Clonal Selection Algorithm for Partitioning and Scheduling of Codesign Systems

[1]Maryam Zomorrodi Moghaddam and [2]Ahmad Kardan
[1]Department of Engineering, Bojnord University, Bojnord, Iran
[2]Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran

***Abstract*- In system-level design, applications are presented as task graphs where tasks; i.e. nodes of the graph, have several implementation options differing in some criteria such as time, area and power. Systems designed with this approach are those that are application specific and for performance reasons are implemented in a hardware/software codesign manner. In this paper the most important design issues in these systems i.e. partitioning and scheduling are investigated. Our approach, namely CSPA, is a heuristic algorithm inspired by the biological immune system and attempts to obtain an optimal design for a given system composed of several hardware and software components. We use a graph representation of the system where nodes are operational components and edges are communication links between them. We propose an immune-based approach which based on artificial immune system and apply the clonal selection algorithm as one of the different types of algorithms inspired by biological systems. To date there is no work in this field that uses the clonal selection algorithm for optimization of partitioning. Empirical results show a suitable improvement by using this approach in comparison with traditional evolutionary algorithms and also traditional immune-based approach.**

***Keywords*-** CoDesign, Hardware/Software Partitioning, Scheduling, Artificial Immune System and CSPA.

I. INTRODUCTION

Embedded systems have important roles in the world of autonomous systems. They are the fundamental elements in most of the innovations in many fields. So optimizing them in the design time is a major task. One of the primary features of these systems is their speciality to needs of each field. They don't be designed to be responsible to several needs that conflict in definitions and tasks. These systems are said mixed since they are usually a combination of hardware and software components.

The most important step in designing such systems is partitioning, that is how computation is divided between hardware and software elements so that overall cost of implementation is minimized. Scheduling in such systems is also of critical importance. Scheduling is the task of determining the starting time for each task of the system. Really those two parts, i.e. partitioning and scheduling, could not be viewed separately, because each part has influence in the other part. Scheduling needs knowledge about the execution of each component, i.e. execution start and end times, and this knowledge is only obtained when we know where components are to be executed and on what platform to calculate its execution time and its possibility for concurrent execution with other components.

These two problems known to be NP-hard. In this paper a novel bio-inspired algorithm is introduced for solving partitioning problem in combination with scheduling.Similarly to the way nervous system inspired the development of artificial neural networks (ANN), the immune system has now led to the em ergence of artificial immune systems (AIS), as a novel computational intelligence system.

The proposed Clonal Selection Partitioning Algorithm named CSPA is based on the clonal selection in the human body immune system. It is designed to solve partitioning problem with the clonal selection of solutions and applying the algorithm to them. Our approach aims at using ideas gleaned from immunology. The artificial immune system (AIS) for hardware/software partitioning is used in [1] under the name "evolutionary immune system".

Some of our work in this paper which applies AIS is based on the approach used in [1]. But we have developed this approach and combined it with the clonal selection theory as another algorithm in this field. On the other hand, our paper more precisely and completely investigates the immune system method.

The remainder of this paper is divided to these parts: first in the following section we consider some works in this area. Then in section III we outline the main components in the immune system and illustrate its role in the optimization problems. Section IV shows our proposed work in more details. In section V we brought some discussion of our algorithm and its ability to escape from local optimal. Section VI illustrates the fitness function used and scheduling method we applied. Finally we investigate experimental results and conclusion remarks.

II. RELATED WORKS

We can divide partitioning algorithms with two categories include: those that their criterion is improvement of a cost function which is a combination of all necessary factors [8]. And those that attempt to optimize one parameter in design while constrain other factors [7]. Our algorithm is in the first class.

On the other hand, algorithms used in the optimizations in this field have been very diverse. A large numbers of them are heuristic approaches [9, 10]. This is because the problem's nature is NP-complete and when the dimensions of the

problem are large, finding an acceptable solution is not possible.

Immune-based systems have become popular in recent years as a novel approach for solving several problems in many fields. The artificial form of clonal selection has been popularized mainly by de castro and Von Zuben, beginning with an algorithm which they called CLONALG (de castro and Von Zuben, 2000). CLONALG currently exists in two forms [16] - one for optimization and one for pattern matching. In [13] theoretical advances in the AIS's field have reviewed And shown that to date these advances include immune networks, clonal selection and negative selection. In [17] the ability of clonal selection to escape from local optima is proved with one technique called receptor editing that occasionally allows accepting less promising regions. The authors in [6] propose another variation of AIS namely DISAS. It is based on discrimination between self and non-self in the human immune body.

However there are a lot of studies in both partitioning and AIS, but our work is a new one which applies the clonal selection mechanism with other features of AIS to partitioning. In the following sections we describe the details of it.

III. Using Artificial Immune System in our Work

Artificial Immune Systems (AIS) have become an increasingly popular area for study by computer scientists in recent years. The term AIS refers to any computer system which is inspired by the natural immune system [3]. Artificial Immune Systems are a group of methods that inspired from the community of immunology [4]. Body immune system which is part of this system is distributed around the body and protects us from attacks of foreign bodies known as *pathogens*.

The job of the immune system is to monitor the body and make classifications as to whether the items it encounters are *self* or non-self. Everything which the immune system categorizes *non-self* must be destroyed. Everything which is a natural and healthy part should remain untouched by the immune system. Antibodies are features inside cells that used for recognition of antigens. The term antigen is said to everything that immune system is capable to recognize it.

The part of antibody that is responsible to recognizing antigen is said *paratope* which also called V shape (because of its variability). The model used in this paper is the discriminated-based immune system such as the model that is pointed in [6] and named DAIS. But we applied clonal selection within it.

For us those parts of the immune system operations that generate final T cells are considerable. Its complete and precise discussion is in [5]. Actually, optimal or near optimal solution for partitioning is the same as final T cells which are generated after cell cloning and maturation. Clonal selection is a process in the immune system where matching cells are proliferated in order to population of best matching cells become great. These are then can response better to incoming pathogens. Fig. 1 illustrates the process of clonal selection. This figure is taken from [16].

The immune system approach is also usually imparted from a segment of immune system operation namely negative selection. The purpose of negative selection mechanism is to give some tolerance to self cells. This part of immune system operations occurs in thymus uses capability of immune system to appear unknown antigens, while do not react to self cells [14].

Self-set in our algorithm is set of solutions that their implementation costs are high. In other word their fitness that is a criterion about the goodness of some solution in the word of genetic algorithms is low. The algorithm attempts to find which selected solutions have maximum distance to self set. This is the same as immune system does about the selection of T cells and tries the population of pathogen distinguisher cells have maximum distance to self cells to restrain the immune system from destroying itself. Matching rule can be hamming distance, r-bit continuous matching and so on. Matching threshold is shown with η.

According to the r-contiguous matching rule used in [13], we have the following definition for matching between two solutions:

Definition 1- an element $e \in \sum^{L}$ with $e = (e_1, e_2, \ldots, e_L)$ and detector $d \in \sum^{L}$ with $d = (d_1, d_2, \ldots, d_L)$, match according to the r-contiguous bit rule, if a position p exists where $e_i = d_i$ for $i = p, \ldots, p + r - 1, p \le L - r + 1$.

In our work, elements in e and detector set d have 0 or 1 values. L is the number of graph nodes.

r is some percent of L.

Also because of two implementation options in our work, hardware and software, the generated strings are binary, and our alphabet is $\sum = \{1,0\}$.

In the phase where clones are created, however we used the hamming distance with the following equation to achieve the similarity between an individual's clones.

$$D = \sum_{i=1}^{L} \delta, where \quad \delta = \begin{cases} 1 & if\ Ab_i \neq S(t)_i \\ 0 & otherwise \end{cases} \qquad (1)$$

In (1):

D is the hamming distance.

Ab_i is the *i'*th gene in antibody or *i'*th node in the process graph.

$S(t)_i$ is the *i'*th gene in the self-set.

The selection of the r-contiguous bit matching rule in the operational graph of the embedded system leads to increasing in speed to reach acceptable solutions. This is because of the more similarity in graphs matched with this model from the communication cost's point of view.

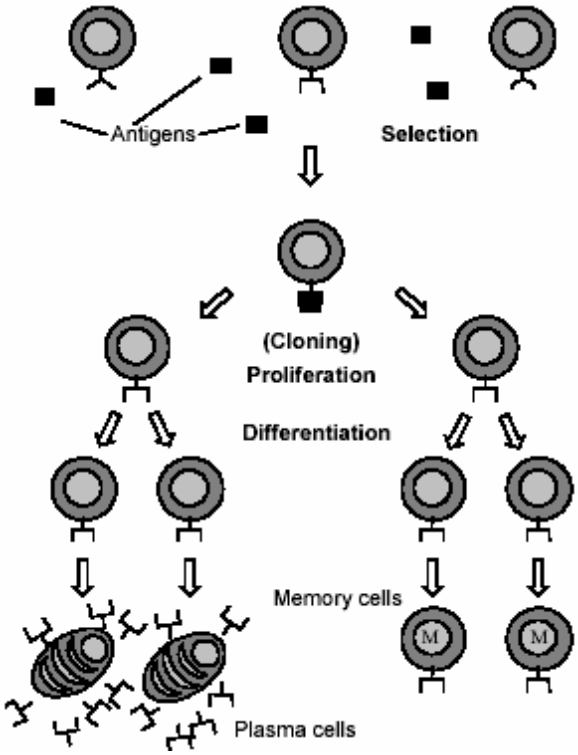

Fig. 1. The clonal selection principle

One of the operations take places in B cells is the generation of random genetic changes that expressed as diverse antibody patterns to system. In addition to somatic hypermutation and receptor editing, a fraction of newcomer cells from the bone marrow is added to the prior lymphocytes in order to maintain the diversity of the population. This may yield to not landing in local optima [16]. Finally we eliminate newly differentiated lymphocytes carrying low affinity antigenic receptors.

IV. PROPOSED APPROACH

Before describing the proposed algorithm, we first introduce symbols and assumptions used in the paper.
Just like the original genetic algorithm suppose there are a population of solutions in each generation, rather than one solution in each step. Also each solution in the population is considered as an individual in the population and the final goal is improvement in the individual's genes in each generation.

Each individual in the set is mapped to a process graph; so that:

For each individual there exists a graph G:
{Node (i); 0 < i < L}

Each individual has various genes that are various nodes in the process graph. Genes have several properties that their mapping to the process graph is the various parameters such as execution time, implementation area, power consumption and so on. In addition, each node has two types of specific implementations that are software implementation and hardware implementation. With these assumptions, each gene is encoded as a binary number which may be zero for hardware implementation and one for software implementation or vice versa.

In CSPA algorithm, we use the process of generating defender cells in the body. These cells are mapped to good solutions in the partitioning problem. So the goal in our application domain is achieving antigenic pattern with a difference that we have not pathogen, but know its features. In this manner, antibodies after production in each step of algorithm are proliferated or if they didn't match gradually decreased. According to [15, 18], iterations in AIS are obtained from the following formula:

$$\frac{dx_i}{dt} = \left[\begin{pmatrix} antigens \\ recognised \end{pmatrix} - \begin{pmatrix} death \\ rate \end{pmatrix} \right] = \left[k_2 \left(\sum_{j=1}^{N} m_{ji} x_i y_j \right) - k_3 x_i \right] \quad (2)$$

Where:
N is the number of antigens, or the same good solutions,
x_i is the concentration of antibody i.
y_j is the concentration of antibody.
k_2 is the stimulation effect and k_3 is the death rate.
m_{ji} is matching function between antibody i and antigen j.

Some symbols used in the paper are as follows:
Each antibody in the individual set of each generation is presented with *Ab(t)* and antigens with *Ag(t)*. The affinity function; that shows how much a solution is good, is presented with *AF(t)*.

The proposed algorithm is as follows:

1- Generate an initial population *Ab(0)* with randomly encoding, from B-cells involving antibodies or in our analogy the partitions. Set the iteration generations equal to zero. Encoding in this work involves binary encoding, with the probability 0.5 for each one.
Gen = 0, t = 0, N = number of antibodies.
2- Initial evaluation – Intrinsic affinity of each immature antibody is calculated. This is done via affinity function and its calculation is in section VI. The value of this function constitute a vector and we have:
Af (t) = { Af_j (j) ; such that j is between 0 and N }
3- M antbodies with highest affinity are selected from *Ab* set and constitute a new set $Ab(t)_{\{M\}}$ of high affinity antibodies.
4- *Clonal Expansion- M* antibodies in $Ab(t)_{\{M\}}$ are cloned according to their affinities and form a set namely *Ac* of clones. This process is performed according to the proportions. The higher the affinity, the larger the clone size for each of the M selected antibodies. This statement is given by equation 3. From the following formula it should be considered that each antibody with smaller number has more antigenic affinity.

$$N_{Ac}(j) = \left\lceil \frac{\alpha . N}{j} \right\rceil, \qquad N_{Ac} = \sum_{j=1}^{M} N_{Ac}(j) \tag{3}$$

In this formula:

N_{Ac} *(j)* is the number of clones for j th selected antibody, i.e. $Ab(t)_{\{M\}}(j)$.

N is the number of total antibodies in the *Gen* generation.

M is the number of superior antibodies which are selected for clonal expansion.

N_{Ac} is the total generated clones and finally,

α is the coefficient factor that used for adjusting to the application domain.

5- *Mutation*- generated clones are go through mutation process and make the Ac^* set. Mutation is actually a change in one of the gene patterns in the individual. It is one kind of clonal expansion. It includes genetic mutation of clones and is inversely proportional to their antigenic affinities and number of iterations in the algorithm. This is viewed in equation 4. The higher the affinity of an individual, the lower mutation rate.

$$Ac^*(j) = Ac(j) \; except \; with \; randomly \; k \; bits \; flipped$$
$$k = \left\lceil \frac{\beta . L}{j * t} \right\rceil \tag{4}$$

In this equation, L is the number of coded information of the cell that is expressed in many applications as binary strings and in our work is the number of the nodes of process graph.

j is the number of selected antibodies according to the affinity function.

β is the coefficient factor that is used for adjusting to the application domain.

6- *Evaluation*- the affinity AF^* with relation to new immature antibodies is calculated.

7- *Learning process*- from AF^*, those individuals with highest affinity are selected and added to MB memory set. Solutions are retained in the memory for the future reference.

8- *Negative selection*- all antibodies are filtered by negative selection. Some of them are excluded, because they match one or more self-individuals, while others are regarded as mature antibodies and enter the system as a learned group. The matching rule could be hamming distance; r-continuous-bits matching rule and so on.

9- *Updating*- worst solutions by K percent are added to the self-set and self-set is updated just like a queue.

10- e antibodies with lowest affinity are selected from *Ab* and removed from current generation.

11- *Selection*- N antibodies of new generation are selected from past antibodies (those generated in the first step) and intermediate antibodies and compose the *Ab(t)* of t generation. This selection can be roulette wheel or tournament selection.

12- If the number of iterations is not reached, Gen = Gen + 1, t = t + 1 and go to step 4. Otherwise best individual between the current generation Gen and the memory MB is selected as final solution.

V. Algorithm Ability to Escape from Local Optimal

Most of the traditional algorithms in partitioning domain and evolutionary methods are not capable to escape from local optimal and so they can result in suboptimal solutions. In this section we argue that our algorithm is capable to escape from local optimal in normal situations. This prove is based on the work in [1].

Suppose there is a local optimum in current population Ab_i that our algorithm is converged to it and individual x_{local} that is a local optimum is in it. Because of similarity in the pattern of various individuals, most algorithms can accept the population as the final solution. But CSPA algorithm can escape according to following method:

We suppose antibody set is $Ab_t = \{x_t^1, x_t^2, \ldots, x_t^N\}$, at the end of the current generation and after mitosis, mutation and final selection for the next generation, in this generation K individuals with lowest affinities are added to current self-set. This is prior to final selection step. Also the (K < N) relation is true. These individuals are: $\{x_t^1, x_t^2, \ldots, x_t^K\}$.

By selecting a suitable K, when the algorithm reaches at local optimal for Ab_i population, it finds one individual $x_t^a \{1 \le a \le K\}$ in the self-set so that:

$$\left| x_t^a - x_{local} \right| < \theta \tag{5}$$

Where θ is a positive number. This is because of similarity between patterns of current generation that led to locality of solutions.

In the next iteration of the algorithm the antibody set is: $Ab_{t+1} = \{x_{t+1}^1, x_{t+1}^2, \ldots, x_{t+1}^N\}$.Because of negative selection, no individual in the generation matches the self-set $\{x_t^1, x_t^2, \ldots, x_t^K\}$ according to matching rule. So if the matching threshold is selected suitable value $\eta \ge 2\theta$, we have the following formula:

$$\left| x_{t+1}^i - x_t^a \right| > 2\theta, \qquad 1 \le i \le N. \tag{6}$$

So:

$$\left| x_{t+1}^i - x_{local} \right| = \left| (x_{t+1}^i - x_t^a) - (x_{local} - x_t^a \right| \ge \left| x_{t+1}^i - x_t^a \right| - \left| x_{local} - x_t^a \right| > 2\theta - \theta = \theta. \tag{7}$$

We consider that because of negative selection, individuals in the *t+1* generation can escape from local optimal. And so on this algorithm is escaped from local optimal step by step.

VI. COST OR AFFINITY FUNCTION

Affinity function in terms of immune systems shows the suitability of one implementation that is the reverse of cost function in our work. To obtain the cost of a system, one should find parameters that lead to cost in digital systems and then enter them into the function.

The partitioning algorithm is guided by this cost function that allows evaluating the quality of a given solution. The characteristics which are taken into account in our cost function are space, execution time and power.

Execution time criterion- It is obvious that execution-time reduction is one of the most important factors that needed to be thought about it in each step of creating a system, especially in design phase. In codesign systems this factor is computed for each node, when constructing process graph and partitioning the system to appropriate component regarding its mapping choice.

In the communication cost of the system, the cost of hardware-hardware interfacing and also software-software nodes is usually negligible and not considered. The only important cost in the node communication is between hardware and software nodes. For each node we also consider the number of executions in each run of the system for execution time and power consumption. But area is considered only one.

With these descriptions we express the execution time such as [9].

$$T = \sum_{i=1}^{N_s} (ts_i * iter_i) + \sum_{i,j} comm_{ij} \tag{8}$$

Where T is the overall execution time of the system, N_s is number of nodes implemented in software, ts_i is the execution time of software node i, $iter_i$ is the iteration of node i, and $comm_{ij}$ is the time required for communication between nodes with various implementation options.Now we consider the influence of each of these factors in the affinity function.

We use normalization to accommodate the conflicting factors in one term. In addition we apply a coefficient which adjusts the importance of various factors with regard to designer needs. We select the cost function as [10] and the affinity function as [11] that is the reverse of the cost function plus 1 to stands between zero and one.

$$\begin{aligned} affinity &= \frac{1}{1+\cos t\_func} \\ \cos t\_func &= \alpha \exp \frac{(T - Time\ \mathrm{Re}\, q)}{M_i \sigma_t} \\ &+ (1-\alpha)\frac{C}{\sigma_c} + (1-\alpha)\frac{P}{\sigma_p} \end{aligned} \tag{9}$$

where:

$$\begin{aligned} \sigma_c &= \cos tHW - CostSW \\ \sigma_t &= \max(TimeSW - Time\ \mathrm{Re}\, q, \\ &Time\ \mathrm{Re}\, q - TimeHW) \\ \sigma_p &= powHW - powSW \end{aligned} \tag{10}$$

Also in this formula T, C and P are execution time, hardware area cost and power consumption for a given partitioning respectively. TimeSW and TimeHw denote overall execution time of the system when all nodes are in software and hardware respectively. CostSW and CostHW denote overall cost of area when all nodes are in software and hardware respectively and finally powSW and powHW are just like the above.

TimeReq is the system time constraint given by designer. α is the coefficient that adjusts cost and time factors. We chose it 0.5. We adaptively select Mi equal to 1 and decrease it with the equation $M_{i+1}=0.96M_i$ like in the [1].

Space criterion- optimizing this factor is needed for producers and customers and for device portability. This factor gives system manufacturers with the reduction in the hardware resources. General purpose processor is not considered as a hardware resource to account for. So we don't consider it in calculating area of the system. The reason is that its area is negligible and also there is one from it in our system and we must have at least one processor to execute the software code. We take into account the area of the special-purpose hardware such as FPAGs and ASICs. We have:

$$C = \sum_i ah_i + \sum_{i,j} Acomm_{ij} \tag{11}$$

where C is the total cost of manufacturing area, ah_i is the area of nodes implemented on the hardware platforms and $Acomm_i$ is the area of buses.

Power criterion- This criterion is of extra importance especially in the mobile systems. Although in every system, its growth leads to lose in energy and needs of recharging the device with some resource of energy to continue its work. Growing up power consumption is not acceptable, especially in battery-enabled devices. We calculate the power consumption with following equation:

$$P = \sum_i (pnode_i * iter_i) + \sum_{i,j} Pcomm_{ij} \tag{12}$$

Where P is the total power consumption of system, $pnode_i$ and $iter_i$ are power consumption of node and number of iterations of that node respectively. And $Pcomm_{ij}$ is the power consumed in node's communication.

Scheduling - Process scheduling means specifying the start time for every entity in the system [12]. System scheduling for executing operations is one of the important problems in partitioning algorithms that in many cases are not considered. Actually we cannot have exact estimation about execution

time of a system without having enough information about scheduling. We used a list scheduling algorithm [1]. This scheduling is applied for each new solution and its result is the execution time of it.

VII. Experimental Results

In this section the performance of the proposed algorithm is evaluated with the results obtained from simulation experiments. Self-set population is selected from the 20 percent of the overall population for each generation. Best solutions that are cloned contain 5 percent of total individuals in current generation. Also *r* selected 0.6N where N is the graph size and we considered G = 20.

Our algorithm was written in C++ code and evaluated on several test samples. We used TGFF software for benchmarking of our work. Our parameters for graph sizes applied to this software were 20, 50 and 100. And graphs were near to these sizes after creation by TGFF. The experiments on each graph size were taken from more than 20 different independent runs. We compared our work with two standard algorithms: genetic and traditional AIS, with the same graphs. The results are shown in table I. In this table the average cost is based on the affinity function and execution time denotes the average total execution time for each graph size.

Table I shows that we have good improvements in final costs. Compared to AIS and genetic algorithm, the solutions have better quality. Increase in execution times compared with AIS is observed, but its rate per graph size is slow. This is due to the extra phases included in this algorithm for clonal selection and learning.

TABLE I
Results of Cspa Compared with Two Other Algorithms

N=	AIS cost	GA cost	CSPA cost	AIS exe. Time (sec)	GA exe. Time (sec)	CSPA exe. Time (sec)
20	0/8622	0/8570	0/8663	1.432	4.269	101
50	0/8974	0/8741	0/9014	3.453	6.25	3.500
100	0/8563	0/8634	0/8740	5.650	9.536	5.874

VIII. Conclusion and Future Works

Partitioning for digital systems that consist of combinations of hardware and software components is one of the most fundamental steps to reach the good performance of these systems. In this paper a heuristic approach inspired by nature called CSPA was proposed. Our algorithm uses the clonal selection of body immune system which is a new idea in this area. Results indicate that the quality of solutions in this approach is better than the evolutionary and traditional immune-based algorithm. In the future works, this approach can also be used in the area of reconfigurable systems to improve several design steps in such systems where they need optimization.

References

[1] Yiguo Zhang, Wenjian Luo, Zeming Zhang, Bin Li, Xufa Wang, "A hardware / software partitioning algorithm based on artificial immune principles," *Applied Soft Computing* 8 (2008) 383–391.

[2] J. Javier Resano, M. Elena Pe´rez, Daniel Mozos, Hortensia Mecha, Julio Septien, "Analyzing communication overheads during hardware/software partitioning*," Microelectronics Journal* 34 (2003) 1001–1007.

[3] Dan W Taylor, David W Corne, "An Investigation of the Negative Selection Algorithm for Fault Detection in Refrigeration Systems," Artificial Immune Systems, Second International Conference, ICARIS 2003, Edinburgh, UK, September 1-3, 2003, Proceedings. Lecture Notes in Computer Science 2787 Springer 2003, ISBN 3-540-40766-9, pp.34-45.

[4] Xian Shen, X. Z. Gao, and Rongfang Bie, "*Artificial Immune Networks: Models and Applications*," *International Journal of Computational Intelligence Systems*, Vol.1, No. 2 (May, 2008), 168–176.

[5] Ashraf E. Alfagih, "Artificial Immune Systems," *CISC*-491/879 - Unconventional Computing, November 29, 2007.

[6] Kazushi Igawa and Hirotada Ohashi, "Discrimination-based Artificial Immune System: Modeling the Learning Mechanism of Self and Non-self Discrimination for Classification," *Journal of Computer Science* 3 (4): 204 -211, 2007, ISSN 1549-3636.

[7] Asawaree Kalavade and Edward A. Lee, "The Extended Partitioning Problem: Hardware/Software Mapping, Scheduling, and Implementation-bin Selection," *Journal of Design Automation of Embedded Systems*, vol 2, no.2 pp 226-163, Mar 1997.

[8] Jörg Henkel et al., "Adaptation of Partitioning and High-Level Synthesis in Hardware/Software Cosynthesis," *ICCAD*' 1994.

[9] D. Saha, R. S. Mitra, Anupam Basu, "Hardware Software Partitioning using Genetic Algorithm," *10th International Conference on VLSI Design*, January 1997.

[10] T. Wiangtong, P. Y. Cheung, and W. Luk, "Tabu Search with Intensification Strategy for Functional Partitioning in Hardware-Software Codesign," *Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* (FCCM'02), 2002.

[11] Y. Zou, Z. Zhuang, H. Chen, "HW–SW partitioning based on genetic algorithm," *Proceedings of the CEC'04*, 2004, pp. 628–633.

[12] Mouloud Koudil, Karima Benatchba, Amina Tarabet, El Batoul Sahraoui, "Using artificial bees to solve partitioning and scheduling problems in codesign," *Applied Mathematics and Computation* 186 (2007) 1710–1722.

[13] J. Timmis, A. Hone, T. Stibor, E. Clark, "Theoretical advances in artificial immune systems," Theoretical Science 403 (2008) 11–32.

[14] Uwe Aickelin, "Artificial Immune Systems (AIS) – A New Paradigm for Heuristic Decision Making," *OR46 (2004) AIS for OR*, Keynote Speech.

[15] Farmer J, Packard N and Perelson A (1986), "The immune system, adaptation, and machine learning," *Physica*, vol. 22, pp. 187-204, 1986.

[16] Leandro N. de Castro, *Member*, *IEEE*, and Fernando J. Von Zuben, *Member*, *IEEE, "*Learning and Optimization Using the Clonal Selection Principle," *IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems*, vol. 6, n. 3, pp. 239-251, 2002.

[17] Leandro Nunes de Castro, Fernando J. Von Zuben, "The Colonial Selection Algorithm with Engineering Applications," *In Workshop Proceedings of GECCO'00*, pp. 36-37, *Workshop on Artificial Immune Systems and Their Applications*, Las Vegas, USA, July 2000.

[18] Uwe Aickelin, "Artificial Immune Systems (AIS) – A New Paradigm for Heuristic Decision Making," *OR46* (2004) AIS for OR, Keynote Speech.