SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
VOLUME 5 NO 4, 2017
ISSN 2054-7390

Collective Behavior Bees for Solving HW/SW Partitioning and Scheduling Problems in RSoC

Yahyaoui Khadidja, Bouchoicha Mohammed
Department of math, Faculty of Exact Sciences, University of Mascara, 29000 Algeria
yahyaouiinfo@gmail.com; Bouchoicham_info@yahoo.fr

ABSTRACT

In the codesign domain, many hardware and software techniques must be developed to satisfy specific constraints in terms of computation time, area, performance, power consumption, etc. This paper introduces an automatic approach To perform HW/SW partitioning and scheduling such that the global application execution time is minimized and the majority area of FPGA (Field programmable gate array) used in RSoC (Reconfigurable System on Chip) is exploited. The used algorithm is inspired by the collective behavior of social insects such as bees. : Honey Bees Mating Optimization (HBMO). Comparing the proposed method with Genetic algorithm, the simulation results show that the proposed algorithm has better convergence performance.

Keywords—RSoC, HW/SW partitioning; scheduling; Honey Bee Mating Optimization; Genetic algorithm.

1 Introduction

Using a Reonfigurable System on Chip (RSoC) is increasingly common in embedded applications. RSoC is a circuit comprising multiple functions such as one or more processors, one or more reconfigurable devices like FPGA (of Field Programmable Gate Arrays), a signal processor DSP (Digital Signal Processor), various peripherals and memory or analog parts. These circuits are increasingly used because of their small size and reduced costs compared to the use of various circuits for performing the same function. However, the use of Field Programmable Gate Arrays (FPGAs) provides specific hardware technology, which can also be reprogrammable thus providing a reconfigurable embedded system. Thus, the corresponding circuit can be modified to adapt its functionality to perform different tasks. Field Programmable Gate Arrays (FPGAs) are defined as semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks;

The decision about the functional blocks to be implemented in software or in hardware in the dynamically reconfigurable embedded systems is based on the experience of the designer and/or making a brief exploration of the design space. This procedure, in addition to not complying with any methodology, does not ensure an optimal result, since for obtaining the best configuration it is necessary to solve an optimization problem which in most of its formulations is NP-hard [1]. In most cases, the HW/SW problem

DOI: 10.14738/tmlai.54.3205
Publication Date: 15th August, 2017
URL: http://dx.doi.org/10.14738/tmlai.54.3205

is driven by optimization aim such as the area of hardware, execution time or power consumption. On the other hand, some of these models use exact algorithms to obtain an exact solution to the problem, but the search time increases proportionally to the problem size. Taking into account that in some cases a near optimal solution is considered as good enough [2]. Consequently, many researchers employed heuristics based on Genetic Algorithms [3][4][5], Simulated Annealing [6], Tabu Search [7], Ant Colony Optimization (ACO) [8] and bee swarms [9] to obtain sub-optimal solutions. The IA technics

are also used to resolve the HW/SW partitioning and scheduling problem. In [10], the authors apply a Fuzzy Logic for Hardware/Software Partitioning in Embedded Systems. A Neural Networks Based Approach for the Scheduling of Reconfigurable Embedded Systems supporting FPGA as reconfigurable component is applied in [11]. However there are a problems which are resolved by the hybridization of some of these algorithms and they have given an optimal results. For the resolution of HW/SW partitioning and scheduling problem is not yet adopted [12, 13].

Scheduling is the process of determining a starting time for each software/hardware task of the system. When it used with partitioning in codesign, it is also allows to determine the execution time of the system under design when a partitioning solution is determined (mapping of the software and hardware implementation tasks on the architecture target supporting a FPGA). The two problems partitioning and scheduling are interrelated and tightly coupled. Each of one is known to be NP-Hard, and the fact that they have to be solved simultaneously complicates their resolution. This is why the only feasible way to solve them is to use heuristics,since exhaustive methods take a prohibitive execution time to find the best solution as soon as the system becomes large [9].

This work describes an automatic partitioning/scheduling approach that uses the combination of global search represented by HBMO as main strategy with two local searches methods: HW1 and HW2 in order to intensify the search in promising zones detected by the HBMO exploration process. The HW1 and HW2 approaches are based respectively on Tabu Search (TS) and Simulated annealing (SA). These algorithms are used as heuristics in the stage of the brood's improvement HBMO. The comparison with the genetic algorithm (GA) has been used.

The paper proceeds as follows. The presentation and formulation of HW/SW partitioning and scheduling problem is described in Section II. Section III presents the proposed Scheduling Approach. The resolution process of the HW/SW partitioning and scheduling problem is detailed in Section IV. The V section, the simulation and experimental results are given. The paper finishes with the conclusion presented in the section VI.

2 HW/SW Partitioning and Scheduling Problem Formulation.

Our algorithm target architecture contains one processor, one dynamically reconfigurable devise of type FPGA, one bus and one shared memory. The bus is used as communication channel connecting processor and configurable devise.

The main goal of partitioning is to select whether to put each task of the application model into SW (task assigned to the processor) or HW (task assigned to the FPGA) such that the whole execution time is minimized and the area occupation is maximized while reducing the overhead. Each application task is defined as a 5-tuple: Ti = (wi, hi, sti, ci, hti) where wi, hi, sti, ci and hti, denote width, height, software execution time, communication cost and hardware execution time of the task. The reconfiguration time

of the task can be considered as a part of its hardware execution time (hti) . In this work, we focus on dependent tasks witch may be represented as a directed task graph (DAG).

In this model, a solution to the partitioning and scheduling is expressed as a set of binary elements: X={x_1, x_2,...,x_n} where the x_i element represents the type of implementation assigned to the task Ti; if x_i = 1, it means implementation on HW, or implementation on SW otherwise.

Basin of the given the above stated, it is possible to define the main objective taking into consideration all the aforementioned imposed constraints as following:

Let $T = \{T1, T2, .. Tn\}$ with i=1 to n

$$x_i = \begin{cases} 1 & if\ \ Ti\ \ is\ assigned\ on\ FPGA \\ 0 & if\ Ti\ is\ assigned\ on\ processor \end{cases} \quad (1)$$

$$Texecution = \sum_{i=1}^{N}[(1 - x_i) * sti + x_i * hti] + \sum_{i=1}^{N-1} c_i \quad (2)$$

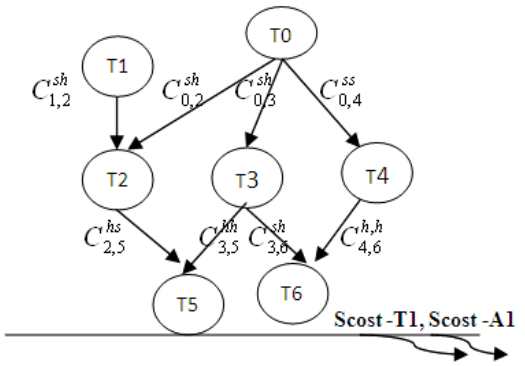
-c_i^{ss} (c_i^{hh}) denotes the communication time between node i and node (i+1) if both tasks blocks are assigned to software (hardware). $1 < i < n$

c_i^{sh} $\left(c_i^{hs}\right)$ denotes the communication time between node i and node (i+1) if task i is assigned to software (hardware) and task i+1 is assigned to hardware (software). 1 < i < n.

In our study the cost c_i^{ss} is ignored because two tasks i and i+1 are executed on the same processor. The total cost communication becomes:

$$c_i = x_i * x_{i+1} * c_i^{hh} + x_i * (1 - x_{i+1}) * c_i^{hs} + (1 - x_i) * x_{i+1} * c_i^{sh} \quad (3)$$

The following figure (fig.1) describes the tasks model used.



Bee-scheduling$_1$ [T_0, T_2, T_1, T_4, T_3, T_6, T_5]234 24
Bee-partitionning1 [hw_0,hw_2,sw_1,sw_4,hw_3,hw_6,sw_5]
Bee-scheduling$_2$ [T_0, T_2, T_4, T_1, T_3, T_5,T_6] 200 30
Bee-partitionning2 [sw_0, hw_2, hw_4,sw_1, sw_3,hw_5,hw_6]
Bee-scheduling3 [T_2, T_0, T_4, T_1, T_3, T_5 , T_6]322 9
Bee-partitionning3 [hw_2,sw_0,sw_4,sw_1,sw_3, sw_5,hw_6]
Bee-scheduling$_4$ [T_0 ,T_2 , T_1 , T_4 , T_3 , T_6 , T_5] 230 24
Bee-partitionning4 [hw_0,sw_2,sw_1,hw_4,hw_3,sw_6,sw_5]

Figure 1 HW/SW task model considered.Example of HW/SW partitioning-scheduling with cost function values.

In the third section, main aspects of our algorithm are described.

3 Proposed Hardware/Software Partitioning Scheduling Approach

The proposed approach combines the Honey Bees Mating Optimization (HBMO) algorithm with the local searches: WH1 based on simulated annealing (SA) and WH2 based on Tabu Search (TS). HBMO was found to outperform some better known algorithms. However, it has not been applied to dynamically reconfigurable embedded systems system scheduling/partitioning HW/SW problems. A honey bee colony consists of the queen(s), drones, worker(s) and broods. The Honey bees Mating Optimization algorithm mimics the natural mating behavior of the queen bee when she leaves the hive to mate with drones in the air. After each mating, the genetic pool of the queen is enhanced by adding sperm to her sperm theca. In [14], the authors give more information for the artificial comportment of the mating honey bees process. The workers are presented as heuristics whose functionality is to improve the broods produced during the mating process. The initial HBMO population is generated randomly and selects the best solution as the queen. To improve the broods produced during the queen's mating, we use two workers WH1 and WH2 based on local searches. The hybrid approach is based on a Honey Bees Mating Optimization (HBMO) that realizes a design space exploration by generating different mappings of the tasks on the processor and the FPGA.

4 Presentation of the Resolution Process

In this section, we present the approach adapted to resolve the issues addressed. To start with, a set of terms are explained in the table 1.

Table 1 Analogy between the natural bees, the artificial bees and the adopted partitioning/scheduling problem

Naturel honey bee entities	Artificial honey bee Entities	Scheduling/partitioning Entities problem (adopted approach)
Bees population	All feasible solutions	scheduling plans of hardware and software tasks with temporal and spatial constraints,
Queen	Best and optimal solution	Best HW/SW scheduling plan in the population
Drones	Incumbent solutions	Remaining HW/SW tasks scheduling plans.
Broods	New trial solutions	Plans gradually improved by crossover process in the mating HBMO stage and by two workers WH1 and WH2 in the HBMO amelioration stage.

The following algorithm presents the improved HBMO approach adopted.

Algorithm: HBMO_HW/SW partitioning scheduling

1. Initialization
2. **Generate the initial population of the honey bees randomly**
3. **Evaluate** the fitness of the scheduling (time execution cost, area cost);
4. **Select** the best plan of HW/SW partitioning of the population of HW/SW partitioning plans which represents the queen;
5. Sperm: Size of the spermatheca**;**
6. **E (t)** and **S (t)**: Energy and Speed in [0.5, 1];
7. **α**: factor of reduction of energy and speed in [0,1];
8. **M**: maximum number of mating flights;
9. **For** i:=0 **to** M (mating flights)
10. **do while** E(t) > 0 and Sperm is not full Select a drone
11. **If** the drone passes the probabilistic condition
12. **Add** sperm (plan) of the drone in the spermatheca
13. **Endif**
14. S(t+1) = **α** *S(t)
15. E(t+1) = **α** * E(t)
16. **Enddo**
17. **For** j = 1 **to** Size of Spermatheca
18. **Select** a sperm from the spermatheca;
19. **Generate** a brood by crossover the queen's genotype with the selected sperm;
20. **Improve the brood's fitness** by applying the **workers (WH1 &WH2)**
21. **If** the brood's fitness is better than the queen's fitness
22. **Replace** the queen with the brood
23. **else** Add the brood to the population of drones
24. **Endif**
25. **Replace** the drones by the broods
26. **Enddo(for)**
27. **Enddo**
28. **Return** The Queen (Best Solution Found).

Broods improvement: The improvement process is realized by two methods: WH1 and WH2.The genotypes resulting from the queen with the drone are improved by workers through WH1 and WH2. This stage of improved HBMO generates broods solutions: HW/SW partitioning. Thus the brood's improvement algorithm is applied to every operation of crosser often queen with the drone.

The next algorithms explain respectively the improvement strategies for the brood solutions by using two heuristic methods (Worker and WH2).

Algorithm WH1 (Worker Heuristic1:SA_ improving broods)

1. **Begin**
2. **Initial solution** s(T,A) , (brood which will be improved);
3. **Put** T1 ← cost-T(s(T))
4. T2←cost-A(s(A)) ; //do one mutation on the brood or One deplacement on the brood
5. s'(T,A)← **Mutation** (s(T,A));
6. **If** (cost-T(s'(T)) < cost-T(s(T))&
7. Cost-A(s'(A))>Cost-A(s(A)) **then** return s'(T,A)
8. **Else**
9. **Generate** r and r' randomly r in [0, 1]
10. **if** $\begin{pmatrix} r < e^{((cost-T(s(T))-cost-T(s'(T)))/T1} \text{and} \\ r' < e^{cost-A(s(T))-cost-A(s'(T))} \end{pmatrix}$
11. **then** return $s'(T, A)$
12. **Else** return s (T,A);
13. **Endif**;
14. **Endif**;
15. **End.**

Algorithm WH2 (Worker Heuristic1:TS_ improving broods)

1. **Begin**
2. Initial solution s (T,A) (brood which will be improved);
3. Insert **s** in the set L of solution ;
4. Scost-T_{min} ← s(T). / /Scost-T_{min} : best solution
5. Scost-A_{max} ← s(A)
6. **While** (Criterion of stop not checked) **do**
7. **Generate** the neighborhood of the current solution by **mutation**
8. **Select** s'(T,A) in this neighborhood although s'(T,A**)** is not present in the Set L.
9. **If** Scost-T(s'(T)) < Scost-T_{min}(s(T)) and Scost-A(s'(A)) > Scost-A_{max}(s(A)) **then begin**
10. Scost-T_{min} ← s'(T)
11. Scost-A_{max} ← s'(A**)** //minimize the cost function time and maximize cost function area
12. **Update** the set L;
13. **End;**
14. **Endif;**
15. **End of while;**
16. **End.**

5 Simulation and Experimental Results

In this section results of the Improved HBMO for HW/ SW partitioning and scheduling are compared to those given by the genetic algorithm (GA). In GA each task is represented by a binary gene which denotes the allocation of this task in the physical resources. Another advantage of this coding strategy is that genetic operators, such as crossover and mutation, can easily be applied without generating invalid chromosomes. It is also easy to extend to loosely coupled multiple-processor mulitple- FPGA architecture.

The JCreator version 4 has been used to implement the approach. These generated graphs are used as the benchmark to evaluate the result quality of our HW/SW partitioning and scheduling. In the graph, Each node i is represented by the quadruplet (sti,hti,ci, hai(wi,hi)) contains software execution time (sti), hardware execution time (hti), cost communication (ci) and hardware area occupied (hai)such the area hai is defined by the product between the width (wi) and the height (hi). It specifies the number of CLBs necessary for the execution of the task in FPGA. The values of software execution time (sti), hardware area occupied (hai) and hardware execution time (hti) were generated randomly. Since the execution time for one node implemented in software is generally greater than in a hardware implementation. The execution time of hardware (FPGA) equals one third to one fifth of the execution time of software [15].

5.1 Parameters values

The parameters of the adopted approach have been selected after thorough testing. A number of different values were tested and the ones selected are those that yielded the best results in both solution quality and computational time. The best values of these parameters are presented in the following tables:

Table 2. HBMO and GA Parameters values

Algorithms	Parameters	Values
HBMO	Population size	50
	Mating flight	100
	Spermatica	6
	Speed	0.80
	Energy	0.70
	Reduction factor α	0.20
Genetic Algorithm	Population size	50
	Probability of crossing	0.60
	Probability of mutation	0.20
	Number of generation	70

Table 3. Workers_ Parameters values

Algorithms	Parameters	Values
Worker_ simulated annealing (WH1)	Initial temperature	Brood-cost
	Final temperature	0
	Number of iterations	Until the amelioration or temperature=0
Worker_ Tabu search (WH2)	Size of list	5
	Number of iterations	5
	neighbood	Mutation and deplacement

5.2 Simulation results

In this section, we study the effectiveness of HBMO approach. The comparisons were made on the values found by the application of each algorithm: Improved HBMO and GA. In each execution evaluations of both objective functions were made: minimization of execution time and maximization of occupation area. This is illustrated by the graphs presented in the Fig.1 and Fig. 2:

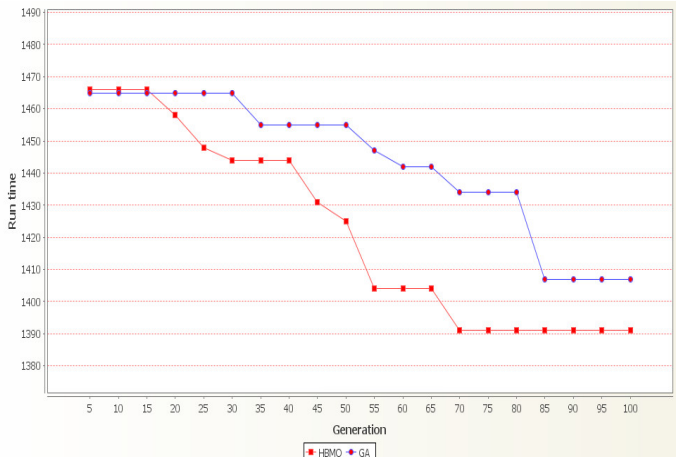


Figure 2. Comparative results between the graphics simulations HBMO (red curve) approach and genetic algorithms GA (blue curve) (case of minimization of execution time)

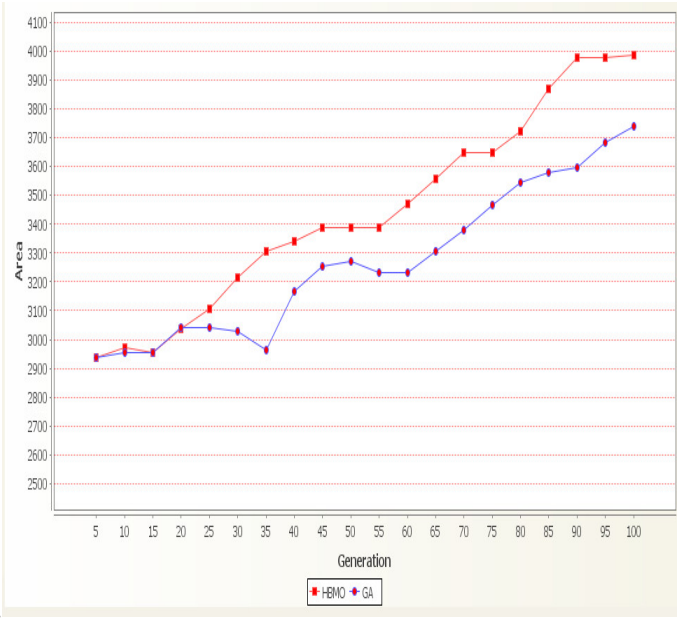


Figure 3 Comparative results between the graphics simulations HBMO approach (red curve) and genetic algorithms GA(blue curve) (case of maximizing the space used for FPGA size = 2000 CLBs.

We note that the effectiveness of HBMO is proven when ever the size of the FPGA increases. The majority of tasks will have hardware implementations. The same applies to the execution time, we note that HBMO converges to a minimum execution time than the GA.

6 Conclusion

In this paper, we described an improved HBMO heuristic for scheduling and HW/SW partitioning applications on reconfigurable System on Chip (RSoC). The application of the adopted method shows that the two problems: HW/SW partitioning and scheduling are hence interrelated and tightly coupled because the partitioning is included in the design space exploration while scheduling allows the evaluation of each solution. To test the validity of these algorithms, the results are obtained by HW/SW partitioning and

scheduling different groups of task graphs on different target systems (using different number of resources: size of FPGA). We compared the adopted approach, HBMO, to GA method, the simulation results obtained solutions showed the optimality for scheduling process and the effectiveness of the adopted approach in the all cases of examples.

REFERENCES

[1] Y. Jing, J. Krang,Jiayi DU and B. HU, Application of improved simulated Annealing optimization Algorithms in hardware/Software partitioning of the system on chip,Sringer.pp532-540,2014.

[2] H. Daz Pando, S. Asensi, R. Seplveda et al. An Application of Fuzzy Logic for Hardware/Software Partitioning Embedded Systems. Computation system Vol. 17 No.1, pp. 25-39.2013.

[3] M.B. Abdelhalim, S.E.-D.Habib,An integrated high-levelhardware/software partitioning methodology, Des Autom Embed Syst 15: pp.(19-50), 2011.

[4] P. Arat,., Z.A. Mann, and A. Orbn. Algorithmic aspects of hardware/software partitioning. ACM Transactions on Design Automation of Electronic Systems (TODAES), 10(1), 136156. 2005.

[5] S. Luo, X. Ma, Y. Lu .: An Advanced Non-Dominated Sorting Genetic Algorithm Based SOC Hardware/Software Partitioning, ACTA ELECTRONICA SINICA, (11):pp. 2595-2599. 2009.

[6] Y. Kang, H. Lu, J. He.: A PSO-based Genetic Algorithm for Scheduling of Tasks in a Heterogeneous Distributed System, Journal of software, 8(6): pp.1443-1450. 2013.

[7] F.Vahid Modifying min-cut for hardware and software functional partitioning. 5th International Workshop on Hardware/Software Co- Design (CODES/CASHE97), Braunschweig, Germany, 4348 .1997.

[8] F. Ferrandi, P. C. Lanzi, C. Pilato, D. Sciuto, A. Tumeo. Ant Colony Optimization for Mapping, Scheduling And Placing in Reconfigurable Systems. NASA/ESA Conference on Adaptive Hardware and Systems (AHS),2013.

[9] M. Koudil , K. Benatchba, ATarabet, E. B. Sahraoui. Using bees to solve partitioning and scheduling problem in codesign . Applied mathematics and computation 186,pp1710-1722. 2007.

[10] H. Daz Pando1, S. Cuenca Asensi, R. Seplveda Lima, J. Fajardo Caldern1 and A. Rosete Surez, An Application of Fuzzy Logic for Hardware/Software Partitioning in Embedded Systems. Computacin y Sistemas Vol. 17 No.1, pp. 25-39, 2013.

[11] G. Rehaiem , H. Gharsellaoui, S. Ben Ahmed. A Neural Networks Based Approach for the Real-Time Scheduling of Reconfigurable Embedded Systems with Minimization of Power Consumption. ICIS 2016, Okayama, Japan June 26-29, 2016.

[12] P.Peng, Z. Kuchcinski, K.Doboli, A.. System level hardware/ software partitioning based on simulated annealing and Tabu Search. Design Automation for Embedded Systems, 2(1), 532. 1997.