# New Ant Colony Algorithm Method based on Mutation for FPGA Placement Problem

Setareh Shafaghi[1] Fardad Farokhi[2], Reza Sabbaghi-Nadooshan[3]

[1]Electrical Engineering Department, Central Tehran Branch, Islamic Azad University, Email: strshafaghi@yahoo.com
[2]Electrical Engineering Department, Central Tehran Branch, Islamic Azad University, Email: fardad.farokhi@gmail.com
[3]Electrical Engineering Department, Central Tehran Branch, Islamic Azad University, Email: r_sabbaghi@iauctb.ac.ir

**Abstract**

Many real world problems can be modelled as an optimization problem. Evolutionary algorithms are used to solve these problems. Ant colony algorithm is a class of evolutionary algorithms that have been inspired of some specific ants looking for food in the nature. These ants leave trail pheromone on the ground to mark good ways that can be followed by other members of the group. Ant colony optimization uses a similar mechanism to solve the optimization problem. Usually the main difficulties of evolutionary algorithm for solving the optimization problem are: early convergence, loss of population diversity, and placing in a local minimum .Therefore, it needs the way that preserves the variation and tries to avoid trapping in local minimum. In this paper by combining ant colony algorithm and mutation hybrid algorithms that leads to the better solution for optimization of FPGA (Field Programmable Gate Array) placement problem is made. They are different types of swarm intelligence algorithm. After designing the algorithm, its parameters tuning have been done by solving several problems, and then the proposed methods have been compared with the other approaches. The results show that in most problems, the proposed hybrid method is able to obtain better solutions and makes fewer errors.

Keywords: Ant colony algorithm, Mutation, FPGA placement, Optimization.



## 1. Introduction

FPGA is a programmable integrated circuit (IC) which includes a large number of functional blocks and programmable interconnection networks that make it easy to implement complex circuits. Using FPGA to implement an integrated circuit has increased enormously in recent years. The primary advantages of FPGA are generating rapid prototypes in a short time and low cost and ease of designing that is a useful tool for digital circuit design by designers due to the ability of reconfiguration [1, 2]. FPGA is just like a computer program that can be changed by changing its codes. The fundamental and main idea of FPGA is simple. The FPGA is a particular integrated circuit that designer can re-configure a large number of schema, without any need to the more production costs [3].

The most important part of each FPGA is CLB (Configurable Logical Block). The other important part is programmable connections which connect different parts of FPGA and allows them to communicate with each other. These connections are programmable and users can define any part of their connection. An island style is general structure of the FPGA, which consists of three main parts described as bellow:

1-CLB is base logic blocks that is used to implement logic functions.

2-IOB (input–output block) is used to connect FPGA with external hardware and desired signal can go inside or outside of FPGA through these blocks.

3-Switch block and communication resources that are used to connect to the routing channel [4, 5].The placement Problem of the FPGA is always a limiting factor for the overall performance. In the placement phase, logic blocks in the network must be placed in

specific positions on the FPGA, An example of the FPGA placement process is shown in Figure 1.

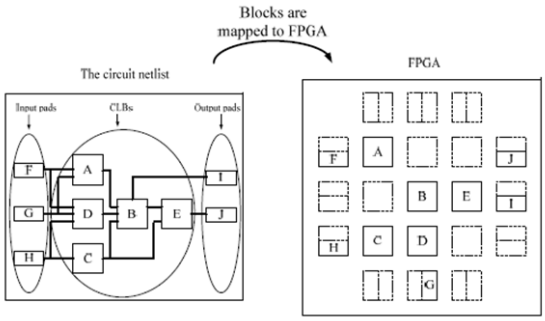


Figure 1. The Process of FPGA Placement [6]

FPGA Placement requires a net list of logic blocks that includes CLB layers and their connections. The result of placement is the physical transmission of all the blocks on the FPGA in order to optimize one or more cost functions [5].Three common optimization criteria are usually considered for placement. Time-length driven placement maximizes loop speed, wire-length driven placement tries to locate blocks close to each other due to the length of the wire and reduces wire and routability- driven placement and balances wire congestion on FPGA [7, 5].

Ant colony algorithm can be used to solve both static and dynamic combinatorial optimization problems. For Static problems the characteristics of the problem are given once, while the problem is being solved the defined problem remained unchanged. One example of this type of problems is the travelling salesman problem, where city location and their relative distances are parts of the problem definition and does not change during the runtime. In contrast the dynamic problems are the cases that the problem characteristic is changed during the run-time and the optimization algorithms are able to adapt more quickly with the changing environment [9, 8]. One example of such problems is the placement problem in FPGA. Because in this case, the logical blocks and their relative distances are defined as parts of the problem and are changed during the run-time.

Up to now, optimal placement in FPGA has been done by various methods including evolutionary algorithms such as ACO. In 2007, Wenayo et al. [6] used the ACO algorithm for FPGA Placement. They presented a new algorithm based on ant colony optimization. It was a new meta-heuristic algorithm that formed by positive feedback mechanisms and random policy decisions based on collective intelligence. Then, they had a comprehensive comparison with SA (simulated annealing) and GA (Genetic Algorithm) algorithm, and SA and GA combination, respectively. Results show that these placement algorithms can catch promising performance and is a potential approach for FPGA placement. In 2009, Wanget al. [10] used ant colony method in order to optimize the placement of symmetrical FPGA. They also considered the routing density by introducing a density factor. This algorithm also obtain promising performance, in terms of routing channel density In 2011, Chopra [4] used ACO algorithm in order to design and routing of FPGA structure and decreased the CPU time consumption. Result show that this is a proper algorithm for complex circuits. But ACO algorithm has long convergence time; to solve this problem we proposed a new ant colony algorithm that uses mutation for rapid convergence and so find optimized solution with minimum error in short time.

The rest of this paper is organized as follows, In Section 2, ACO algorithm and its framework is introduced. In section3 the proposed algorithm and its methodology is described. Then, in Section 4, simulation results are presented and finally a comparison between the obtained results is done and conclusions are given in Section 5.

## 2. ACO meta-heuristic algorithm

The type of algorithms that often called meta-heuristic, are not designed for a specific problem; these algorithms provide a general approach for organizing the search in the solution space. Examples of known meta-heuristic algorithms are the evolutionary algorithms like simulated annealing, Tabu search, and etc. Meta-heuristic algorithms can obtain optimal solutions during some iteration in two steps:

1. A set of solutions are made using the probabilistic parameters model.
2. Selected solutions are evolved to be better solutions in order to decrease the cost function.

A good meta-heuristic algorithm is defined with inspiring of the exploratory behaviour of ants to find food. This algorithm called ACO that act successfully to solve dynamic problems. The main innovation of ACO includes a new types of the two mentioned steps described above.

1. A set of artificial ants are considered for a structure, called the construction graph. They are creating new solutions by using local information, called pheromones. Then, they are stored in the construction graph.
2. When ants make the solutions; they are using the data collected during pheromone values construction phase [11, 12].

### *2.1. The application of Ant colony algorithm*

Ant colony algorithm was proposed by Dorigo et al [12]. They introduced a stochastic optimization algorithm as a method, inspired by nature to solve

the optimization problem in early 1990[4]. The algorithm is inspired by the behaviour of ants looking for food, and it was named ant colony optimization. This is a simple but efficient method to solve routing problem for finding the shortest path. Ant colony optimization is meta-heuristic approach in which artificial ant colonies contributes to find a better solution for the defined problem. Some of the applications of ant colony algorithm are routing problem in telecommunication networks, routing the wired network[13], graph coloring, sequential ordering, vehicle routing[12], and solving travel salesman problem[12,13]which is a symbolic presentation for the optimal design and placement of integrated circuits[14] and optimization of FPGA placement problem[6,10].

### *2.2. Principles of Ant Colony Optimization*

In ant colony algorithm, each agent is an artificial ant. When ants are looking for food outside the colony, at first, their path will be chosen randomly. During movement each ant leaves amount of pheromone on the ground and thus it specified crossed path. Of course, the pheromone is evaporated quickly, but in the short term as a track it remains on the ground. In other words, ants during the movement selected a path with more pheromone and leave more pheromone on the way that will reinforce the path. Random choosing of ways is led to find the optimal path. Thus, when an ant is attracted to the optimum route, it puts some pheromone on the path. This increases the amount of pheromone and attracts more ants. Other ants also add path pheromone intensity. This process is repeated until optimal route is selected. Thus, the probability of selecting a path that are chosen by previous ants, will be increased. But over the path, there is the pheromone evaporation. Pheromone evaporation deletes the information of abandoned route. Ant algorithms have memory; it means that the selected ants' path will remain after several iterations; because pheromones evaporation on the path needs several iterations to be evaporated completely [13, 14]. Figure 2 shows the ant's mechanism for finding the shortest path in which white point is the food source and black points are the nests. At first all ants search around nest randomly for food source and finally they find the shortest path.

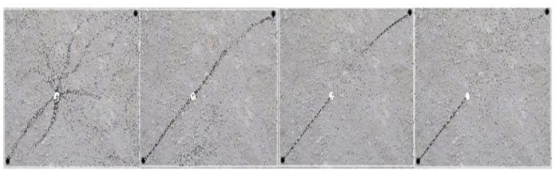

Fig.2. The shortest path finding mechanism used by ants

## 3. The proposed algorithms and method

The aim of these algorithms is finding the most proper position for each CLB to be placed on FPGA, in order to minimize the cost function as far as possible.

### *3.1. Mathematical description of FPGA problem*

A set of modules as M= $\{M_1, M_2, \ldots, M_n\}$ and a set of net lists as N = $\{ N_1, N_2 , \ldots , N_n\}$ is considered. We associate a set of modules $M_i \in M$ to each netlists $N_{Mi}$ as $N_{Mi} \subseteq N$. Similarlya net lists $N_i \in N$ and a set of modules$M_{Ni}$ as $M_{N\ i} \subseteq M$ is associated. Also a set of locations is considered by $L = \{L_1, L_2, ..., L_k\}$ that $k \geq n$. The aim of solving placement problem is specifying a location $L_i$ for each $M_i \in M$ so that cost function would be optimized. Usually each module is considered as a point and $M_i$ is assigned to a location $L_i$ and its position is described by coordinates $(x_i,y_i)$.Placement algorithm should decrease a defined cost function in design process which could be for instance the wire length cost function[15].

### *3.2. Conventional ant colony algorithm*

In general, conventional ant colony algorithm consists of the following sections:

#### *3.2.1. Generating the initial population*

In order to implement the first population, m artificial ants create several parallel paths. At this stage, these initial solutions are created randomly.

#### *3.2.2. Creating a route due to probability rule*

Each ant is placed on randomly selected points and as a solution named *i*. It is assumed that k ant are located in I position and wants to go to *j* position. Therefore; the next position to move likely is selected according to a probability rule. Since the initial pheromone level is the same for all paths; probability is the same for all solutions which may take by the ants. But in each iteration of the algorithm, due to the evaporation and deposition of pheromone for different solutions, the probability of selection will vary. Equation 1 shows the chosen probability for ant k to go from i position to j position [16].

$$P_{ij}^{k}(t) = \frac{\tau_{ij}^{\alpha}(t)}{\sum_{L} \tau_{il}^{\alpha}(t)} \qquad (1)$$

In the above formula, α factor shows the effect of the pheromone. So, the choice is depend on the pheromones.

*3.2.3 Fitness evaluation and fitting ant selection*

After each ant passes its path completely, the solution is evaluated by a cost function that is given in Equation 2.At this stage the fitness of each solution was measured. Some of the best solutions are selected as appropriate ant for deposition.

$$\begin{aligned}&\text{cost function}\\&= \max(\Delta x, \Delta y)\\&\times \sum_{i=1}^{n} (|x_{1,i} - x_{2,i})| + |(y_{1,i} - y_{2,i})|\\&\times \max(w_x, w_y)\end{aligned} \quad (2)$$

*3.2.4. Pheromone updating and generating new solutions*

After evaluating solutions by equation (2), pheromone will be updated. In general; updating the pheromone level is done in two steps: at first, some percentage of the pheromone in all paths will be evaporated. ρ is a parameter that called pheromone evaporation factor. At the second step, pheromones are needed for deposition. Here for two good ants that have the less error, all pheromone levels are deposited. One of these two ants is the best ant in the current iteration and the second is the best ant up to now. Updating the pheromone τ is done using equation (3).Then the same original number of ant is created as new solutions and these new solutions are evaluated and the loop is repeated in order to reach the stop condition.

$$\tau_{new} = (1 - \rho)\tau_{old} \quad (3)$$

Finally the ant colony algorithm flowchart is presented in Figure 3and its steps are discussed briefly as follows:

1.Initial Ant production: At this stage the ant colony algorithm is initialized. Ants are placed on the primary positions and the pheromones are initialized.

2-FitnessEvaluation: the fitness of all ants is evaluated based on the cost function. Then pheromones are added to the specific route of the ants.

3-Ant distribution: at last the ants are distributed based on pheromone levels

4-Stopping criteria: The process continues up to the maximum number of ants. All passed routes should be evaluated by the ants in iteration and if a better solution is found that solution must be replaced in the solution lists. The best solution is selected among all iterations as the final solution.

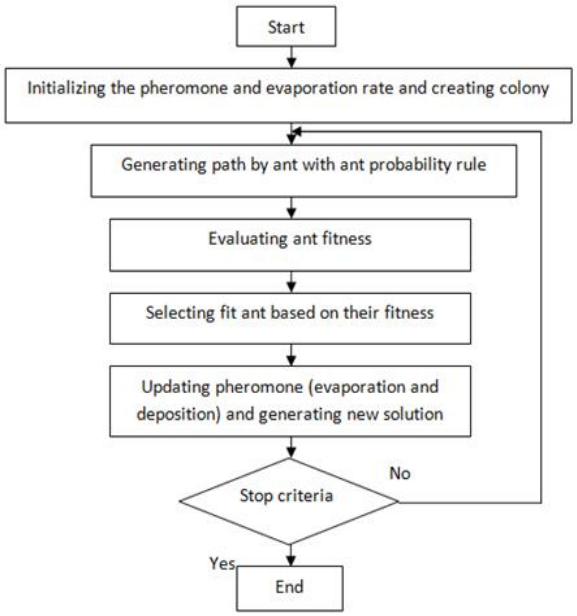


Fig.3. ant colony algorithm Flow chart

*3.3. Ant colony algorithm beside mutation*

The ant colony algorithm is able to achieve the optimal solution for solving the placement problem. But in order to improve the quality of the solutions obtained by this algorithm, using parallel algorithms and combination of them is necessary. In this paper, for improving the ant colony, make population diversity, avoiding of getting algorithm into local minima and the absence of early convergence, a new heuristic ant colony algorithm beside mutation and ant colony algorithm with mutation is proposed. These algorithms' steps are completely similar to the algorithm of the previous section; just the only difference is at the end of the algorithm in pheromone updates step that produced new solutions that leads to the production of better random answers. In ant colony algorithm with mutation, production of new solution is such that after m new solution was obtained by the probability rule, for all m solution, mutation will be occurred, like the type of the mutation that is considered in the genetic algorithm after the crossover step. This method leads to the solution of the stochastic that consequently prevents from early convergence of ant colony, and placing the algorithm on the local optimum.

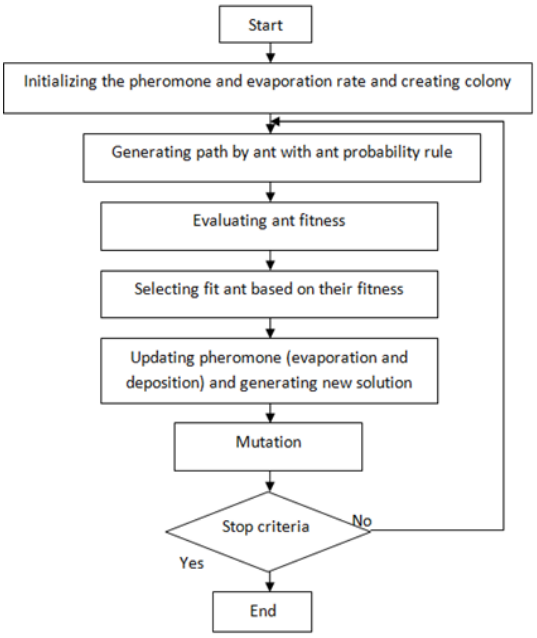


Figure 4: Flow chart of ant colony algorithm beside mutation

### *3.4. Ant colony algorithm with mutation*

Producing new solutions in the ant colony algorithm with mutation are based on the following steps:

A) New solutions generation using probability rule: Here from m initial solution m1 number of them (which m1is obtained from equation 4) is formed the new solutions using probability rule of selection in ant colony algorithm.

$$m1 = (1 - \gamma) \times m \quad (4)$$

B) Generating new solutions using mutation: Among m initial solutions, new solutions up to m2 numbers are created using mutation (which m2 is obtained from equation 5). Such that based on the best solution that is constructed from the first iteration up to previous iteration, m2 number solution is generated by mutation similar to that best solution.

$$m2 = (\gamma) \times m \quad (5)$$

It is obvious that $m1 + m2 = m$ that is the total number of solutions. $\gamma$ is a parameter that regulates the number of ants for both above equation that is desired to be $0 < \gamma < 1$

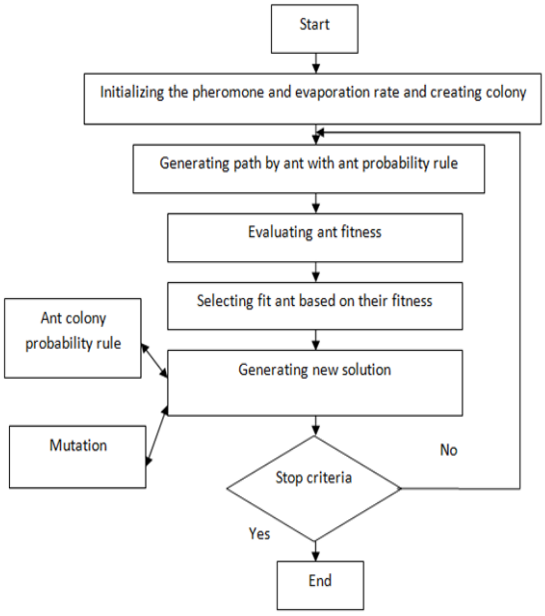


Figure 5.Ant colony algorithm with mutation Flow chart

## 4. Simulation results

### *4.1. The parameters tuning*

Simulations were done using MATLAB software on a computer with Intel (R) Core processor (TM) i3 with 2.27 Ghz and 3.00 GBRAM. All the proposed algorithms are done for a series of test FPGA circuits for better bench marking; those are Microelectronics Center of North Carolina (MCNC) benchmarks, which consist of optimized test circuit that is collected from industry and university. These test circuits include a standard library which represents circuits with various complexities from simple to advanced industrial circuits [17, 18].

Simulations are done on 5 benchmarks with different characteristics. Table 1 shows the structure of the FPGA benchmarks that are used in the simulations.

Table.1
FPGA benchmark circuits parameters

| circuit | function | Number of logic block | size |
|---|---|---|---|
| B9 | Logic | 61 | 10×10 |
| Cc | Logic | 37 | 12×12 |
| COMP | Logic | 52 | 10×10 |
| F51m | Arithmetic | 47 | 10×10 |
| Pcler8 | Logic | 55 | 10×10 |

Table 2 .shows the initialized parameter values of the algorithm in order to find the best answer.

Table 2. ACO parameters settings Table

| parameter | value |
|---|---|
| Population size | 100 |
| Evaporation rate | 0.5 |
| Deposition rate | 1 |
| α | 1 |

In both ant colony algorithms beside mutation and ant colony algorithm with mutation, mutation rate is considered as 0.5. γ is set as 0.3 for ant colony algorithm with mutation.

### *4.2. Simulation results with the proposed algorithm*

Table 3 shows CPU time for running each algorithm. It can be seen that the running time of ant colony algorithm beside mutation and ant colony algorithm with mutation is shorter than the conventional ant colony algorithm.

Table.3
CPU time of each algorithm.

| Circuit | ACO | ACO-MUT | ACO+MUT | Average |
|---|---|---|---|---|
| B9 | 407.93 | 334.81 | 336.89 | 363.21 |
| Cc | 251.82 | 195.09 | 195.85 | 214.25 |
| COMP | 322.48 | 266.23 | 273.74 | 287.49 |
| F51m | 281.78 | 228.59 | 246.81 | 252.39 |
| Peler8 | 337.78 | 275.46 | 296.69 | 303.31 |

Table 4 shows initial wire length and final wire length and also wire length reduction percentage. Maximum wire length reduction is obtained after the implementation of ACO + mut algorithms for all of test circuits. Cc circuit has the maximum wire length reduction. Clearly, to reduce the length of the wires, logic block should be located nearby. Figure 6 shows position of configurable logic blocks on Cc benchmark, from left to right, before and after applying the algorithm, respectively.

Table.4
Wire length reduction

| circuit | ACO | | | ACO-mut | | | ACO+mut | | |
|---|---|---|---|---|---|---|---|---|---|
| | Initial wire length | Final wire length | Reduction percentage | Initial wire length | Final wire length | Reduction percentage | Initial wire length | Final wire length | Reduction percentage |
| B9 | 3207 | 2685 | 16.28% | 3207 | 2217 | 30.87% | 3207 | 2186 | 31.84% |
| Cc | 1493 | 1113 | 25.45% | 1493 | 696 | 53.38% | 1493 | 681 | 54.39% |
| Comp | 2238 | 1787 | 20.15% | 2238 | 1438 | 35.75% | 2238 | 1374 | 38.61% |
| F51m | 2024 | 1685 | 16.75% | 2024 | 1217 | 39.87% | 2024 | 1174 | 41.97% |
| Peler8 | 2729 | 2198 | 19.46% | 2729 | 1835 | 32.76% | 2729 | 1789 | 34.45% |

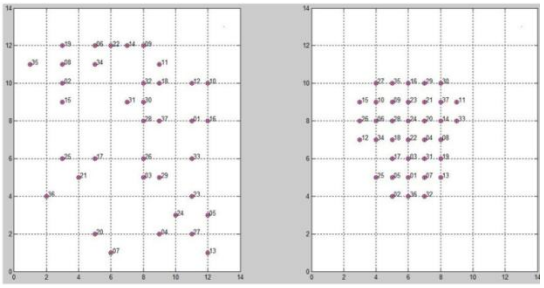

Fig.6. Cc logic blocks position, from left to right, before and after applying the algorithm, respectively

Table 5 shows error reduction percentage in the cost function for running of each algorithm.

Table.5
Error reduction percentage in the cost function

| Circuit | ACO | ACO-MUT | ACO+MUT | Average |
|---|---|---|---|---|
| B9 | 42.12% | 46.16% | 46.22% | 44.83% |
| Cc | 54.54% | 68.08% | 74.17% | 65.60% |
| COMP | 45.78% | 45.95% | 50.14% | 47.29% |
| F51m | 53.94% | 54.83% | 56.96% | 55.24% |
| Peler8 | 43.41% | 44.64% | 55.31% | 47.78% |

Fig.7 shows error reduction in the cost function as described in Section 3.2.3 from left to right after applying conventional colony algorithm, ant colony algorithm beside mutation and ant colony algorithm with mutation on the circuit Cc that has the best impact, respectively.

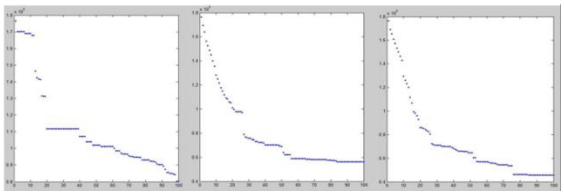

Fig.7. Reduction of error rate for Cc circuit

Fig.8 shows error reduction of the cost function as described in Section 3.2.3 on the circuit B9 that has the worst impact. From left to right after applying conventional colony algorithm, ant colony algorithm beside mutation and ant colony algorithm with mutation, respectively.

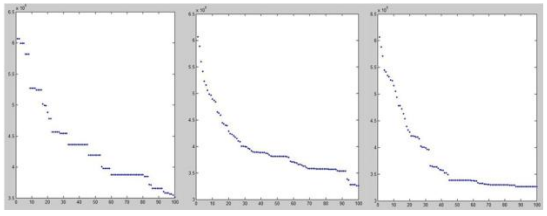

Fig.8. Reduction of error rate for circuit B9

One useful type of FPGA is Xilinx XC4000. The XC4000 families of Field-Programmable Gate Arrays (FPGAs) provide the benefits of custom

CMOS VLSI, while avoiding the initial cost, time delay and inherent risk of a conventional masked gate array XC4000-family devices have generous routing resources to accommodate the most complex interconnect patterns. This FPGA includes a 14×14 matrix. Placement of a BCD counter (X74_168) 4-bit on it has been investigated by various algorithms [19]. The results are as follows:

Fig.9 shows the initial location of BCD counter 4-bit logical block on the FPGA.

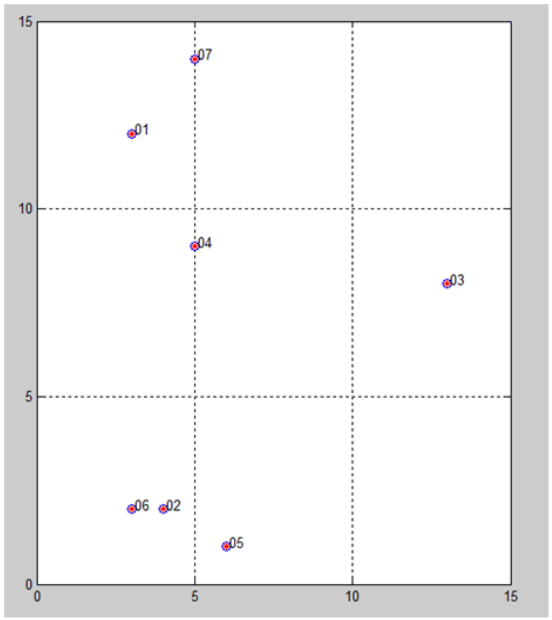


Figure 9. Initial position of the BCD counter logic blocks

Figures 10, 11 and 12 show final position of BCD counter block and the error rate reduction after 200 iterations, after applying conventional ACO algorithm and ant colony algorithm beside mutation and ant colony with mutation, respectively.

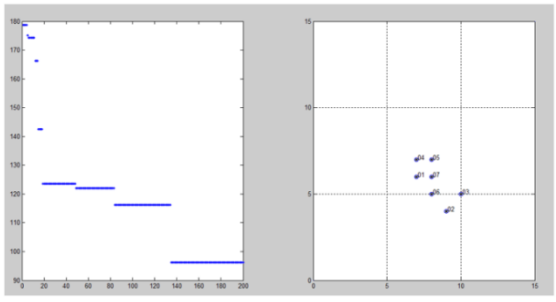

Fig.10. Final position of BCD counter logic block and error reduction after applying the conventional ACO algorithm.

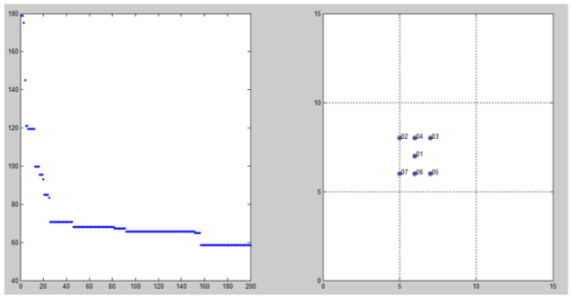

Fig.11. Final position of BCD counter logic block and error reduction after applying the conventional ACO beside mutation algorithm.

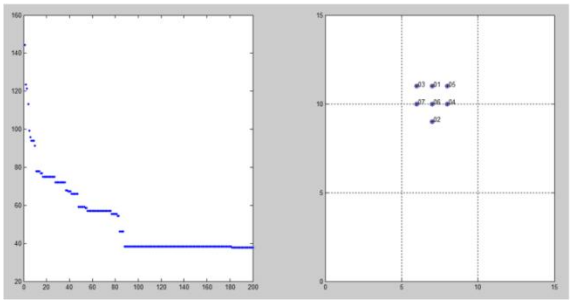

Fig.12. The final position of BCD counters logic block and error reduction after applying the conventional ACO with mutation

Table 6 shows a comparison between the different algorithms that their aim is decreasing the error.

Table.6
Comparing proposed Algorithms with previous algorithms for error reduction

| ACO+MUT | ACO-MUT | ACO | Best result in [21] | Best result in [20] |
|---|---|---|---|---|
| 73.72% | 67.26% | 46.17% | 44.60% | 37.04% |

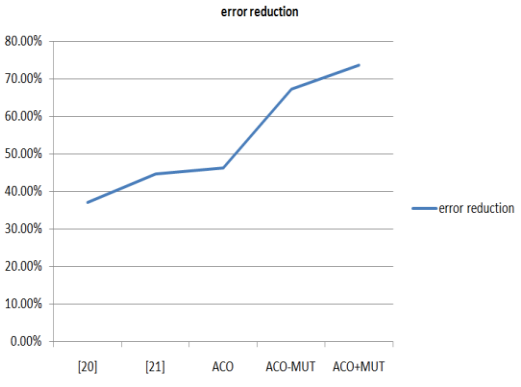


Fig.13. Cost function error reduction by each algorithm

Table.7
Comparison of the proposed algorithm with [15] in terms of reducing wire length

| Circuit | Circuit function | Number of CLB | [15] | ACO+mut wire length reduction |
|---|---|---|---|---|
| Cm138 | Logic | 43 | 57.11% | 59.90% |
| tcon | Logic | 82 | 46.55% | 49.07% |
| X2 | Logic | 58 | 48.63% | 51.53% |

## 5. Conclusion:

In this paper, we proposed two new algorithm setups for optimizing the FPGA placement problem based on ant colony algorithm and mutation. Mutation is used in order to speed up convergence and reduce the probability of trapping in local optimum. In this paper error reduction rate, algorithm run time speed and wire length reduction are investigated. According to the obtained values, it is shown that ant colony algorithm beside mutation has better performance than conventional ant colony algorithm, but ant colony algorithm with mutation among all three algorithms has the best performance. They have not only shorter in run time but also the cost function error and wire length is reduced. Clearly based on simulation results which are given in Table 6 and 7, all three algorithms presented in this paper in terms of error rate reduction and wire length reduction compared to the cost of the algorithms presented in [20] ,[21] and [15] have better results.

## References:


[1] B. Premalatha and S. Umamaheswari "survey of online hardware task scheduling and placement algorithm for partially reconfigurable computing systems" International Journal of computing and Corporate Research, Vol.2, Issie.3, 2012.

[2] E. M .Vasconcelos de Lima, A. C. Cavalcanti and L. dos Anjos Formiga Cabral, "A New Approach to VPR Tool's FPGA Placement", Proceedings of the World Congress on Engineering and Computer Science (WCECS), 2007.

[3] M. Yang, A.E.A. Almaini, L. Wang and P.J .Wang, "FPGA Placement Using Genetic Algorithm with Simulated Annealing", 6th international conference on ASIC, Vol.2, pp.808-810, 2005.

[4] V. Chopra, and A. Singh, "Solving FPGA Routing using Ant Colony Optimization with Minimum CPU Time" International Journal of Computer Science & Technology (IJCST), Vol.2, Issue.4, pp.223-226, 2011.

[5] X. Shi, "FPGA Placement Methodologies: A Survey" Dept. of Computing Science, University of Alberta, 2009.

[6] X. Wenyao, K. Xu and X. Xinmin, "A Novel Placement Algorithm for Symmetrical FPGA" Institute of Electronic Circuit and Information System, Zhejiang University", 7th international conference, IEEE press, pp. 1281- 1284, 2007.

[7] S. J .Lee and K. Raahemifar, "FPGA Placement Optimization Methodology Survey", Canadian Conference on Electrical and Computer Engineering (CCECE), pp.001981- 001986, 2008.

[8] M. Dorigo and T. Stützle, "Ant Colony Optimization",2nd edition, Vol.146, In International Series in Operations Research and Management Science, pp.227-263, Springer Verlag, NewYork, 2010.

[9] K.E Parsopoulos and M.N. Vrahatis. "Particle Swarm Optimization and Intelligence: Advances and Applications", IGI Global: Hershey, PA, 2010.

[10] K. Wang, N Xu, "Ant Colony Optimization for Symmetrical FPGA Placement", 11th IEEE international conference on Computer Aided Design and Computer Graphics, pp.561-563, 2009.

[11] M. Dorigo, M. Zlochin, N. Meuleau and M. Birattari. "Updating ACO pheromones using stochastic gradient and cross-entropy methods." In Applications of Evolutionary Computing, Springer Berlin Heidelberg, pp.21-30, 2002.

[12] M. Dorigo, G. Di Caro and L. M. Gambardella, "Ant algorithms for discrete optimization", Artificial life 5, No.2, pp.137-172, 1999.

[13] F. Ducatelle, "Adaptive routing in ad hoc wireless multi-hop networks", Universita della Svizzera italiana, Lugano, Switzerland, 2007.

[14] M. Shokouhifar, Sh. Sabet, "PMACO: A Pheromone-Mutation based Ant Colony Optimization for Traveling Salesman Problem", International Symposium on Innovations in Intelligent Systems and Applications (INISTA), 2012.

[15] Z. Baruch, O. Creț, H. Giurgiu, "Genetic Algorithm for FPGA Placement", In Proceedings of the 12th International Conference on Control Systems and Computer Science (CSCS12), Vol. 2, pp.121-126, 1999.

[16] M. Dorigo and G. Caro, "The Ant Colony Optimization Meta-Heuristic", Universit e Libre de Bruxelles.

[17] www.cse.wustl.edu/~jain/cse567-08/ftp/fpga/

[18] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0", 1991.

[19] www.datasheetcatalog.org/datasheets/400/287785_DS.pdf.

[20] P.K .Rout and D.P. Acharya, "Novel PSO based FPGA Placement Techniques", International Conference on Computer & Communication Technology (ICCCT'10), pp.630-634, 2010.

[21] P.K .Rout, D.P Acharya and G. Panda, "Digital Circuit Placement in FPGA based on Efficient Particle Swarm Optimization Techniques", 5th International Conference on Industrial and Information Systems, pp.224-227, 2010.