❐   989

An Area-Optimized Chip of Ant Colony Algorithm Design in Hardware Platform Using the Address-Based Method

E. Shafigh Fard[1], K. Monfaredi[2], M. H. Nadimi[3]
[1]Faculty of Computer Engineering, Najafabad branch, Islamic Azad University, Isfahan, Iran
[2]Engineering Faculty, Department of Electrical and Electronic Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran
[3]Faculty of Computer Engineering, Najafabad branch, Islamic Azad University, Isfahan, Iran

Article Info	ABSTRACT
	The ant colony algorithm is a nature-inspired algorithm highly used for solving many complex problems and finding optimal solutions; however, the algorithm has a major flaw and that is the vast amount of calculations and if the proper correction algorithm and architectural design are not provided, it will lead to the increasing use of hardware platform due to the high volume of operations; and perhaps at higher scales, it causes the chip area not to work because of the high number of problems; hence, the purpose of this paper is to save the hardware platform as far as possible and use it optimally through providing a particular algorithm running on a reconfigurable chip driven by the address-based method, so that the comparison of synthesis operations with the similar works shows significant improvements as much as 1/3 times greater than the other similar hardware methods.

Corresponding Author:

E. Shafigh Fard,
Faculty of Computer Engineering, Najafabad Branch,
Islamic Azad University, Isfahan, Iran.
Email: shafighfard@azaruniv.edu

1.   INTRODUCTION

Imitating the nature and its experiences is one of the most common methods used in artificial intelligence. Algorithms and evolutionary computations refer to a set of methods whose problems have been solved being inspired by the evolution of animals, plants, and insects in nature. This type of algorithms which is used to find the optimal solution include complex and time-consuming calculations causing these types of problems to be remained unanswered in the past; but since the early 1990s, the speed of ant colony algorithm began to improve using parallel software methods [1, 2, 3]. By 2002, all methods were software-based and mostly along with the parallel processing technique [4, 5] and in a few cases were run in parallel on a graphic processor composed of multiple cores [6]. From the 2000s onwards, methods for reconfigurable on-chip hardware implementation began to be used by Dr. Scheuermann's workgroup [7] and later, a work was presented using the implementation based on CMOS technology [8] as well as 2 works were presented based on the reconfigurable chip with the comprehensive use of all on-chip IPs and cores [9, 10] and in 2012, a work which can be considered as a compromise of hardware and software was presented in which besides the flexibility of software, the hardware speed has been also added [11]. In the following, sections 2-4 have been provided as follows: In section 1, the evolutionary algorithms, particularly the ant colony algorithm are introduced; in section 2, the method presented by this paper will be discussed; section 3 is devoted to evaluation, simulation, and comparison with previous methods; and section 4 includes conclusions and future works.

2. THE ANT COLONY ALGORITHM

In a group collective behavior model, a group of people are working together to achieve the ultimate goal. This method is much more beneficial than when they act independently. The ant colony can be defined as an organized set of organisms which collaborate with each other using pheromones and the information exchanged by updating pheromones [12]. In computational collective intelligence, creatures such as ants, termites, bees, fish, and birds groups are modeled. In this type of structures, every living thing is doing a very simple task, but their cooperation with each other shows complex behaviors [13]. The overall non-linear behavior of a community is obtained from combining individual behaviors of that community; in other words, there is a very complex relationship between the collective and individual behaviors of a community. The collective behavior also depends on the interaction between individuals, so that interactions enhance the experiences of individuals and thereby cause the progress of the community. When an ant in city i wants to go to the next city, e.g. city j, the probability distribution is used by the ant to select the next city [M. Dorigo, G. Di Caro, L. Gambardella, 1999].

$$pij = \frac{[\tau i,j]^\alpha [\eta i,j]^\beta}{\sum[\tau i,j]^\alpha [\eta i,j]^\beta} \tag{1}$$

However, in the probability distribution (equation 1), there should be a link between i and j, so it can be said that node i is the neighbor of node j. As the equation implies, the studied js should belong to the set N containing the nodes that have not been previously selected, because pijs which have been already selected are assumed to be zero. After calculating the pijs which must be calculated, q_0 in the interval (0,1] is a random variable compared with q which is a parameter of the ant colony algorithm, so that to go from node r to node s using the path u;

If $q_0 > q$,

If s is not among the remaining cities

$$P_k(r,s) = \sum_{u \in jk(r)} \frac{Ph(r,s)^\alpha . D(r,s)^\beta}{Ph(r,u) . D(r,u)}$$

If s is one of the remaining cities

$$P_k(r,s) = 0 \tag{2}$$

If $q_0 < q$,

If the best mode is selected

$$S = Arg\max\left\{ Ph(r,u)^\alpha, D(r,u)^\beta \right\}$$

If the next mode is selected

$$s \in jk(r)$$
$$s = \text{next city} \tag{3}$$

After searching for a solution and completing the iteration, ants perform the final update operations termed as the global update. Here, the ant that has made the shortest tour is allowed to modify the pheromone of edges belonging to its tour; this increase in the pheromone is according to equation 4.

$$\tau ij^{new} = (1-\rho)\tau ij^{old} + \rho\Delta\tau ij \tag{4}$$

Based on the global update, only the pheromone of edges belonging to the shortest path is changed and the highest value of pheromone is assigned to edges with the shortest length.

3. THE PROPOSED METHOD

In this section, a framework is presented for modeling ant colony algorithm, which takes advantage of the hardware design based on the programmable system-on-chip (PSoC) to cover a wide range of applications. The results of all software on the desktop processor have been compared with the modeling results of the address-oriented architecture based on the programmable system-on-chip. The presented method is aimed at reducing the processing time of the algorithm. The proposed architecture is completely hardware-oriented which has been implemented using the address-based method. The efficiency of the presented architecture has been evaluated by several benchmark problems. Implementations have been performed using various design tools such as Xilinx ISE and MaxPlusII as well as VHDL (VHSIC hardware description language); and the completely software cases have been simulated in Matlab.

3.1 Parallel Ant Colony Algorithm in Hardware Platform

Here, the ants' performance in the reconfigurable chip platform is described in accordance with the proposed idea. The following flowchart (Figure 1) shows the ants' performance.
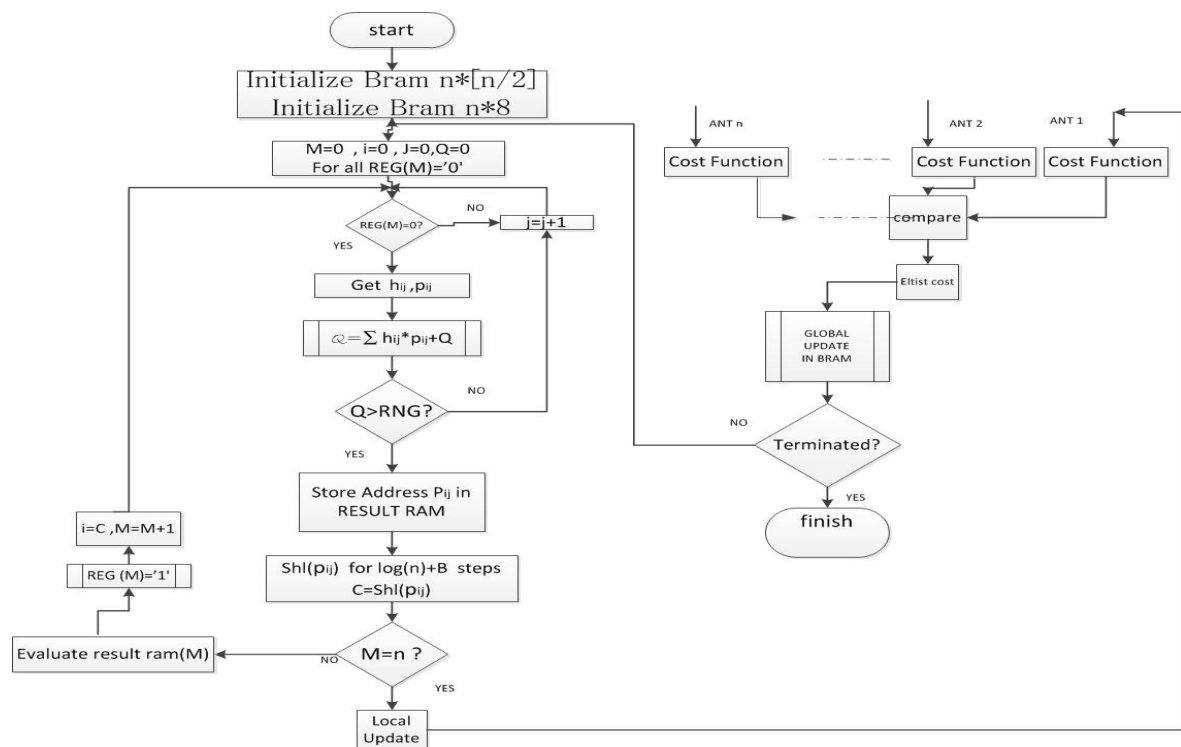

Figure 1. The flowchart of the proposed ant colony algorithm in the hardware platform

As observed in Figure 2, firstly, 2 RAM blocks are initialized; one of these RAMs is used to store the information of pheromone between two nodes and the other one, despite being stored in a RAM block, is in the application of ROM, because it holds heuristic coefficients to prevent any changes in the distance between two nodes during the execution of the program; and most importantly, Result RAM which is used as the control of nodes select or deselect has as many one-bit units as the number of nodes that all are firstly set to zero; namely, no node has been selected and the field related to a node becomes 1 with each choice. In all ant colony algorithms that are implemented in the hardware platform pheromone values stored in the memory are considered as an n*n matrix like Figure.

In the pheromone matrix, the values of i, j makes up columns and rows of the matrix and each ant is assigned to a row; in the mentioned flowchart, ant number 1, in the beginning, is assigned to the zero-th column and row and begins its search operations and reads the pheromone and heuristic coefficient values of cell i, j from the related memories and carries out operations in accordance with the flowchart; selecting each node causes the value of M, which has been initially set to zero, to be incremented one unit, and the address associated with the selected node is stored in a memory called Result and the ant finds its next row based on

the node currently found so that using left-shift operations which is a kind of substitute for multiplication operations to lower the power consumption, it goes from one row to another in each transition.
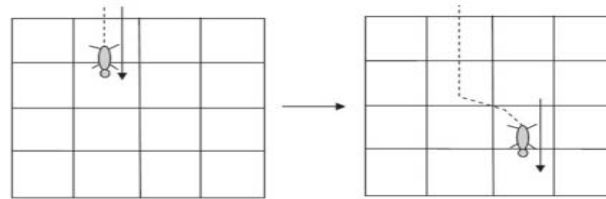

Figure 2. The movement of ants in the pheromone matrix

However, if any node dos not meet the condition of being greater than the random number generated by LFSR method [14], the ant just tries to alter the column frontward and stays in the same row to select a node as well as by selecting a node, checks if the condition M=n is met or not, which it shows whether all nodes have been selected or not. If M=n, the process of creating a solution for that iteration is completed and the local update phase and then the phase of assessing the competence of ants begins. By comparing the cost functions of all the ants, the lowest cost function is selected and consequently the pheromone memory is updated; after the update, as mentioned in the previous section "familiarity with the ant colony algorithm", this algorithm can be repeated to infinity times; it should mentioned that after doing all phases, at the end, the iteration condition is always checked and if the termination of operations is true, the process of program execution ends up.

3.2 The Framework of Ant Colony Optimization Algorithm Architecture Based on the Programmable System-On-Chip

Modifications on the Ant Colony Algorithm: Some modifications have been conducted on the algorithm to reduce hardware costs; for example, simple register shift operations have been used instead of multiplication operations as well as a series of simple changes have been conducted in the update process without defecting the main program to prevent numbers from being displayed as decimal ones.

The architectural structure: The framework designed for the algorithm has consisted of a reconfigurable chip so that the ant colony parameters are defined in two blocks of memory on reconfigurable chip and all arithmetic operations of the algorithm are hardware modeled on FPGA logics. Figure 3 shows the presented framework and the connections between different blocks; as shown in the figure, there are two independent memories including main parameters of the algorithm along with evaluation, update, and city selection units each of which has consisted of a series of sub-blocks.
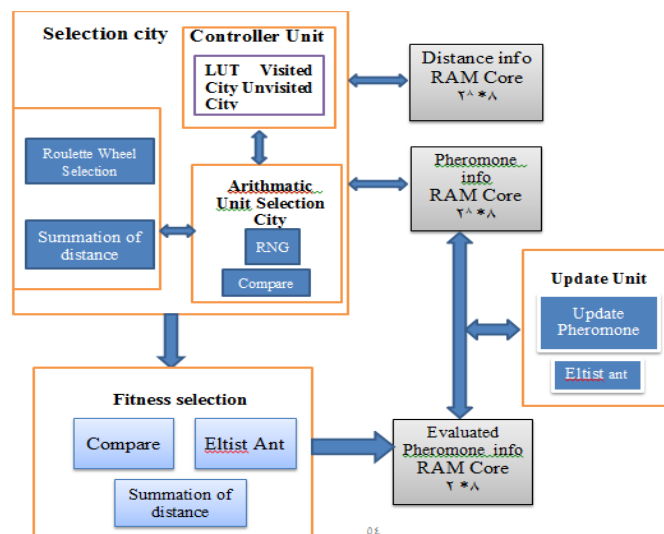

Figure 3. The architectural structure of the ant colony algorithm based on the programmable system-on-chip

Inter-chip memories have been selected and the hardware core of the ant colony optimization algorithm has been modeled on FPGA logics.

Memories: The distance info RAM Core (the memory of heuristic information) which stores the information including a 100% enlargement of the distance between nodes and can be modeled as an n*n matrix; for n=16, it can be considered that the diameter of the matrix is entirely zero, because the distance of a node from itself is zero. The pheromone info RAM Core which stores the value of pheromone secreted between nodes and is displayed as an n*n matrix. Both of memories mentioned above are of the Ram block type and initialized with default values; however, the distance info RAM Core is a BRAM type in the application of ROM. The memory of the control unit which has been simulated as a multiplexer uses LUTs distributed in FPGA platform; this N*1 multiplexer is firstly initialized as n 0-bit inputs to convey the concept of deselecting the desired city, which takes the flag with value 1 in the multiplexer after repeating the operation and meeting the condition of selecting the relevant node. The result memory including the best and most optimal results is a RAM block which contains the elements (nodes) addresses in the memory; for example, it has a 16*8 memory for a 16-fold node.

The next city selection block: As shown in figure 3, this block consists of two blocks as 1- the control unit and 2- the arithmetic unit including sub-blocks for generating random numbers, comparison, and a series of arithmetic operations based on the Roulette Wheel law. In the following, the blocks and their functions are discussed in detail.

The control unit: As mentioned in section 2, ants respectively and node to node carry out traversal operations from the nest based on the laws of probability to reach the food; however, to prevent the repetitive selection of a node in the path, it should be controlled that the traversed nodes are separated from the ones not traversed. So as shown in figure 4, a logic circuit is used, in which a flag is sequentially defined for each node. The flag is stored in a 16-bit array which operates in the form of a 16*1 multiplexer; and each ant starts the search from the cell array 1 (the cell array 0 is related to the source and has been already selected).
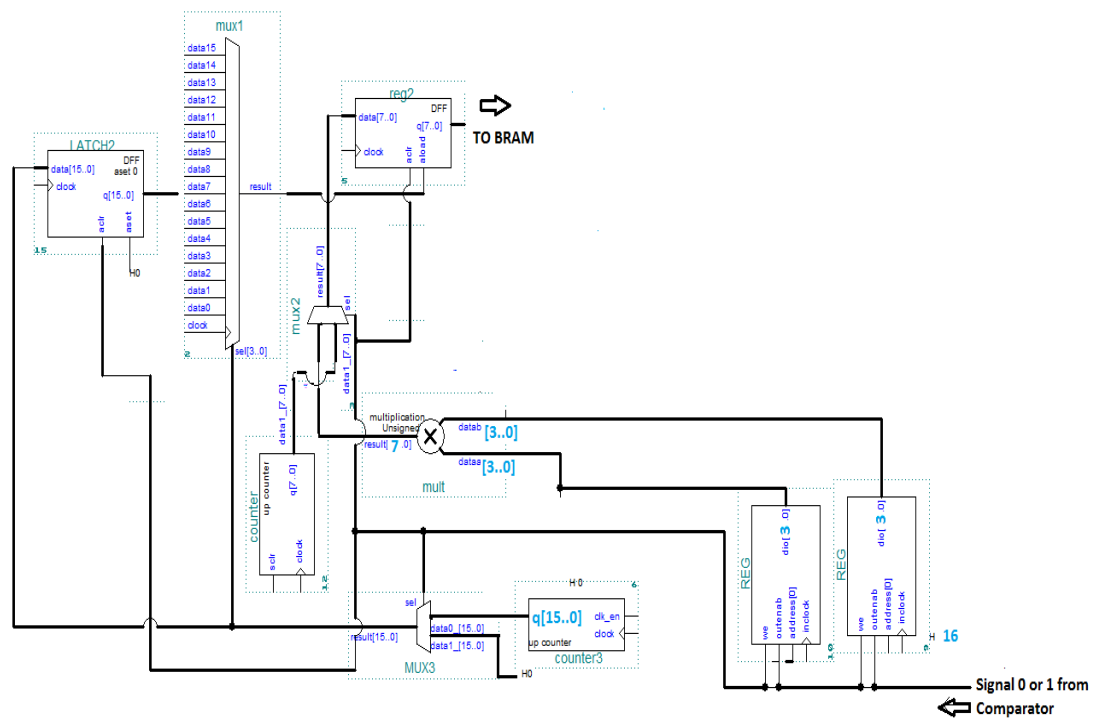

Figure 4. The block diagram of the control unit

If the flag is 0, it means that the relevant node has been selected; and if it is 1, the node can be a candidate for being elected as the next city; hence, using this method, the number of comparisons is reduced, because there is no need to check whether a node has been already selected or not. The further explanation that can be given for the control unit is that when a flag related to a node is checked to ensure the select or deselect of the node, if the flag has not been already selected; namely, the flag is 0, the system will go towards BRAM to find the parameters related to the relevant node. According to the rules previously stated,

the desired node meets the conditions and been selected, the value obtained from the comparator has become 1 (active high); and since the idea proposed by this paper is address-based, the address of desired node stored in a register is multiplied by 10 Hex or 16 binaries, because it has been designed in the memory so that for example, the addresses H"0000" to H"0001" and H"0010" to H"0011" are respectively assigned to nodes 0 and 1, and so forth. As observed in figure 4, when a node is selected, the value of address in the register which should be read from BRAM is reset and multiplied by the multiplexer whose selector value is 1; and the line 1 of multiplexer entered the address register which is an address counter is not selected, because the selector is zero; on the other hand, the selector of multiplexer 16*1 selects line 1 through the selector; in other words, an overall reset is conducted in multiplexer 16*1 and checking the flags is restarted from the source node to the node newly selected; but if we consider the case when the desired node has not been selected and the signal output from the comparator is 0, all checks will be performed in the same sequential manner and no mutation will take place, because no reset has been occurred in registers through the signal 0 resulted from the comparator; and selecting line 1 of both multiplexers, whether multiplexer 16*1 or 8*1, the register of incremental counter is selected and operations are sequentially performed. It should be also noted that there must be a separate control unit for each ant.

   The calculations and logic unit (the ant colony laws of probability): Most rules and operations of the ant colony algorithm needed to find the next node is performed in this block. As shown in figure 5, two main parameters (the pheromone and heuristic coefficient) are respectively taken from BRAM and ROM according to the relevant address.
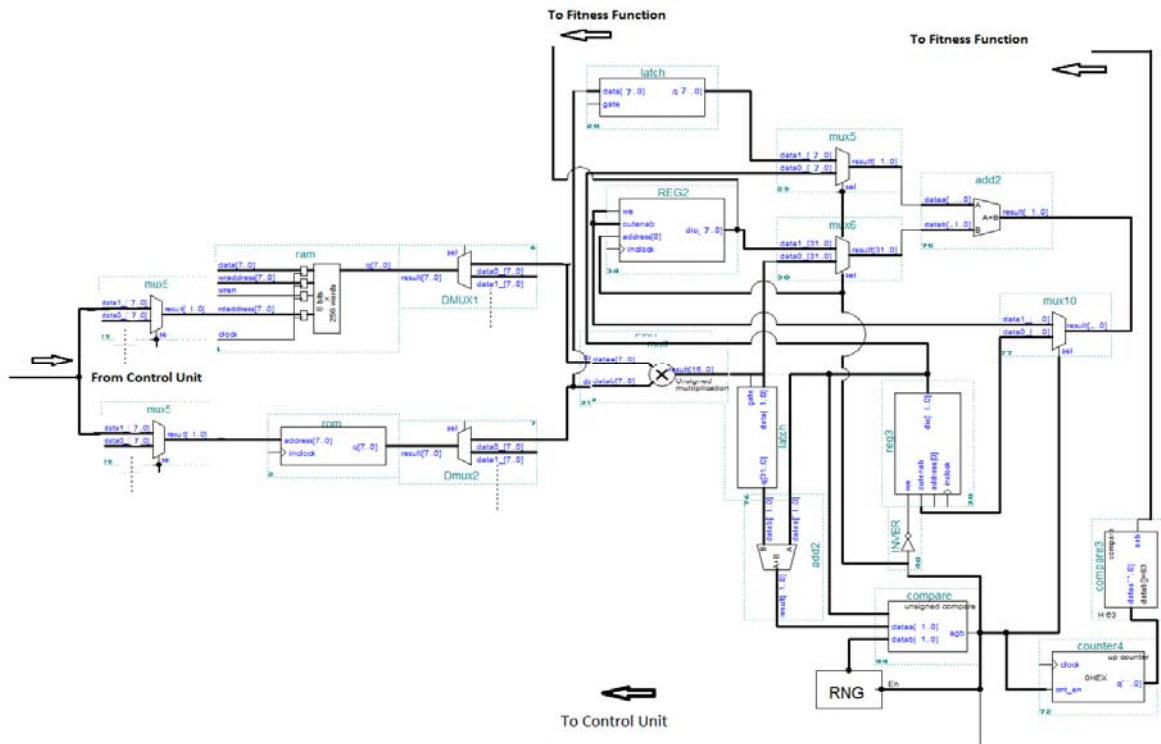

Figure 5. The unit of calculations block

After the control block which is responsible for detecting whether any node has been already selected or which address should be selected, it's time to start the calculations stage to select the node. According to figure 5, there should be a calculations unit for each ant so that the nodes selection operations are performed in parallel with each other. After receiving the desired parameters, the search and qualifying operations mainly begins by multiplying the heuristic value by the amount of pheromone between the current node and the other node which is being checked in terms of becoming a candidate. After multiplying the pheromone by the heuristic coefficient, the resulting value is compared with the parameter obtained from the random number generating unit (which is considered as a separate module and attempts to generate a random number in the specified interval by changing each clock. The resulting value is of utmost importance,

because the comparison signals, 1 or 0, are referred to the control unit so that the control unit function is based on the comparison signal (Table 1).

Table 1. The control unit function based on the comparison signal

Comparison Signal Status	How the next address is selected in the memory block	How the next node is selected in the multiplexer (the control unit)
0; the value of random number is higher.	The checked node is not selected and the count operation is occurred and no reset is performed in the running registers.	The count operation is occurred to check the next node.
1; the value of random number is lower.	The current node is selected; and to select the next node, the shift operation from the current address is performed as many as the number of LOG (N); and the reset operation is occurred in running registers.	The overall reset is occurred in the multiplexer, and the selector starts to check node 0.

To identify and clarify the performance of calculations unit and the way of determining the cost function, figure 4 has been magnified to show clearly how one of the Roulette Wheel operations or the cost function is performed and selected. As shown in the following figure (figure 6), after the comparison was performed once and the comparator produced the result as signal 1 (which means the selection of current node), lines 1 are selected. When the multiplexer lines become 1, the cost of selected node is processed along with the previous costs as the current final cost and if the output signal of comparator is 0, the lines 0 of multiplexers are selected and a part of the Roulette Wheel operation will be ready for the next clock; in this way that by selecting zero lines, the values of $\sum [\tau ij]^\alpha [\eta i, j]^\beta$ are calculated to be used in the process of selecting the next node in the next clocks.
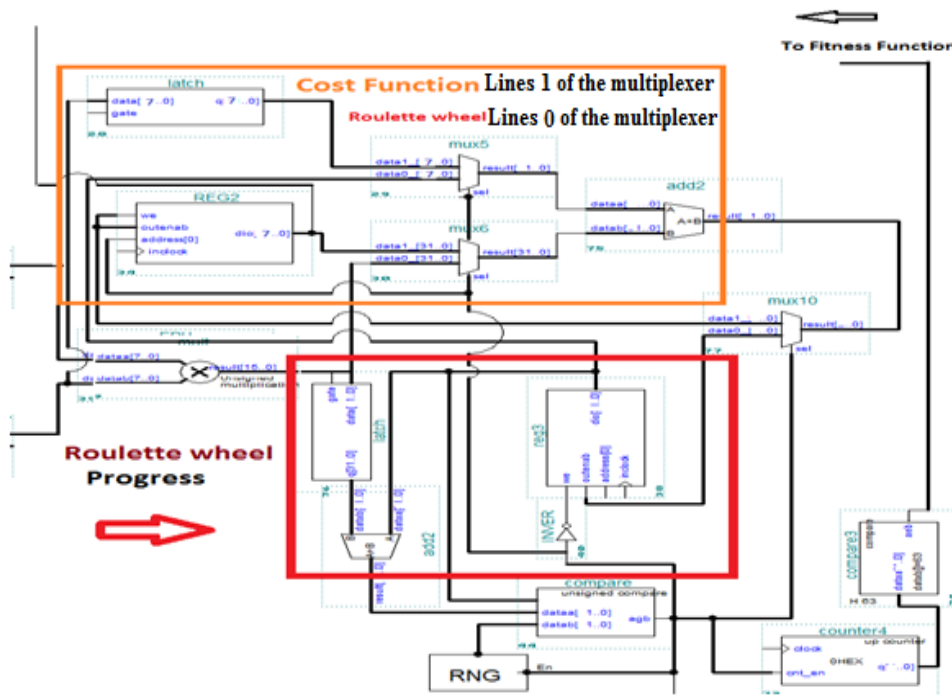

Figure 6. Magnification of the calculations unit

The competence evaluation unit: At this stage, after several iterations (depending on the condition of completion of the algorithm), the system enters the phase of competence evaluation. At this stage, each ant enters the evaluation phase while it has a cost function located in an n-bit register (depending on the number of nodes) and a $n * \lceil n/2 \rceil$ memory containing the addresses of selected nodes. Firstly, through the comparison of ants' costs functions, the minimal value is selected; and by selecting the lowest cost function, the solution related to the desired ant enters the update phase.
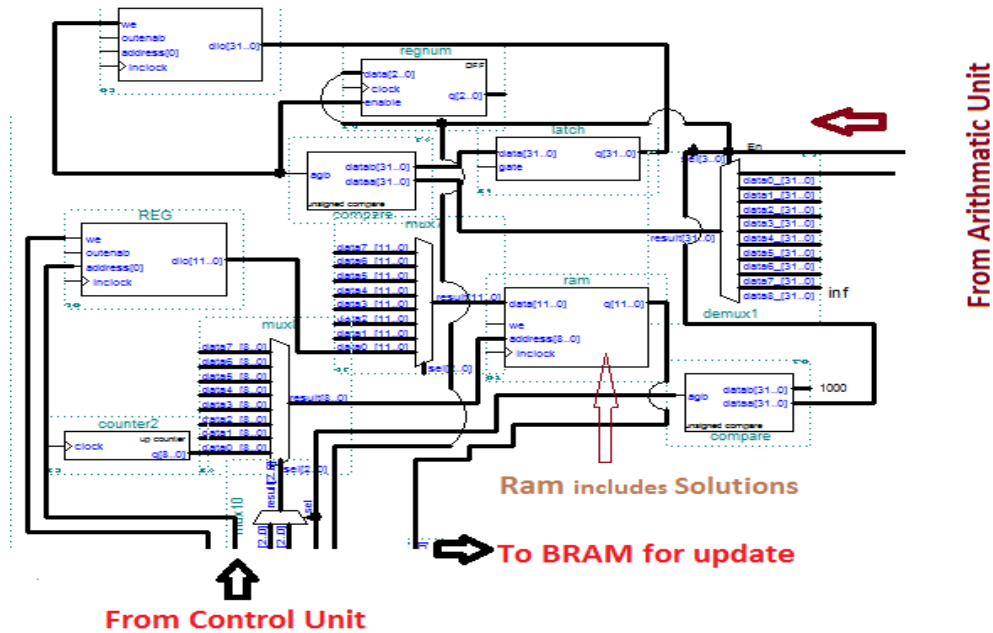
Figure 7. The evaluation unit

The global update unit: In this block, after the solutions available in RAM (figure 6) are compared with each other in terms of the cost function, the addresses of the best solution (considering the performance of the best ant) are sent to Bram which includes the pheromone parameter to perform the global update operations on the memory using the pheromone parameters.

4.   EVALUATION AND ANALYSIS OF THE RESULTS

In order to determine the efficiency of the system proposed based on the SOPC, it has been simulated and synthesized on the Virtex7chip, from Xilinx series. All private and public blocks have been written by VHDL (VHSIC hardware description language). For the synthesis operations, the ISE Simulator tool available in the ISE Xilinx software has been used. Here, the results of resource consumption obtained from synthesizing the presented method are stated. Since the number of nodes (with respect to the occupancy of bit vector) and the number of ants (due to the creation of solutions pipeline) play an essential role in the occupation and use of on-chip resources, several tests have been conducted by altering the number of nodes (n) and number of ants (m) to show their changes effects on the increase and decrease of on-chip resources consumption. For this purpose, the synthesis operations have been conducted in a fixed number of 64 nodes with three different pipelines. Figures 8-10 show the results obtained from the regression function. Table 2 shows a comparison between the results obtained from the presented method and a similar work conducted by Dr. Scheuermann's workgroup; and Table 3 shows the comparison of resources needed by the method of system-on-chip with external memory and 24-bit vector for the heuristic coefficient
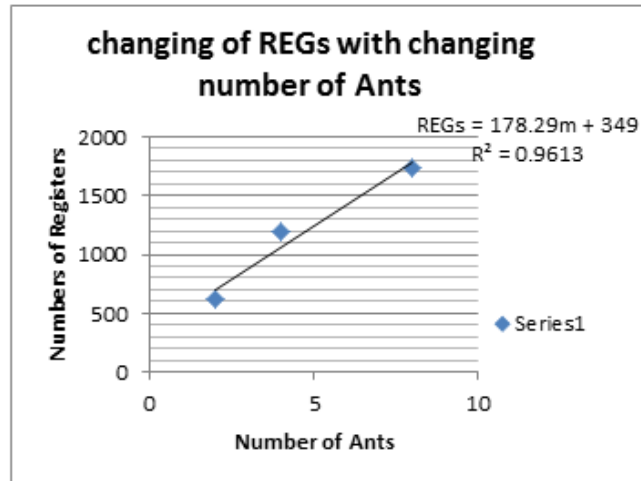
Figure 8. The results of regression function for estimating the number of registers in $n = 64$
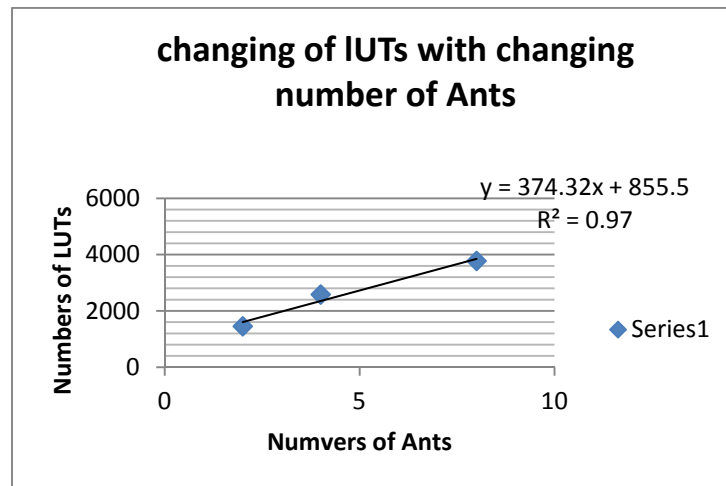

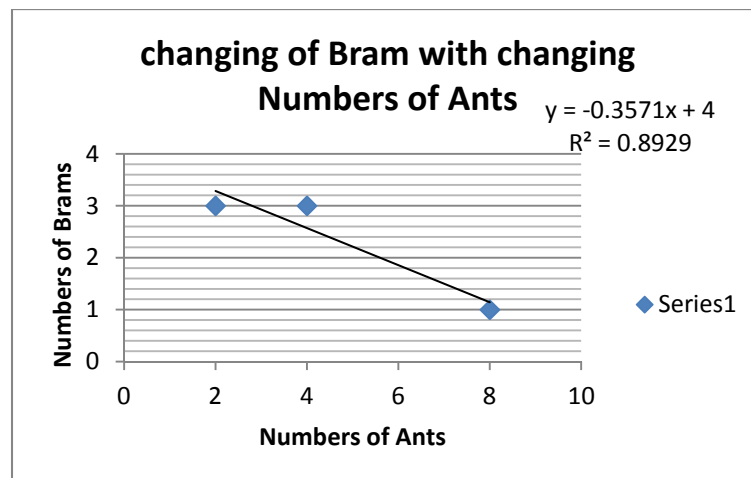Figure 9. The results of regression function for estimating the number of LUT in $n = 64$

Figure 10. The results of regression function for estimating the number of memory blocks in $n = 64$

Table 2. The comparison of resources used by the presented method and the ones by Dr. Scheuermann's workgroup

Bit Vector	The number of BRAMs with n=64 m= the number of ants	The number of LUT with n=64 m= the number of ants	The number of registers with n=64 m= the number of ants	Resources / The method
6-bit	BRAMs=2.4802m+1.5297	LUTs=1516.m+97.2749	REGs=542.244m+244.764	Population based on the ant colony optimization on FPGA.
8-bit	BRAMs = -0.3571m + 4	LUTs = 374.32m + 855.5	REGs=178.29m + 349	The proposed work

Table 3. The comparison of resources needed by the method of system-on-chip with external memory and 24-bit vector for the heuristic coefficient

IO	The number of LUT M=1	The number of registers M=1	Resources / The method
26	4511	434	Architect for high speed ant colony optimization.
25	452	150	The proposed work

5.   CONCLUSION AND FUTURE WORKS

In this paper, to realize the ant colony optimization algorithm, a modeling based on the system-on-chip with address orientation was proposed, which is applicable as a real-time optimizer. This architecture uses a set of parallel structures and pipelines that enable it to fulfill the optimization of various problems by the ant colony algorithm. The presented method is entirely hardware and the area used by the chip shows significant improvements as much as 1.3 times greater than the other similar hardware methods. As the future works, it can be pointed to making the method scalable which can be conducted using the network-on-chip technology. The other interesting work can be providing a hardware-software combination method based on the proposed hardware core to preserve the flexibility of the algorithm more.

REFERENCES

[1]   Dorigo M. Optimization, Learning and natural   algorithms, *Phd, thesis*, Politecbico di Milano, Italy 1992.
[2]   Dorigo M. Parallel ant system: an experimental study, unpublished manuscript, *cited by M. Dorigo 1993*.
[3]   Dorigo M, Di Caro. Gambardella, Ant algorithms for discrete optimization, *Artificial Life* 5 (2), 1999. 137–172
[4]   Pedemonte M, Nesmachnow S, Cancela H. "Survey on parallel ant colony optimization", *Applied Soft computing journal 2011*.
[5]   Delisle P, Gravel M, Krajecki M, Gagné C, Price W. "Comparing parallelization of an ACO: message passing vs. shared memory", in: *Proceedings of the 2nd International Workshop on Hybrid Metaheuristics,* Lecture Notes in Computer Science vol. 3636, 2005. 1–11.
[6]   Fu J, LEI L, Zhou G. "Parallel ant colony optimization algorithm with gpu acceleration based on all-in-roulette selection", *in Proceedings of the 3rd international workshop on advanced computational intelligence*, 2010, pp 260-264
[7]   Scheuermann B, Janson S, Middendorf M. "Hardware-oriented ant colony optimization", *Journal of Systems Architecture* 53 (7) 2007. 386–402.
[8]   Gheysari K, Khoei A, Mashoufi B. "High speed ant colony optimization CMOS Chip", in *Journal of Expert Systems with Applications*, Elsevier science, Spring 2011.
[9]   Yoshikawa M, Terai H. "Architecture for high –speed ant colony optimization .*proceedings*", of IEEE International Conference on information Reuse and integration ,las Vegas,NV,2007. 1-5.
[10] Yoshikawa M and Terai H, Hardware-oriented ant colony optimization considering intensification and diversification in: Bednorz, W. editor. *Advances in greedy algorithms*, I-Tech, 2008. 359-68.
[11] Merkle D, Middendorf M. "Fast ant colony optimization on runtime reconfigurable processor arrays", *Genetic Programming and Evolvable Machines* 3 (4) 2002. 345–361
[12] Bai H, OuYang D, Li X, He L, Yu H. "Max-min ant system on gpu with cuda", in: *Proceedings of the 2009 Fourth International Conference on Innovative Computing*, Information and Control, IEEE Computer Society, 2009. pp. 801–804.
[13] Duan H, Yaxiang Yu, Jie Zou, Xing Feng. "Ant Colony Optimization algorithm design and its FPGA implementation", *IEEE International Symposium on Intelligent Signal Process and Communication Systems 2012*.
[14] Martin P,"an analysis of random number generators for a hardware implementation of genetic programming using FPGA and Handek-C", IN Proc.Geetic and Evolutionary Computation Conf, July 2002, pp. 837-844