Preprints

2nd IFAC Workshop on Fractional Differentiation and its Applications

19 - 21 July, 2006
Porto, Portugal

IFAC

isep
Instituto superior de engenharia do porto

INSTITUTO POLITÉCNICO DO PORTO

FCT

Proceedings of the 2nd IFAC
Workshop on Fractional Differentiation and its Applications
Porto, Portugal, July 19-21, 2006

FRACTIONAL-ORDER EVOLUTIONARY DESIGN OF DIGITAL CIRCUITS

Cecília Reis * **J. A. Tenreiro Machado** *
J. Boaventura Cunha ** **Lino Figueiredo** *

* *Institute of Engineering of Porto, Rua Dr. António Bernardino de Almeida, Porto, Portugal*
** *University of Trás-os-Montes and Alto Douro, Engenharias II, Vila Real, Portugal*

Abstract: This paper analyses the performance of a Genetic Algorithm (GA) in the synthesis of digital circuits using a new approach. The novel concept extends the classical fitness function by introducing a fractional-order dynamical evaluation. The dynamic fitness function results from an analogy with control systems where it is possible to benefit the proportional algorithm by including a differential component. For this purpose the non integer derivative is approximated through Padé fractions. The experiments reveal superior results when comparing with the classical fitness method.

Keywords: Fractional Systems, Genetic Algorithms, Digital Circuits

1. INTRODUCTION

In the last decade genetic algorithms (GAs) have been applied in the design of electronic circuits, leading to a novel area of research called Evolutionary Electronics (EE) or Evolvable Hardware (EH) Zebulum et al. (2001). EE considers the concept for automatic design of electronic systems. Instead of using human conceived models, abstractions and techniques, EE employs search algorithms to develop good designs Thompson and Layzell (1999).

One decade ago Louis and Rawlins (1991) applied GAs to the combinational circuit design problem. They combined knowledge-based systems with the GA and defined a genetic operator called masked crossover. This scheme leads to other kinds of offspring that can not be achieved by classical crossover operators.

John Koza (1992) adopted genetic programming to design combinational circuits.

In the sequence of this work, Coello et al. (1996) presented a computer program that automatically generates high-quality circuit designs. They use five possible types of gates (AND, NOT, OR, XOR and WIRE) with the objective of finding a functional design that minimizes the use of gates other than WIRE.

Miller et al. (1997) applied evolutionary algorithms for the design of arithmetic circuits. The technique was based on evolving the functionality and connectivity of a rectangular array of logic cells, with a model of the resources available on the Xilinx 6216 FPGA device.

Kalganova et al. (1998) proposed a new technique for designing multiple-valued circuits.

In order to solve complex systems, Torresen (1998) proposed the method of increased complexity evolution. The idea is to evolve a system gradually as a kind of divide-and-conquer method. Evolution is first undertaken individually on simple cells. The evolved functions are the

basic blocks adopted in further evolution of more complex systems.

A major bottleneck in the evolutionary design of electronic circuits is the problem of scale. This refers to the very fast growth of the number of gates, used in the target circuit, as the number of inputs of the evolved logic function increases. This results in a huge search space that is difficult to explore even with evolutionary techniques. Another related obstacle is the time required to calculate the fitness value of a circuit Vassilev and Miller (2000); Miller et al. (1997). A possible method to solve this problem is to use building blocks either than simple gates. Nevertheless, this technique leads to another difficulty, which is how to define building blocks that are suitable for evolution.

Gordon and Bentley (2002) suggest an approach that allows evolution to search for good inductive bases for solving large-scale complex problems. This scheme generates, inherently, modular and iterative structures, that exist in many real-world circuit designs but, at the same time, allows evolution to search innovative areas of space.

The idea of using memory to achieve better fitness function performances was first introduced by Sano and Kita (2000). Their goal was the optimization of systems with randomly fluctuating fitness function and they developed a Genetic Algorithm with Memory-based Fitness Evaluation (MFEGA). The key ideas of the MFEGA are based on storing the sampled fitness values into memory as a search history, introducing a simple stochastic model of fitness values to be able to estimate fitness values of points of interest using the history for selection operation of the GA.

Following this line of research, and looking for better performance GAs, this paper proposes a GA for the design of combinational logic circuits using fractional-order dynamic fitness functions.

The area of Fractional Calculus (FC) deals with the operators of integration and differentiation to an arbitrary (including noninteger) order and is as old as the theory of classical differential calculus Oldham and Spanier (1974); Miller and Ross (1993). The theory of FC is a well-adapted tool to the modelling of many physical phenomena, allowing the description to take into account some peculiarities that classical integer-order models simply neglect. Nevertheless, the application of FC has been scarce until recently, but the advances on the theory of chaos motivated a renewed interest in this field. In the last two decades we can mention research on viscoelasticity/damping, chaos/fractals, biology, signal processing, system identification, diffusion and wave propagation, electromagnetism and automatic control Oustaloup (1995); Méhauté (1991); Machado (1997); Westerlund (2002).

Table 1. Gate sets

Gate set	Logic gates
Gset a	{AND,XOR,WIRE}
Gset b	{AND,OR,XOR,NOT,WIRE}

Bearing these ideas in mind the article is organized as follows. Section 2 describes the adopted GA as well as the fractional-order dynamic fitness functions. Section 3 presents the simulation results and finally, section 4 outlines the main conclusions and addresses perspectives towards future developments.

2. THE ADOPTED GENETIC ALGORITHM

In this section we present the GA in terms of the circuit encoding as a chromosome, the genetic operators and the static and dynamic fitness functions.

2.1 Problem Definition

The circuits are specified by a truth table with input bits ordered according with the Gray Code. The goal is to implement a functional circuit with the least possible complexity. Two sets of logic gates have been defined, as shown in Table I, being Gset a the simplest one (*i.e.*, a RISC-like set) and Gset b a more complex gate set (*i.e.*, a CISC-like set).

For each gate set the GA searches the solution space, based on a simulated evolution aiming the survival of the fittest strategy. In general, the best individuals of any population tend to reproduce and survive, thus improving successive generations. However, inferior individuals can, by chance, survive and also reproduce. In our case, the individuals are digital circuits, which can evolve until the solution is reached (in terms of functionality and complexity).

2.2 Circuit Encoding

In the GA scheme the circuits are encoded as a rectangular matrix $\mathbf{A}$ (row $\times$ column $= r \times c$) of logic cells as represented in figure 1, having inputs $\mathbf{X}$ and outputs $\mathbf{Y}$.

Each cell is represented by three genes: $<$*input1*$>$ $<$*input2*$>$ $<$*gate type*$>$, where *input1* and *input2* are the circuit inputs, if they are in the first column, or one of the previous outputs, if they are in other columns. The *gate type* is one of the elements adopted in the gate set. The chromosome

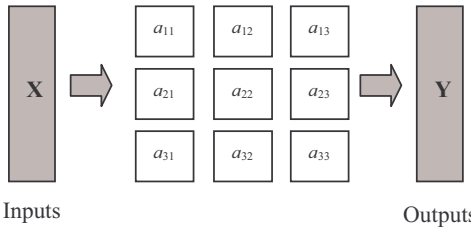

Fig. 1. A 3 × 3 matrix **A** representing a circuit with input X and output Y

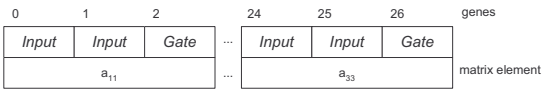

Fig. 2. Chromosome for the 3 × 3 matrix of fig. 1

is formed by as many triplets of this kind as the matrix size demands. For example, the chromosome that represents a 3 × 3 matrix is depicted in figure 2.

2.3 The Genetic Operators

The initial population of circuits (strings) is generated at random. The search is then carried out among this population. The three different operators used are reproduction, crossover and mutation, as described in the sequel.

In what concerns the reproduction operator, the successive generations of new strings are reproduced on the basis of their fitness function. In this case, a tournament selection is used to select the strings from the old population, up to the new population.

For the crossover operator, the strings in the new population are grouped together into pairs at random. Single point crossover is then performed among pairs. The crossover point is only allowed between cells to maintain the chromosome integrity.

The mutation operator changes the characteristics of a given cell in the matrix. Therefore, it modifies the gate type and the two inputs, meaning that a completely new cell can appear in the chromosome. Moreover, an elitist algorithm is applied and, consequently, the best solutions are always kept for the next generation.

To run the GA we have to define the number of individuals to create the initial population P. This population is always the same size across the generations, until the solution is reached.

The crossover rate CR represents the percentage of the population P that reproduces in each generation. Likewise the mutation rate MR is the percentage of the population P circuits that can mutates in each generation.

2.4 The Fitness Functions

The goal of this study is to find new ways of evaluating the individuals of the population in order to achieve GAs with superior performance.

The calculation of the F_s in (1) is divided in two parts, f_1 and f_2, where f_1 measures the circuit functionality and f_2 measures the circuit simplicity. In a first phase, we compare the output $\mathbf{Y}$ produced by the GA-generated circuit with the required values $\mathbf{Y}_R$, according with the truth table, on a bit-per-bit basis. By other words, f_1 is incremented by *one* for each correct bit of the output until f_1 reaches the maximum value f_{10}, that occurs when we have a functional circuit (eq. 1a and 1b).

Once the circuit is functional, in a second phase, the GA tries to generate circuits with the least number of gates. This means that the resulting circuit must have as much genes $<gate\ type>$ $\equiv <wire>$ as possible. Therefore, the index f_2, that measures the simplicity (the number of null operations), is increased by *one* (*zero*) for each *wire* (*gate*) of the generated circuit (eq. 1d), yielding:

- First phase, circuit functionality:

$$f_{10} = 2^{ni} \times no \tag{1a}$$

$$\begin{gathered} f_1 = f_1 + 1, \\ if\ \{bit\ i\ of\ \mathbf{Y}\} = \{bit\ i\ of\ \mathbf{Y_R}\}\ , \\ i = 1, ..., f_{10} \end{gathered} \tag{1b}$$

- Second phase, circuit simplicity:

$$f_2 = f_2 + 1,\ if\ gate\ type = wire \tag{1c}$$

$$F_s = \begin{cases} f_1, & F_s < f_{10} \\ f_1 + f_2, & F_s \geq f_{10} \end{cases} \tag{1d}$$

where $i = 1, \ldots, f_{10}$, and ni and no represent the number of inputs and outputs of the circuit.

The concept of dynamic fitness function F_d results from an analogy with control systems, where we have a variable to be controlled, similarly with the GA case, where we master the population through the fitness function. The simplest control system is the proportional algorithm; nevertheless, there can be other control algorithms, such as, for example, the proportional and the differential scheme.

In this line of thought, applying the static fitness function corresponds to using a kind of proportional algorithm. Therefore, to implement a proportional-derivative evolution the fitness function needs a scheme of the type:

$$F_d = F_s + K\ D^{\mu}\left[F_s\right] \tag{2}$$

where $0 \leq \mu \leq 1$ is the differential fractional-order and K is the 'gain' of the dynamical term.

The generalization of the concept of derivative $D^{\mu}[f(x)]$ to noninteger values of μ goes back to the beginning of the theory of differential calculus. In fact, Leibniz, in his correspondence with Bernoulli, L'Hôpital and Wallis, had several notes about its calculation for $\mu = 1/2$ Oldham and Spanier (1974); Miller and Ross (1993). Nevertheless, the adoption of the *FC* in control algorithms has been recently studied using the frequency and discrete-time domains Oustaloup (1995); Méhauté (1991); Machado (1997).

The mathematical definition of a derivative of fractional order μ has been the subject of several different approaches. For example, Eq. (3) represent the Grünwald-Letnikov definition of the fractional derivative of order μ of the signal $x(t)$:

$$D^{\mu}\left[x\left(t\right)\right] = \lim_{h \to 0} \frac{1}{h^{\mu}} \sum_{k=0}^{\infty} \frac{(-1)^k\, \Gamma\left(\mu+1\right)}{k!\Gamma\left(\mu-k+1\right)} x\left(t-kh\right) \tag{3}$$

where Γ is the gamma function and h is the time increment. This Grünwald-Letnikov formulation Machado (1997) inspired a discrete-time calculation algorithm, based on the approximation of the time increment h through the sampling period T and a r-term truncated series yielding the equation:

$$D^{\mu}\left[x\left(t\right)\right] \approx \frac{1}{T^{\alpha}} \sum_{k=0}^{r} \frac{(-1)^k\, \Gamma\left(\mu+1\right)}{k!\ \Gamma\left(\mu-k+1\right)} x\left(t-kT\right) \tag{4}$$

This technique was adopted in Reis et al. (2005) with $r = 50$.

In this paper the fractional derivative is calculated through a Padé fraction approximation of Euler transformation:

$$\begin{aligned} D^{\mu}(z) &= \left(\frac{1}{T}\right)^{\mu} Padé\left\{(1-z^{-1})^{\mu}\right\}_{m,n} \\ &= \left(\frac{1}{T}\right)^{\mu} \frac{P_m(z^{-1})}{Q_n(z^{-1})} \\ &= \left(\frac{1}{T}\right)^{\mu} \frac{p_0 + p_1 z^{-1} + \cdots + p_m z^{-m}}{q_0 + q_1 z^{-1} + \cdots + q_m z^{-n}} \end{aligned} \tag{5}$$

where m, $n \in \aleph$ are the orders of the polynomials and z^{-1} represents the discrete time sampling.

Table 2. Circuits truth tables

PC4

A_3	A_2	A_1	A_0	P
0	0	0	0	0
0	0	0	1	1
0	0	1	1	0
0	0	1	0	1
0	1	1	0	0
0	1	1	1	1
0	1	0	1	0
0	1	0	0	1
1	1	0	0	0
1	1	0	1	1
1	1	1	1	0
1	1	1	0	1
1	0	1	0	0
1	0	1	1	1
1	0	0	1	0
1	0	0	0	1

M2 − 1

S_0	I_1	I_0	O
0	0	0	0
0	0	1	1
0	1	1	1
0	1	0	0
1	1	0	1
1	1	1	1
1	0	0	0
1	0	0	0

FA1

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	1	0	1
0	1	0	1	0
1	1	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	1	0

3. PERFORMANCE EVALUATION

A reliable execution and analysis of a GA usually requires a large number of simulations to provide that stochastic effects have been properly considered. Therefore, in this study $n = 100$ simulations for each case are executed.

The experiments consist on running the GA in order to generate a combinational logic circuit with the gate sets presented in Table 1, $CR = 95\%$, $MR = 20\%$ and $P = 100$ circuits, using the fitness scheme described previously.

In this article three case studies are adopted corresponding to a 4-bit parity checker ($PC4$), a 2-to-1 multiplexer ($M2-1$) and a 1-bit full adder ($FA1$) as follows:

- the $PC4$ circuit, has 4 inputs $\mathbf{X} = \{A_3, A_2, A_1\ A_0\}$ and 1 output $\mathbf{Y}_R = \{\mathrm{P}\}$. The matrix $\mathbf{A}$ size is 4×4, and the length of each string representing a circuit (i.e., the chromosome length) is $CL = 48$,
- the $M2-1$ circuit, has 3 inputs $\mathbf{X} = \{S_0, I_1, I_0\}$ and 1 output $\mathbf{Y}_R = \{\mathrm{O}\}$. The matrix $\mathbf{A}$ size is 3×3, and $CL = 27$,
- the $FA1$ circuit, has 3 inputs $\mathbf{X} = \{\mathrm{A}, \mathrm{B}, C_{in}\}$ and 2 outputs $\mathbf{Y}_R = \{\mathrm{S}, C_{out}\}$. The matrix $\mathbf{A}$ size is 3×3, and $CL = 27$.

Table 2 presents the Boolean truth Tables for the circuits under study.

The implementation of the differential fractional order operator adopts Eq. (5) with $m = n = 4$ and $T = 1$s.

Having these ideas in mind, a superior GA performance means achieving solutions with a smaller number N of generations, in order to accelerate convergence and a smaller variability, deviation in order to reduce the stochastic nature of the algorithm.

Due to the huge number of possible combinations of the GA parameters, in the sequel we evaluate only a limited set of cases. Therefore, *a priori*, other values can lead to different results. Nevertheless, the authors developed an extensive number of numerical experiments and concluded that the following cases are representative.

We analyze the GA performance when adopting a dynamical scheme for the fitness function and the simulations investigate the differential scheme $\mu = \{0.0,\ 0.25,\ 0.5,\ 0.75,\ 1.0\}$ in F_d for gains in the range $K \in [0,1]$.

Figures 3-5 show the average $AV(N)$ and the standard deviation $SD(N)$ of the required number of generations to achieve a solution versus K with F_d, for the $PC4$, $M2-1$ and $FA1$ circuits, using the Gsets {a, b}, respectively. For comparison the charts include the case $\mu = 0$, that corresponds to the static function F_s. We verify that F_d produces better results than the classical F_s.

In conclusion, the modification of the standard fitness function concept, by introducing the dynamical effects improves significantly the GA performance as it was presented in Reis et al. (2005) when using Taylor Series. Applying the Padé approximation we have achieved similar results for $AV(N)$ and $SD(N)$ with a faster computational scheme.

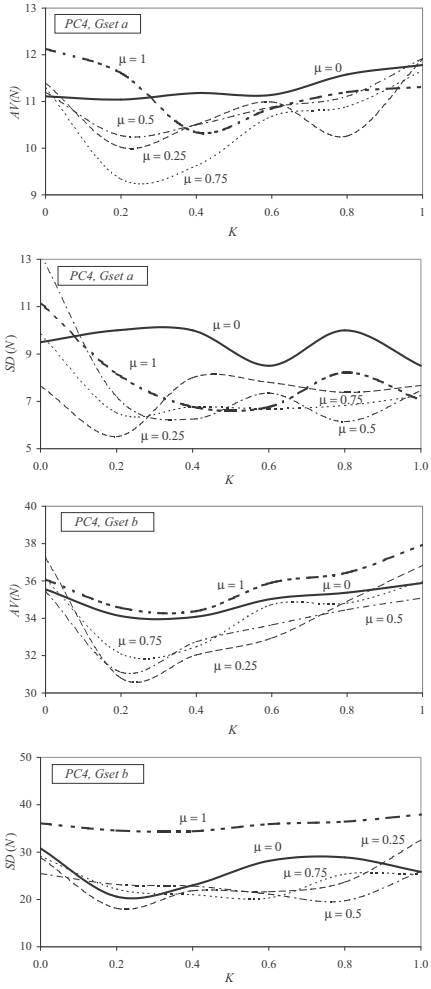

Fig. 3. Average number of generations $AV(N)$ and standard deviation $SD(N)$ to achieve a solution for the $PC4$ circuit versus $K \in [0,1]$, with Gsets {a, b} and F_d

4. CONCLUSIONS

The concept of fractional-order dynamical fitness function is an important method to outperform the classical static approach. The Padé approximation, having the advantage of being a faster implementation technique, leads to similar results as the Taylor Series method in terms of $AV(N)$ and $SD(N)$. The tuning of the 'optimal' parameters (μ, K) was established through a numerical evaluation. Therefore, future research will address the problem of establishing a more systematic design method. Furthermore, these conclusions encourage further studies using other fractional order dynamical schemes.

REFERENCES

C. Coello, A. Christiansen, and A. Aguirre. Using genetic algorithms to design combinational logic circuits. *Intelligent Engineering through Artificial Neural Networks*, pages 391–396, 1996.

T. Gordon and P. Bentley. Towards development in evolvable hardware. In *Proceedings of the 2002 NASA/DOD Conference on Evolvable Hardware*, pages 241–250, 2002.

T. Kalganova, J. Miller, and N. Lipnitskaya. Multiple valued combinational circuits synthesised using evolvable hardware. In *Proceedings of the Seventh Workshop on Post-Binary Ultra Large Scale Integration Systems*, 1998.

J. Koza. *Genetic Programming. On the Programming of Computers by means of Natural Selection.* MIT Press, 1992.

S. Louis and G. Rawlins. Designer genetic algorithms: Genetic algorithms in structure design. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

J. Machado. Analysis and design of fractional-order digital control systems. *SAMS Journal Systems Analysis, Modelling, Simulation*, pages 107–122, 1997.

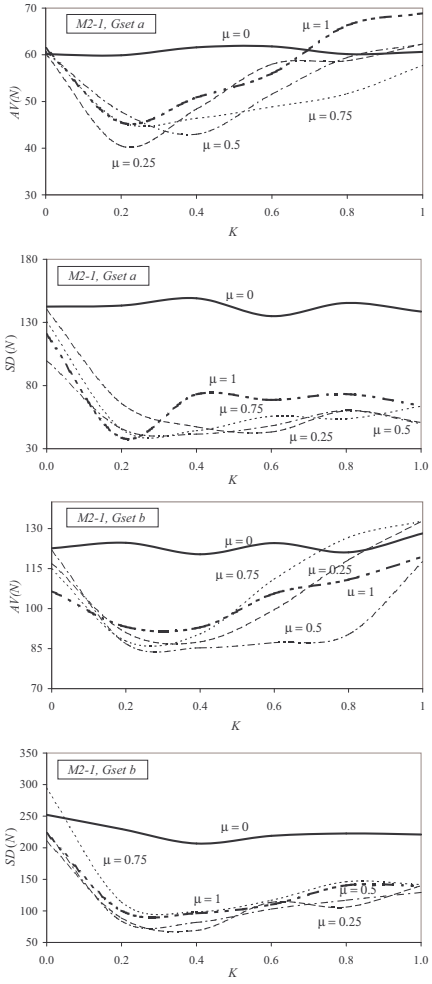

Fig. 4. Average number of generations $AV(N)$ and standard deviation $SD(N)$ to achieve a solution for the $M2-1$ circuit versus $K\in$ [0,1], with Gsets {a, b} and F_d

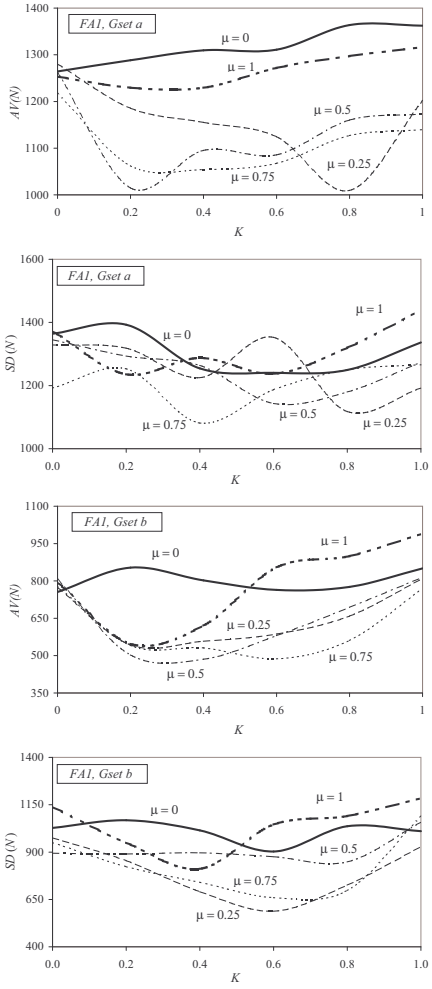

Fig. 5. Average number of generations $AV(N)$ and standard deviation $SD(N)$ to achieve a solution for the $FA1$ circuit versus $K\in$ [0,1], with Gsets {a, b} and F_d

A. Méhauté. *Fractal Geometries: Theory and Applications*. Penton Press, 1991.

J. Miller, P. Thompson, and T. Fogarty. *Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications*. Wiley, 1997.

K. Miller and B. Ross. *An Introduction to the Fractional Calculus and Fractional Differential Equations*. John Wiley & Sons, 1993.

K. Oldham and J. Spanier. *The Fractional Calculus: Theory and Application of Differentiation and Integration to Arbitrary Order*. Academic Press, 1974.

A. Oustaloup. *Dérivation Non Entier: Théorie, Synthèse et Applications*. Ed. Hermes, 1995.

C. Reis, J. Machado, and J. Cunha. Evolutionary design of combinational circuits using fractional-order fitness. In *Proceedings of the Fith EUROMECH Nonlinear Dynamics Conference*, pages 1312–1321, 2005.

Y. Sano and H. Kita. Optimization of noisy fitness functions by means of genetic algorithms using history of search. In *Proceedings of the PPSN VI*, pages 571–581, 2000.

A. Thompson and P. Layzell. Analysis of unconventional evolved electronics. *Communications of the ACM*, pages 71–79, 1999.

J. Torresen. A divide-and-conquer approach to evolvable hardware. In *Proceedings of the Second International Conference on Evolvable Hardware*, pages 57–65, 1998.

V. Vassilev and J. Miller. Scalability problems of digital circuit evolution. In *Proceedings of the Second NASA/DOD Workshop on Evolvable Hardware*, pages 55–64, 2000.

S. Westerlund. *Dead Matter Has Memory!* Causual Consulting, 2002.

R. Zebulum, M. Pacheco, and M. Vellasco. *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. CRC Press, 2001.