International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:1, No:7, 2007

Synthesis of Digital Circuits with Genetic Algorithms: A Fractional-Order Approach

Cecília Reis, J. A. Tenreiro Machado and J. Boaventura Cunha

***Abstract*—** This paper analyses the performance of a genetic algorithm using a new concept, namely a fractional-order dynamic fitness function, for the synthesis of combinational logic circuits. The experiments reveal superior results in terms of speed and convergence to achieve a solution.

***Keywords*—** Circuit Design, Fractional-Order Systems, Genetic Algorithms, Logic Circuits.

I. 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) [1]. EE considers the concept for automatic design of electronic systems, employing search algorithms to develop good designs.

One decade ago Sushil and Rawlins [2] applied GAs to the combinational circuit design problem. In the sequence of this work, Coello, Christiansen and Aguirre [3] presented a computer program that automatically generates high-quality circuit designs. Miller, Thompson and Fogarty [4] applied evolutionary algorithms for the design of arithmetic circuits. Kalganova, Miller and Lipnitskaya [5] proposed a new technique for designing multiple-valued circuits.

In order to solve complex systems, Torresen [6] proposed the method of increased complexity evolution. The idea is to evolve a system gradually as a kind of divide-and-conquer method. The evolved functions are the basic blocks adopted in further evolution of a larger and more complex system.

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 [7]. A possible method to solve this problem is to use building blocks either than simple gates.

Manuscript received January 24, 2004.

Cecília Reis is with the Electrical Engineering Department, Institute of Engineering, Polytechnic Institute of Porto, Porto, Portugal (e-mail: cecilia@dee.isep.ipp.pt).

J. A. Tenreiro Machado is with the Electrical Engineering Department, Institute of Engineering, Polytechnic Institute of Porto, Porto, Portugal (e-mail: jtm@dee.isep.ipp.pt).

J. Boaventura Cunha is with the Engineering Department, University of Trás-os-Montes and Alto Douro, Vila Real, Portugal (e-mail: jboavent@utad.pt).

The idea of using memory to achieve better fitness function performances was first introduced by Sano and Kita [8]. 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 [11-12]. The theory of FC is a well-adapted tool to the modeling 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.

Bearing these ideas in mind this 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.

II. The Adopted Genetic Algorithm

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

A. Problem Definition

To design combinational logic circuits a GA strategy is adopted. The circuits are specified by a truth table and 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 1, 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).

Table 1 Gate sets

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

B. Circuit Enconding

In the GA scheme the circuits are encoded as a rectangular matrix A of logic cells as represented in figure 1.

Each cell is represented by three genes: *<input1><input2><gate type>*, where *input1* and *input2* are one of 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 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.

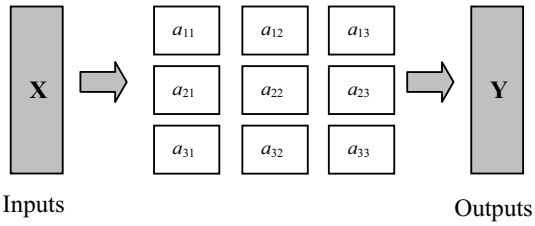


Figure 1: A 3 × 3 matrix A representing a circuit with input **X** and output **Y**.

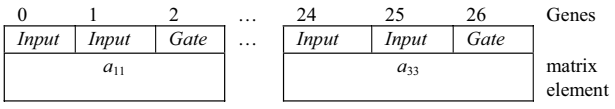


Figure 2: Chromosome for the 3 × 3 matrix of figure 1.

C. The Genetic Operators

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

In what concern the reproduction operator, the successive generations of new strings are reproduced on the basis of their fitness function. In this case, it is used a tournament selection 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, it is applied an elitist algorithm 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* that can mutates in each generation.

D. The Static and the Dynamic Fitness Functions

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

In this paper we propose two concepts for the fitness functions, namely the static fitness function F_s and the dynamic fitness function F_d.

The calculation of F_s in (1) is divided in two parts, f_1 and f_2, where f_1 measures the functionality and f_2 measures the simplicity. In a first phase, we compare the output **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_{10} is incremented by *one* for each correct bit of the output until f_1 reaches the maximum value f_{10}, which occurs, when we have a functional circuit. 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>* ≡ *<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, yielding:

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

$$f_1 = f_1 + 1 \text{ if } \{\text{bit } i \text{ of } \mathbf{Y}\} = \{\text{bit } i \text{ of } \mathbf{Y_R}\}\ ,\ i = 1, \ldots, f_{10} \tag{1b}$$

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

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

where *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, like the differential and the integral schemes. Therefore, applying the static fitness function corresponds to using a kind of proportional algorithm. If we want to implement a proportional-derivative or a proportional-integrative evolution the fitness function needs a scheme of the type:

$$F_d = F_s + K_I\, I^{\lambda}\left[F_s\right] + K_D\, D^{\mu}\left[F_s\right] \tag{2}$$

where $0 \le \lambda \le 1$ is the integral fractional-order parameter, $0 \le \mu \le 1$ is the differential fractional-order parameter, , and K_I, K_D are the integral and the differential 'gains' of the dynamical term, respectively.

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

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

where Γ is the gamma function and h is the time increment. This formulation [13] 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:

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

III. Experiments and Simulation Results

Reliable execution and analysis of a GA usually requires a large number of simulations to provide a reasonable assurance that stochastic effects have been properly considered. Therefore, in this study are developed $n = 1000$ simulations for each case under analysis.

The experiments consist on running the GA to generate a typical combinational logic circuit, namely a 2-to-1 multiplexer (*M2-1*) and a 4-bit parity checker (*PC4*), using the fitness scheme described previously. The circuits are generated with the gate sets presented in Table 1 for $CR = 95\%$, $MR = 20\%$ and $P = 100$ and implementation of the differential/integral fractional order operator adopts Eq. (5) with a series truncation $r = 50$ terms.

Having a superior GA performance means achieving solutions with a smaller number N of generations and a smaller standard deviation in order to reduce the stochastic nature of the algorithm.

The first set of simulations investigate a differential scheme ($\mu = \{0, 0.25, 0.5, 0.75, 1\}$) and an integral scheme ($\lambda = \{0, 0.25, 0.5, 0.75, 1\}$) in F_d for gains $10^{-3} \le K_D \le 10^2$ and $10^{-3} \le K_I \le 10^2$, respectively.

Figures 5 to 8 show the average number of generations to achieve a solution $AV(N)$ and the standard deviation $SD(N)$ for the differential PD^{μ} (i.e., $K_I = 0$) and integral schemes PI^{λ} (i.e., $K_D = 0$), for the *M2-1* and *PC4* circuits, using Gset a and Gset b, respectively. The charts include the plots for $\mu = 0$ and $\lambda = 0$, that is without dynamics, in order to ease the comparison.

Since we achieved better and more stable results for $\mu = 0.25$ and $\lambda = 0.25$, we have investigated the combination of these parameters. Therefore, the second set of simulations investigates a proportional-integral-differential $PI^{\lambda}D^{\mu}$ scheme. Due to the large number of possible combinations of $\{\lambda, \mu, K_I, K_D\}$ we establish $\lambda = \mu = 0.25$ and $10^{-3} \le K_D = K_I \le 10^2$.

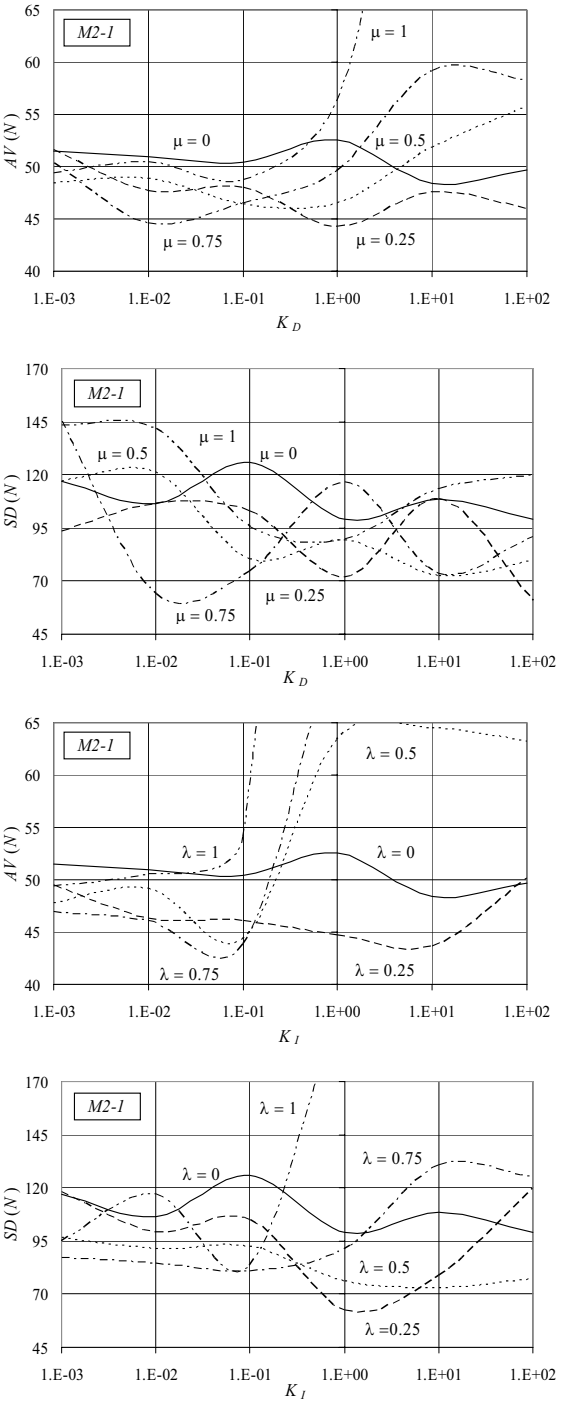
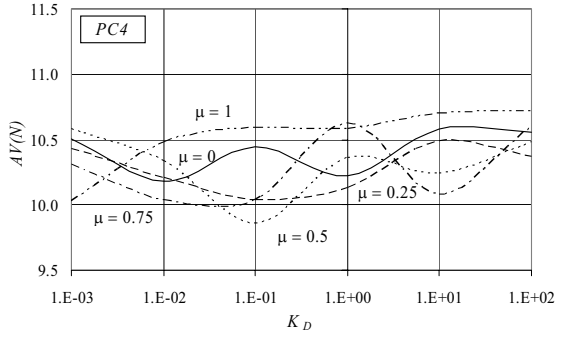
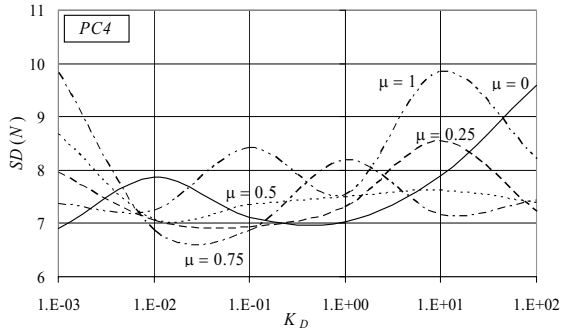
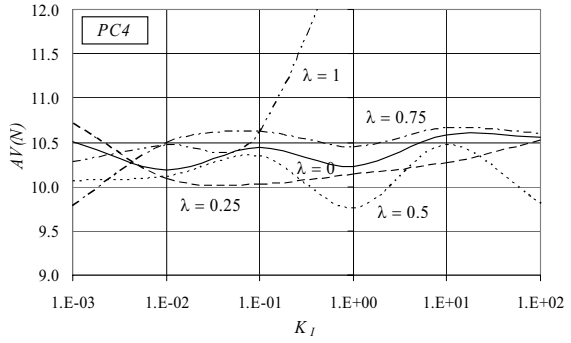


Figure 5: *M2-1* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the PD^{μ} and PI^{λ} schemes with Gset a.







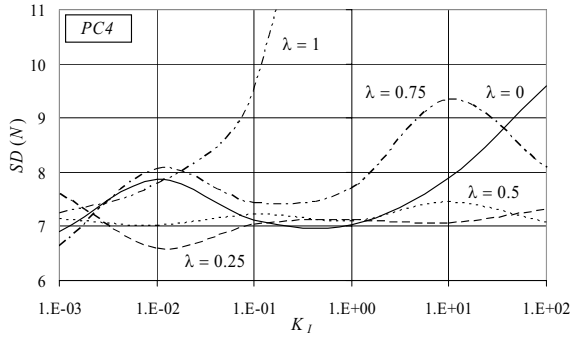
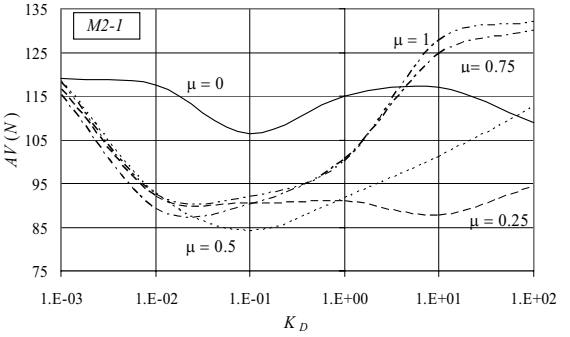
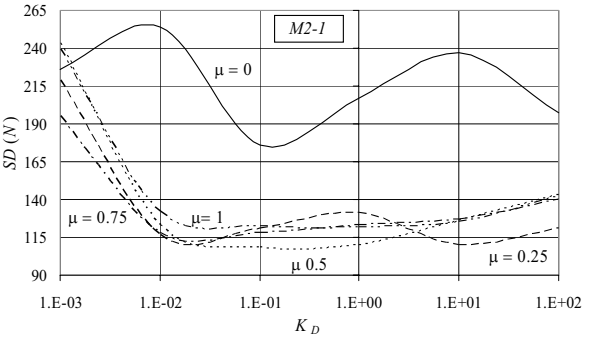
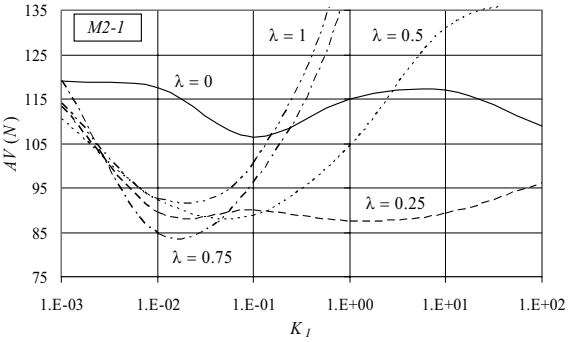


Figure 6: *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the PD^{μ} and PI^{λ} schemes with Gset a.







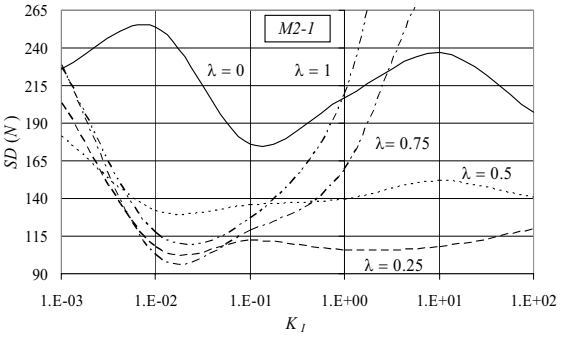


Figure 7: *M2-1* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the PD^{μ} and PI^{λ} schemes with Gset b.

Figures 9 and 10 show the average number of generations to achieve a solution $AV(N)$ and the standard deviation $SD(N)$ for the proportional-integral-differential $PI^{\lambda}D^{\mu}$ scheme, for the *M2-1* and *PC4* circuits, using Gset a and Gset b, respectively.

Comparing the previous $PD^{1/4}$ and $PI^{1/4}$ schemes with the $PI^{1/4}D^{1/4}$ case, we verify that the inclusion of both differential and integral actions improves slightly the results.

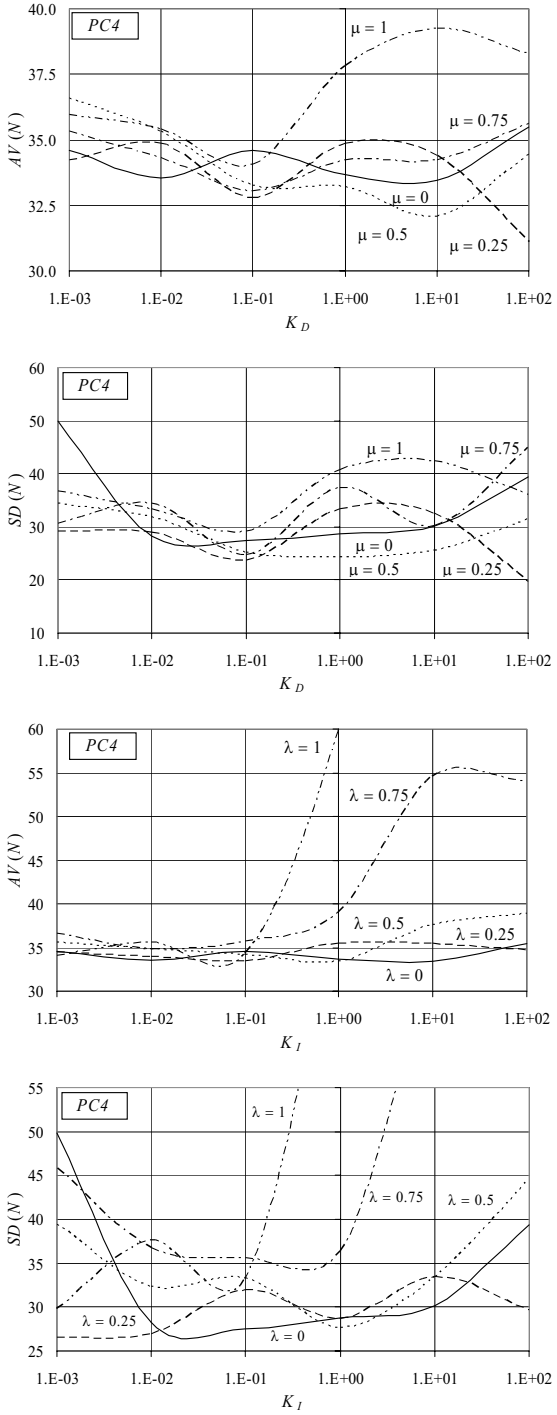


Figure 8: *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the PD^{μ} and PI^{λ} schemes with Gset b.

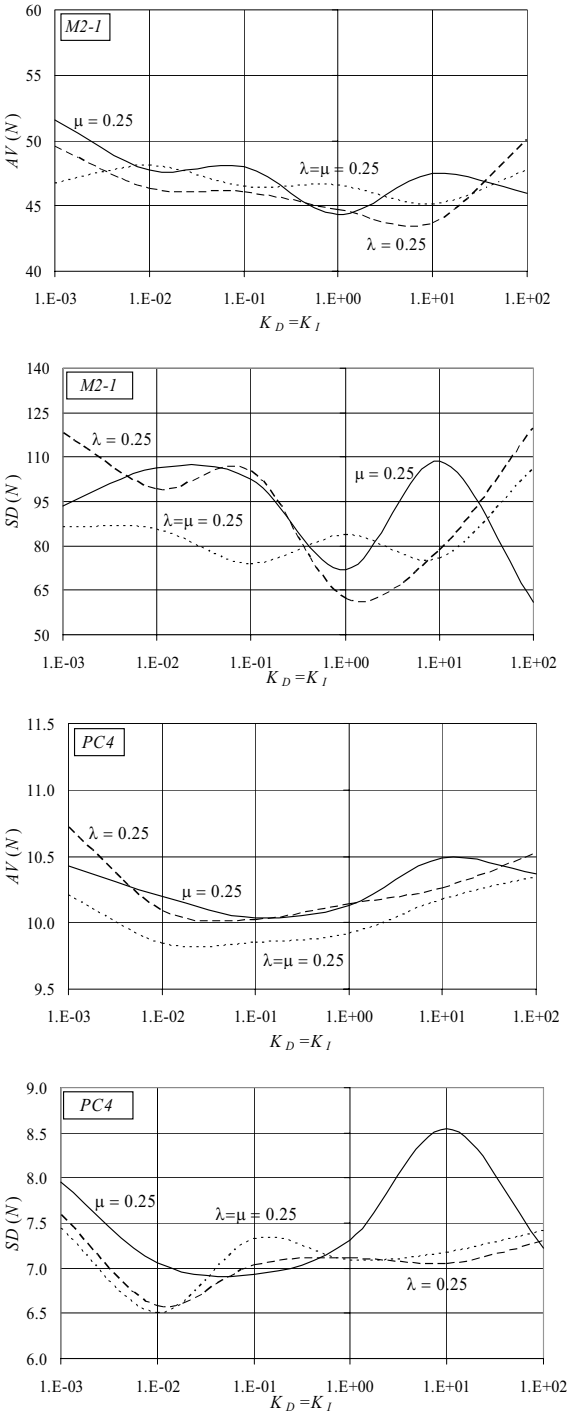


Figure 9: *M2-1* and *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the $PI^{\lambda}D^{\mu}$ scheme with Gset a.

We conclude that the F_d concept produces better results than the classical F_s. Moreover, the results reveal that, as it was expected from previous studies [9,10], the RISC-like set Gset a presents better performance than the CISC-like gate set Gset b for all values of (λ, μ, K_I K_D).

IV. Conclusion

The new concept of fractional-order dynamic fitness function of the GA, demonstrates to be an important method that outperforms the traditional static fitness function

approach. The tuning of the 'optimal' parameters (λ, μ, K_I, K_D) was established by numerical evaluation; therefore, future research will address the problem of having a more systematic design method. These conclusions encourage further studies using fractional order dynamical schemes.

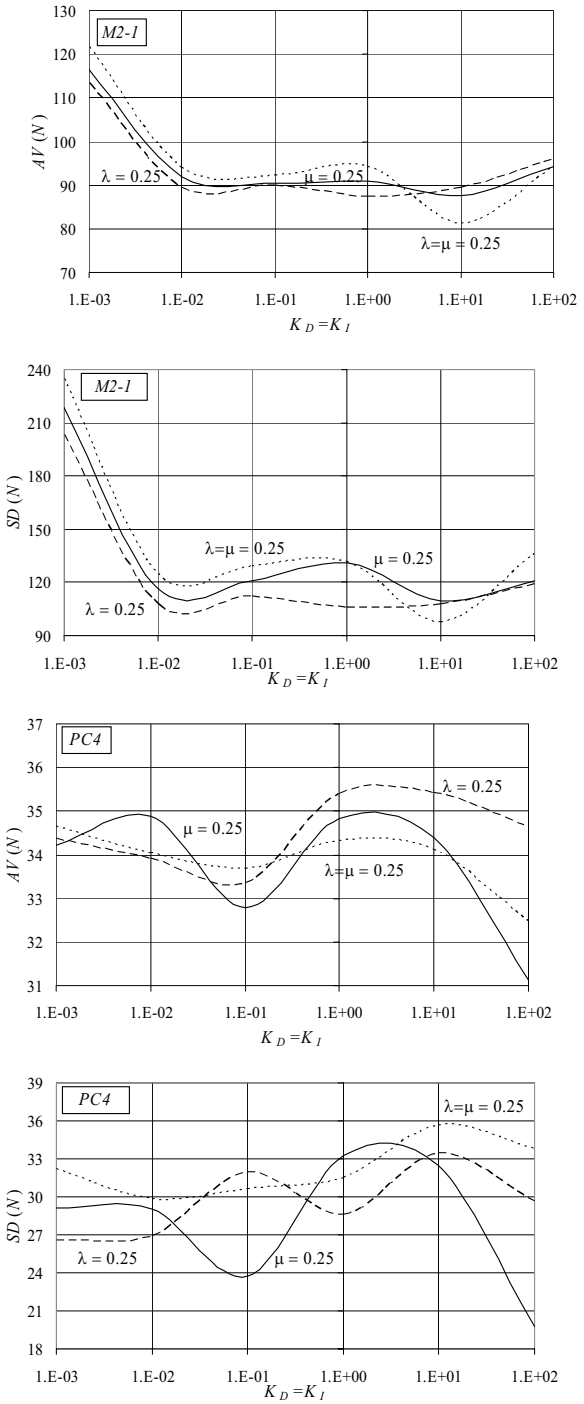


Figure 10: *M2-1* and *PC4* average number of generations to achieve a solution $AV(N)$ and standard deviation $SD(N)$ for the $PI^{\lambda}D^{\mu}$ scheme with Gset b.

REFERENCES

[1] Zebulum, R. S., Pacheco, M. A. and Vellasco, M. M., Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms, CRC Press, 2001.
[2] Louis, S.J. and Rawlins, G. J., "Designer Genetic Algorithms: Genetic Algorithms in Structure Design," in Proc. of the Fourth Int. Conference on Genetic Algorithms, 1991.
[3] Coello, C. A., Christiansen, A. D. and Aguirre, A. H., "Using Genetic Algorithms to Design Combinational Logic Circuits", Intelligent Engineering through Artificial Neural Networks. Vol. 6, 1996, pp. 391-396.
[4] Miller, J. F., Thompson, P. and Fogarty, T, Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications. Chapter 6, 1997, Wiley.
[5] Kalganova, T., Miller, J. F. and Lipnitskaya, N., "Multiple_Valued Combinational Circuits Synthesised using Evolvable Hardware," in Proceedings of the 7th Workshop on Post-Binary Ultra Large Scale Integration Systems, 1998.
[6] Torresen, J., "A Divide-and-Conquer Approach to Evolvable Hardware," in Proceedings of the Second International Conference on Evolvable Hardware. Vol. 1478, 1998, pp. 57-65.
[7] Vassilev, V. K. and Miller, J. F., "Scalability Problems of Digital Circuit Evolution," in Proc. of the Second NASA/DOD Workshop on Evolvable Hardware, 2000, pp. 55-64.
[8] Y. Sano and H. Kita, Optimization of Noisy Fitness Functions by means of Genetic Algorithms using History of Search with test of Estimation, Proceedings of CEC, 2002.
[9] Cecília Reis, J. A. Tenreiro Machado, and J. Boaventura Cunha, "Evolutionary Design of Combinational Logic Circuits", Journal of Advanced Computational Intelligence and Intelligent Informatics, Fuji Technology Press, Vol. 8, No. 5, pp. 507-513, Sep. 2004.
[10] Cecília Reis, J. A. Tenreiro Machado, and J. Boaventura Cunha, "Synthesis of Logic Circuits Using Fractional-Order Dynamic Fitness Functions", Proceedings of the ICCI'2004 – International Conference on Computational Intelligence, 2004, pp. 77-79.
[11] K. B. Oldham and J. Spanier. The Fractional Calculus: Theory and Application of Differentiation and Integration to Arbitrary Order. Academic Press, New York, 1974.
[12] K. S. Miller and B. Ross. An Introduction to the Fractional Calculus and Fractional Differential Equations. John Wiley & Sons, New York, 1993.
[13] J. A. Tenreiro Machado. Analysis and Design of Fractional-Order Digital Control Systems. *SAMS Journal Systems Analysis, Modelling, Simulation*, vol. 27: 107−122, 1997.

Cecília Reis was born in November 24, 1967. She graduated in Electrical Engineering - Industrial Control from Institute of Engineering of Polytechnic Institute of Porto, Portugal, in 1992 and received the Master's degree in Electrical and Computer Engineering from the Faculty of Engineering of the University of Porto, Portugal, in 1995. Presently she teaches at the Institute of Engineering of the Polytechnic Institute of Porto, Department of Electrical Engineering. Her research interests include digital systems, evolvable hardware, genetic algorithms and fractional-order systems.

J. A. Tenreiro Machado was born in October 6, 1957. He graduated and received the Ph.D. degree in Electrical and Computer Engineering from the Faculty of Engineering of the University of Porto, Portugal, in 1980 and 1989, respectively. Presently he is Coordinator Professor at the Institute of Engineering of the Polytechnic Institute of Porto, Department of Electrical Engineering. His main research interests are robotics, modelling, control, genetic algorithms, fractional-order systems and intelligent transportation systems.

J. Boaventura Cunha was born in November 21, 1961. He graduated in Electronics Engineering from the University of Aveiro in 1985 and received the Ph.D. degree in Electrical and Computer Engineering from the University of Trás-os-Montes e Alto Douro, Portugal, in 2002. Presently he is an Assistant Professor at the Engineering Department of the University of Trás-os-Montes e Alto Douro and he is a researcher at the CETAV Institute. His main research interests are automation, modelling and control systems.