Hardware/Software Codesign Approach for heterogeneous MPSoC system

Mouna RIABI , Yassine MANAI, and Joseph HAGGEGE,

National Engineering School of Tunis, El Manar University, Tunis, Tunisia

Summary
The aim of this paper is to provide Hardware/Software (HW/SW) Codesign process for heterogeneous Multi-Processor Systems on Chip (MPSoC). For this process, an efficient HW/SW partitioning procedure is proposed. Two heuristics algorithms based on Genetic Algorithms (GA) and Artificial Bee Colony (ABC) are used to solve multi-objective optimization problem taking into account different constraints (time execution, available area and hardware multipliers). The proposed approaches are then validated on a controller system for an asynchronous motor. A comparison between GA and ABC algorithm is performed to retrieve those with the best time/area performances. The best solution is then implemented on a Zedboard platform.

Key words:
Codesign process, Hardware/Software Partitioning, Multi-Processor System on Chip (MPSoC), Heuristic Algorithms.

1. Introduction

Today's, embedded systems are more complex than ever. In fact, the need to improve performance, power and other design constraints of these systems represents a challenge to designers.
These challenges requires the use of advanced technologies. Thus, with the emergence of heterogeneous Multi-Processor System on Chip (MPSoC), new perspectives for implementing more functionalities into embedded applications are opened and the focus of design is better developed[1]. Indeed, the advantage to include both hardware and software resources help to reach tradeoff between various metrics in Hardware/Software (HW/SW) Codesign process. Where, hardware resources are used to accelerate the processing time by exploiting parallel implementations and software resources gives more flexibility. Additionally MPSoC architectures provide a high level of scalability compared with monolithic cores, particularly in terms of power and performance [2].
In addition to the technological aspects, researches in this issue are focused on exploring new automatic Codesign methodologies in order to develop this phase in the process, then, the goal of Codesign is to improve the performance and power of the product as well as satisfying other design constraints [3].
The most critical phase in HW/SW Codesign is the partitioning [4]. It requires to explore the optimal solution to choose which tasks are to be implemented in software and which ones in hardware, in order to solve the problem of managing the heterogeneity of embedded systems.
Previous researches use different approaches based on optimization algorithms to solve the HW/SW partitioning problem.
These approaches can be split into exact and heuristic algorithms. The exact methods, such as Integer Linear Programming (ILP)[5], Satisfiability Modulo Theories (SMT) [6] treat the systems with low complexity and limited number of tasks. However, as HW/SW Partitioning is an NP-hard problem [7,8], efficiently heuristic algorithms produce best solutions in term of computing time.
The heuristic methods are also divided into two classes: heuristics based on single solution such as Tabu Search which is considered in [9], also, Simulated Annealing in [10], and heuristics based on populations of solutions like the evolutionary algorithms, such as Genetic Algorithm (GA) in [11,12,13], NSGAII in [2], and algorithms based on swarm intelligence [14] where Ant Colony in [15,16], Artificial Bees in [17].
Several previous works, treated the HW/SW partitioning problem for signal and image processing [18,19] or multimedia applications [20,21], and deal with mono-objective problems. Otherwise, we notice the few number of such applications for MPSoC control systems [7] and consider muti-objective constraints.
In this paper, we propose a novel HW/SW Codesign approach for heterogeneous MPSoC applications, where the target system is presented in the modality of task graphs.
The main contributions of this work is to propose two heuristic methods based on Genetic Algorithm and Artificial Bee Colony (ABC) to resolve HW/SW partitioning problem. This approaches takes into account heterogeneous constraints and give optimal architectural solutions. In addition, a new HW/SW Codesign process is investigated for design embedded systems dedicated to control system.
It was tested on a speed controller system. The MPSoC platform used in this work consists of a Zync 7000 and Cortex A9 soft-core processor.
The other parts of this paper are arranged as follows. Section 2 present the proposed HW/SW Codesign methodology. Section 3 describes the specification of the control system. Section 4 identifies the different parameters

Manuscript received January 5, 2018
Manuscript revised January 20, 2018

corresponding to the cost estimation. Section 5 explains the proposed algorithms for HW/SW partitioning, Section 6 shows experimental results. Finally, we finish by a conclusion.

2. Proposed Hardware/Software Codesign Approach

HW/SW Codesign consists of a succession of steps starting with specification of embedded systems to the synthesis and experimental validation [7]. It aims to decide what is the best and optimal assignment of each component of an embedded system to hardware resources or software ones [22]. In order to automate this process, a set of steps and rules should be followed.

As shown in Fig. 1, this method is decomposed into four main steps, as follows:

- The specification step which consists to decompose the system into subsystems and verifies its functionality.
- The second step is the cost estimation defining the characteristics of each subsystem and their corresponding granularity level. In this step, the target system will be described by a task graph.
- The difficulty of the third step is mainly linked to the search for an optimized HW/SW partitioning of the functional modules respecting different architectural constraints and control performances. This goal is reached with the help of heuristic methods based on genetic algorithm and artificial bee colony.
- Finally, in the fourth step an experimental validation allows the verification of obtained optimal solution.

To perform these steps, different tools are used like Xilinx Vivado, Matlab/Simulink and HDL Workflow Advisor.

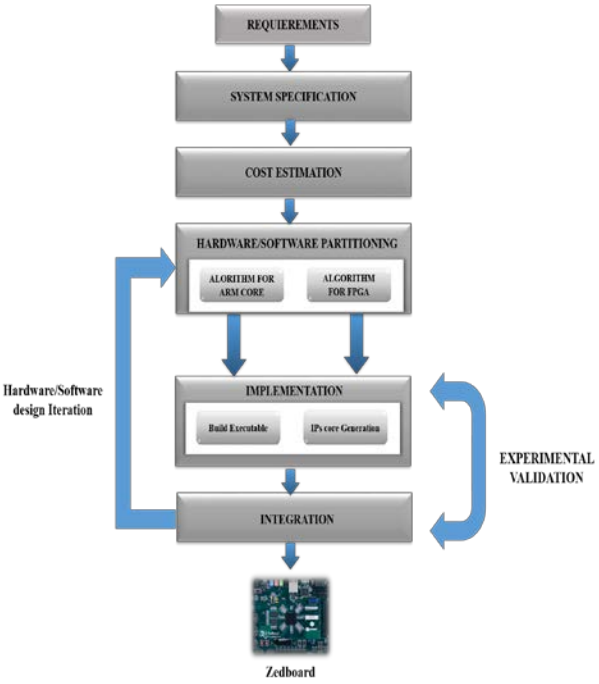

Fig. 1 Hardware/Software Co-design method

3. System Specification

3.1 Hardware specification

The digital control unit is based on Xilinx ZynqTM-7000 All Programmable SoC (AP SoC). Combining a dual Cortex-A9 Processing System (PS). It consist of 53,200 LookUp Tables (LUTs), 106,400 Flip Flops (FF), 4.9MB bloks RAM, 220 DSP.

The structure of the developed control system is first overviewed and depicted in Fig. 2. It is composed of the following principal subsystems :

- Speed controller ;
- Current controller ;
- Estimator ;
- Park transform ;
- Park inverse transform.

These subsystems are divided into functional modules as described in the next section.

The complete control system is designed and its functionality is verified at first in continuous time domain and finally in the discrete time and fixed point domain. Matlab/Simulink Tools are used to simulate the behavior of the asynchronous motor with the corresponding numerical values:

$$\begin{cases} R_s = 1; L_s = 0.25H \\ R_r = 0.7; L_r = 0.25H \\ J = 0.7kg.m^2 \\ M = 0.24H \\ T_r = 0.153N.m \end{cases} \quad (1)$$

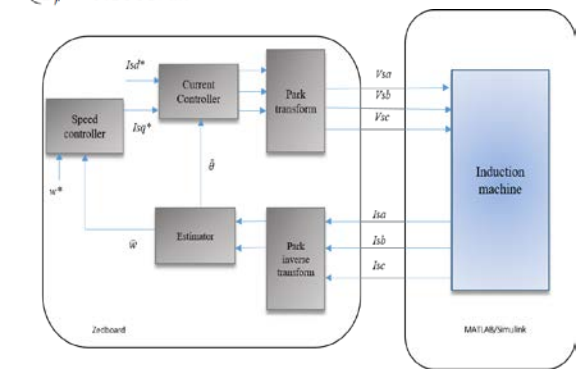

Fig. 2 Overview of control system

3.2 Granularity Level

Partitioning in Codesign process requires completion of the specification steps with the decomposition of the system into subsystems and the definition of the characteristics of each one and the corresponding granularity level [7]. To do this, we choose a coarse granularity level, as shown in Fig. 3, which consists of independent and reusable functional modules. Thus, using this level reduces the number of partitioned blocks and preserve the physical meaning of each subsystem [23]. Even with the coarse grain, high throughput and low power consumption can be satisfied [24].

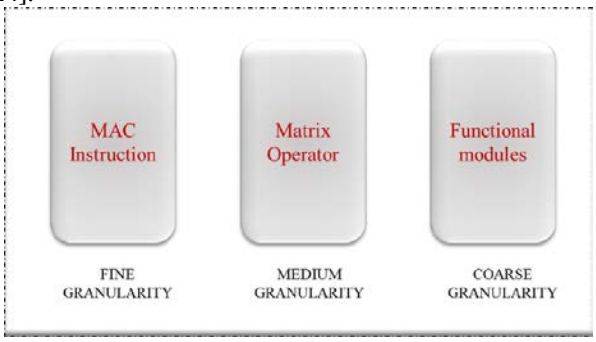

Fig. 3 Granularity level

4. Cost estimation

4.1 Performances estimation

In this step of HW/SW codesign process, an accurate performances estimation of the functional modules is crucial. These performances estimation can correspond to different ressources such as time/area/ memory/power consumption [25].
To do this, different approaches are used, depending on target architecture and type of ressources. For software ones, the profiling method is used to make the performances analysis. For hardware ressources, the use of techniques based on behavioural description or gate-level description can provide a good accuracy[23].
In our case, the choice of high level granularity allow the use of profiling method to predict software performance. In the other hand, the generated report of Xilinx allows the determination of estimated parameters in the case of hardware performance.
These HW/SW performance, which characterize the functional modules, are described by the following parameters:

- $\boldsymbol{A_i}$: consumed resources of module *i*, in the case of hardware implementation (LUT & FF).
- $\boldsymbol{t_{hi}}$: execution time taken by the module *i* when executed in hardware.
- $\boldsymbol{t_{si}}$: execution time taken by the module *i* when executed in software.
- $\boldsymbol{HM_i}$: consumed hardware multiplier (DSP units) of module *i*, in the case of hardware implementation.
- $\boldsymbol{I_i}$ / $\boldsymbol{O_i}$: number of inputs and outputs of module *i*.

The set of metrics of each functional modules M_i is presented in Table 1, where the execution time is given in microseconds, the area correspond to the number of LUTs and number of Flip Flops, and Hardware multiplier is the number of DSP blocks.

4.2 Task graph

A directed acyclic graph G = (V, E), called the task graph of the system, is necessary 26]. So, in order to prepare the partitioning process, the controller system is described through a Directed Graph presented in Fig. 4. The modules Mi are represented by nodes Vi, and the edges Ei represent communication between them. Additionally, each of the modules has a number which indicate its scheduling order.

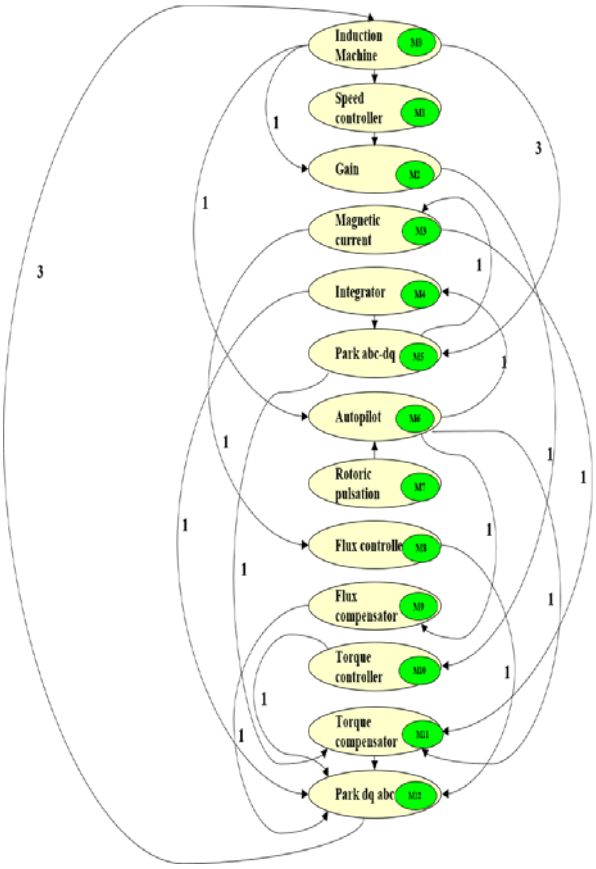

Fig. 4 Direct Acyclic Graph

5. Hardware/Software Partitioning

The main goal of the HW/SW partitioning process is to find the optimal assignements of the functional modules to hardware or software resources while respecting the considered architectural constraints. In this case, HW/SW partitioning consists in a multi-objective optimization problem.

5.1 Cost function

In order to formalize this problem, a set of metrics defining the cost function parameters are described as follows :

- $M = \{M_1, M_2, \ldots, M_n\}$: *n* functional modules group (*n* is here equal to 12).
- (*w1, w2, …, wn*) : 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).
- *Area:* total consumed hardware resources.
- $A_{\mu p}$: hardware resources consumed by the processor.
- HM_{blocks}: total consumed hardware multipliers (DSP units).
- $HM_{\mu p}$: total consumed hardware multipliers (DSP units) by the processor.
- T_{exe}: total execution time .
- T_{com}: communication time between module i and the other modules.

where *Area*, HM_{blocks},T_{exe} are derived from expression (8) below.
In order to optimize the cost function, we must meet the following requirements must be achieved :

$$\begin{cases} Area < S \\ HM_{blocks} < HM \\ T_{exe} < T \end{cases} \quad (2)$$

where *S* and *HM* are respectively the area and number of hardware multiplier available in the ZedBoard.
T corresponds to the maximum time delay that satisfies the control performance requirements.

Table 1: Metrics of functional modules

Functional Modules	A_i	t_{hi} *(µs)*	t_{si} *(µs)*	HM_i
M1 : Speed controller	630 LUT 662 FF	466	682	0
M2 : Gain	654LUT 727FF	598	490	1
M3 : Magnetic current	654 LUT 727 FF	453	332	0
M4 : Integrator	669 LUT 727 FF	626	700	1
M5 : Park abc-dq	5499 LUT 3656 FF	657	995	8
M6 : Autopilot	669 LUT 725 FF	642	406	0
M7 : Rotoric pulsation	1188 LUT 774 FF	643	516	3
M8 : Flux controller	630 LUT 662 FF	550	627	0
M9 : Flux compensator	653 LUT 726 FF	719	479	2
M10 : Torque controller	630 LUT 662 FF	375	664	0
M11 : Torque compensator	673 LUT 740 FF	583	461	3
M12 : Park dq-abc	5486 LUT 3640 FF	999	1143	9

The communication model used in (7) is detailed as follow:

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

$$C_i^{hs} = \text{Re}ad\_cycle \times O_i \quad (3)$$

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

$$C_i^{sh} = Write\_cycle \times O_i \quad (4)$$

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

$$C_i^{hh} = Clock\_cycle \quad (5)$$

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

$$C_i^{SS} = Clock\_cycle \times O_i \quad (6)$$

Where

$$Read\_cycle = Write\_cycle = 3 \times Clock\_cycle$$

$$T_{com} = \begin{bmatrix} (1-w_i)\times(1-w_{i+1})\times C_i^{ss} \\ +w_i \times w_{i+1} \times C_i^{hh} \\ +(1-w_i)\times w_{i+1} \times C_i^{sh} \\ +w_i \times (1-w_{i+1}) \times C_i^{hs} \end{bmatrix} \quad (7)$$

$$\begin{cases} Area(x_1,w) = \sum_{i=1}^{n} w_i \times A_i + \sum_{i=1}^{n} (1-w_i) \times A_{\overline{BB}^t} + (x_1)_i \\ HM_{blocks}(x_2,w) = \sum_{i=1}^{n} w_i \times HM_i + \sum_{i=1}^{n} (1-w_i) \times HM_{\overline{BB}^t} + (x_2)_i \\ T_{exe}(x_3,w) = \sum_{i=1}^{n-1} \begin{bmatrix} w_i \times w_{i+1} \times \max(t_{hi}, t_{hi+1}) \\ +w_i \times (1-w_{i+1}) \times \max(t_{hi}, t_{si+1}) \\ +(1-w_i) \times w_{i+1} \times \max(t_{si}, t_{hi+1}) \\ +(1-w_{i+1}) \times \max(t_{si}, t_{si+1}) \\ +w_i \times t_{hi} \\ +(1-w_i) \times t_{si} \end{bmatrix} \\ +\left[w_n \times t_{hn} + (1-w_n) \times t_{sn} \right] + T_{com} + (x_3)_i \end{cases} \quad (8)$$

5.2 Proposed Approaches

The multi-objective optimization problem described by the above set of equations deals with two heuristics approaches.

5.2.1 First approach based on Genetic Algorithm

Genetic Algorithms were successfully used for solving several optimization problems in the field of embedded systems [2].

In our case, this algorithm is adopted and improved in order to find the optimal solution for HW/SW partitioning with the optimization of three principal parameters: Area, Execution Time and Hardware Multiplier.

As shown in Fig. 5, the developed approach follows the main steps of binary genetic algorithms [27] with several modifications to accommodate the multi-objective problem and meets different constraints.

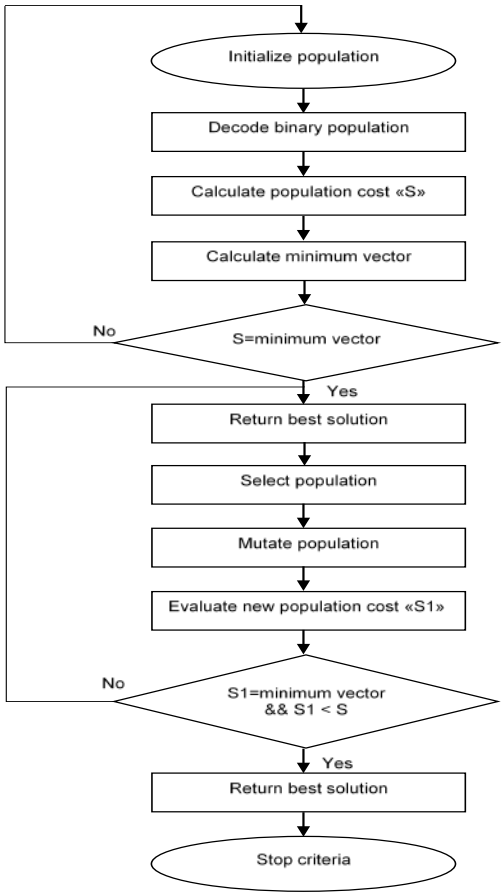

Fig. 5 Main program of proposed genetic algorithm

In order to get diversity, an initial population of chromosomes is generated randomly. After decoding the vector of chromosomes, the population is evaluated using the multi-objective function. The best solution is a vector that contains the minimum for each 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.

The obtained results represent the optimized three parameters (area, execution time, hardware multiplier) and the binary vector which represents the modules to be implemented : the target is software resources, if it is equal to zero, and the target is hardware resources, if it is equal to one. In our case the vector size corresponds to the modules number (n = 12).

5.2.2 Approach based on Artificial Bee Colony

In this paper, we improve the basic ABC algorithm definition with several modifications in order to get optimal 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 that corresponds to the quality (fitness) of the solution [27].
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 [28]. The latter steps, which are applied on the three parameters to be optimized in the HW/SW partitioning process, are described by the flowchart in the Fig. 6.

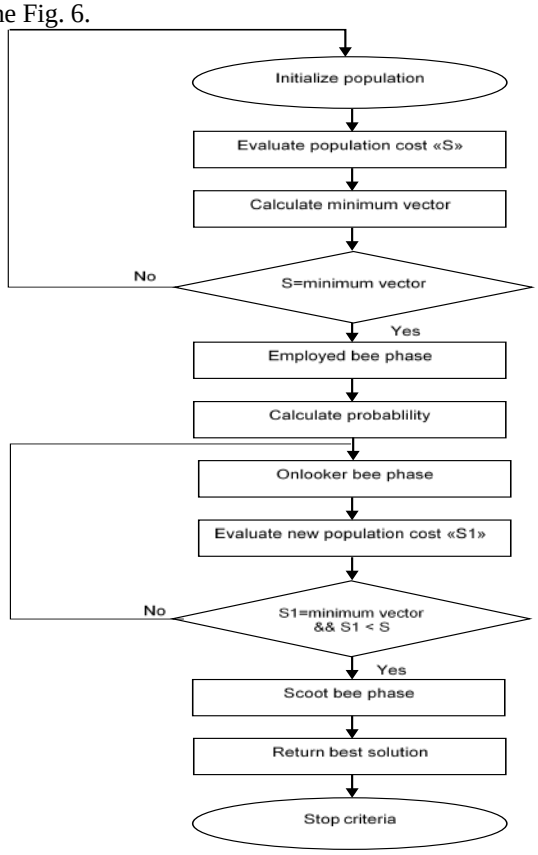

Fig. 6 Main program of proposed ABC algorithm

5.2.3 Simulation Results of partitioning

The HW/SW partitioning tests were simulated in Matlab taking into account the requirement of the MPSoC zedboard including zynq 7000 FPGA and two ARM Cortex soft core. The Genetic Algorithm configuration was done considering the parameters in Table 2:

Table 2: Configuration parameters of GA

Number of generation	*Number of population*	*Selection rate*	*Mutation rate*
1000	200	0.5	0.15

In the other hand, ABC algorithm depends on the four control parameters that are configured in Table 3:

Table 3: Configuration parameters of ABC algorithm

Colony size	*Number of food source*	*Limit*	*Maximum cycle*
24	12	100	1000

Table 4 and Table 5 represents an example of optimization solutions for the two proposed heuristic algorithms.
It can be seen from these results that the implementation of more modules in hardware resources provides a reduction of the execution time but it increases the consumed hardware resources.

Table 4: Examples of optimization solution of GA

	Binary Vector	*Texe*	*Area*	*HM*
S1	000000001000	7744	653	2
S2	000000010100	7126	1260	0
S3	110000000100	7093	1914	1
S4	100100010100	6843	2559	1
S5	100110000100	6579	7428	9

Table 5: Examples of optimization solution of ABC Algorithm

	Binary Vector	*Texe*	*Area*	*HM*
S6	100000000000	7356	642	2
S7	000100000100	7139	1332	2
S8	100000010100	6917	1890	12
S9	100100010100	6851	2614	58
S10	101110000100	6738	8136	50

A comparison between the two algorithms based on the same partitioning and scheduling solution of modules is shown in Table 6. Where we notice that genetic algorithm give us a better optimization in term of area/execution time.

Table 6: Comparison between optimization solution of GA and ABC algorithm

	Genetic Algorithm	*ABC Algorithm*
Binary Vector	100100010100	100100010100
Area	2559	2614
Execution Time	6843	6851

However, S3 gives better results compared to S4 because it is a more homogeneous solution. The execution time value

is still acceptable knowing that the number of modules implemented in software is higher compared with S4. Figure. 7 shows the scheduling diagram of S3.

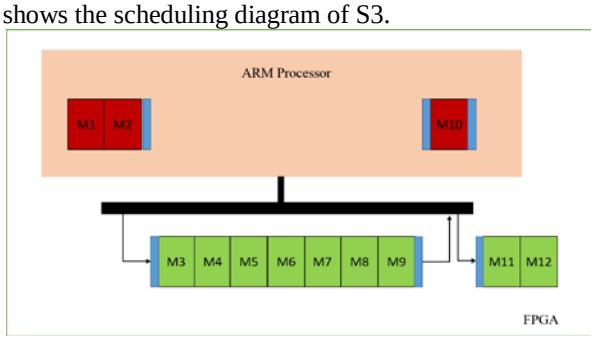

Fig. 7 Scheduling diagram of S3

6. Experimental Validation

Experimental validation of S3 has been carried out using the platform of Zedboard and Matlab Simulink to implement the partitioned system. After implementation of solution using the HDLWorkflow, a verification test with processor in the loop method shows the good matching of the desired and the measured speed signal as presented in Fig. 8. In addition, Figure 8 shows that when applying load torque at instant 2.5, this disturbance is rejected by the system.

In this work, the most important validation concerns the quality of solution obtained by both Genetic algorithm and ABC algorithm, which allows the designer to choose the best appropriate allocation and scheduling for his application.

The contribution of this work, is to obtain the best tradeoffs area/execution time respecting the constraint of system architecture.

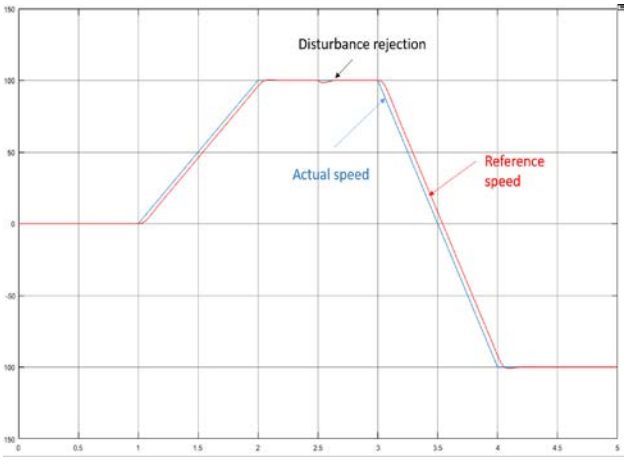

Fig.8 Validation of the asynchronous motor : Estimation of speed

7. Conclusion

A hardware/software Codesign approach was proposed for controller system based on MPSOC architecture. The contribution in this approach is the automation of the HW/SW partitioning process. To reach this purpose, two heuristics algorithms, genetic algorithm and artificial bee colony algorithm, have been proposed to solve the corresponding multi-objective optimization problem. In early stage, the system specification is occured taking into account the available resources in zedboard platform. The choice of the optimal result is based on respecting the area and time execution constraint. The optimal solution given by the proposed algorithms was tested and implemented to MPSoC architecture. As an application, an asynchronous machine controller system for asynchronous motor is considered. Results were found satisfactory in terms of area/time execution for genetic algorithm in comparison with ABC algorithm. Taking into account other architectural constraints is one of the direct perspectives in this work. Another one is to propose other approaches or hybrid algorithm for multi-objective partitioning problem.

References

[1] G. Sapienza, G. Brestovac, R. Grgurina, T. Seceleanu, “On applying multiple criteria decision analysis in embedded systems design,”Des Autom Embed Syst, DOI 10.1007/s10617-016-9171-7, 2016.

[2] 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.

[3] E. Azari, H. Koc, “Improving Performance through Path-Based Hardware/Software Partitioning,” ISBN: 978-1-4673-6832-2, 2015 IEEE.

[4] I. Mhadhbi, S. Ben Othman, S. Ben Saoud, “An Efficient Technique for Hardware/Software Partitioning Process in Codesign,” Hindawi Publishing Corporation Scientific Programming, Vol. 2016, Article ID 6382765, 2016.

[5] S. Banerjee, E. Bozorgzadeh, N. D. Dutt, “Integrating Physical Constraints in HW-SW Partitioning for Architectures With Partial Dynamic Reconfiguration,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, Issue 11, pp. 1189-1202, 2006.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[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] I. Alouani, B. L. Mediouni, S. Niar, "A Multi-Objective Approach for Software/Hardware Partitioning in a Multi-Target Tracking System," IEEE DOI: 10.1109/RSP.2015.7416556, 2016.

[19] T. Kryjak, M. Komorkiewiez, M. Gorgon, "Real-time hardware–software embedded vision system for ITS smart camera implemented in Zynq SoC," J Real-Time Image Proc, DOI 10.1007/s11554-016-0588-9, 2016.

[20] T. D. Ngo, K. J. M. Martin, J. P. Diguet, "Move Based Algorithm for Runtime Mapping of Dataflow Actors on Heterogeneous MPSoCs," J Sign Process System, DOI 10.1007/s11265-015-1088-z, 2015.

[21] N. Govil, R. Shrestha, S. R. Chowdhury, "PGMA: An algorithmic approach for multi-objective hardware software partitioning," Microprocessors and Microsystems, 54 (2017) 83–96, 2017.

[22] M. Riabi, Y. Manai, J. Haggège, "New Multi-Objective Approaches for Hardware/Software Partitioning Problem in Codesign Process," Proceedings of Engineering and Technology-PET, Vol.22 pp.96-102, 2017.

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

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

[25] A. Ouyang, X. Peng, J. Liu, A. Sallam, "Hardware/Software Partitioning for Heterogenous MPSoC Considering Communication Overhead," Int J Parallel Prog, DOI: 10.1007/s10766-016-0466-x, 2016.

[26] A. B. Trindade, L. C. Cordeiro, "Applying SMT-based verification to hardware/software partitioning in embedded systems," Des Autom Embed Syst, DOI: 10.1007/s10617-015-9163-z, 2015.

[27] 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.

[28] 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.

Mouna Riabi was born on Mai 1988 in Tunisia. She received her degree in industrial systems computing engineer in 2013, from the "Higher Institute of Computer (ISI), University El Manar, Tunisia". She is currently a PhD Student in the "Automatic Control research laboratory" (LA.R.A) in the "National School of Engineers of Tunis" (ENIT), Tunisia.Her current research interests are in the areas of system-on-chip design, embedded systems, partitioning and scheduling methodologies.

Yassine Manai was born on December 1979 in Tunisia. He received the Master degree in Automatic and Signal Processing and the Doctorate degree in Electrical Engineering from the "National School of Engineers of Tunis" (ENIT) Tunisia in 2005 and 2009 respectively. His Doctorate thesis is prepared within the framework of unit research "Automatic Control research laboratory" (LA.R.A) about Embedded System Architectures Design and Synthesis by the use of Heterogeneous Platforms. His research interests are embedded systems, stability and stabilization of Takagi-Sugeno fuzzy systems, and its applications.

Joseph Haggège was born in 1975, in Tunis, Tunisia. He graduated from National School of Engineers of Tunis in 1998. He received the PhD degree in Electrical Engineering 2003 and the Habilitation in 2010. He is currently Senior Lecturer at the National School of Engineers of Tunis (ENIT), Tunisia. His research interests are in the area of meta-heuristics optimization, embedded systems and robust digital control.