Open Access Library Journal
2020, Volume 7, e6209
ISSN Online: 2333-9721
ISSN Print: 2333-9705

A Prototype of FPGA Based on Genetic Algorithm Core Connected to a Cluster

Ahmed D. Alwzan[1*], Abdul Sattar Mohammed Khidhir[2*], Muhmood Shuker Thabit[3]

[1]Northern Technical University, Technical Engineering College, Mosul, Iraq
[2]Northern Technical University, Computer Center, Mosul, Iraq
[3]Northern Technical University, Mosul Technical Institute, Mosul, Iraq
Email: *ahmedalwazan2017@gmail.com

How to cite this paper: Alwzan, A.D., Khidhir, A.S.M. and Thabit, M.S. (2020) A Prototype of FPGA Based on Genetic Algorithm Core Connected to a Cluster. *Open Access Library Journal*, 7: e6209.
https://doi.org/10.4236/oalib.1106209

Received: March 4, 2020
Accepted: April 5, 2020
Published: April 8, 2020

Abstract

In this paper, it was proposed to design and implement a system of parallel processing to find the required solutions from selective algorithms as quickly. Motives for writing paper are the current performance of computer systems that depend on evolutionary algorithms (EA), and the wide spread of the (EA), and its application to a very wide range of scientific fields, and by taking advantage of the Field programmable gate array (FPGA) board due to its high speed of implementation. The values were validated and the Genetic Algorithm (GA) was used as a functional model and implementation. Also, in the most important stages, the process of calculating fitness function, which is considered an executive criterion for the (GA), with terminal computers with high speeds and medium specifications, was done for the purposes of calculating fitness function independently of Board. Identical results were obtained at 100% accuracy by applying the work to a non-linear quadratic.

Subject Areas

Engineering/Computer Engineering

Keywords

Evolutionary Algorithms, Genetic Algorithm, FPGA, Parallel GA

1. Introduction

Presently, the need for fast, high-performance computing is pressing since most computational problems require an excessive amount of computing time [1]. Although FPGA speed and capacity have increased under Moore's Law, there has been no increase in design productivity, creating a large gap between the ability of FPGAs and the ability to use them effectively. This gap increases the cost of market development time for FPGA-based designs, with insufficient design tools being one of the main factors [2]. In my research, I applied the fitness function calculation to the parallel genetic algorithm (PGA) to take advantage of the high speed of FPGA. EA are population-based metaheuristic optimization algorithms that use biology-inspired mechanisms and "survival of the fittest" theory to refine a set of solutions iteratively. GAs are a subclass of EAs where the elements of the search space are binary strings or arrays of other elementary types [3]. Tabassum *et al.* found that GAs have many advantages that make them one of the most preferred and widespread algorithms in Artificial Intelligence [4]. GAs have many applications; for instance, Metawa *et al.* proposed an intelligent model based on the GA to organize bank lending decisions in a highly competitive environment with a credit crunch constraint [5]. Cerrada *et al.* built up a durable system for the multi-class fault diagnosis in spur gear by selecting the best set of condition parameters on frequency and time-frequency domains, which are extracted from vibration signals by using a GA based on the k-means algorithm [6]. Deng *et al.* developed a new strategy to improve the genetic algorithm for solving the traditional combinatorial optimization problem, the traveling salesman problem [7]. Qiu *et al.* presented a GA-based optimization algorithm for a chip multiprocessor equipped with Phase-change memory (PCM) in green clouds [8]. Paul Graham *et al.* described a Splash 2 GA, a parallel genetic algorithm that improved symmetrical traveling salesman problems using Splash 2. In this study, each processor consisted of four FPGAs and associated memories that performed 6.8 to 10.6 times the speed of equivalent software [9]. Tang *et al.* implemented a GA using the domain programming gate FPGA array, which improved the speed of the genetic algorithm in parallel with the devices [10]. Finally, Torquato *et al.* proposed full implementation of optimized parallel GA based on FPGA to minimize system processing time, which is the main objective of this project. Torquato's results showed that the full parallel GA implementation yielded about 16 million generations per second and acceleration between 17 and 170,000 m/s^3 as supported by several works in the literature [11].

2. Theory

A GA is a heuristic search and optimization technique that uses algorithms to mimic the process of natural evolution [12]. Drawing from the theory of evolution by natural selection [13], GAs have important applications for problems related to optimization, ma-chine learning, game theory, design automation, evolvable hardware, distributed sys-terms, network security, and bioinformatics. A GA is an iterative procedure that works on groups of solution representations called chromosomes. Each chromosome consists of smaller segments of data called genes and a set of chromosomes forms a population. Genes are usually initialized randomly in each chromosome. The basic iterative work of the GA is an evolution from one population, (k), to the next population, (k+1). The solution of the optimization problem evolves toward a better solution [14]. Usually, a simple GA consists of three operations: selection, crossover, and mutation. **Figure 1** below shows a flowchart of the genetic algorithm. The GA follows 6 steps:

1) Initialization: the initial population of candidate solutions is usually generated randomly across the search space.

2) Evaluation: once the population is initialized or an offspring population is created. Then, the fitness function is calculated for each element created.

3) Selection: the algorithm creates more copies of those solutions with higher fitness values, thus imposing a "survival of the fittest" mechanism on the candidate solutions.

4) Crossover: parts of two or more parent chromosomes are combined to create new solutions.

5) Mutation: while recombination operates on two or more parental chromosomes, mutation acts locally to randomly modify a solution.

6) Replacement: the offspring population created by selection, recombination, and mutation replaces the original parental population [15].

3. System Model

The system consists of five parts which are shown in **Figure 2**. This system includes FPGA, Arduino, a server PC, a router, and three client PCs.

1) FPGA used Xilinx Spartan 6 (SP 601). In this part of the model, we used 24 numbers from a random number generator (RAD) to increase the population in the genetic algorithm and we can use the Equation (1), is to generate these numbers.

$$\text{RAD}(0) \le \text{NOT RAD}(7)\ \text{XOR RAD}(6) \tag{1}$$

where RAD is random number generator, NOT is inverse gate and XOR is deferential gate.

2) Arduino (UNO), an open-source platform used for constructing and programming electronics [16], was used to transfer the RAD and the constructor in FPGA by connected warring in the J13 Hader in SP601 and pin 4 in Arduino.

3) The PC server has the following specifications: CORE i5, RAM 8 G, CPU 2.5MHZ. The PC server was used to redirect the RAD to several PC clients using JAVA software.

4) The router is a local area network using router type TP _link and was used to send and receive between the server and client's PC.

5) The client PCs had the following specifications: 1) CORE i3, RAM 4 G, CPU 2.5MHZ, 2) CORE i5, RAM 6 G, CPU 2.4MHZ, and 3) CORE i7, RAM 8 G, CPU 2.6MHZ. These PCs were used to receive the RNG and calculate the fitness function and send the result to the server.

In the following **Figure 3**, a flowchart for all the stages of the proposed work system.

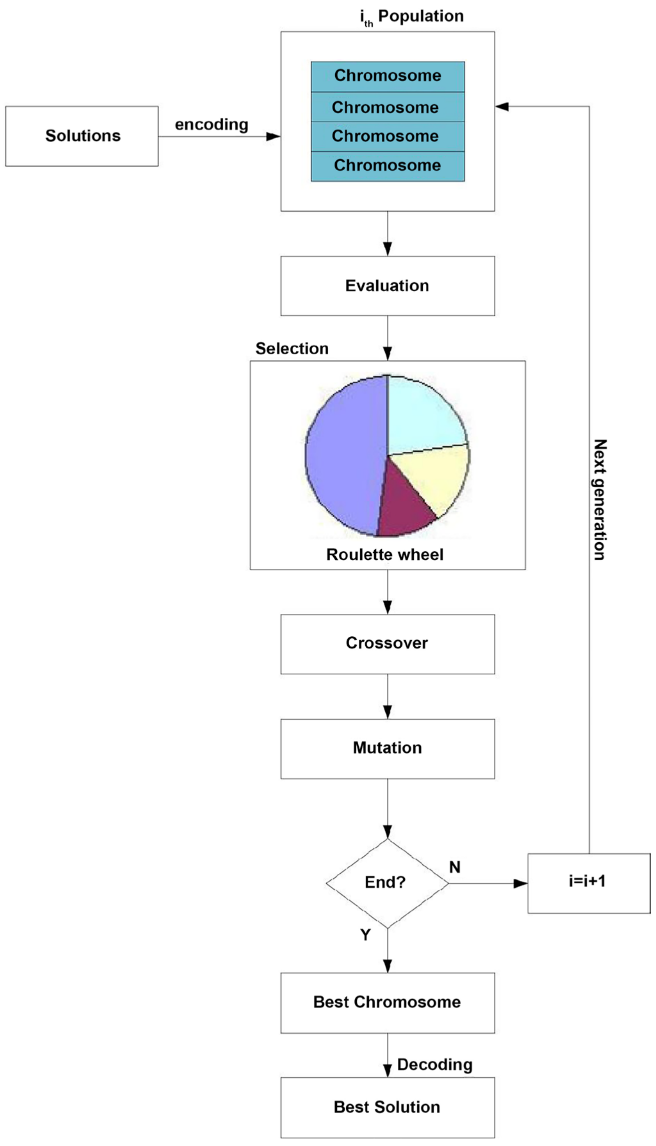


Figure 1. Flowchart of the genetic algorithm [15].

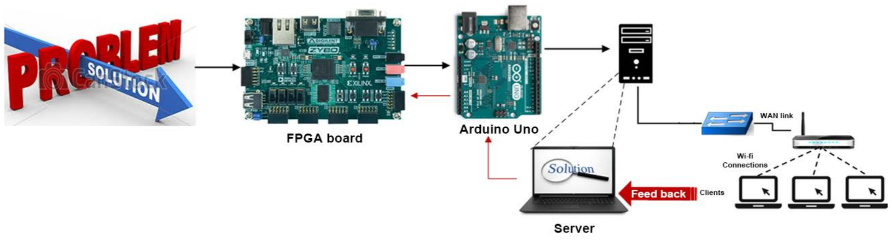


Figure 2. Proposed system schematic diagram.

4. Result

Through this study, 24 RAD is generated inside the FPGA, as shown in Figure 4 below.

After creating random numbers within the FPGA, a 10-bit header was attached to inform the server computer to prepare to receive the beginning of the

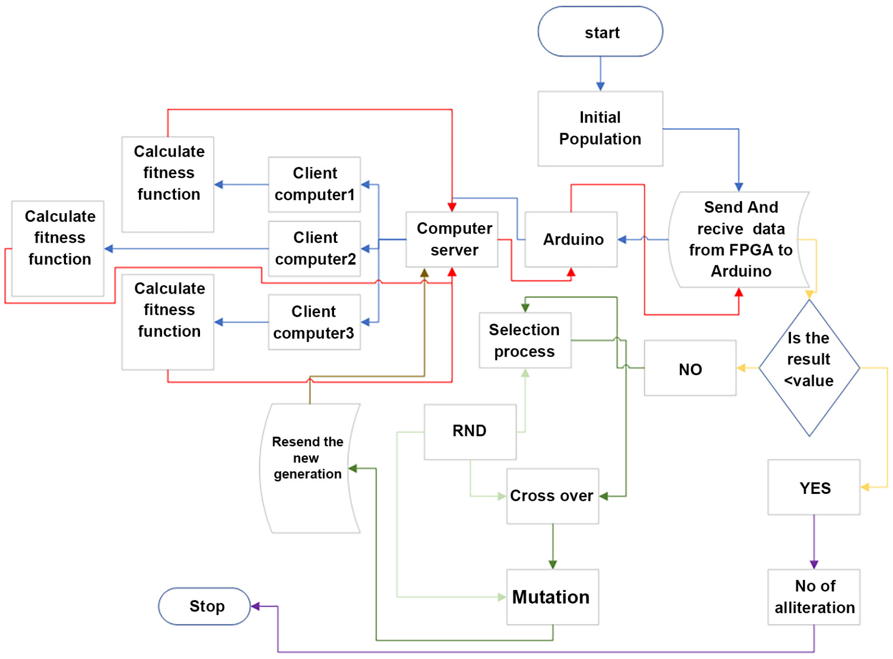
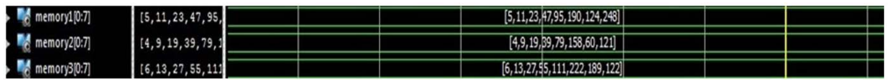


Figure 3. Flowchart for all the stages.



```
memory1[0:7]  [5,11,23,47,95,   [5,11,23,47,95,190,124,248]
memory2[0:7]  [4,9,19,39,79,1   [4,9,19,39,79,158,60,121]
memory3[0:7]  [6,13,27,55,111   [6,13,27,55,111,222,189,122]
```

Figure 4. The 24 random numbers generated inside the board.

packet. In addition, 2 bits were attached to the beginning of each of the three random numbers as shown in Table 1. After receiving the frame completely from the server computer, the header or the 2-bit that is used for identifying each group is ignored and each group is sent to the client computer after converting all numbers from binary to integers, using the following Equation (2):

$$f(x,y) = \left[(x+2y-7)\right]^\wedge 2 + \left[(2x+y-5)\right]^\wedge 2 \tag{2}$$

In the following Figure 5, the numbers are sent after creating the frame to be sent to the Board of Arduino. Figures 6-8 below show the results of the values obtained from all client computers.

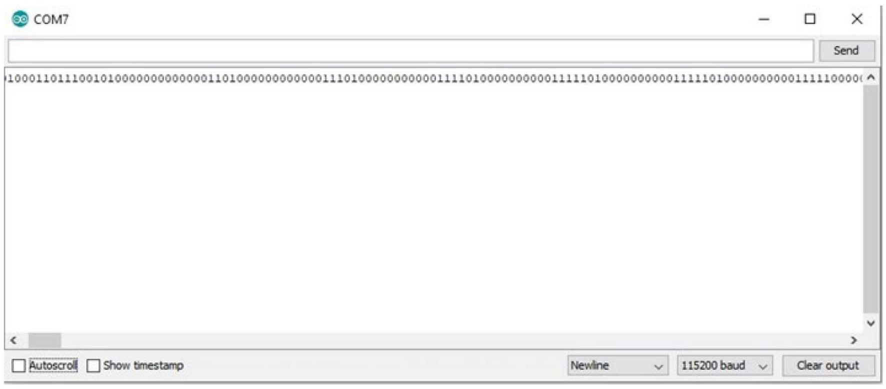
In the following Figure 9, the results are re-sent to the FPGA board. For comparison purposes, we note that there is a difference between the work proposed by us and the research done by researchers [17], where we note their suggestion of a parallel method for the genetic algorithm by proposing the same numbers and taking a specific set of these numbers while we suggest different groups of numbers be in one and thus the process of obtaining the correct solution or The closest solution to the correct more quickly compared to the research referred to above.

5. Conclusion

This work contributes to the implementation on FPGA and this paper proposed

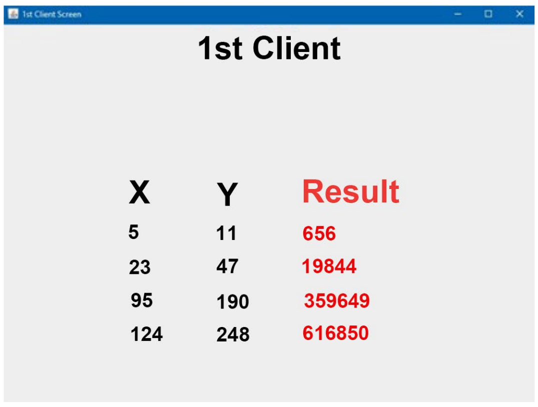
Table 1. Header and indication of each group.

header and indication	
Header	1000110111
Identification group 1	00
Identification group 2	01
Identification group 3	10



```
1000110111001010000000000000011010000000000000111010000000000000111101000000000000111110100000000000111111010000000000000111111000000
```

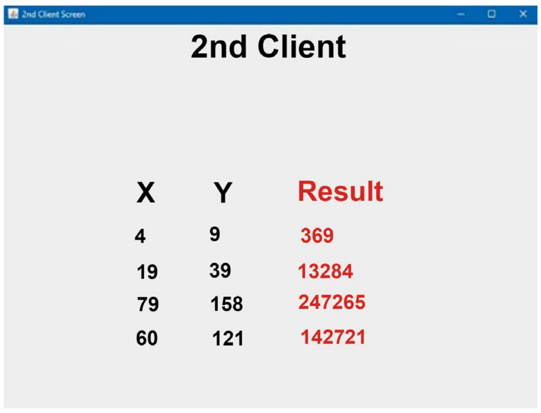
Figure 5. Send data to the Arduino board.



1st Client

```
X      Y      Result
5      11     656
23     47     19844
95     190    359649
124    248    616850
```

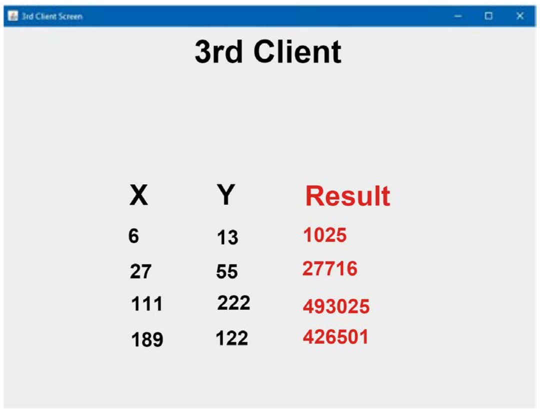
Figure 6. Results of client 1.



2nd Client

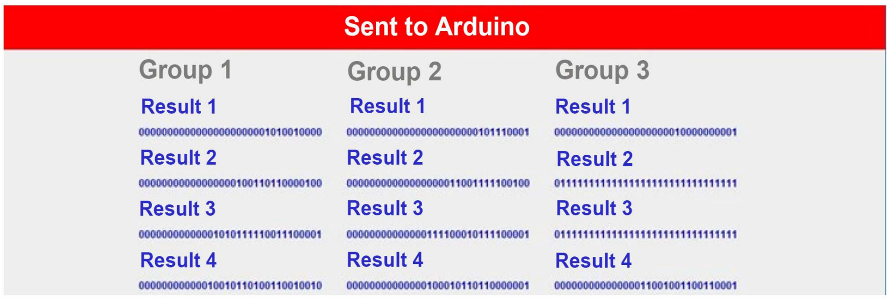
```
X      Y      Result
4      9      369
19     39     13284
79     158    247265
60     121    142721
```

Figure 7. Results of client 2.



X	Y	Result
6	13	1025
27	55	27716
111	222	493025
189	122	426501

Figure 8. Results of client 3.



	Group 1	Group 2	Group 3
Result 1	00000000000000000000001010010000	00000000000000000000000101110001	00000000000000000000001000000001
Result 2	00000000000000000100110110000100	00000000000000000001100111100100	01111111111111111111111111111111
Result 3	00000000000001010111110011100001	00000000000000111100010111100001	01111111111111111111111111111111
Result 4	00000000000010010110100110010010	00000000000000100010110110000001	00000000000000011001001100110001

Figure 9. Results send to FPGA bord.

system to calculate and solve complex equations, such as high order equations in less time than other systems and with high accuracy, using PCs with only medium specifications. The proposed system adopted an enhanced genetic algorithm based on an FPGA board. The proposed system used a microcontroller based on Arduino (Uno), PCs, a Wi-Fi module, and an FPGA board. The experiments were conducted to calculate values for non-linear equations of the second order. The experiments' results indicate that the proposed genetic algorithm can calculate the value of the equation with 100% accuracy.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Quartian, E.S., Sibaroni, Y. and Nhita, F. (2016) Parallel Genetic Algorithm for Traveling Salesman Problem on Graphic Processing Unit. *Journal of Information Technology Education*, **3**, 139-146. https://doi.org/10.12988/jite.2016.6829

[2] Hoo, C.H. (2017) ParaDiMe: A Distributed Memory FPGA Router Based on Speculative Parallelism and Path Encoding. *IEEE 25th Symposium on Field-Programmable Custom Computing*, CA, USA, March 2017, 172-179. https://doi.org/10.1109/FCCM.2017.34

[3] Malhotra, R., Singh, N. and Singh, Y. (2011) Genetic Algorithms: Concepts, Design

for Optimization of Process Controllers. *Computer and Information Science*, **4**, 39-54. https://doi.org/10.5539/cis.v4n2p39

[4] Tabassum, M. and Mathew, K. (2014) A Genetic Algorithm Analysis towards Optimization Solutions. *International Journal of Digital Information and Wireless Communications*, **4**, 124-142. https://doi.org/10.17781/P001091

[5] Metawa, N., Hassan, M.K. and Elhoseny, M. (2017) Genetic Algorithm Based Model for Optimizing Bank Lending Decisions. *Expert Systems with Applications*, **80**, 75-82. https://doi.org/10.1016/j.eswa.2017.03.021

[6] Cerrada, M., Zurita, G., Cabrera, D., Sánchez, R., Artés, M. and Li, C. (2015) Fault Diagnosis in Spur Gears Based on Genetic Algorithm and Random Forest. *Mechanical Systems and Signal Processing*, **70-71**, 87-103. https://doi.org/10.1016/j.ymssp.2015.08.030

[7] Deng, Y., Liu, Y. and Zhou, D. (2015) An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP. *Mathematical Problems in Engineering*, **2015**, Article ID: 212794. https://doi.org/10.1155/2015/212794

[8] Qiu, M., Member, S., Ming, Z., Li, J., Gai, K. and Zong, Z. (2015) Phase-Change Memory Optimization for Green Cloud with Genetic Algorithm. *IEEE Transactions on Computers*, Vol. 9340, 1-13. https://doi.org/10.1109/TC.2015.2409857

[9] Graham, P. and Nelson, B. (1995) A Hardware Genetic Algorithm for the Traveling Salesman Problem on Splash 2. Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Vol. 975, 352-361. https://doi.org/10.1007/3-540-60294-1_129

[10] Oliver, J. (2013) Hardware Implementation of Genetic Algorithms Using FPGA. *Journal of Chemical Information and Modeling*, **53**, 1689-1699. https://doi.org/10.1021/ci400128m

[11] Torquato, M.F. and Fernandes, M.A.C. (2019) High-Performance Parallel Implementation of Genetic Algorithm on FPGA. *Circuits, Systems, and Signal Processing*, **38**, 4014-4039. https://doi.org/10.1007/s00034-019-01037-w

[12] Bhattacharjya, R.K. (2013) Introduction to Genetic Algorithms. No. November, 1-90.

[13] Mirjalili, S. (2001) Genetic Algorithm. Vol. 13, No. 1, 89-98.

[14] Holland, J.H. (1992) Genetic Algorithms. *Scientific American*, **267**, 66-73. https://doi.org/10.1038/scientificamerican0792-66

[15] Hermawanto, D. (2013) Genetic Algorithm for Solving Simple Mathematical Equality Problem.

[16] Sastry, K., Goldberg, D. and Kendall, G. (2005) Genetic Algorithms. In: Burke, E.G. and Kendall, G., Eds., *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Springer, Berlin, 97-125. https://doi.org/10.1007/0-387-28356-0_4

[17] Ochi, L.S.D., Figueiredo, L.M.A. and Rosa, M.V. (1997) Design and Implementation of a Parallel Genetic Algorithm for the Travelling Purchaser Problem. https://doi.org/10.1145/331697.331750