FPGA on FPGA: Implementation of Fine-grained Parallel Genetic Algorithm on Field Programmable Gate Array

A. AL-Marakeby
Systems and Computers Engineering Dept.
Faculty of Engineering , Al-Azhar University,
Cairo, Egypt.

ABSTRACT
Many optimization problems have complex search space, which either increase the solving problem time or finish searching without obtaining the best solution. Genetic Algorithm (GA) is an optimization technique used in solving many practical problems in science, engineering, and business domains. Parallel Genetic Algorithm (PGA) has been widely used to increase speed of GA, especially after the spread of parallel platforms such as GPUs, FPGA, and Multi-Core Processors. In this paper, we introduce a type of PGA called Fine-grained Parallel Genetic Algorithm, which has the advantages of maintaining better population diversity, and inhibiting premature. Fine-grained PGA is implemented on Field Programmable Gate Array, and the system is used to solve the classical TSP problem. The results show the advantages of the Fine-grained PGA over sequential GA, and the advantages of Field Programmable Gate Array as a parallel platform.

Keywords
Parallel Genetic algorithm, FPGA, TSP, Parallel Processing.

1. INTRODUCTION
Genetic algorithms (GA) are abstract implementations of natural evolutionary processes used to solve search and optimization problems [10]. For complex problems, GA needs a long time to find solutions. Parallel GA comes into being and becomes a hit because it can reduce the time requirements dramatically [5]. Recently, there has been increased interest in parallel versions of the algorithms, in particular where the population has a spatial structure [10]. Most research in parallel processing depends on the multi-core processors, Graphical Processing Units (GPUs), or Field Programmable Gate Array (FPGA). Wei and Ying have used the multi-core processors to analyze the effect of Adaptive Migration on the performance of Distributed Parallel Genetic Algorithm [9]. Many researchers have applied Genetic Algorithm on Field Programmable Gate Array (FPGA) [4] [6]. Mohamed .et.al have optimized Parallel Genetic Algorithms for Graphical Processing Units (GPU)[7]. In this paper, the Field Programmable Gate Array (FPGA) is used to implement the Parallel Genetic Algorithm. The hardware design using FPGA has the advantage of implementing special processing cores for the specified task. The processor core is designed especially for implementing Genetic Algorithm tasks such as selection, crossover, mutation, and fitness function estimation. This design reduce the required resources and increase the speed of the system. The parallelization of Genetic Algorithm can be done in different ways. Master/Slave , Fine-grained , Coarse grained, and hybrid parallel genetic algorithms are the most common techniques used in parallelization of Genetic Algorithm[7][11]. In this research the Fine-grained Parallel Genetic Algorithm is used, which has the advantages of maintaining better population diversity, and inhibiting premature [3] [5]. This paper is organized as follow: Parallel Genetic Algorithm and different type of PGA are discussed in section 2. The Fine-grained Parallel Genetic Algorithm and the solution of Traveler Salesman Problem are given in section 3. Section 4 presents the hardware design for Fine-grained PGA. Section 5 gives the experiments and results. Finally, section 6 gives the conclusion.

2. PARALLEL GENETIC ALGORITHM
The parallel genetic algorithm (PGA) can combine the high speed of parallel computers parallelism with the GA inherent parallelism, which accelerates the search process of GA, maintains the diversity of population prevents premature, and solves such a complicated problem efficiently and effectively[11]. In PGA, there is always a selection-crossover-mutation cycle as in GAs, but you must meet new terms there. They are a deme, a migration and a topology [12]. A ***deme*** is one separated population, ***Migration*** means an exchange of individuals between the demes, and the topology represents the connections between nodes/demes. PGA can be classified into four classes:

2.1 Master/Slave PGA:
are typically used for problems involving expensive to compute fitness function, where the master node runs the entire algorithm while the slaves execute the fitness evaluations[7]. In Master/Slave PGA, the fitness function is evaluated in parallel using the slave nodes. The remaining tasks are performed serially on the master node[11]. In this way, a master process distributes the fitness evaluations among different enabled processors and collects the fitness evaluations in order to support the normal process of the genetic algorithm[3].

2.2 Coarse-grained PGA(CPGA)*:*
In a coarse-grained model, the GA population is divided into multiple subpopulations (or demes). Each subpopulation evolves independently, with only occasional exchanges of individuals between subpopulations. This isolation promotes diversity, thereby helping to prevent premature convergence across the population as a whole[10]. Fig.1 shows the different subpopulation and the migration between adjacent nodes.

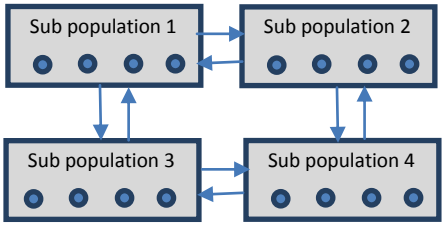


Fig.1 Coarse-grained PGA

2.3 Fine-grained PGA (FPGA):

In this model each processor is allocated only one individual. That is to say that each subpopulation is composed of only one individual and it communicates with another one, with 1 Hamming distance between them[11]. It has advantages of maintaining better population diversity, inhibiting premature, keeping the utmost parallelism, and therefore outperforms all other traditional genetic algorithms when dealing with high-dimensional variable spaces [5]. Fig.2 shows a connect 4 Fine-grained PGA.

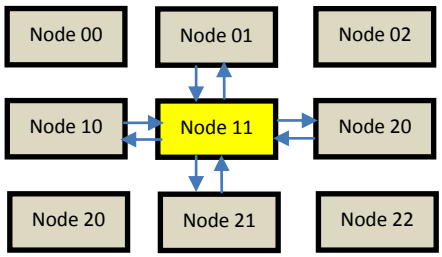


Fig.2 Fine-grained PGA

2.4 Hybrid PGA:

The hybrid model simply utilizes two or more of the Master/Slave, coarse-grained and fine-grained in a hierarchal method. Hybrid models are not the most common due to the need for additional new parameters to account for a more complex topology structure[7].

3. FINE-GRAINED PARALLEL GENTIC ALGORITHM.

Fine-grained PGA is more suitable for custom hardware design. While each node has a single individual, it is better to design a simple processor core to perform the required tasks. In this research we used Fine-grained PGA to solve the classical traveling salesman problem. The traveling salesman problem (TSP) is a typical example of a very hard combinatorial optimization problem. The problem is to find the shortest tour that passes through each vertex in a given graph exactly once[2]. The connection model used in this design is the connect 4 model. Each node (processor) is communicating with 4 adjacent nodes. The random initialization of individuals, fitness evaluation, selection, crossover , and mutation are performed for each node as shown in Algorithm.1[1]. Solving TSP with GA adds some constraints on individuals' values during initialization, crossover and mutation. Each city should appear only once in the chromosome and can't be repeated. So random initialization of values, or simple crossover by combining a part of chromosome 1 with chromosome 2 can repeat cities or delete others.

```
for each node do in parallel
        generate an individual randomly
end parallel do

while not stop_criterion_satisfied do
  for each node do in parallel
        evaluate the fitness of the individual
        get the fitness values of four neighbouring
        individuals

        find out the optimum fitness value

        get the neighbouring individual
        corresponding to optimum fitness

        uniform crossover with the local
        individual according to the crossover rate


        mutate the individual according to the
        mutation rate

   end parallel do
  test the stopping criteria
end while
```

Algorithm.1 Fine-grained PGA [1]

A swap operation is used during initialization or mutation to solve this problem. A chromosome is initialized with values contains all cities and 2 random values are selected and swapped with each others. This ensure that there is no repetition or removal of cities. During crossover, the moon crossover technique is used. Before the addition of the part of the second chromosome, a scanning is done for the first part to cancel repeated values. This is illustrated in fig.3[2]. The cost function used for the TSP is simple summation of the distances between cities. A table contains all locations of the cities is used to calculate this cost value. The cost function is given by equation(1).

$$\sum_{i=1}^{N} \sqrt{(X_i - X_{i-1})^2 + (Y_i - Y_{i-1})^2} \qquad (1)$$

Where Xi, Yi the coordinates of city (i).

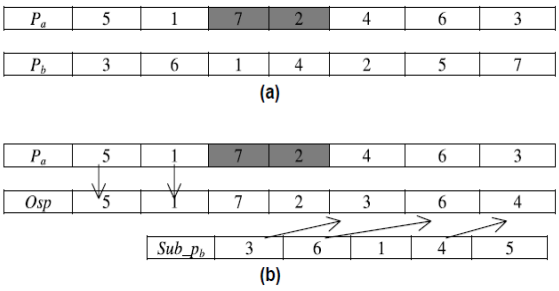


Fig.3 Illustration of the moon crossover[2]

4. HARDWARE DESIGN

Field Programmable Gate Array (FPGA) provide a cheap and efficient way to implement parallel algorithms. FPGAs maintain the performance of ASICs (Application Specific ICs) while avoiding their high development cost and inability to accommodate design modifications after production[8]. Custom processor cores are designed to run the Fine-grained PGA on the Field Programmable Gate Array. A single node or processor core in Fine-grained PGA performs all GA tasks in addition to the communication with neighbors. Fig.4 shows the different components and I/Os of a single processor core for Fine-grained PGA.

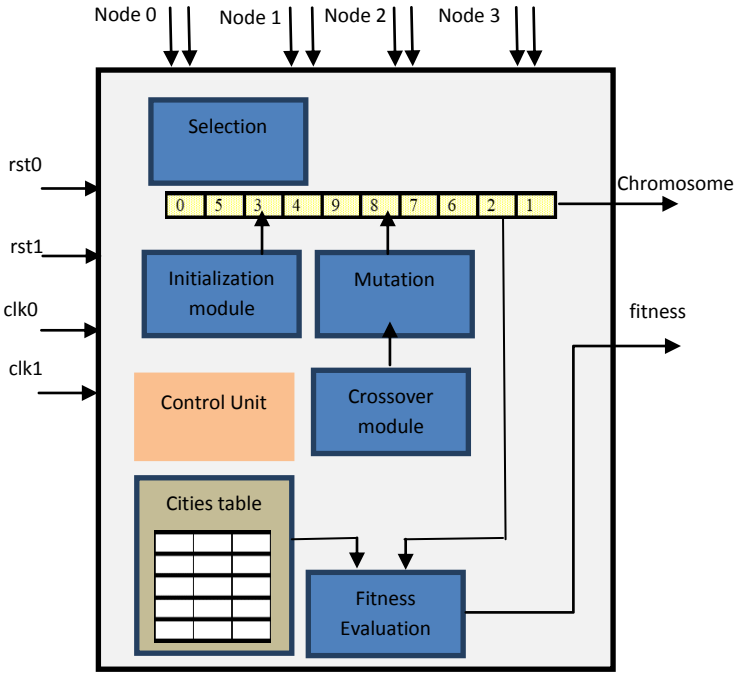


Fig.4 Genetic Algorithm processing core

Input/Output signals: the processor core has reset signals and clocks. One reset signal initialize the chromosome and start the GA cycle while the other starts the GA cycle without initializing the values of the chromosome. One clock is used as the system clock, and the other clock used for random number generation.The outputs of the processor core are the fitness value and the chromosome.

Neighbors connections: each processor core communicates with its four neighbors. This is shown in the figure as node 0 – node 3 inputs. Each node has two signals: the fitness value and the chromosome.

Initialization module: this module initialize the values of the chromosome randomly. The chromosome is initialized to a sequence of arranged cities, and then several random swap operations are applied to it. A simple counter with different clocks and random initial values is used as random number generator.

Fitness evaluation module: this module reads the current chromosome values and extracts the corresponding cities locations from the cities table. Addition, Subtraction , and square root operations are executed to calculate the fitness value. An integer square root module is used to avoid floating point calculations while the cities coordinates are large and the approximation is acceptable.

Selection module: the fitness value of the current chromosome with the fitness of the four neighbors are compared. The best two values are selected and the corresponding chromosomes are chosen.

Crossover module: The best two chromosomes are merged using a single point moon crossover described in section 3.

Mutation module: two addresses are chosen randomly and the values of these addresses are swapped. The new chromosome generated after mutation are stored instead of the old chromosome. The GA cycle is started again until the stop condition is achieved.

5. EXPERIMENTAL RESULTS

A PC-based system is developed to simulate the solution of TSP using Fine-grained PGA. The PC-based software is easily modified to test the effect of different parameters on the performance of the system. The parallelization of the system is simulated on the PC-based system using sequential execution. The implementation of the system on FPGA chip has been done after many experiments and parameter tuning on the PC-based version. The processor core shown in fig.4 is implemented on Altera DE2 board. A main controller unit is designed to organize the communication between processor cores, repeat the GA cycles, and store the best fitness and chromosome. The total required resources for a single core is 3% of the FPGA chip total logic elements. A cities table consists of 16 cities with different location is used to test the system. The chromosome consists of 16X4 bits register. The cost function and the chromosome are displayed on 7 segments display of the board. The system runs with 27 MHz clock. A single GA cycle needs (in average) of 440 time clocks for all selection, crossover, mutation and the main controller processing. The crossover has not a fixed time while it depends on the arrangements of cities inside the two best chromosomes. The speed of the system is in the range of 61,300 generations/s. a PC based program with the same parameters are running in average speed of 3300 generations/s. Fig. 5 shows this comparison.

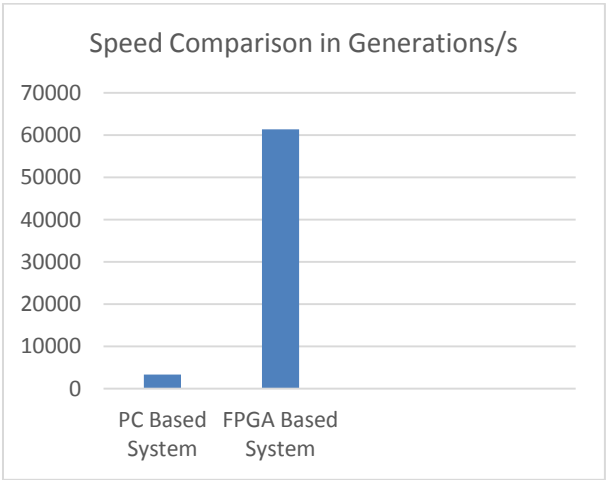
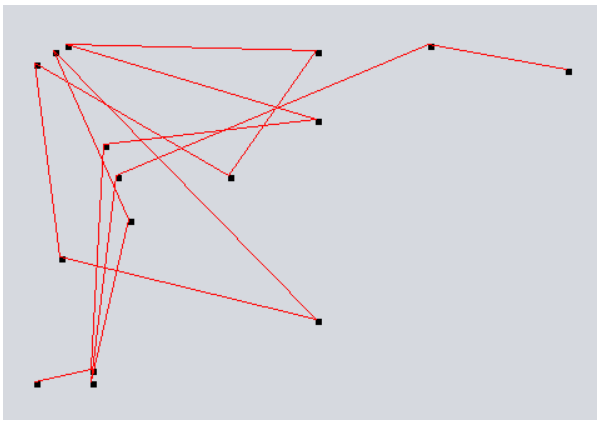
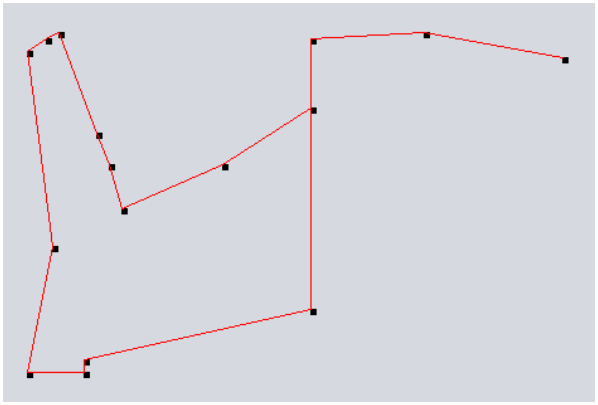


Fig.5 Speed Comparison between PC based system and FPGA based system

While the single core consumes only 3% of the chip resources, the multi-processor cores require additional resources due to the communication elements and the main controller . A system with 9 processor cores has consumed 63% of the chip logic elements. Fig.6 shows a randomly initialized, and a final solution for the TSP problem.



a. Randomly Initialized path for TSP



b. The Best path after running GA.

Fig.6 Solution of the travelling salesman problem

6. CONCLUSION

Field Programmable Gate Array (FPGA) is an efficient platform for implementing Parallel Genetic Algorithm. Fine-grained Parallel Genetic Algorithm increases the speed of processing GA and improve diversity of population. The designed processor core for Fine-grained PGA has a high performance in the terms of speed and consumption of FPGA chip resources. Many improvements can be added to this system to obtain better performance as a future work. A pipeline architecture can be used to increase the speed of the system. While each module of the processor core is running at a specific time, and it is idle otherwise, the pipeline architecture can utilize the module all the time. The communication between process cores are consuming chip resources, and special designed buses can organize the communication with less resources consumption.

7. REFERENCES

[1] A. Muhammad, A. Bargiela, G. King, Fine-Grained Parallel Genetic Algorithm: A Stochastic Optimisation Method, Proc. of 1st World Congress on Systems Simulation, p.199-203, Singapore, September 1997

[2] Chiung Moon , Jongsoo Kim , Gyunghyun Choi , Yoonho Seo , An efficient genetic algorithm for the traveling salesman problem with precedence constraints, European Journal of Operational Research -140 (2002).

[3] Giovanni Cantor , Jonatan Gómez, Maintaining Genetic Diversity in Fine-grained Parallel Genetic Algorithms by Combining Cellular Automata, Cambrian Explosions and Massive Extinctions, IEEE Congress on Evolutionary Computation , 2010

[4] H. Emam , M. A. Ashour , H. Fekry , A. M. Wahdan Introducing an FPGA based - genetic algorithms in the applications of blind signals separation, Proceedings of The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications 2003

[5] Jian-Ming Li, Xiao-Jing Wang, Rong-Sheng He, Zhong-Xian Chi , An Efficient Fine-grained Parallel Genetic Algorithm Based on GPU-Accelerated, International Conference on Network and Parallel Computing - 2007

[6] Miroslav Joler, Damir Malnar, and Silvio E. Barbin, Real-Time Performance Considerations of an FPGA-Embedded Genetic Algorithm for Self-Recovery of an Antenna Array, ICECom, Conference Proceedings, 2010

[7] Mohamed Wahib ,Asim Munawar, Masaharu Munetomo , Kiyoshi Akama, Optimization of Parallel Genetic Algorithms for nVidia GPUs, IEEE Congress on Evolutionary Computation (CEC), 2011

[8] M S Hamid and S Marshall, FPGA Realisation Of The Genetic Algorithm For The Design Of Grey-Scale Soft Morphological Filters, International Conference on Visual Information Engineering, 2003.

[9] Wei Li , Ying Huang ,A Distributed Parallel Genetic Algorithm Oriented Adaptive Migration Strategy, 8th International Conference on Natural Computation, 2012

[10] Xiaodong Li, Michael Kirley, The Effects of Varying Population Density in a Fine-,grained Parallel Genetic Algorithm, Proceedings of the Congress on Evolutionary Computation, 2002.

[11] XUE Shengjun , GUO Shaoyong , BAI Dongling, The Analysis and Research of Parallel Genetic Algorithm. Wireless Communications, Networking and Mobile Computing, 2008 .

[12] Zdenˇek Konfrˇst ,Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives , Proceedings of the 18th International Parallel and Distributed Processing Symposium 2004.