Neurocomputing 128 (2014) 316–327

Contents lists available at ScienceDirect

Neurocomputing

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

# An efficient crossover architecture for hardware parallel implementation of genetic algorithm

Rasoul Faraji *, Hamid Reza Naji

*Department of Electrical and Computer Engineering, Graduate University of Advanced Technology, Haftbagh BLV, Kerman, Iran*

A R T I C L E I N F O

*Article history:*
Received 22 March 2013
Received in revised form
24 August 2013
Accepted 25 August 2013
Communicated by R.W. Newcomb
Available online 18 October 2013

*Keywords:*
Genetic algorithm
FPGA
Crossover operator

A B S T R A C T

In this article a new architecture for hardware implementation of genetic algorithm in reconfigurable embedded systems is presented. The main idea is based on the efficient use of a genetic algorithm's crossover operator to enhance the speed of algorithm to reach an optimal solution. In this article a new crossover called DSO and also two new architectures for implementation of crossover operators are introduced to provide suitable solutions for solving the problems related to fitness function of the genetic algorithm. At first, some optimum operators are selected and then utilized in a new parallel architecture to increase the speed and accuracy of algorithm convergence. Finally, based on reusability of existing resources, the main idea of the article is introduced to improve the performance of the algorithm and finding the optimal solution. The properties of FPGAs such as flexibility and parallelism help this purpose.


## 1. Introduction

The genetic algorithm is an efficient search-optimization technique inspired by natural evolution for finding approximate solutions. This algorithm is a stochastic search based on iteration. Complex optimization problems with too many solutions need a method of finding an appropriate solution in the least possible time. Meta-heuristics algorithms like genetic algorithm can be a suitable approach according to its characteristics. The characteristics that drive the genetic algorithm to approach this goal are:

(1) The genetic algorithm uses stochastic search technique in sets of population that leads to wide access of existing solutions. These sets of population are placed in the global search space. The global search selects a series of regions from the entire search space for detailed exploration by the local search. In fact, the local search accelerates finding optimal solutions.
(2) The use of stochastic variables in the main operators of the algorithm (such as mutation and crossover) and also applying probabilistic transition rules, help the convergence of algorithm to the appropriate solution.
(3) The genetic algorithm uses the inherited fitness function information and doesn't need of additional auxiliary knowledge [1]. In fact in this algorithm when decisions are made and changes occur then fitness function is used to determine whether new changes improve the results or not.

FPGAs provide suitable platforms for hardware implementation of software algorithms. The low execution time of FPGA in comparison with its software counterparts is the main reason of using FPGA as a platform to implement the genetic algorithm. A feature which enhances the performance of FPGAs is its parallel processing capability. This feature enables the algorithm to enhance its performance by running different parts of algorithm simultaneously which is very important in real time applications. Note that, the parallel implementation requires a large amount of hardware resources and high rate of memory access, so we prefer to use a combination of pipeline and parallel method [2]. In order to have an optimum performance the main operators of this algorithm can be implemented using pipeline.

In recent years, new methods for hardware implementation of genetic algorithms are proposed to improve its accuracy [3,4], increase the speed of operation [1,5–7], reduce the resources used for implementation [8], improve the searching space [9], having a general-purpose design [10] and enhance the performance of floating-point fitness functions [11]. Considering the features of FPGAs, this article tends to introduce a new method that increases the speed and accuracy of algorithm using this technology. We chose some optimum crossover operators with new architecture to achieve a better performance. The main idea is to use multiple crossovers in the form of a single crossover which runs simultaneously with other crossovers to enhance the probability of convergence to get an optimal solution. The generations produced by this crossover are somehow a kind of anticipation of the future generations.

The paper is organized as follows: In Section 2, the principles of genetic algorithms are described, a general overview of the

* Corresponding author. Tel.: +98 3426226611; fax: +98 3426228011.
*E-mail addresses:* R.Faraji@kgut.ac.ir (R. Faraji), hamidnaji@ieee.org (H.R. Naji).

0925-2312/$ - see front matter Crown Copyright © 2013 Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.08.035

proposed architecture is presented in Section 3. Section 4 provides implementation of the proposed algorithm. Simulation results and comparisons are presented in Section 5. Section 6 provides conclusion and finally related future works are discussed in Section 7.

## 2. Principles of genetic algorithms

In genetic algorithm, at first some chromosomes are randomly selected through the initial population using a special search method. Then a value is assigned to the selected chromosomes according to the fitness function and two chromosomes that have the highest fitness value are chosen as initial parents. These parents are then reproduced to form a new generation using crossover and mutation operators in a repeated process. Crossover

```
Parameters available to the designer :
l : individual length
n : population size
gens_num : number of generations
cc : convergence_criteria
   begin
initial_population := randomly search initial individuals to the best global individual ;
       population := call local search (initial _population) ;
          do:
             fitness  := evaluate (population) ;
             parents := select (population) ;
new_gen := mutation(crossover(parent)) ;
             fitness := evaluate (new_gen) ;
                      if fitness (new_gen) fitter than fitness (parent)
                         then   population := new_gen ;
             generation := generation +1 ;
repeat until (fitness(new_gen [i] /= cc) OR (generation ≤ gens_num))
return fittest individual (population) ;
```

**Fig. 1.** The Pseudocode of the genetic algorithm.

operator is one of the most important operators in genetic algorithm and it combines chromosomes to produce a new generation by providing general heuristics.

Crossover and mutation operators attempt to perturb the characteristics of parents to generate distinct offspring with better characteristics than their parents. If the produced generations have more appropriate values than their parents based on fitness function, they will substitute their parents in the next reproduction process. This process repeats until some criteria such as range of convergence, number of produced generations and evolution time are achieved. The Pseudocode of genetic algorithm is shown in Fig. 1.

In this algorithm, the local search interacts with the global search to have an optimal search. After producing new offspring and replacing the old generations, the algorithm would stop when criteria are satisfied. Now if the solutions do not get to the convergence range and the stopping condition does not satisfy, then the algorithm will be repeated infinite times. To solve this problem we use number of produced generations (gen-number) and after a specified number of generations algorithm stops whether it is converged or not. So the number of produced generations is an appropriate criterion for evaluation of the results.

## 3. The proposed algorithm

Fig. 2 shows the main structure of the proposed hardware implementation of genetic algorithm. To optimize the selection process we use two searches that interact in a hierarchical manner, namely global search and local search. The global search selects some regions of initial population from entire search space randomly, and then the selected regions which are consist of

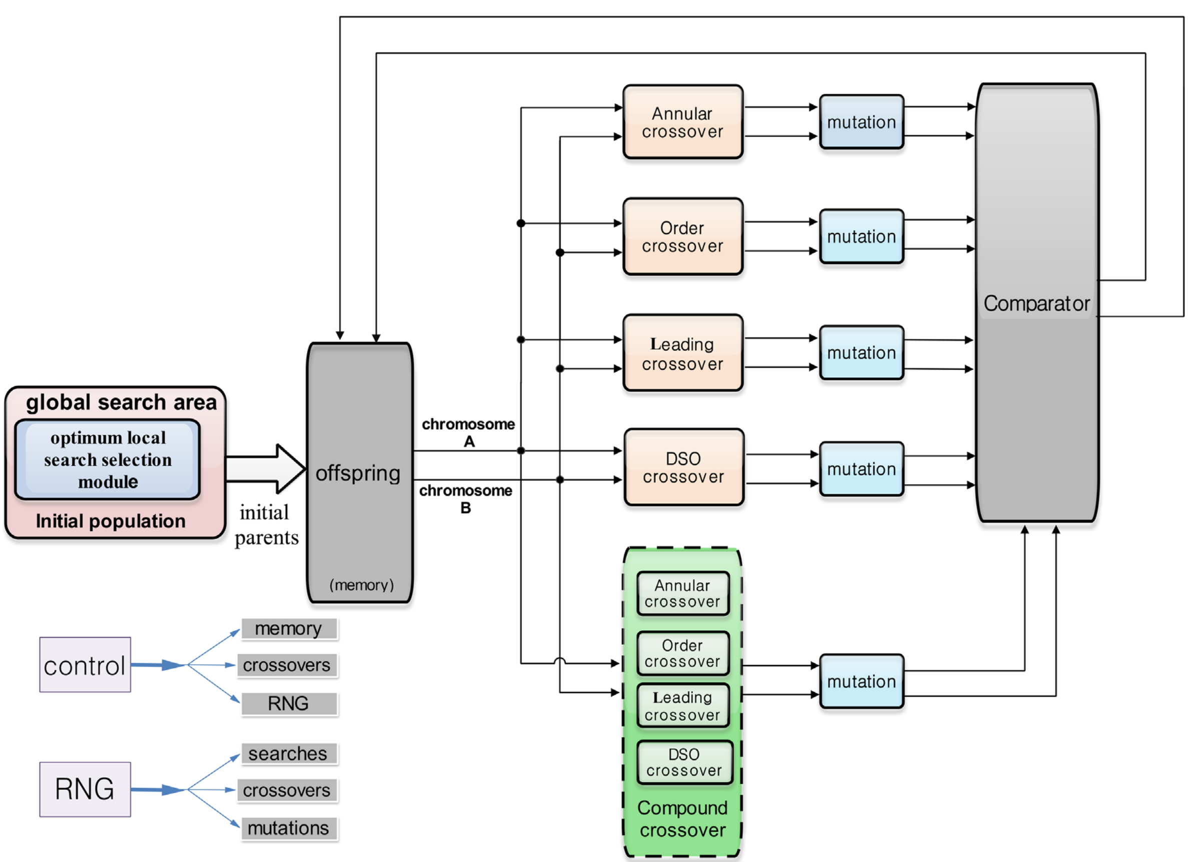


**Fig. 2.** The proposed hardware architecture of genetic algorithm.