To overcome this, various algorithms for automatic alignment have been devised. However, existing automatic-alignment algorithms are only capable of aligning fibers according to three degrees of freedom (DOF). Thus, we have devised a new algorithm based on GAs that is capable of aligning fibers according to five or more DOF. Figure 11-8 shows automatic fiber alignment system realized our algorithm. The advantages of the algorithm are summarized below:

1. *Improvement in transmission efficiency*

   Unless two optical fibers are connected in an optimal way, there will be a reduction in transmission efficiency. However, as alignment is possible according to a greater DOF, there are no reductions in transmission efficiency.

2. *Improvement in reliability*

   Automatic alignment reduces the number of tasks to be conducted by technicians, which leads to the improved reliability of products.

3. *Improvement in production efficiency*
   The fiber alignment process is often a critical part in the entire production process, and so by reducing fiber-alignment times, production efficiency can be improved considerably.
4. *Cost reductions*
   Less expensive optoelectric components are often inferior in terms of precision. However, variations from the desired specifications for each component can be compensated by using our alignment algorithm for greater DOF connections. This makes it possible to reduce the overall costs of a system.

Conventional alignment systems cannot achieve connections with 5 DOFs because such precise fiber alignment is impossible given practical time constraints. However, our algorithm successfully carried out alignments for 4 different initial conditions in a few minutes with developed fiber alignment system shown in Figure 11-9.

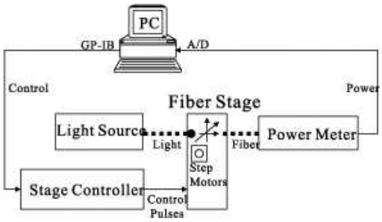


Figure 11-8. Automatic fiber alignment system

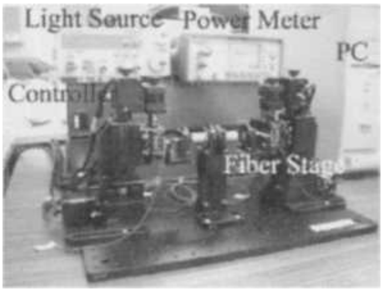


Figure 11-9. Fiber alignment system for development

6. EVOLVABLE MICHELSON INTERFEROMETER SYSTEM

In recent years, advances in spectroscopic technology are making important contributions to the ultra-trace analysis of environmental pollutions such as dioxin and other air pollutants. The Michelson interferometer (Michelson, 1981) is widely used as an instrument for spectroscopic analysis, as well as for environmental pollution and ultra-trace analyses, as a component of Fourier-Transform Infrared spectroscopy (FTIR) (Griffiths, 1975). Although it is commonly used within chemical laboratories for conventional analyses, recently there is an increasing demand for portable instruments for on-site analysis.

However, there are two major problems in using Michelson interferometers for on-site analysis. The first problem is that to achieve high-level analytic performance it is necessary to optimally align the component mirrors and prisms with micron-meter precision. Conventional Michelson interferometers are usually set manually, and often include automatic dynamic alignment methods to compensate for small deviations in component positioning. However, this method is not capable of adjusting for the kind of large misalignments that occur during transportation to outdoor locations. This has made the on-site use of these interferometers, which are extremely sensitive to variations, impossible.

The second problem is that conventional interferometers are too bulky for on-site use. When aligning the components of an interferometer, it is necessary to adjust at least three alignment axes (such as the focus position, the rotation angle, and the tilt angle of a mirror). Because the alignment axes of the components are interdependent, however, achieving the optimal alignment for all components is extremely difficult. Using larger and heavier holders (which are less sensitive to discrepancies), is one way to avoid the problem of interdependency, although this creates a trade-off situation between the compactness desirable for on-site use and the ease of independent adjustment mechanisms.

In order to overcome these problems, we propose an automatic alignment method using GA for the Michelson interferometer (Figure 11-10). Specifically, our research shows that: (1) GA is more robust than a hill-climbing method for the automatic alignment of interferometers, (2) the proposed system can optimally adjust the mirrors quickly, and (3) alignment interdependency is not a problem for the compact mirror holders utilizing GA. In our experiments with the developed system (Figure 11-11), we have been able to realize an automatically-adjustable Michelson interferometer system, which is capable of adjusting for the large misalignments that might arise at setting (and previously had to be adjusted for by technicians), in a practical time of approximately 3 minutes.

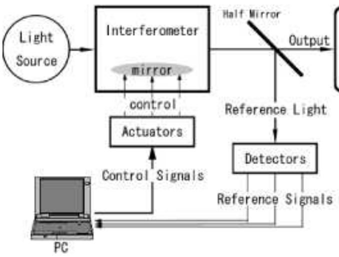


Figure 11-10. Evolvable Michelson interferometer system

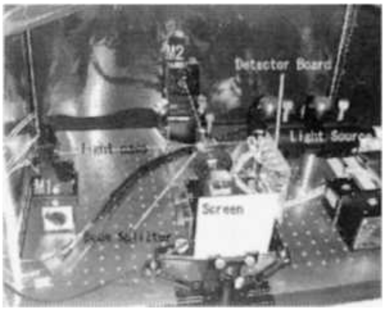


Figure 11-11. Developed evolvable Michelson interferometer system

7. SUMMARY

This chapter has described five evolvable optical systems. The adavantage common to all these systems is the inclusion of GA in order to automatically execute the micron-meter precision alignment of optical components, such as mirrors and prisms, very quickly and very efficiently. Although the fitness function for the GA used in the present six systems was light intensity, it is helpful to develop other fitness functions in order to make automatic alignment even more stable and faster. The present systems are simply early examples of how mechanical evolution can be utilized in a wide variety of industrial applications in the future.

References

Goldberg, D. E. 1989. *Genetic algorithms in search, optimization and machine learning*. Addison Wesley.

Griffiths, P. R. 1975. *Chemical Infrared Fourier Transform Spectroscopy.* John Wiley & Sons.

Grosso, R. P., and M. Yellin, 1977. "The membrane mirrors as an adaptive optical element," *J. Opt. Soc. Am.*, Vol. 67, No. 3, 399-406.

Hayes, J. 2001. *Fiber Optics Technician's Manual 2nd edition.* Delmar Publishers.

Maiman, T. H. 1960. "Stimulated Optical Radiation in Ruby," *Nature*, Vol. 187, 493-494.

Mendes, M., et al. 1998. Femtosecond UV Laser Micromachining of $Al_2O_3 - TiC$ Ceramics. In *Proc. of International Congress on Applications of Lasers & Electro-Optics (ICALEO) 1998*, 19-27.

Michelson, A. A. 1981. "The Relative Motion of the Earth and the Luminiferous Aether," *Amer. J. Sci.*, 22, 120-129.

Murakawa, M., et al. 2000. An evolvable laser system for generating femtosecond pulses. In *Proc. of the Second Genetic and Evolutionary Computation Conference (GECCO2000)*, 636-642, Morgan Kaufmann.

Murakawa, M., H. Nosato, and T. Higuchi, 2003. Automatic Optical Fiber Alignment System using Genetic Algorithms. In *Proc. of the 6th International Conference on Artificial Evolution*, INRIA France, 223-234.

Murata, N., et al. 2005. An Automatic Multi-objective Adjustment System for Optical Axes Using Genetic Algorithms. In *Proc. of the 5th International Conference on Intelligent Systems Design and Applications*, ISDA 2005, 546-551.

Nosato, H., et al. 2002. Automatic Adjustment of a Michelson Interferometer Using Ganetic Algorithms. In *Proc. of 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02)*, Vol. 2, 610-616.

Nosato, H., et al. 2003. Automatic adjustments of a femtosecond-pulses laser using genetic algorithms. In *Proc. of 2003 Congress on Evolutionary Computation (CEC 2003)*, 2096-2101.

Nosato, H., et al. 2004. Automatic Wave-Front Correction of a Femtosecond Laser Using Genetic Algorithm. In *Proc. of the 2004 IEEE International Conference on Systems, Man & Cybernetics (SMC2004)*, 3675-3679.

Chapter 12

HARDWARE PLATFORMS FOR ELECTROSTATIC TUNING OF MEMS GYROSCOPE USING NATURE-INSPIRED COMPUTATION

Didier Keymeulen[1], Michael I. Ferguson[1], Luke Breuer[1], Wolfgang Fink[1], Boris Oks[1], Chris Peay[1], Richard Terrile[1], Yen-Cheng[2], Dennis Kim[2], Eric MacDonald[3], and David Foor[4]

[1]Jet Propulsion Laboratory, MS 303-300, 4800 Oak Grove Dr., Pasadena, CA 91109, USA didier.keymeulen@jpl.nasa.gov; [2]Mechanical and Aerospace Engineering Department, University of California, Los Angeles, CA 90095-1597; [3]University of Texas at El Paso, 500 West University Dr., El Paso, TX, 79968-0523; [4]Texas A&M University-Kingsville, 700 University Blvd, Kingsville, TX, 78363

Abstract: We propose a tuning method for Micro-Electro-Mechanical Systems (MEMS) gyroscopes based on evolutionary computation to increase the accuracy of MEMS gyroscopes through electrostatic tuning. The tuning method was tested for the second generation JPL/Boeing Post-resonator MEMS gyroscope using the measurement of the frequency response of the MEMS device in open-loop operation. We also report on the development and preliminary results of a hardware platform for integrated tuning based on "switched drive-angle" of MEMS gyroscopes whereby the same gyro is operated with its drive direction first at 0° and then at 90°. The control of this device is implemented through a digital design on a Field Programmable Gate Array (FPGA). The hardware platform easily transitions to an embedded solution that allows for the miniaturization of the system to a single chip.

Key words: MEMS, tuning, genetic algorithm, simulated annealing, FPGA.

1. INTRODUCTION

Future NASA missions would benefit tremendously from an inexpensive, navigation grade, miniaturized inertial measurement unit (IMU), which surpasses the current state-of-the art in performance, compactness (both size and mass), and power efficiency. Towards this end, under current development at JPL's MEMS Technology Group are several different designs for environment tolerant (Ferguson, 2005), high performance, low mass and volume, low power MEMS gyroscopes. The accuracy with which the rate of rotation of micro-gyros can be determined crucially depends on the properties of the resonant structure. It is both difficult and expensive to attempt to achieve these desired characteristics in the fabrication process, especially in the case of small MEMS structures, and thus one has limited overall sensor performance due to unavoidable fabrication inaccuracies.

The accuracy with which a given vibratory gyroscope can determine the true inertial rate is crucially dependent on the properties of the two degree of freedom resonator. The problem with tracking orientation using only gyros is drift, such as gyro bias which creates a steadily growing angular error, gyro white noise, and gyro bias instability (Stanney, 2002). Today's commercial-grade gyros (devices used in automobiles, camcorders) can only be used for a minute before the drift becomes distracting and the user needs to reset the orientation tracker. Tactile-grade gyros used in short-range missile guidance are good enough for head-tracking for more than 20 minutes. Navigation-grade gyros are required for space applications. Unfortunately, the price, weight and power ratio between navigation, tactile and commercial-grade gyros follows the performance ratio. MEMS gyroscopes will gradually be closing in on the performance using techniques presented in the chapter.

One way to reduce the rate drift is to increase the sensitivity or scale factor relating volts out to radians per second of inertial input (Hayworth, 2003). The scale factor is maximized when the resonant frequencies of the two modes of freedom of the MEMS gyroscope are identical. Symmetry of construction is necessary to attain this degeneracy. However, despite a symmetric design, perfect degeneracy is never attained in practice. Many methods have been developed for tuning MEMS post-resonator gyroscopes. For example, (Leland, 2003) and (Painer, 2003) use adaptive and closed-loop methods, while (Chen, unpublished) changes the frame of the pick-off signal. Our approach of gyro tuning is achieved through an electrostatic biasing approach (Hayworth, 2004). This approach consists of applying bias voltages to built-in tuning pads to electrostatically soften the mechanical springs. Because of the time-consuming nature of the tuning process when performed manually, in practice any set of bias voltages that produce degeneracy is viewed as acceptable at the present time. Thus, a need exists for reducing the time necessary for performing the tuning operation, and for finding the optimally tuned configuration which employs the minimal maximum tuning voltage.

This chapter describes the application of evolutionary computation to this optimization problem. Our open-loop and closed-loop methods used the following fitness function for each set of bias voltages applied to the built-in tuning pads: the frequency split between the two modes of resonance of the MEMS gyroscope. Our open-loop evaluation proceeds in two steps. First, it measures the open-loop frequency response using a dynamic signal analyzer. Second, it evaluates the frequency of resonance of both modes by fitting Lorentzian curves to the experimental data. The process of setting the bias voltages and the evaluation of the frequency split is completely computer automated. The computer controls a signal analyzer and programmable power supplies through General Purpose Interface Bus (GPIB). Our method has demonstrated that we can obtain a frequency split of 52mHz fully automatically in one hour, compared with 200mHz obtained manually by humans in several hours.

The closed-loop method, also called "switched drive-angle" method, is based on controlling the gyro in a closed-loop along one axis and measuring the resonance frequencies along this axis at a given set of bias voltages, then swapping and driving the other axis, thereby extracting the resonant frequency of both axes. An evolutionary algorithm is then applied iteratively to modify the bias voltages until the resonant frequency of each axis is equal. A major advantage of this closed-loop approach is that the resonant frequencies can be extracted quickly (~1 second) as compared to the open-loop control system, which takes two orders of magnitude longer. The design of the closed-loop control approach is realized on an FPGA with augmented portability for future designs and implementations.

This chapter is organized such that Section 2 describes the mechanics of the MEMS micro-gyro, Section 3 describes the evolutionary computation applied to open-loop measurements of the resonance frequencies, Section 4 describes the closed-loop hardware platform and the results of our preliminary experiments, and Section 5 describes future directions and summarizes the project results.

2. MECHANISM OF THE JPL MEMS MICRO-GYROSCOPE

The mechanical design of the JPL MEMS micro-gyro can be seen in Figure 12-1. The JPL/Boeing MEMS post-resonator gyroscope (PRG), is a MEMS analogue to the classical Foucault pendulum. A pyrex post, anodically bonded to a silicon plate, is driven into a rocking mode along an axis (labeled as X in Figure 12-1) by sinusoidal actuation via electrodes beneath the plate. In a rotating reference frame, the post is coupled to the Coriolis force, which exerts a tangential “force” on the post. Another set of electrodes beneath the device senses this component of motion along an axis (labeled as Y in the figure) perpendicular to the driven motion. The voltage that is required to null out this motion is directly proportional to the rate of rotation to which the device is subjected, and the voltage scale is reduced proportionally to the frequency split between the two modes of resonance. A change in capacitance occurs as the top plate vibrates due to the oscillating gap variation between this plate and the electrodes underneath. This change in capacitance generates a time-varying sinusoidal charge that can be converted to a voltage using the relationship V = Q/C. The post can be driven around the drive axis by applying a time-varying voltage signal to the drive petal electrodes labeled D1– (minus sign), D1+, D1in–, and D1in+ in Figure 12-1. Because there is symmetry in the device, either of the two axes can be designated as the drive axis. Each axis has a capacitive petal for sensing oscillations as well: driving axis: labeled S1+ and S1– in Figure 12-1, sensing axis: labeled S2+ and S2– in Figure 12-1. The micro-gyro has additional plates that allow for electrostatic softening of the silicon springs, labeled B1, BT1, B2, and BT2 in Figure 12-1. Static bias voltages can be used to modify the amount of softening for each oscillation mode. In an ideal, symmetric device, the resonant frequencies of both modes are equal; however, unavoidable manufacturing imperfections in the machining of the device can cause asymmetries in the silicon structure of the device, resulting in a frequency split between the resonant frequencies of these two modes. The frequency split reduces the voltage scale used to measure the rate of rotation to which the device is subjected, and thus the sensitivity for detection of rotation is decreased. By adjusting the static bias voltages on the capacitor plates, frequencies of resonance for both modes are modified to match each other; this is referred to as the tuning of the device using an electrostatic biasing approach (Hayworth, 2004).

In order to extract the resonant frequencies of the vibration modes, there are two general methods: 1) open-loop and 2) closed-loop control (Hayworth, 2003). In an open-loop system, we are measuring the frequency response along the drive axis over a 50Hz band and extracting from the measurement the frequency split. A faster method is a closed-loop control, whereby the gyro is given an impulse disturbance and is allowed to oscillate freely between the two resonance frequencies, using a hardware platform to control the switch of the drive-angles.

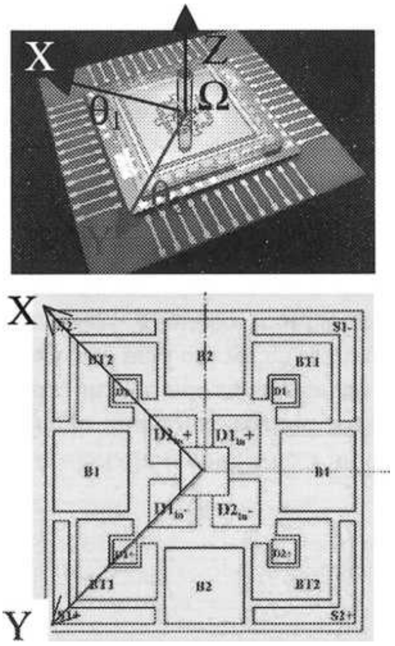


Figure 12-1. A magnified picture of the JPL MEMS micro-gyroscope with sense axis Y (S2–, S2+ electrodes used to sense, D2–, D2+, D2in– and D2in+ used to drive along the sense axis), drive axis X (D1–, D1+, D1in–, and D1in+ used to drive, S1–, S1+ electrodes used to sense along the drive axis) and the electrodes used for biasing (B1, B2, BT1, BT2) (picture courtesy of C. Peay, JPL)

3. EVOLUTIONARY COMPUTATION USING OPEN-LOOP MEASUREMENT

3.1 Instrumentation Platform for Open-loop Frequency Response

The open-loop measurement consists of exciting the drive axis with a sine wave at a given frequency and measuring the resulting amplitude. This is done repeatedly throughout the frequency spectrum (frequency range from 3,300Hz to 3,350Hz; 50Hz span; 800 points). Because of cross-coupling between the different axes, two peaks in the amplitude response will appear at two different frequencies, showing the resonant frequencies of both axes (Figure 12-4). This takes approximately 1.4 minutes to complete using our instrumentation platform (Figure 12-2) and must be repeated at least three times to average out noise.

The platform includes one GPIB programmable power supply for DC voltage, a GPIB signal analyzer to extract frequency responses (from 3.3kHz to 3.35kHz) of the gyro in open-loop, and a computer (PC) to control the instruments and to execute the evolutionary optimization algorithms. The power supply DC voltage controls the electrostatic bias voltages (connected to the plates B1, BT1, B2, and BT2 in Figure 12-1) that are used to modify the amount of damping to each oscillation mode. The GPIB signal analyzer generates a sine wave with a variable frequency (from 3300 Hz to 3350 Hz with a stepsize of 62.5 mHz – 800 points, 50Hz span) on the drive electrode (D1–, D1+, D1in–, and D1in+ in Figure 12-1) and measures the response signal on the sense electrode (S1–, S1+ in Figure 12-1) along the drive axis X.

A PC runs the instrument control tool, the measurement tool, and the evolutionary computation tool. The instrument control software sets up the static bias voltages using the GPIB power supply DC voltage and measures the frequency response along the X axis using the GPIB signal analyzer as shown in Figure 12-2. The software calculates the frequency split using peak fitting algorithms. Finally, the evolutionary computation software determines the new DC bias voltages from the frequency split. This procedure is repeated until a satisfactory (user-defined) frequency split is obtained.

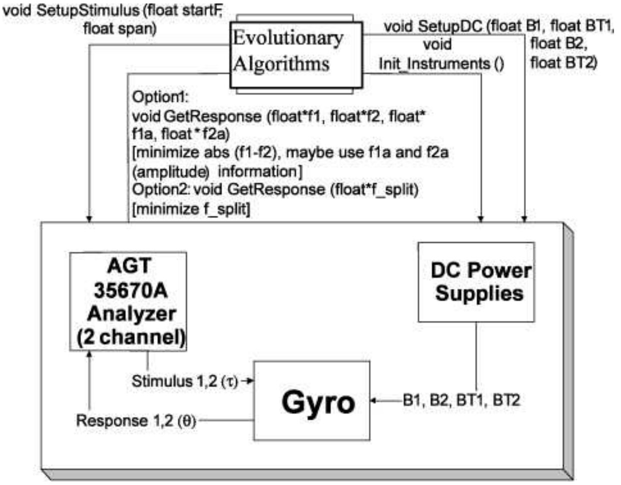


Figure 12-2. Software interface between the modified simulated annealing/modified genetic algorithm (dynamic hill-climbing) and the instrumentation platform using a GPIB programmable power supply DC voltage and a signal analyzer. The modified simulated annealing and the modified genetic algorithm are running on a PC, which controls the bias voltages and receives the frequencies of both resonance modes

3.2 Results of Evolutionary Computation

The MEMS post-resonator micro-gyroscope is subject to an electrostatic fine-tuning procedure performed by hand, which is necessary due to unavoidable manufacturing inaccuracies. In order to fine-tune the gyro, 4 bias voltages applied to 8 capacitor plates have to be determined within a range of –60V to +15V. The manual tuning took several hours and obtained a frequency split of 200 mHz.

In order to fully automate the time-taking manual fine-tuning process, we have established a hardware/software interface to the existing manual gyro-tuning hardware setup using commercial-off-the-shelf (COTS) components described in Section 3.1.

We developed and implemented two stochastic optimization techniques for efficiently determining the optimal tuning voltages and incorporated them in the hardware/software interface: a modified simulated annealing related algorithm (Metropolis, 1953; Kirkpatrick, 1983) and a modified genetic algorithm with limited evaluation (dynamic hill-climbing) (Holland, 1975; Yuret, 1993). These optimization techniques have also been used for other space applications (Terrile, 2005).

3.2.1 Simulated Annealing Approach

We were able to successfully fine-tune both the MEMS post-resonator gyroscope and MEMS disk-resonating gyroscope (a different gyro design not discussed here) within one hour for the first time fully automatically. After only 49 iterations with the modified simulated annealing related optimization algorithm, we obtained a frequency split of 125mHz within a 1V-discretization of the search space, starting with an initial split of 2.625Hz, using a 50Hz span and 800 points on the signal analyzer for the MEMS post-resonator gyroscope (Figure 12-3A). For the MEMS disk-resonating gyroscope we obtained a frequency split of 250mHz/500mHz within a 0.1V–/0.01V-discretization of the search space, starting with an initial split of 16.125Hz/16.25Hz, after 249/12 iterations using a 200Hz span and 800 points on the signal analyzer (Figure 12-3B). All three results are better than what can be accomplished manually but worse than the results obtained by dynamic hill-climbing (modified genetic algorithm). The reason for this is that instead of the peak fitting algorithm employed in the modified genetic algorithm approach, a simplified, direct peak-finding procedure was used in the simulated annealing approach.

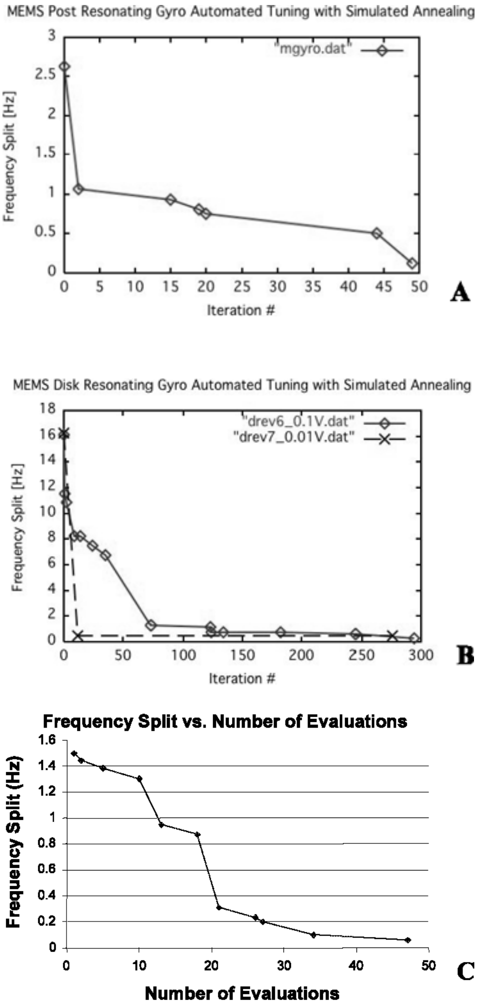


Figure 12-3. Frequency split as a function of number of evaluations: simulated annealing iterations (A) for the MEMS post-resonator gyroscope; (B) for the MEMS disk-resonating gyroscope. (C) Dynamic hill-climbing algorithm (modified genetic algorithm)

3.2.2 Genetic Related Algorithm Approach

We were also able to fine-tune the MEMS post-resonator gyroscope within one hour fully automatically using a modified genetic algorithm: dynamic hill-climbing. Figure 12-3C shows the progress of the optimization algorithm aimed at minimizing the frequency split. Each evaluation is a proposed set of bias voltages. Our optimization method only needed 47 evaluations (51 min.) to arrive at a set of bias voltages that produced a frequency split of less than 100mHz.

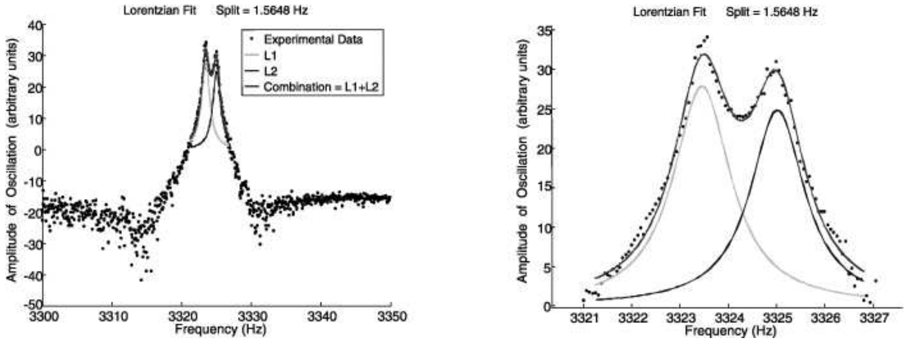


Figure 12-4. Frequency response (top: 50Hz band, bottom: 6Hz band) before tuning using the modified genetic algorithm. The frequency split is 1564.8mHz. The Y axis is measured in dB. The initial values of the four bias voltages are: B1 = 14.00V, BT1 = 14.00V, B2 = 14.00V, and BT2 = 14.00V. The right picture shows a zoomed-in display of the frequency split over a 6Hz band

Figures 12-4 and 12-5 show the frequency response for the unbiased micro-gyro respectively before and after tuning using the dynamic hill-climbing and the peak-fitting algorithm. After optimization of the bias voltages (Figure 12-5), the frequency split has been minimized to less than 100mHz and the two peaks are indistinguishable on an HP spectrum analyzer at 62.5mHz/division (50Hz span, 800 points) setting, which was used during the optimization process.

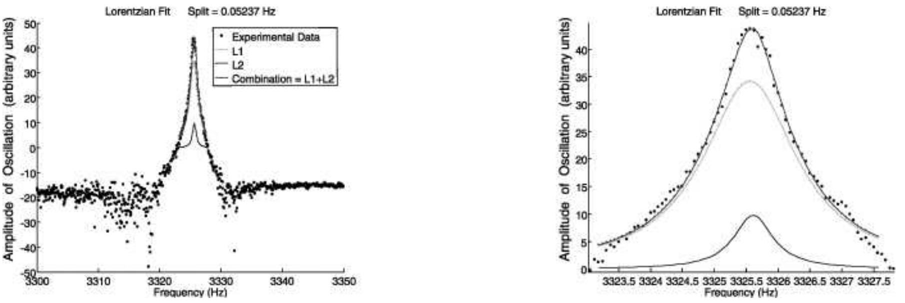


Figure 12-5. Frequency response (top: 50Hz band, bottom: 5Hz band) after tuning using the modified genetic algorithm. The Y axis is measured in dB. The tuning frequency split is 52mHz. The optimized values of the four bias voltages are: B1 = 4.00V, BT1 = 4.00V, B2 = 14.00V, and BT2 = –16.00V. The right picture shows a zoomed-in display of the frequency split over a 4Hz band

The frequency split of 52mHz was verified using a higher resolution mode of the signal analyzer.

4. HARDWARE PLATFORM USING “SWITCHED DRIVE-ANGLE” METHOD

The principle of “switched drive-angle” electrostatic biasing is based on measuring the resonance frequencies of the drive axis at a given set of bias voltages then swapping and driving the other axis, thereby extracting the resonant frequencies of both axes. An algorithm is then applied iteratively to modify the bias voltages until the resonant frequency of each axis is equal. A major advantage of this closed-loop approach is that the resonant frequencies can be extracted quickly (~1 second) as compared to the open-loop control system, which takes two orders of magnitude longer. The design of the electrostatic biasing approach is realized on an FPGA with augmented portability for future designs and implementations.

4.1 Control of the MEMS Micro-gyro

The “switched drive-angle” approach requires a closed-loop control whereby the gyro is given an impulse disturbance and is allowed to oscillate freely. This so-called “pinging” of the vibration mode allows the gyroscope to immediately settle to its natural frequency. The corresponding frequency, F1, is measured from the sensing plate under the drive axis X. Because the device is relatively symmetric, the drive and sense axes are swapped and the other mode is pinged to get F2. The difference in the frequencies, i.e., frequency split, is determined very quickly using this technique (about 1.5 seconds), roughly 50 times faster than from the open-loop control method. This ability to quickly swap the drive axis with the sense axis is a feature of our FPGA Gyro Digital System (GDS).

The circuitry of the closed-loop control system includes a drive loop and a sense rebalance loop (Chen, unpublished). The drive loop takes the input from the “drive sense” petal (S1–, S1+ electrodes along the drive axis), and outputs the forcing signal to the “drive-drive” petal electrodes (D1–, D1+, D1in- and D1in+ electrodes along the drive axis). The sense rebalance loop receives input from the “sense-sense” petal (S2–, S2+ electrodes along the sense axis), and forces or rebalances the oscillations back along the drive axis with a forcing signal to the “sense drive” (D2–, D2+, D2in– and D2in+ electrodes). The magnitude of this forcing function in the rebalance loop is related to the angular rate of rotation. The closed-loop control also has several scaling coefficients, denoted as Ki, which allow for a mixing of the sensed signals from both axes and a swapping of the drive- and sense-axis, thus permitting the tuning algorithm to measure the resonance frequency along the X- or Y-axis, or, indeed, any axis between X and Y (Hayworth, 2003).

The drive loop implements an Automatic Gain Control (AGC) loop combined with finite impulse response (FIR) filters. Because the amplitude of the freely oscillating drive axis will naturally decay, the AGC is implemented in a way to lightly drive or damp (depending on the circumstance), the drive axis so that the amplitude of the driven signal is constant and the gyroscope is maintained in an oscillation mode at the natural frequency. The optimal parameters of the FIR filters and the AGC loop to maintain the oscillation of the gyroscope have been determined by the UCLA team using a DSP measurement system and a UCLA MatLab modeling tool (M'Closkey, 2005).

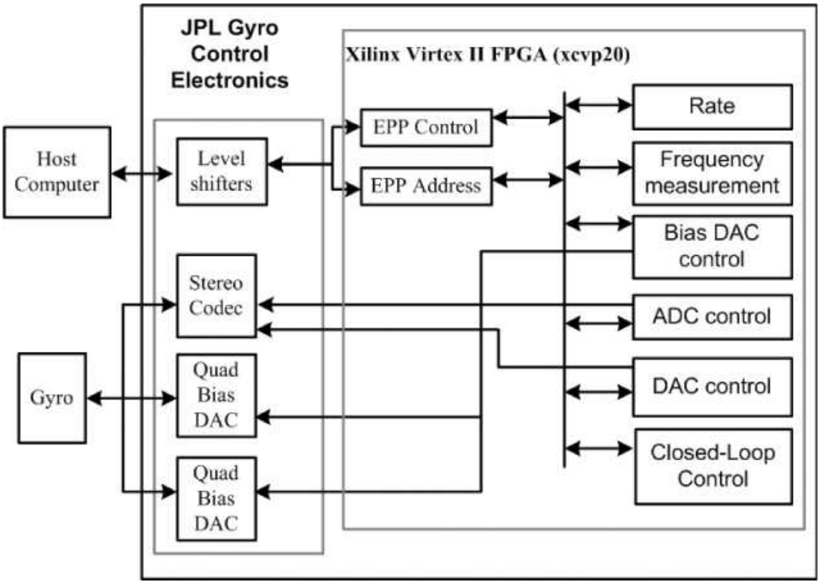


Figure 12-6. Block diagram of the entire closed-loop control system

4.2 Gyro Digital System (GDS)

The system used to implement the control, operation, and observability of the micro-gyro is referred to as the Gyro Digital System (GDS). Figure 12-6 illustrates the implementation of the analog and digital systems used to control the micro-gyro. The key circuit elements that allow proper operation of the micro-gyro include the audio codec (Stereo Digital-to-Analog Converter, DAC), high voltage Analog-to-Digital Converters (ADCs), IEEE-1294 Enhanced Parallel Port (EPP) interface replaced by a UART interface, frequency measurement, and the Digital Signal Processor (DSP) functionality integrated into a Xilinx Virtex II FPGA.

The audio codec is used to translate the analog sensing signals for both the drive and the sense axes. Its stereo capabilities allow for two inputs and two outputs. The high-voltage DACs are utilized for the setting of the electrostatic bias voltages on the gyroscope, which range from +15V to –60V. The parallel port interface allows for user input/output capabilities. The user can configure the coefficients for the finite impulse response (FIR) filters along with the scaling coefficients (K1 through K8) and automatic gain control (AGC) proportional integral (PI) coefficients (Kp and Ki). The codec is configured through this interface as well.

4.3 Results

Using this FPGA digital control system, the micro-gyro was operated for a period of several hours and provided a frequency measurement that was stable to 1 mHz.

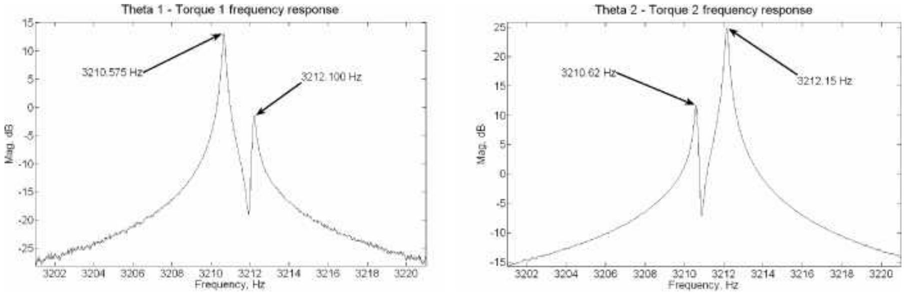


Figure 12-7. Bode magnitude of the experimental frequency response data for a non-tuned MEMS micro-gyroscope (B1=B2=BT1=BT2=14V). Left drive axis on the axis X. Right drive axis on the axis Y

This FPGA system has not yet been tested in the mode where the drive- and sense-axes are swapped, but we have performed experiments using a DSP platform controlled by a Simulink environment running on a PC that demonstrates the feasibility of the closed-loop approach. In Figure 12-7 we show the frequency response of a non-tuned MEMS gyroscope (B1=B2=BT1=BT2=14V) with two peaks for each of the resonance frequencies when drive axis is along axis X (left) and when drive axis is along axis Y (right). Using a closed-loop control, the UCLA team was able to find the correct AGC and FIR filter parameters to maintain the gyro in an oscillating mode at the natural frequency. The DSP platform measured the frequency of both modes by swapping the drive- and sense-axes (F1 = 3210.62Hz and F2 = 3212.15Hz) as shown on Figure 12-7. Keeping the value of the AGC and FIR parameters constant and changing the value of the DC bias voltage, we were able to maintain the gyro in an oscillation mode and to extract both resonance frequencies, which have changed due to the update DC bias voltage.

The next step is to couple the FPGA frequency measurement with the genetic algorithm and the simulated annealing running on the PC. The ultimate goal is to implement the GA and the SA on a microprocessor integrated into a FPGA.

5. CONCLUSION

The tuning method for MEMS micro-gyroscopes based on evolutionary computation shows great promise as a technology to replace the cumbersome, manual tuning process. We demonstrated, using an open-loop measurement, that we can for the first time fully automatically obtain a four times smaller frequency split at a tenth of the time, compared to human performance. We also showed that the "switched drive-angle" system has the option of swapping the drive- and sense-axes, thus decreasing the time required for tuning by more than a factor of fifty compared to the open-loop approach. Additionally, a future design will include a microprocessor on-chip to allow for in situ retuning of the MEMS micro-gyroscope if there is an unexpected change in the behavior due to radiation, temperature shift, or other faults.

The novel capability of fully automated gyro tuning, integrated in a single device next to the gyro, enables robust, low-mass and low-power high-precision Inertial Measurement Unit (IMU) systems to calibrate themselves autonomously during ongoing missions, e.g., Mars Ascent Vehicle.

Acknowledgements

The work described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Special thanks to Tom Prince, who has supported this research through the Research and Technology Development grant entitled "Evolutionary Computation Technologies for Space Systems".

References

Chen, Y., et al. "A control and signal processing integrated circuit for the JPL-Boeing micromachined gyroscopes", (submitted to IEEE).

Ferguson, M. I., et al. 2005. Effect of Temperature on MEMS Vibratory Rate Gyroscope. In *Proceedings of the IEEE Aerospace Conference*, Big Sky, March 2005.

Hayworth, K. 2003. Continuous Tuning and Calibration of Vibratory Gyroscopes. In *NASA Tech Brief*, Oct 2003 (NPO-30449).

Hayworth, K., et al. 2004. "Electrostatic Spring Softening in Redundant Degree of Freedom resonators", patent US 6,823,734 B1, JPL and Boeing, Nov. 30, 2004.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: The University of Michigan Press, 1975.

Kirkpatrick, S., C. D. Gelat, and M. P. Vecchi. 1983. "Optimization by Simulated Annealing", *Science*, 220, 671-680, 1983.

Leland, R. P. 2003. "Adaptive mode tuning vibrational gyroscopes", *IEEE Trans. Control Systems Tech.*, vol. 11, no. 2, 242-247, March 2003.

M'Closkey, R. and D. Kim. 2005. "Real-time tuning of JPL-Boeing MEMS gyro", *Personal Communication*, JPL, March 2005.

Metropolis, N., et al. 1953. "Equation of State Calculation by Fast Computing Machines", *J of Chem. Phys.*, 21, 1087-1091, 1953.

Painer, C. C. and A. M. Shkel. 2003. "Active structural error suppression in MEMS vibratory rate integrating gyroscopes", *IEEE Sensors Journal*, vol. 3, no. 5, 595-606, Oct. 2003.

Stanney, Kay, Ed., 2002. *Handbook of Virtual Environment Technology*. Lawrence Erlbaum Associates, 2002.

Terrile, R. J., et al. 2005. Evolutionary Computation Technologies for Space Systems. In *Proceedings of the IEEE Aerospace Conference*, Big Sky, March 2005.

Yuret, D. and M. de la Maza. 1993. Dynamic Hill Climbing – Overcoming limitations of optimization techniques. In *Proceedings of the 2nd Turkish Symposium of AL and ANN*, 254-260.

INDEX