Computers and Electrical Engineering 39 (2013) 1367–1379

Contents lists available at SciVerse ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Hardware implementation of the elitist compact Genetic Algorithm using Cellular Automata pseudo-random number generator [☆]

Marco A. Moreno-Armendáriz [*], Nareli Cruz-Cortés, Carlos A. Duchanoy, Alejandro León-Javier, Rolando Quintero

Instituto Politécnico Nacional, Centro de Investigación en Computación, Av. Juan de Dios Batíz s/n, Unidad Profesional Adolfo López Mateos, Col. Nueva Industrial Vallejo, Mexico City 07738, Mexico

A R T I C L E I N F O

Article history:
Received 9 January 2012
Received in revised form 20 March 2013
Accepted 21 March 2013
Available online 13 April 2013

A B S T R A C T

In this paper the design and implementation of two versions of the compact Genetic Algorithm (cGA), with and without mutation and elitism, and a Cellular Automata-based pseudo-random number generator on a Field Programmable Gate Arrays (FPGAs) are accomplished. The design is made using a Hardware Description Language, called VHDL. Accordingly, the obtained results show that it is viable to have this searching algorithm in hardware to be used in real time applications.

1. Introduction

Genetic Algorithms (GAs) are very well known optimization techniques originally proposed in [1]. GA have demonstrated to be successful in the solution of complex numerical and combinatorial optimization problems, for single and multiple objectives [2] simulating natural evolution over populations of candidate solutions. GA handle a set of potential solutions instead of only one, but GA must evaluate the objective function several times, thus one of their main disadvantages is the high computing time required to solve complex problems. Some strategies have been proposed to deal with this drawback; for example, parallel implementations, efficient operators design, and hardware implementations, among others.

On the other hand, compact Genetic Algorithms (cGAs), are a kind of probabilistic model-building Genetic Algorithms or Estimation Distribution Algorithms (EDAs) [3]. cGA operates on probability vectors by replacing the variation operators (crossover and mutation) that describes the distribution of a hypothetical population of solutions. It is known that the cGA obtains solutions of the same quality as the simple GA with binary representation and uniform crossover but with the advantage of an important reduction in the memory requirements, i.e. it only needs to store the probability vector (instead of the entire population) [4]. Therefore, the cGA may be useful in memory-constrained applications such as evolvable hardware [5].

Some efforts have been made in order to improve the cGA performance; for example in [6] the authors proposed using *elitism*, *mutation* and, *champion resampling* demonstrating the improvement without increasing the algorithm complexity.

In this paper, a novel and efficient design of a cGA in a hardware platform is shown. This design presents the following features: modularity, concurrency, minimal resource consumption, real time execution, and high scalability properties.

The main contributions of this work can be summarized as follows:

[☆] Reviews processed and recommended for publication to Editor-in-Chief by Associate Editor Dr. Florin Balasa.
[*] Corresponding author. Tel./fax: +52 5557296000; ×56525.
E-mail address: mam_armendariz@cic.ipn.mx (M.A. Moreno-Armendáriz).

0045-7906/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compeleceng.2013.03.016

- A very useful guide to other researchers on how using this information to develop their own implementation in the hardware of their choice.
- Two pseudo-random numbers generators implemented in hardware and its evaluation in terms of the used resources and the average of iterations required to reach their the solution.

Nowadays there exists different computing machines to implement parallelism, nevertheless in this study we are looking for a portable and autonomous evolvable hardware to be used in real-time tasks [7], therefore we select a FPGA to develop our design.

Also, it is important to mention that the population sizes in GA are problem-dependent, however in most of the cases they are from hundreds to thousands of individuals. Some exceptions are called "micro Evolutionary Algorithms" which have especially small population sizes, i.e. from 3 to 5, but in most of the cases they need additional mechanism to maintain the diversity.

The Genetic Algorithms are inherently parallel mainly due to their population-based nature. Thus, each individual of the population could be executed in a parallel manner. However, the compact Genetic Algorithm (cGA) has not a population, instead it only has a Probability Vector (PV), then, its parallelization is slightly different.

In the cGA implementation considered in this work, each PV element is executed in parallel. Specifically, the tasks performed in parallel are the following:

- Each PV element compares its value against a random number to generate 2 bits, one for each individual.
- Each PV element updates its value.
- Each PV element evaluates its value to determine if it has converged.

The remaining of this paper is organized as follows: in Section 2 we present the previous related work. In Section 3 the compact Genetic Algorithm is explained. Section 4 presents our hardware design proposal. The experimental results are shown in Section 5, and finally the conclusions and future work are presented in Section 6.

2. Related work

In this section we will present some of the most representative works including those related to evolvable hardware as well as those studies related to the compact Genetic Algorithms.

The term *evolvable hardware* is a research area that includes both, the design and implementation of Evolutionary Algorithms into hardware platforms to execute specific tasks; as well as the usage of Evolutionary Algorithms to generate hardware designs.

We will refer only to the former, that is, those works where the authors designed and implemented Evolutionary Algorithms into hardware. More than that, we will refer only to the *compact Genetic Algorithms* (cGAs) (i.e. excluding other kind of Evolutionary Algorithms).

In 2001 Aporntewan and Chongstitvatana [8] proposed a cGA implementation in a FPGA using the language Verilog (Hardware Description Language). The authors showed that their design runs 1000 times faster than its software version executed in a workstation. The design is composed of five modules: random number generator, probability register, comparator, buffer, and fitness function evaluator. It is based on three basic operations: addition, subtraction, and comparisons. The probability vector updating is executed in parallel. It was tested on the Max-One problem with 32 bits and implemented in a Xilinix FPGA 23.57 MHz. using 15,210 gates and a population size equal to 256. This cGA executes one generation per three clock cycles for the Max-One problem. For other more complicated problems, one generation would take $3 + e$ clocks, where e is the number of clocks used to evaluate the fitness function.

In [9,6], the authors argued the cGA is very weak to optimize problems interesting for evolvable hardware's researchers. So, to overcome this issue, they proposed some modifications to the cGA by incorporating *elitism*, *mutation* and *resampling*. They demonstrated it increases cGA exploration capabilities without increasing the hardware complexity. It was implemented with VHDL in a XC400 BORG and Virtex xc2v1000 whose architecture is composed by the following five modules: Random number generator, registers for the vector and mutation probabilities, buffer (2-ports RAM storing the individuals), modules INC/DEC that update each probability vector element and, a register indicating which is the winning individual. It was tested using a Max-One problem with 32-bits and 255 individuals. Their results were compared against the ones in [8], reporting similar results with a very small increment of the consumed resources. Furthermore, the algorithm was implemented on software and tested on the De Jong functions [10]. Additionally, the authors conducted experiments on some dynamic optimization problems.

In [11] a cellular compact Genetic Algorithm implemented in a FPGA was proposed. It consists of a set of identical cGA. Each is called a *cell* and interacts only with its four neighbors. Each cGA cell exchanges probability vectors with its neighbors in an asynchronous schema. The probability vectors are combined by using an equation proposed by the authors. They argued the cellular cGA parallelization is straight forward and suited to be implemented in a FPGA. They experimented with the Max-One problem and two numerical optimization problems demonstrating that their proposal is better than the simple cGA.