Conférence Internationale en Automatique & Traitement de Signal (ATS-2017)
Proceedings of Engineering and Technology – PET
Vol.22 pp.96-102

New Multi-objective Approaches for Hardware/Software Partitioning Problem in Co-design process

Mouna Riabi[1], Yassine Manai[2] and Joseph Haggège[3]
Department of Electrical Engineering
National School of Engineers of Tunis
University of El Manar
Bp 37, le Belvédère 1002 Tunis, Tunisia
[1]mouna.riabi@gmail.com
[2]yacine.manai@gmail.com
[3]joseph.haggege@enit.rnu.tn

***Abstract*—Two algorithms for Hw/Sw partitioning problem resolution with multi-objective optimization were presented in this paper. This work has been conducted respecting several constraints for AC drive application. Initially, a new approach based on binary Genetic Algorithm (GA) was introduced in order to get the best compromise area/execution_time. Then, Artificial Bee Colony (ABC) used for the first time in this field, was proposed for improving the given solution. The simulation results concluded that our proposed approaches are efficients and gave better tradeoffs area/execution_time in comparison with recent results as well as Non-dominated Sorting Genetic Algorithm II (NSGAII).**

Keywords—Hardware/Software partitioning, System-on-Chip, heuristic approaches, FPGA, soft core processor.

I. INTRODUCTION

The Co-design process is used for mixed system design. It consists of a succession of steps starting with specification of the system modules to their synthesis [1]. Hardware/Software partitioning is an essential step in co-design method. It refers to the partitioning of a system specification into separate hardware and software modules [2]. Also, it decides what the best assignment of each component of an embedded system to hardware resources or software ones taking into account a various conflicting constraints.

The Hw/Sw Co-design is evolved in a way to automate all its phases especially the Hw/Sw partitioning process [3]. For this purpose, several approaches are proposed to solve different types of partitioning problems, with one objective optimization, and a multi-objective one which is the currently treated issue.

As the embedded system complexity increases and classified as NP-hard problem [1], [4], there is various techniques used to obtain exact solutions to these problems, including integer linear programming (ILP), Satisfiability Modulo Theories (SMT) [5]. Researchers have also used the meta-heuristic approaches which become crucial for obtaining an efficient partitioning of the control algorithm between software and hardware resources. These approaches are based on optimization algorithms, such as tabu search [6], [7], genetic algorithm [8], [9], [10] and ant colony algorithm [3], [11].

The previous cited works have focused on optimizing multi-objectives partitioning problem, citing the example of processing time, area, power consumption, which presents the major issue to be addressed [12], [13], [14].

Several proposed works have treated the problem for signal and image processing applications [13]. Otherwise, we notice the minority of these propositions for control system.

This paper will contribute in proposing Hw/Sw partitioning procedure for System-on-Chip (SoC) field programmable gate array (FPGA) which is based sensorless AC drive applications.

As it is known as a classical solution, simple, universal and robust, the genetic algorithm is proposed to solve multi-objective Hw/Sw partitioning problem.

Then, optimal partitioning solutions are obtained via Artificial Bee Colony algorithm which is considered, to our best knowledge, as a new method in this field.

The proposed approaches take into account heterogeneous constraints such as available area, execution time, memory and hardware multipliers. They are simulated and compared with a previous algorithm maintaining the same target architecture in [12].

The rest of this paper is organized as follows. Section 2 defines the related work. Section 3 explains the proposed algorithms for Hw/Sw partitioning. Section 4 presents the system modeling and section 5 shows simulated results and provides discussion that are presented to clarify the algorithms contribution. Finally, we terminate with a conclusion.

II. RELATED WORK

The primary work that has treated the Hw/Sw partitioning problem in embedded system is presented in [15]. Then, automatic method is used to solve partitioning problem and respond to the conditions above. It uses algorithms implemented in heterogeneous system that help designers to find the best partitioning solutions. Two kinds of automatic approaches exist to solve partitioning problem in co-design: exact search methods and approached ones.

ISSN 2356-5608

In the following, more details concerning types of approaches are presented.

A. Exact search methods

These algorithms are simple to implement. They allow for sure to determinate and evaluate all possible solutions, except that is only in theory. In [16], a dynamic programming algorithm is used to calculate the exact solution for Hw/Sw partitioning problem. But, it is mentioned that these kind of methods are suitable for small-sized problems. Otherwise, they become inadequate for larger-sized problems.

In our case, and for the partitioning problem, it is no longer possible to use this type of algorithm as the computing time grows with the number of tasks in heterogeneous systems [17].

B. Approached methods

Since the Hw/Sw partitioning is considered as NP-hard for more cases [16], heuristic approaches are generally used to perform the system partitioning and hardware exploration.

In literature, proposed meta-heuristic methods are divided into two classes: Firstly, meta-heuristics based on single solution such as Tabu search which is considered in [16] to refine solution for bi-objective partitioning problem. Also, Simulated Annealing in [6] which is combined with Tabu search in order to solve Hw/Sw partitioning problem for reconfigurable system. Secondly, the class of meta-heuristics based on population of solution, starting with evolutionary algorithms in [8] where the author used Genetic Algorithm (GA) for the partitioning and scheduling of task graph taking into account the timing constraint and the configuration delay. Also, in [12], Non-dominated Sorting Genetic Algorithm II aims to solve partitioning problem in Co-design process. Arriving to algorithms based on swarm intelligence [18], where Ant colony optimization is the most popular algorithm applied on Hw/Sw partitioning problem [19], [3]. Moreover, artificial bees are used to solve scheduling and partitioning problem and gives better results in comparison with genetic algorithm and exact method [17]. We notice that evolutionary algorithms are more developed in this field and allows an improvement of the different decision variables.

In the rest of the paper we develop two evolutionary methods with the consideration of system constraints, multi-objective optimization and the communication delay which include all problems treated in the presented previous works.

III. PROPOSED APPROACHES

A. First approach based on Genetic Algorithm

As the main goal of this paper is to solve Hw/Sw partitioning problem, several improvements are made to genetic algorithm developed in [20]. This algorithm is used to find the optimal assignment of modules to hardware or software resources with the optimization of four principal parameters: Area, Execution Time, Memory block and Hardware Multiplier.

As shown in Fig.1, the developed approach follows the main steps of binary genetic algorithm [20] with several modifications to accommodate the multi-objective problem and meet different constraints.

In order to get diversity, an initial population of chromosomes is generated randomly. After decoding the vector of chromosome, the population is evaluated using the multi-objective function. The best solution is a vector that contains the minimum of the four parameters, it is calculated and retained to the next generation. Then, the use of operators (selection and mutation) will guarantee the improvement of solutions. This procedure is iterated until finding the best solution provided when the maximum generations are not reached or the load limits are not exceeded.

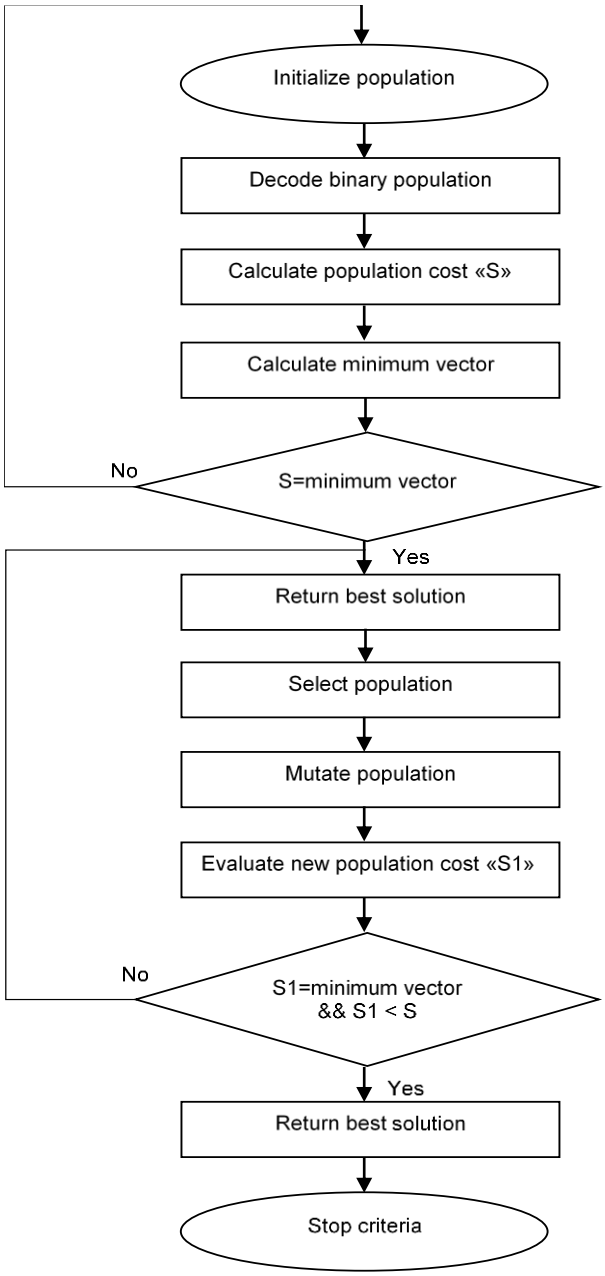

Fig. 1 Main program of proposed genetic algorithm

ISSN 2356-5608

The obtained results represent the optimized four parameters (area, execution time, memory, hardware multiplier) and the binary vector which represents the modules to be implemented in software resources, if it is equal to 0, or in hardware resources, if it is equal to 1. In our case the vector size corresponds to the modules number (n = 15) [20].

B. Approach based on Artificial Bee Colony

1) Basic Artificial Bee Colony

The artificial bee colony algorithm was proposed in 2005 by Karaboga [21]. It is a new meta-heuristic that has enriched the number of optimization methods based on swarm intelligence.

The colony of artificial bees is inspired by the intelligent behavior of a honey bee colony in nature. It is composed of three groups of bees: employed bees, onlooker bees and scout bees. The employed bees exploit new sources of food and memorize them. Based on the information submitted by the latter, the onlooker bees make a decision for choosing the best sources to exploit. For the scout bees, they are carrying out random search of sources. As described by Karaboga: In the Artificial Bee Colony (ABC) algorithm, each cycle of the search consists of three steps: sending the employed bees onto the food sources and then measuring their nectar amounts; selecting of the food sources by the onlookers after sharing the information of employed bees and determining the nectar amount of the food; determining the scout bees and sending them onto possible food sources.

There is minimal control parameters in ABC algorithm which are described as below:

- Population_Number: The number of colony size corresponding to the sum of employed bees and onlooker bees,
- Food_Number: The number of food sources equals to the half of the colony size,
- Limit: A food source which could not be improved through "limit" trials is abandoned by its employed bee,
- Max_Cycle: The number of cycles for foraging, a stopping criteria.

The main steps of basic ABC algorithm are given below [21].

```
Initialize
Repeat
   Employed Bee phase;
   Onlooker Bee Phase;
   Scout Bee Phase;
   Memorize the best solution;

Until (max_Cycle).
```

2) ABC algorithm for Hw/Sw partitioning

While several modifications to the original ABC algorithm have been made to solve global optimization problem, Hw/Sw partitioning approach has not been explored in the literature yet to the best of our knowledge.

In this paper, we join the basic ABC algorithm definition with several modifications in order to get better results of optimization solutions for Hw/Sw partitioning.

The main goal of proposed ABC approach is to solve a multi-objectives Hw/Sw partitioning problem taking into account different constraints that give better compromise area/execution_time.

The solutions of multi-objective optimization problem correspond to the food sources positions. Each position is qualified by the nectar amount which corresponds to the quality (fitness) of the solution.

The main steps of the proposed ABC algorithm are described as follows: Firstly, an initial population is generated randomly where the number of employed bees equals to the number of positions. Then, three search processes are repeated until reaching the maximum number of cycles; employed bees process, onlooker bees process and scout bees process. In the employed and onlooker bees processes, new sources are generated and their nectar amount are tested until reaching the best solution. In the scoot bees process, the food sources which nectar is abandoned by the bees is replaced with a new source [21]. The latter steps, which are applied on the four parameters to be optimized in the Hw/Sw partitioning process, are described by the flowshart in the Fig.2.

For more detail, the main program is outlined below.

```
Repeat
    Initialize the population Xij, i = 1. .NP (Number of Population), J = 1. .D (Parameters Number);
    Generate randomly the binary vector Mi (Mi = 0 if the task i must be assigned to the SW and Mi = 1 if the task i must be assigned to HW) [20];
    Evaluate population;
Until (best solution = the minimum vector of four parameters)
    Memorize the best solution;
Cycle = 1
Repeat
    Employed Bee Phase:
        Produce new solution Vij for "employed bees";
        Apply the selection process;
        Calculate the probability Pij for solutions Xij;
    Onlooker Bee Phase:
      Repeat
        Produce a new solution Vij for "onlooker bees" from the solutions Xij, using the probability Pij and evaluate them;
        Apply the selection process to the four parameters;
      Until (best new solution = the minimum vector of four parameters)
        Compare new solution with the previous one;
```


ISSN 2356-5608

Scoot Bee Phase:

Determine the abandoned solution whose trial counter exceeds the "limit" value and replace it with a new solution Xij;

Return the best solution and the corresponding binary vector;

Cycle = Cycle +1

Until (MCN = Max cycle or limits constraints reached).

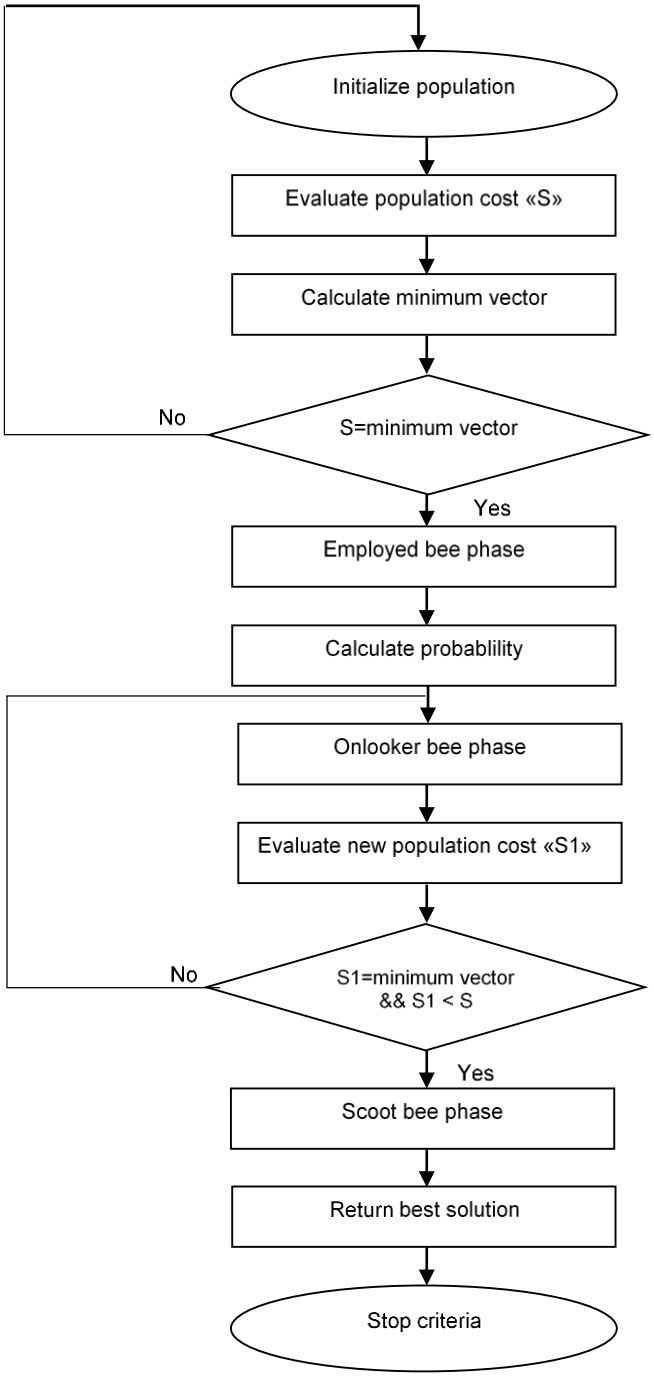

Fig. 2 Main program of proposed ABC algorithm

IV. SYSTEM MODELING

A. Target architecture

The used system is based on an FPGA platform that consists on the Xilinx Virtex-5 embedding a Microblaze soft-core processor. It is decomposed into subsystems in specification step. The goal of the latter is to define the characteristic and the granularity level for each module in order to simplify the partitioning process [12].

In this paper we keep the same granularity level used in [12] which define the functional modules characterized by a set of metrics used as cost function parameters where :

- *Ai*: consumed resources of module i, in the case of hardware implementation (6-bLUT & FF),
- *Hi*: memory blocks used by module i, in the case of software implementation,
- *thi*: execution time taken by the module i when executed in hardware,
- *tsi*: execution time taken by the module i when executed in software,
- *HMi*: consumed hardware multiplier (DSP units) of module i, in the case of hardware implementation,
- *Ii/Oi*: number of inputs and outputs of module i.

B. Multi-objective cost function

In this part, the multi-objective character of this optimization problem is described as in [20].

In the following, basic terms and parameters used in (7) are defined to conceive the quality of the obtained solutions.

- $M = \{M_1, M_2, \ldots, M_n\}$: *n* functional modules group (*n* is here equal to 15),
- $(w_1, w_2, \ldots, w_n)$: binary vector that represents the solution of the partitioning problem where $w_i \in \{0,1\}$. $w_i=1$ (resp. $w_i=0$) means that the i^{th} module has to be implemented in Hw (resp in Sw),
- $y = [y_1, y_2, \ldots, y_{n-1}]$: this parameter depends on the scheduling and the data dependency between the different modules. Each of these components is characterized by a binary value. $y_i=1$ (resp. $y_i=0$) when M_i can be executed in parallel (resp.in series) with M_{i+1},
- $A_{\mu p}$: hardware resources consumed by the Microblaze soft-core, the associated bus and the peripherals. Note that the Sw implementation of one module requires the whole use of $A_{\mu p,}$
- *Area*: total consumed hardware resources,
- HM_{blocks}: total consumed hardware multipliers (DSP units),
- $HM_{\mu p}$: total consumed hardware multipliers (DSP units) by the processor,
- *Mem*: total memory use,
- T_{ex}: total execution time,
- T_{com}: communication time between the module *i* and the other modules.

ISSN 2356-5608

The communication model used in (6) is detailed as follow [22]:

- C_i^{hs} : communication time from a Hardware module M_i to a software module M_{i+1}.

$$C_i^{hs} = \mathrm{Re}ad\_cycle \times O_i \quad (1)$$

- C_i^{sh} : communication time from a software module M_i to a hardware module M_{i+1}.

$$C_i^{sh} = Write\_cycle \times O_i \quad (2)$$

- C_i^{hh} : communication time from a hardware module M_i to a hardware module M_{i+1}.

$$C_i^{hh} = Clock\_cycle \quad (3)$$

- C_i^{ss} : communication time from a software module M_i to a software module M_{i+1}.

$$C_i^{ss} = Clock\_cycle \times O_i \quad (4)$$

Where [22]

$$Read\_cycle = Write\_cycle = 3 \times Clock\_cycle \quad (5)$$

$$T_{com} = \sum_{i}^{n-1} (1-y_i) \times \begin{bmatrix} (1-w_i) \times (1-w_{i+1}) \times C_i^{ss} \\ +w_i \times w_{i+1} \times C_i^{hh} \\ +(1-w_i) \times w_{i+1} \times C_i^{sh} \\ +w_i \times (1-w_{i+1}) \times C_i^{hs} \end{bmatrix} \quad (6)$$

$$\begin{cases} Area(x_1, w) = \sum_{i=1}^{n} w_i \times A_i + \sum_{i=1}^{n} (1-w_i) \times A_{\mu p} + (x_1)_i \\ HM_{blocks}(x_2, w) = \sum_{i=1}^{n} w_i \times HM_i + \sum_{i=1}^{n} (1-w_i) \times HM_{\mu p} + (x_2)_i \\ Mem(x_3, w) = \sum_{i=1}^{n} (1-w_i) \times H_i + (x_3)_i \\ T_{ex}(x_4, w) = \sum_{i=1}^{n-1} \begin{bmatrix} y_i \times w_i \times w_{i+1} \times \max(t_{hi}, t_{hi+1}) \\ +y_i \times w_i \times (1-w_{i+1}) \times \max(t_{hi}, t_{si+1}) \\ +y_i \times (1-w_i) \times w_{i+1} \times \max(t_{si}, t_{hi+1}) \\ +y_i \times (1-y_i) \times (1-w_{i+1}) \times \max(t_{si}, t_{si+1}) \\ +(1-y_i) \times (1-y_{i-1}) \times w_i \times t_{hi} \\ +(1-y_i) \times (1-y_{i-1}) \times (1-w_i) \times t_{si} \end{bmatrix} \\ +(1-y_{n-1}) \times \left[w_n \times t_{hn} + (1-w_n) \times t_{sn} \right] + T_{com} + (x_4)_i \end{cases} \quad (7)$$

In order to optimize the cost function, we must considered the following requirements:

$$\begin{cases} Area < S \\ Mem < H \\ HM_{blocks} < HM \\ T_{ex} < T_{alg} \end{cases} \quad (8)$$

where S, H, and HM are respectively the area, memory size, and number of hardware multiplier available in the FPGA. Talg corresponds to the maximum time delay that satisfies the control performance requirements (bandwidth and stability margin) [20].

V. SIMULATION RESULTS AND ANALYSIS

To make a fare comparison between the two proposed algorithms and Non-dominated Sorting Genetic Algorithm II (NSGAII), the same multi-objective function proposed as in [12] is implemented depending on the functional modules characteristics. In this field, we test the proper operation of both genetic algorithm and artificial bee colony.

Assume that each module i requires resources for a hardware implementation or software one, we should respect the different constraints defining the limit of consumed resources. The proposed algorithms GA and ABC are simulated in Matlab following the same requirements presented in [22], where area constraints in terms of LUTs is 7500, the maximum hardware multipliers is fixed to 288 DSP, the memory used doesn't exceed 4752 Kb, and the maximum execution time is set to 3791 clock_cycle.

A random w_i vector generated by the similar manner for both GA and ABC algorithm in order to make the results comparable in terms of scheduling and partitioning. The latter schedule the modules to be mapped in hardware or software resources as it details in Fig.3 below:

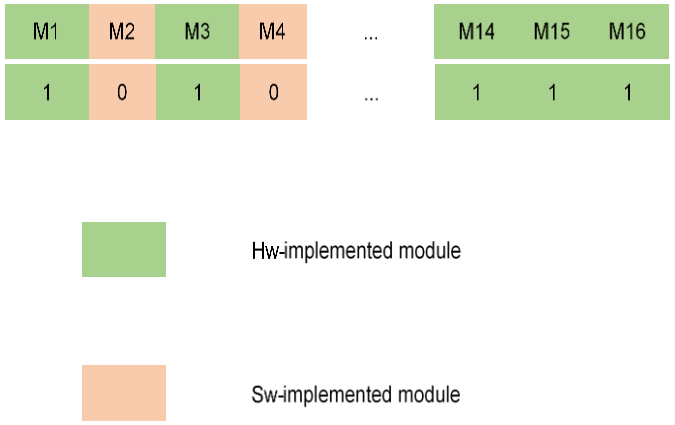

Fig. 3 Example of scheduling and partitioning representation of modules

The two algorithms aims to obtain the best solution of Hw/Sw resources assignment with the optimization of four principal parameters: area, execution time, hardware multipliers and memory blocks.

The GA configuration was done considering the following parameters values: number of generations = 1000, population = 100, selection rate =0.5, and mutation rate = 0.15.

ISSN 2356-5608

For the given problem, the solution produced by ABC mainly depends on the following control parameters; the number of colony size, the number of food sources, limit and the number of maximum cycle which are respectively configured as below:

NP = 20, FoodNumber = 10, Limit = 100 and maxCycle = 1000.

In the following, we focus on area/execution_time tradeoffs which depend on the binary vector. The Table I shows the quality of approximate solutions with different way of modules partitioning and scheduling which are represented by the binary vector generated randomly in each execution.

We notice that the implementation of more modules in hardware resources procures a reduction of the execution time but it increases the consumed resources. In addition the Fig.4 shown the four solutions respecting the constraints limits for both execution time and area. The results of GA and ABC are close and provide a good number of feasible solutions.

TABLE I
EXAMPLE OF OPTIMIZATION SOLUTIONS

	Binary Vector	GA		ABC algorithm	
		Area	***Tex***	***Area***	***Tex***
S1	111111111111100	7488	1461	7500	1486
S2	101011111101011	7401	2067	7428	2095
S3	011101101111000	7105	2360	7128	2415
S4	001001101111000	7019	3228	7027	3226

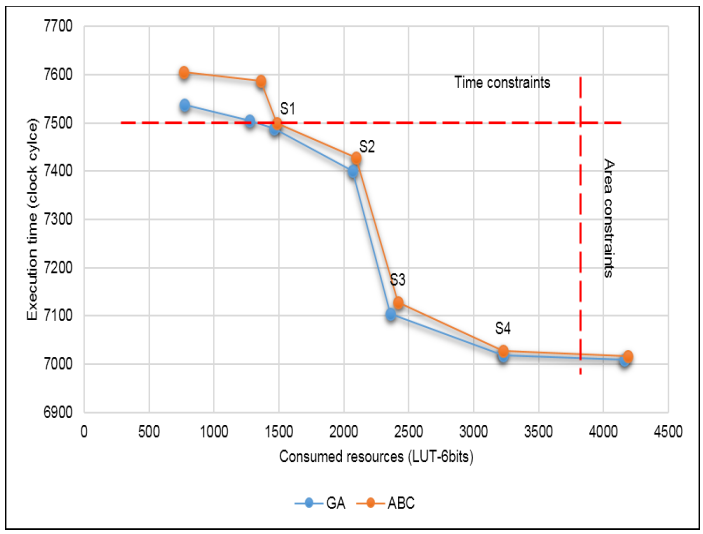

Fig. 4 Example of feasible solutions

We can also explore the contributions of ABC algorithm and genetic algorithm by comparing them with NSGAII. Firstly the proposed algorithms can reduce the execution time for about 2 times less than NSGAII as shown in Table II.

Secondly, the latters can implement the system with minimal hardware resources at the same time it reduces the used memory blocks compared to NSGAII.

In conclusion, the proposed algorithms may refine the NSGAII solution for better ones by optimizing 3 of the 4 objectives parameters as shown in Table II.

TABLE II
COMPARISON BETWEEN OPTIMIZATION SOLUTION OF GA, ABC AND NSGAII

	NSGAII [12]	**GA**	**ABC algorithm**
Binary vector	111111000111111	011101101111000	011101101111000
Area	7134	7105	7128
Hardware Multiplier	33	178	184
Memory	103	99	102
Execution time	5768	2360	2415

VI. CONCLUSION

In the embedded system design, especially in the partitioning process, each task can be implemented in multiple ways in the software and in the hardware, where variant implementations have different optimization performances.

This paper has introduced a multi-objective Hw/Sw partitioning problem on control system, which can achieve multiple conflicting design goals.

We optimize the solution of the partitioning problem in order to obtain the best tradeoffs area/execution_time while meeting several constraints. In this case, we have proposed two approaches, a classical heuristic based on genetic algorithm which rapidly generates a global solution and a new one in this field based on bee colony.

It has been shown that our heuristic algorithms refine the solution obtained by NSGAII and give us a better tradeoffs area/execution_time.

Generalizing the implementation of these algorithms in multiprocessor system-on-chip (MPSoC) is one of the direct perspectives of this work. Another one is to propose other approaches or hybrid algorithms for multi-objective partitioning problem.

REFERENCES

[1] Y. Manai, Contribution à la conception et la synthèse d'architectures de systèmes embarqués utilisant des paltes-formes hétérogènes. Ph.D.Thesis, Ecole Nationale d'Ingénieurs de Tunis, Tunisia, 2009.

[2] W. Ahmed, "Concept-based partitioning for large multidomain multifunctional embedded systems," ACM Transactions on Design Automation of Electronic Systems, vol. 15, no. 3, Article 22, 2010.

[3] Y. Manai, J. Haggege, M. Benrejeb, "New approach for hardware/software embedded system conception based on the use of design patterns," Journal of Software Engineering & Applications, pp. 525-535, 2010.

[4] B. Knerr, M. Holzer, M. Rupp, "A fast rescheduling heuristic of SDF graphs for HW/SW partitioning algorithms," 1st International Conference on Communication System Software and Middleware, New Delhi, India, 2006.

ISSN 2356-5608

[5] M. Yuan, Z. Gu, X. He, “Hardware/Software partitioning and pipelined scheduling on runtime reconfigurable FPGAs,” ACM Transactions on Design Automation of Electronic Systems. vol. 15, no. 2, Article 13, 2010.

[6] P. Liu, J. Wu, Y. Wang, “Integrated heuristic for Hardware/Software co-design on reconfigurable devices,” 13th International Conference on Parallel and Distributed Computing, Applications and Technologies Beijing, pp. 370-375, 2012.

[7] L. Cui, “A novel approach to hardware/software partitioning for reconfigurable embedded systems,” Journal of Computers. vol. 7, no. 10, pp. 2518-2525, 2012.

[8] B. Mei, P. Schaumont, S. Vernalde, “A hardware-software partitioning and scheduling algorithm for dynamically reconfigurable embedded systems,” 11th ProRISC workshop on Circuits, System and signal Processing, Veldhoven, Netherlands, 2000.

[9] B. Knerr, M. Holzer, M. Rupp, “Novel genome coding of genetic algorithms for the system partitioning problem,” International Symposium on Industrial Embedded Systems, Lisbon, pp. 134-141, 2007.

[10] A. Farmahini Farahani, M. Kamal, M. Salmani-Jelodar, “Parallel-genetic-algorithm-based hw/sw partitioning,” International Symposium on Parallel Computing in Electrical Engineering, Bialystok, 2006.

[11] G. Wang, W. Gong, R. Kastner, “Application partitioning on programmable platforms using the ant colony optimization,” Journal of Embedded Computing 1, pp. 1-18, 2005.

[12] I. Bahri, L. Idkhajine, E. Monmasson, “Hardware/Software codesign guidelines for system on chip FPGA-based sensorless AC drive applications,” IEEE Transaction on Industrial Informatics, vol. 9, no. 4, pp. 2165-2176, 2013.

[13] I. Alouani, B. Lotfi Mediouni, S. Niar, “A Multi-objective approach for software/hardware partitioning in reconfigurable embedded systems,” International Symposium on Rapid System Prototyping, Amsterdam, pp. 119-125, 2015.

[14] H. Youness, A. Hussein, A. Mahfoz, “A new hardware/software partitioning technique,” International Conference on Computer Engineering & Systems, Cairo, pp. 113-118, 2015.

[15] P. Arato, S. Juhasz, Z. Adam Mann, D. Papp, “Hardware-Software partitioning in embedded system design,” IEEE International Symposium on Intelligent Signal Porcessing, pp. 197-202, 2003.

[16] W. Shi, J. Wu, S.K. Lam, T. Srikanthan, “Algorithms for bi-objective multiple-choice hardware/software partitioning,” Computers and Electrical Engineering, pp. 1-16, 2016.

[17] M. Koudil, K. Benatchba, A. Tarabet, E.B. Sahraoui, “Using artificial bees to solve partitioning and scheduling problems in codesign,” Applied Mathematics and Computation, pp. 1710-1722, 2007.

[18] A. Farmahini-Farahani, M. Kamal, S. Mehdi Fakhraie, S. Safari, “HW/SW partitioning using discrete particle swarm,” 17th ACM Great Lakes Symposium on VLSI, Stresa, Lago Maggiore, Italy, 2007.

[19] G. Wang, W. Gong, R. Kastner, “A new approach for task level computational resource bi-partitioning,” 15th International Conference on Parallel and Distributed Computing and Systems, Marina del Rey, USA, 2003.

[20] M. Riabi, Y. Manai, J. Haggege, “Hardware/Software partitioning approach for embedded system design based on genetic algorithm,” International Journal of Scientific Research & Engineering Technology, vol. 3, issue 3, pp. 20-25, 2015.

[21] D. Karaboga, B. Basturk, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,” Journal of Global Optimization, pp. 459-471, 2007.

[22] I. Bahri, “Contribution of FPGA-based system-on-chip controllers for embedded AC drive applications,” Ph.D.Thesis, Universite De Cergy Pantoise, France, 2011.

ISSN 2356-5608