Journal of Theoretical and Applied Information Technology
31st January 2012. Vol. 35 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A DECADE OF PRODUCTIVE FPGA UTILIZATION WITH GENETIC ALGORITHMS

[1] BHARATHI NAVANEETHAKRISHNAN, [2] NEELAMEGAM P

[1]Asstt Prof., School of computing, SASTRA University, Thanjavur –613401

[2] Prof., School of EEE, SASTRA University, Thanjavur –613401

E-mail: bharathi_n@cse.sastra.edu , neelkeer@eie.sastra.edu

ABSTRACT

Genetic algorithms are one of the best ways to deal with the optimization problems. They are precisely suitable for mixed combinatorial problems. As genetic algorithms find the solution by producing more number of population generations based on selection, crossover, mutation etc.,it can be further improved by exploiting computation power of Field programmable gate arrays. The FPGAs are highly used reconfigurable hardware, which increase the speed of genetic algorithms. In this paper, the exploitation of FPGA to implement genetic algorithm based optimization problems in different frontiers for the past decade is studied.

Keywords: *Field Programmable Gate Array(FPGA), Genetic Algorithm(GA), Selection, Mutation, Crossover, fitness function, Chromosome mapping.*

1. INTRODUCTION

GA requires less information about the problem. GA works well on mixed discrete/continuous problems. Genetic Algorithms is useful in problem domains about which there is no sufficient knowledge for systematic solution. A genetic algorithm for an optimization problem consists of two major components. First, GA maintains a population of individual corresponds to a candidate solution and the population is a collection of such potential solutions. Two of the most common genetic algorithm implementations are 'simple' and 'steady state'. Simple genetic algorithm is a generational algorithm in which the entire population is replaced each generation.. In steady state genetic algorithm only a few individuals are replaced each 'generation'. This type of replacement is often referred to as overlapping populations. Genetic algorithms operate on a population of solutions. We must encode solutions to the given problem in a structure that can be stored in the computer. This object is a genome (or chromosome). The simple algorithm is given as

1. Represent the solution space as chromosome.
2. Generate the initial population(t) randomly.
3. Determine fitness of population(t) based on objective function.
4. Repeat

   Select parents from population(t)

   Perform crossover on parents creating population(t+1)

   Perform mutation of population(t+1)

   Determine fitness of population(t+1)

until best individual is good enough

The operators of GA are selection, crossover, mutation, replacement, fitness functions, scaling etc. The selection operator selects the individuals to perform crossover. Roulette wheel selection, tournament selection, rank selection, threshold selection are in use. Crossover is a genetic operator that combines two chromosomes to produce a new chromosome. One point crossover, two point crossover, arithmetic crossover, uniform crossover and heuristic crossover are some of the types of crossover in practice. Alteration of one or more gene values of the individual chromosome is called as mutation. Its types are flip bit, boundary, non-uniform, uniform and gaussian. Replacement operator is to identify the individuals to replace with new individuals. The various replacement schemes are replace-worst, replace-best, replace-parent, replace-random and replace most similar. Replace-worst and replace most similar are the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

frequently using replacement schemes. Fitness function is for ranking the individual chromosomes on the basis of how much fit it is to participate in successive genetic generations. The conversion of raw objective score to scaled fitness score is called scaling. The existing scaling techniques are linear scaling, sigma truncation scaling, and sharing. We code in high-level language like C and executed in Microprocessor, but it is much slower than the hardware implementation of FPGA. Even though the VHDL code is not optimized to exploit the maximum benefit of FPGA, it will be faster than their microprocessor counter part. This paper depicts 30 different applications with which GA operators are implemented in FPGA. This serves researchers in GAs to pick any one way for their application. Initially chromosome mapping of various applications is explained, followed by their selection, crossover and mutation techniques. Finally their performance analysis is discussed.

2. CHROMOSOME MAPPING

In GA, a binary string commonly represents an individual in the population. The mapping between solutions and binary strings is called a "chromosome mapping". D.V. Coury et. al.,[1] investigated the problem of estimating the frequency of a distorted electrical signal as a GA based optimization problem implemented in FPGA. A binary encoding is followed to map the Amplitude, frequency and phase of the electrical power system signal as $\psi = \{A, f, \theta\}$ [1]. Encoding represents the parameters as sequence of bits with 8 bit, 24 bit and 12 bit for amplitude, frequency and phase respectively. O.Hachour [7] deals with the intelligent path planning of Autonomous Mobile Robots(AMR) in an unknown environment, using hardware based genetic algorithms. In that work, the chromosomes are mapped as the path and positions are the genes. H. Emam et. al.,[3] used hardware genetic algorithm (HGA) for blind signal separation with filter coefficients mapped as chromosomes of length 64bit, 48 bit and with 16 bit fitness value. In that, the fitness function reflects the likeness between the output of the estimated model and the real output. Vavouras, M et. al.,[24] specified a fully functional prototype that supports variable population size, member and fitness value widths. Delbem, A.C et. al[2], Souza, S.A et.al[21][22] proposed FPGA based GAs for measuring the frequency deviation, as well as the voltage magnitude and phase angle of a sinusoid wave by representing the chromosome as 3 sequences of bits for amplitude, frequency and phase. Hamid M.S.et.al[8] applied GA in optimizing a grey-scale soft morphological filter implemented in FPGA with chromosome consist of 3 parts the filter value bits, the rank bits, and the soft morphological operation bits. Zhang X et.al [31] designed a hardware-based architecture to perform the Genetic Algorithm in a system, called FPGA-based Genetic Algorithm Kernel. In that, there are 4 memory components used in the Genetic Algorithm Kernel, and the Population Replacement is performed by transferring each individual in the Inter_Population Memory component and its fitness value in the Inter_Fitness Memory component to the Population Memory component and the Fitness Memory component respectively. Pedraza.C et.al [17] demonstrated a parallel genetic programming (PGP) Boolean synthesis implementation based on a cluster of FPGAs and represented the chromosomes as 2D Tree for balancing the load. 2D tree chromosome representation is sequential bits that are divided into several segments distributed over number of textures[30].

Jewajinda, Y.et.al[10] realized a new way of implementing GA as cellular compact genetic algorithm(CCGA). It employs on probability vectors by replacing the crossover and mutation operators with the probability model estimation. CCGA approach is to parallelize or divide a large problem into smaller tasks and to solve the task simultaneously using multiple genetic algorithms. Wang.J et.al[26] analyzed the finite resource optimization using FPGA resource list and task list with scheduling list as population producing FPGA resource utilization and task schedule. Lei.T.et.al [14] designed GA model and implemented in VHDL using a Xilinx XC2S100 FPGA by dividing the hardware into six modules: control state machine, storage module, selection module, crossover module, mutation module and random data generation module. Koo.J.H.et.al [12] used genetic algorithm to find optimal solution for image enhancement. In t hat, the chromosome mapping of 152 bits is divided as 3 level structured mapping with first level S1 (4 bits) represents number of filters for image enhancement, second level S2 (8 bits) represents type of filter, third level S3 (133 bits) represents parameter values for filters and 7 dummy bits to make it multiples of 8.

Wang.L.et.al [27] generates the key-sequence in AES using GA on FPGA Hardware. The 128 bits cipher key is divided into two parts, which are

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the parents. In two 64-bit parents, zero bit and first bit of second parent represent how many times the parents should rotate right and if the second bit of the parent is 1, the first parent should bitwise not, if the third bit of the second parent is 1, the second parent should bitwise not. Qu.L et.al.[18] implemented adaptive genetic algorithm (AGA) to optimize the parameters of PID controller and used altera FPGA 1P1C6F256C8 to implement PID controller. Skliarova.I.et.al [20] applied GA to optimize the traveling sales person problem in which the path is the chromosome and the cities are the genes. Esmaeilian-Marnani,A.et.al [4] suggested a new control method using GA and FPGA for polarization control used in fiber optic communication. Kher. S.et.al [11] proposed a dynamic crossover (DC) mechanism whose performance is tested by implementing in hardware (FPGA) with convergence rate and higher fitness as the performance metric. Since the updates are carried out along with the population generation, the convergence is faster.

Fernando.P.R.et.al[6] proposed a robust parameterized genetic algorithm IP core,that is readily synthesizable using standard FPGA design tools and that can be easily integrated into any design. Rubio-Solar.M.et.al [19] presented two implementations of a GA: a sequential one and a distributed one. The distributed implementation realized in ring topology model in which nodes exchange their best chromosomes after a determined number of generations. The encoded chromosome structure consists of a bit string, whose length depends on the problem size (number of CLBs and nets, size of the FPGA, etc). The chromosome consists of a set of coordinates (Xn, Yn), which represents each CLB position on the FPGA. Martin.P [16] shows how a GP system can be implemented in FPGA using a high level language to hardware compilation technique. Low.KS .et.al.,[15] focussed a problem with four dimensional inputs and three output clusters. The sepal length, sepal width, petal length, and petal width are measured in millimeters on fifty iris specimens from each of the three species, namely Iris setosa, Iris versicolor, and Iris virginica. The chromosome is formulated by concatenating the integer codes of the features. It uses 7 bits representation and yields an overall 28-bit chromosome. Wang J.et.al [25] mainly focused on demonstrating the possibility and efficiency of the FPGA implementation of evolvable characters recognizer and the benefits of self-adaptive mutation rate control scheme. To self-adapt the mutation rate, rather than fix it, the mutation rate control parameters are also encoded into the chromosome as additional genes.

Lau.W.S.et.al [13] proposed a hardware-assisted combinational logic circuit learning system. The GPP Logic Circuit Synthesizer consists of software Evolution Engine (EE), an FPGA-based logic circuit evaluation engine, and a Multi-Logic-Unit Processor (MLP). Ferlin.E.P.et.al.,[5] suggested reconfigurable parallel architecture for GA. The chromosomes are encoded with 25 genes (one gene for each logic cells) and each gene has 7 bits. Each gene is responsible for the configuration of a LC, and three fields compose it: address A, address B and the function. Vasicek..Z et.al.,[23] proposed accelerator for a given instance of Cartesian Genetic Programming (CGP) (i.e. a reconfigurable graph consisting of $u \times v$ programmable nodes)and it is implemented as a reconfigurable circuit on the FPGA. Its configuration is defined using a bit stream, which is stored in a configuration register implemented also in the FPGA. In order to evaluate a candidate chromosome, a controller has to store the chromosome into the configuration register and activate the fitness unit (FU).

Yang.M.et.al [29] resolves placement problem in which the chromosome structure is L1, L2,.,LN where L represents the configurable logic block number, *N* depends on *K*, the size of an FPGA. (ie) N = K x K. this approach utilizes the advantage of GA and fast convergence of simulated annealing (SA). Wong..CC.et.al., [28] suggested a fuzzy system design based on the concepts of GA to control three-wheeled mobile robot so that it can move to any direction and spin at any given rotating rate. Each individual of the population is represented by a parameter set to determine a fuzzy system.

3. SELECTION AND FITNESS FUCTION

Selection gives prefrence to better individuals, allowing them to pass on their genes to the next generation. The goodness of each individual depends on its fitness. [1][5][12][13][16] The selection process is based on tournament operator in which four individuals {a,b,c,d} are chosen randomly from the current population, parent1 is chosen from {a,b} and parent2 is chosen from {c,d} based on their fitness. They followed the sinusoidal model as cost function in which the 3 parameters {A, f, θ}, the input signal u and the time instant n are involved.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

O.Hachour [7] suggested the fitness function based on *n* is the code number of paths designed to be candidates of selection of two paths*; m* is the code number of all paths. Vavouras, M. et. al., [24] implemented the optimization of six different fitness functions on the XUPV2P platform. Delbem, A.C et. al.[2], Souza, S.A et.al [21][22] applied selection as tournament operator and the fitness function is based on the individual of the population *c*, the number of points *n*, measured signal for a point, signal calculated for the point using parameters *cf*, *cv* and *c*φ from chromosome *c* for measuring the frequency deviation. Wang.J et.al.,[26] employed roulette-wheel selection with schedule length is the fitness function. Lei.T.et.al [14] also implemented selection-processing module using fitness values stored in memory and roulette-wheel selection. Qu.L et.al.[18] selects the two individuals by the use of random number generator module which generates two random address signals for RAM1 and RAM2 respectively.

Skliarova.I.et.al [20] determined the fitness function of a tour corresponds to its length with roulette wheel selection. Fernando.P.R.et.al.,[6] uses proportionate selection scheme to select parents from current population. In that A threshold fitness value is computed from the sum of the fitnesses of all the individuals in the current population and a random number. A cumulative sum of the fitnesses of the individuals in the current population is computed and compared to the threshold fitness value. The individual whose fitness causes the cumulative fitness sum to exceed the scaled fitness threshold is selected as the parent. Low.KS .et.al [15] choosed roulette wheel selection scheme for choosing the parents for mating. Wang J.et.al., [25] describes a training set, that includes 16 test vectors from A to P.The fitness unit evaluates the circuits uploaded to the virtual reconfigurable circuit unit by reading its output vectors and comparing them against the expected output vectors. Yang.M.et.al [29] randomly selected based on the fitness.

4. CROSSOVER

Crossover represents mating between individuals. the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability. D.V.Coury et.al., [1] suggested the crossover process for representing the offspring from five possible values such as ψ of parent1, ψ of parent2, mean value between ψ of parent1and ψ of parent2, mean - δ and mean + δ where δ is the distance between parents. [5][6][7][14][24][26] followed single point or two-point crossover to reproduce better chromosome for next generation. Delbem, A.C et. al.[2], Souza, S.A et.al [21][22] realized the crossover based on five points including the parents and their mean values, choosing one value with probability 20%. Zhang X et.al [31] suggested the crossover operation according to the condition if the crossover probability is less than the operation threshold PC, the component performs one point crossover. Otherwise simply copies the two parents as offspring. Harik.G.et.al [9], Jewajinda, Y.et.al [10] followed probability model instead of crossover because successive application of crossver results in decorrelation of the population's genes and this state can be easily represented as probability vector which is faster way for crossover. Koo.J.H.et.al [12] adapted multipoint crossover and the crossover probability was changed dynamically based on linear interpolation.

Wang.L.et.al [27] determined the crossover place by adding 25 to bit positions 4 to 7 in both parents. Qu.L et.al.,.[18] calculates the crossover probability from the biggest crossover probability, the smallest crossover probability, the biggest generation, current generation, the average fitness value in current generation and the bigger fitness value among two crossover individual. Skliarova.I.et.al., [20] used Partially mapped crossover (PMX) with different crossover probabilities. Esmaeilian-Marnani,A.et.al [4] takes the half upper part of the population matrix as parents. Crossover between these parents makes new populations substituting for the half lower part of the population matrix. Kher. S.et.al [11] suggested a crossover mechanism that dynamically updates the number of crossover points and location of crossover on the basis of fitness values. Rubio-Solar.M.et.al [19] followed n-point crossover 12 points for 260 bits. Low.KS .et.al [15] performed crossover operation in parallel on all the parents. This will result in all *m* offsprings being produced instead of 2 offspring at a time. The crossover probability is taken as 40%. Lau.W.S.et.al [13] has crossover probability as 0.1. Yang.M.et.al., [29] choose a random cut point and divide the individual into two halves, left segment and right segment. A heuristic is used to avoid duplication of offspring. Wong.CC.et.al., [28] uses the crossover probability: pc = 0.9.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. MUTATION

Mutation is needed to avoid the premature and to keep the population assorted. It spawns random changes in the population. With some low probability, a portion of the new individuals will have some of their bits flipped. Its purpose is to maintain diversity within the population and inhibit premature convergence. The mutation is implemented as either adding or subtracting 1 from the parameters [1]. The path planning for mobile robots suggested mutation rate as 0.1 [7]. It is a bit is changed from 0 to 1 or 1 to 0. Vavouras, M et. al [24] used the mutation rarely with low probability generating a random number for each bit and flipping this bit only if the random number is less than or equal to the mutation probability. [2][21][22] followed technique in which a value of 1 is to add or subtract to each gene of the chromosome according to a mutation rate. Zhang X et.al [31] suggested the mutation operation according to the condition If the mutation probability is less than the operation threshold PM, one bit of both offspring is mutated, Otherwise not executed. Wang.J et.al.,[26] applied mutation based on mutation probability with randomly selecting task assignments for mutation. Lei.T.et.al [14] implemented mutation based on small probability and by choosing the ramdom data for anti-operation on the mutation bit. Koo.J.H.et.al [12] changed the mutation probability dynamically using linear interpolation. Wang.L.et.al [27] determined mutation bit by adding 1 to the values in bit positions from 8 to 13 in first parents and bit positions 14 to 18 in second parents. Qu.L et.al.[18] calculates the mutation probability from the biggest mutation probability, the smallest mutation probability, the biggest generation, current generation, the average fitness value in current generation and the fitness value of mutation individual. Skliarova.I.et.al [20] implements the mutation operator by randomly picking two cities in a path and reverses the order of the cities between them. Esmaeilian-Marnani,A.et.al [4] applied mutation on population matrix, by selecting random elements to be changed from 1 to 0 or from 0 to 1. Fernando.P.R.et.al.,[6] generated a 4-bit random number and compares it with the selected mutation threshold to decide if mutation should be performed. Rubio-Solar.M.et.al., [19] taken the mutation rate as 4/gene length. Wang J.et.al [25] performed 2 mutation operators (1) to the configuration bits strings and (2) to their additional genes that decide the mutation rates. The bit-mutation probability [13] is 0.002, [15] 4% and [28] 0.5. Ferlin.E.P .et.al.,[5] executes a point-mutation operation which complements a random bit with probability. Yang.M.et.al [29] followed pair-wise interchange, according to the probability of mutation rate.

6. PERFORMANCE ANALYSIS

D.V.Coury et. al., [1] approach is having an issue of sine function. It needs large number of multiplications, which increases the running time. Instead the performance is improved by use of look up table, storing 1024 sine function values, which are calculated in workstation in advance. The FPGA based GA implementation for path planning [7] of AMR is showing flexibility and can be changed on the fly to meet the different requirements of the users. The HGA approach for blind signal processing [3] is showing amazing real time performance and because of the need of 360 generations execution time is highly reduced. Implementation on a number of different high-end FPGAs [24] outperforms other reconfigurable systems with a speedup ranging from 1.2x to 96.5x. Delbem, A.C et. al.[2], Souza, S.A et.al[21][22] observed frequency estimation based on GAs is faster and has better immunity against noise, at a very small cost and fast enough to work in real time by implementing in FPGA. Hamid M.S.et.al [8] got the inference of good utilization of the device resources. And the optimization process was performed in a very short time. Zhang X et.al [31] proved that the developed genetic Algorithm kernel design for hardware is suitable for any kind of FPGA. Pedraza.C et.al [17] got the performance improvement of up to x500 increase in speedup over an HPC implementation.

Jewajinda, Y.et.al [10] observed the CCGA is more suitable for hardware-based applications where improved quality of search is needed. In addition, CCGA resolves a scalability issue of genetic algorithm with problem size by increasing network size. Wang.J et.al., [26] studied resource utilization for different FPGA and experiment results are tabulated. Lei.T.et.al [14] got the inference of 1000 times better performance if GA is implemented in FPGA with 20MHz than a workstation with 200MHz. Koo.J.H.et.al [12] applied various filters for image enhancement using genetic algorithm and the experimental results shows that proposed system has superior to impulse noise reduction, contrast enhancement, and blurring. Wang.L.et.al [27] proposed cryptographic system of AES based on reconfigurable hardware and genetic algorithm and it is implemented on Virtex-E FPGA,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

which is flexible, improves security level by generating key sequence in every round encryption and has high encryption speed. Qu.L et.al.[18] used AGA to optimize parameters of PID controller. The simulation results show that AGA improves global search capabilities, the precision of PID parameters optimization and has the merit of flexible design, self-tuning on line, high reliability, low development cycle and high speed.

Skliarova.I.et.al [20] performed the comparative analysis between software and hardware implementation. The software version was C++ and executed on a PentiumIII/800MHz/256MB running Windows2000. The hardware part was executed on an XCV812E FPGA with a clock frequency of 40 MHz. The experiment results show that the PMX crossover operator is faster in FPGA for 10-50 times than in software. Kher. S.et.al [11] tested the dynamic crossover (DC) against various static crossover methods. The experimental results show that for a linear and a nonlinear objective function, DC outperforms all static crossover mechanism. Fernando.P.R.et.al., [6] got the inference as the gate-level Verilog implementation of the GA core is advantageous because it can be directly used by commercial layout tools for chip layout generation. The availability of preset modes and scan-chain testability provides some basic fault tolerance to an ASIC designed using the proposed GA design. Rubio-Solar.M.et.al [19] obtained results show us that the main benefit of the distributed model is a large reduction of the execution time. Martin.P [16] observed the performance of the FPGA implementation is better than the equivalent software implementation without using parallel fitness evaluations. Low.KS.et.al [15] proposed an approach, which has been applied to the unsupervised clustering problems. The results have shown that the developed system is very flexible and scalable. Its speed advantage makes it a potential practical approach for real time data clustering.

Wang J.et.al [26] observed the hardware system could evolve the target 16 characters recognizer from scratch in relatively short time when compared to the same algorithm described in C language running on an AMD Athlon64 3200+ CPU. It is found in [13] that the speedup ratio increases with the number of tournaments taken in the evolution. In the experiments using two PEs, [5] gives a performance improvement of 54%, or a speedup of 1.85, reaching 92% of the ideal value is observed. In [23] a significant speedup of evolution was obtained in comparison with a highly optimized software implementation of CGP. In [29] experimental results show that the proposed GASA is effective in improving the quality of placement for the tested MCNC benchmark circuits. It consumes less CPU time than GA. In [28] experimental results indicated that the omni-directional mobile robot had a desirable full mobility and smooth motion.

7. CONCLUSION

GAs can be used where optimization is needed. Perhaps the greatest value of genetic algorithms is in the fact that they are based on the theory of ever-evolving optimization as a response to changing environments. To conclude genetic algorithm can be used for an optimization problem for which straight forward algorithms fails or no such highly suitable domain specific algorithms. GA can also employed to the problems in which little optimization worth a lot. It can also be combined with existing heuristics in order to realize the utmost benefit.

REFRENCES:

[1] Coury, D.V. Oleskovicz, M. Delbem, A.C.B. Simoes, E.V. Silva, T.V. de Carvalho, J.R. Barbosa, D.: "Frequency Relaying Based on Genetic Algorithm Using FPGAs". *15th International Conference on Intelligent System Applications to Power Systems*, pp 1 – 7 (2009)

[2] Delbem, A.C.B. Simoes, E.V. Souza, B.F. Oleskovicz, M. Souza, S.A. Coury, D.V.: "A Fast and Efficient Method for Frequency Deviation Measurement Based on Genetic Algorithms using a FPGA Approach", *Transmission & Distribution Conference and Exposition*, IEEE/PES, pp: 1 – 6. (2006)

[3] Emam, H. Ashour, M.A. Fekry, H. Wahdan, A.M.: "Introducing an FPGA based genetic algorithms in the applications of blind signals separation", *Proceedings of 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications*. pp:123 – 127. (2003)

[4] Esmaeilian-Marnani,A. Mamdoohi, G. Abas, A.F. Samsudin, K. Hidayat, A. Ibrahim, N.H. "Strategy on implementing genetic algorithm on FPGA for polarization control application", *International Conference on*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Computer and Communication Engineering, 11-12 May 2010, pp:1-5, (2010).

[5] Ferlin.E.P, Lopes.H.S, Erig Lima.C.R, Cichaczewski.E : "Reconfigurable parallel architecture for genetic algorithms: application to the synthesis of digital circuits", *Proceedings of the 3rd international conference on Reconfigurable computing: architectures, tools and applications*, LNCS Springer-Verlag Berlin, Heidelberg, pp: 326-336, (2007)

[6] Fernando.P.R, Katkoori.S, Keymeulen.D, Zebulum.R, Stoica.A.: "Customizable FPGA IP Core Implementation of a General-Purpose Genetic Algorithm Engine", *IEEE Transactions on Evolutionary Computation*, Volume: 14 Issue:1, pp: 133 – 149, (2010).

[7] Hachour.O,: "The Proposed Genetic FPGA Implementation For Path Planning of Autonomous Mobile Robot" , *International Journal of Circuits , Systems and Signal* Processing, vol2 , pp:151- 167, (2008).

[8] Hamid M.S. Marshall S.: "FPGA realisation of the genetic algorithm for the design of grey-scale soft morphological filters". *IEEE International Conference on Visual Information Engineering*, pp: 141 – 144. (2004).

[9] Harik.G, Lobo.F, Goldberg.D: "The compact Genetic Algorithm", *IEEE Transaction on Evolutionary Computation*, vol. 3, pp. 287-309, (1999).

[10] Jewajinda, Y. Chongstitvatana, P.: "FPGA Implementation of a Cellular Compact Genetic Algorithm". *Proceedings of the 2008 NASA/ESA Conference on Adaptive Hardware and Systems*, Noordwijk, 22-25 June 2008, IEEE Computer Society Washington, pp: 385 – 390, (2008)

[11] Kher. S, Ganesh.T.S, Ramesh.P, Somani.A.K,: "Greedy Dynamic Crossover Management in Hardware Accelerated Genetic Algorithm Implementations using FPGA", *11th International Conference on Computer Modelling and Simulation*, pp: 47 – 52 (2009)

[12] Koo.J.H, Kim.T.S, Dong.S.S, Lee.C.H,: "Development of FPGA based adaptive image enhancement filter system using genetic algorithms", *Proceedings of the 2002 Congress on Evolutionary Computation*, 12 - 17May 2002, pp: 1480 – 1485 (2002)

[13] Lau.W.S, Li.G, Lee.K.H, Leung.K.S, Cheang.S.M: "Multi-logic-unit processor: A combinational logic circuit evaluation engine for genetic parallel programming", *Proceedings of the 10th European Conference on Genetic Programming*, volume 4445 of LNCS. 167–177 (2005).

[14] Lei.T, Ming-cheng.Z, Jing-xia.W,: "The hardware implementation of a genetic algorithm model with FPGA", *Proceedings of IEEE International Conference on Field-Programmable Technology*, 16-18 Dec. 2002, pp: 374 – 377.(2002)

[15] Low.KS, Krishnan.V, Zhuang.H, Yau.WY : "On-Chip Genetic Algorithm Optimized Pulse Based RBF Neural Network for Unsupervised Clustering Problem", *Advances in Natural Computation LNCS series*, Springer Berlin / Heidelberg, Volume: 4222, pp:851-860. (2006)

[16] Martin.P: "A Hardware Implementation of a Genetic Programming System Using FPGAs and Handel-C", *Genetic Programming and Evolvable Machines*, Vol: 2, Issue: 4 pp: 317-343, (2001)

[17] Pedraza.C, Castillo.J, Martínez.I.J, Huerta.P, Bosque.L.J, Cano.J: "Genetic Algorithm for Boolean minimization in an FPGA cluster", *J Supercomput*, pp:1-9, (2010).

[18] Qu.L, Huang.y, Ling.L : "Design of Intelligent PID Controller Based on Adaptive Genetic Algorithm and Implementation of FPGA", *Proceedings of the 5th international symposium on Neural Networks: Advances in Neural Networks part II*, Springer-Verlag Berlin, Heidelberg, pp: 542 – 551 (2008)

[19] Rubio-Solar.M, Vega-Rodriguez.M.A, Perez.J.M.S, Gomez-Iglesias.A, Cardenas-Montes.M, :"A FPGA Optimization Tool Based on a Multi-island Genetic Algorithm Distributed over Grid Environments", *Eighth IEEE International Symposium on Cluster Computing and the Grid*, pp.65-72 (2008)

[20] Skliarova.I, Ferrari.A.B,: "FPGA-based Implementation of Genetic Algorithm for the Traveling Salesman Problem and its Industrial Application", *Proceedings of the 15th international conference on IEA/AIE '02: developments in applied artificial intelligence*, Springer-Verlag London, pp: 77-87. (2002).

[21] Souza, S.A. Oleskovicz, M. Coury, D.V. Silva, T.V. Delbem, A. Simoes, E.V.: "FPGA implementation of Genetic Algorithms for frequency estimation in power systems". *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, 2008 IEEE, pp: 1 – 6. (2008)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[22] Souza S.A, Oleskovicz.M, Coury.D.V, Silva.T.V, Delbem.A.C.B, Simões.E.V,: "An efficient frequency estimation methodology using genetic algorithms in FPGA". *Proc. 2007 The 33rd Annual Conf.of the IEEE Industry Electronics Society*, pp. 2020-2025 (2007)

[23] Vasicek.Z and Sekanina.L: "Hardware Accelerators for Cartesian Genetic Programming", *Proceedings of the 11th European conference on Genetic programming*, LNCS Springer-Verlag Berlin, Heidelberg, pp:230-241(2008)

[24] Vavouras, M. Papadimitriou, K. Papaefstathiou, I.: "High-speed FPGA-based Implementations of a Genetic Algorithm". *IEEE International symposium on Systems, Architectures, Modeling, and Simulation,* SAMOS '09, pp: 9 – 16. (2009).

[25] Wang J, Piao.C.H, Lee.C.H,: "FPGA Implementation of Evolvable Characters Recognizer with Self-adaptive Mutation Rates", *Adaptive and Natural Computing Algorithms LNCS*, Springer Berlin / Heidelberg, Vol 4431/2007, pp: 286-295, DOI: 10.1007/978-3-540-71618-1_32. (2007)

[26] Wang.J, Sin Ming Loo.S: "Case study of finite resource optimization in FPGA using genetic algorithm", *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, ACM New York,pp: 989-992 (2009)

[27] Wang.L, Wang.Y, Yao.R, Zhang.Z,: "Hardware Implementation of AES Based on Genetic Algorithm", *Advances in Natural Computation LNCS,* Springer Berlin / Heidelberg, Vol 4222/2006, 904-907, (2006).

[28] Wong.CC, Lin.YH, Lee.SA, Tsai.CH,:"GA-based Fuzzy System Design in FPGA for an Omni-directional Mobile Robot", *Journal of Intelligent and Robotic Systems*, Volume 44, pp: 327 – 347, (2005)

[29] Yang.M, Almaini. A. E. A, Wang.P,: "Fpga Placement Optimization By Two-Step Unified Genetic Algorithm And Simulated Annealing Algorithm", *Journal of Electronics-China*, Vol. 23, Number 4, pp: 632-636, DOI: 10.1007/s11767-005-0198-3 (2006)

[30] Yu Q, Chen C, Pan C.: "Parallel genetic algorithms on programmable graphics hardware". *Lecture notes in computer science*. Springer, Berlin, pp 1051–1059, DOI: 10.1007/11539902_134 (2006)

[31] Zhang X Shi C and Hui F.: "FPGA-Based Genetic Algorithm Kernel Design". *Proceedings of the 7th international conference on Evolvable systems: from biology to hardware*, LNCS 4684 -Springer-Verlag Berlin Heidelberg, pp. 426–432 (2007).