Vol.5pp. 17-23 International Journal of Scientific Research& Engineering Technology (IJSET)

Hardware/Software partitioning approach for embedded system design based on Genetic Algorithm

Mouna Riabi, Yassine Manai, and Joseph Haggège

Laboratoire de Recherche en Automatique(L.A.R.A), Ecole Nationale d'Ingénieur de Tunis
Université de Tunis El Manar, Tunisie

mouna.riabi@gmail.com
yacine.manai@gmail.com
joseph.haggege@enit.rnu.tn

***Abstract*—In this paper, a new method is proposed to resolve Hardware/Software (Hw/Sw) partitioning problem for AC drive applications using Field Programmable Gate Array (FPGA) based controllers that are dedicated to power electronics and drives applications. This work aims to optimize a multi-objective problem. The proposed Hw/Sw partitioning approach is based on binary Genetic Algorithm (GA). This procedure takes into account the heterogeneous constraints such as the control requirements and the hardware resources. In order to evaluate our proposed algorithm, we made a comparison between the obtained results and those given by Non-Dominated Sorting Genetic Algorithm (NSGAII).**

Keywords—Hardware/Software partitioning ; system-on-chip ; genetic algorithm ; field programmable gate array ; soft core processor.

I. INTRODUCTION

Electronic systems, that are manifesting in most of the industrial fields, telecommunications transport, automotive, banking, are composed of both hardware and software.

The hardware/software Co-design is born from the need to develop applications that follow the requirements of better performance, rising cost, testability and verification, as well as reducing time-to-market mainly for complex real-time systems.

The embedded system designers gave a great importance to hardware/software partitioning in the co-design process. The latter aims to optimize the allocation of resources and to seek a compromise performance / Area / energy consumption, etc. Indeed, the architectural decisions and allocation choices during this stage have a significant impact on the quality and the cost of the final product.

Hardware/software partitioning is known as NP-difficult problem [10]. Numerous heuristics such as, tabu search, simulated annealing, ant colony algorithm [5, 8], and genetic algorithm [2, 9], have been developed to solve it.

Different implementations are proposed such as in [3, 4, 7] which combine two heuristics and we notice that the approaches which are based on genetic algorithms are extensively used in this research field.

In this work the basic mechanisms of GA are analyzed and modified to achieve a better exploitation of their potential. This method is used to solve multi-objective optimization problem taking into account the architectural constraints (e.g., available area, execution time, memory and hardware multipliers) of a system on ship field programmable gate array based sensorless AC drive applications.

The remainder of this paper is organized as follows: Section II introduces the system studied and briefly explains the concept of partitioning in co-design. Section III presents the proposed method for Hw/Sw partitioning applied to area optimization then to a multi-objective cost function. Simulation results are discussed in section IV. Finally, the conclusion and perspectives are given in section V.

II. PRELIMINARIES

A. Target architecture

In this paper, the reference architecture and all its assumptions are same as that in [1] for fair performance comparison in algorithmic aspects.

The sensorless control system structure is overviewed and depicted in Fig. 1. It contains a power stage, an analog-to-digital converter board, a digital control unit and a host pc.

The power stage is composed of a salient pole synchronous motor loaded by a powder brake and electrically fed by a voltage source inverter [1]. Regarding the digital control unit, it is based on an FPGA platform that consists on the Xilinx Virtex-5 embedding a Microblaze soft-core processor.

B. Functional modules

Partitioning requires completion of the specification step which decomposes the system into subsystems and defines their characteristics and their corresponding granularity level.

In this paper we keep the same functional modules that can be manageable due to their limited number. At the same time, their physical meaning is preserved. These modules are characterized by a set of metrics subsequently defining the cost function parameters. These metrics are cited in [1] where

- A_i: consumed resources of module i, in the case of hardware implementation (6-bLUT & FF).
- H_i: memory blocks used by module i, in the case of software implementation.

ISSN:2356-5608

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

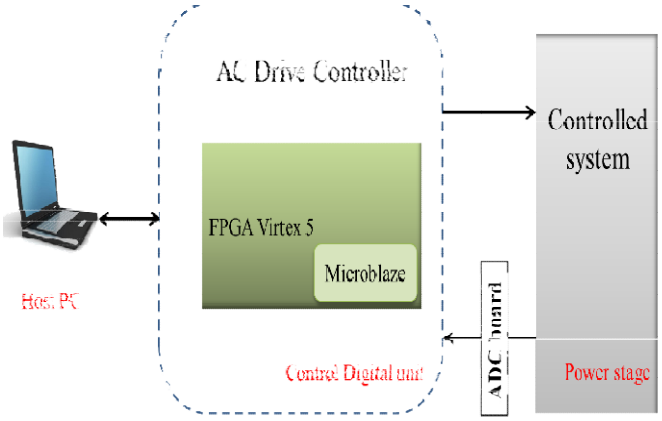

Fig. 1. The target architecture of AC drive Application

C. Partitioning in Codesign

In embedded system Co-design, the term partitioning can essentially be described as the assignment of the system's tasks to hardware resources or software ones. In the final solution we have to meet several requirements or constraints for execution time, area, power consumption, etc.

Partitioning can be classified as fine-grained (if it partitions the system specification at the basic-block level) or as coarse-grained (if system specification is partitioned at the process or task level) [11].The system functionality can be modeled by a set of task graphs, each task graph is composed by nodes that cover the functional object of the system, and edges that ensure data transfers between different processes [12]. Depending on the granularity of the graph representation, the nodes may stand for a single operational unit Multiplier–Accumulator (MAC) that will be represented in this paper.

III. Proposed Approach

A. Basic genetic algorithm

In this section, the use of genetic algorithm in solving partitioning problem is briefly introduces. This algorithm is inspired from natural reproduction of chromosomes. In this fact, several operations reveals to improve the GA's performances: selection, mutation and crossover.

These operations occur during the reproductive phase and ensure a persistent diversity in the number of individuals. Fig. 2 presents the genetic algorithm principle.

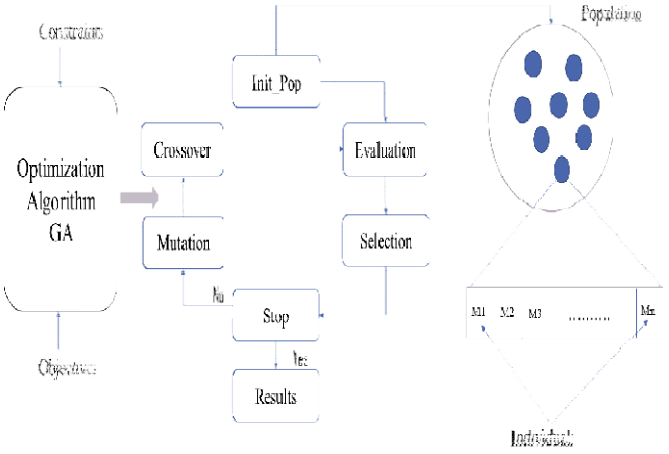

Fig. 2. Genetic Algorithm principle

Depending on the problem formulation there are a variety of mechanisms implementations. In this paper we join the basic genetic algorithm definition with several modifications in order to get better results of optimization solutions.

B. Partitioning with area minimization

As a first step, our approach based on genetic algorithm aims to the minimization of a main resource in the hardware target which is the area consumption. The addressed problem is to optimize the allocation of resources taking into account one goal.

1) Area cost function

The partitioning problem is solved by applying our approach to the cost function described below.

$$Area(x_1, w) = \sum_{i=1}^{n} w_i.A_i + \sum_{i=1}^{n} (1 - w_i).A_{\mu p} + (x_1)_i$$

Where the consumed resource (6-bLUT) of each module is defined by A_i, in the case of hardware implementation and A_{up} in the case of software implementation.

The constrained hardware-software partitioning problem requires finding an assignment for each partitioning variable w_i such that the objective function is minimized.

The limits of area resources depends on the target used in this field, in our case we consider an FPGA platform that consists on the Xilinx Virtex V with à maximum of 7500 6-bLUTs [1].

2) Task graph with MAC_Operation

An embedded system is a set of smartcells each one is modeled with a state space representation. For each smartcell, state vector is programmed with a recursive functions based on MAC operation [5]. In our case, the area cost function is represented as follow:

ISSN:2356-5608

$$Area(x_1, w) = \sum_{i=1}^{n} w_i.A_i + \sum_{i=1}^{n} (1-w_i).A_{\mu p} + (x_1)_i$$

$$A_{temp1} = w_1.A_1 + (1-w_1).A_{\mu p_1} + (x_1)_1$$

$$A_{temp2} = A_{temp1} + w_2.A_2 + (1-w_2).A_{\mu p_2} + (x_1)_2$$

M

$$Area = A_{temp(n-1)} + w_n.A_n + (1-w_n).A_{\mu p_n} + (x_1)_n$$

A Direct Acyclic Graph (DAG) using the MAC approach given in [10] can model the system given by the latter recursive functions as shown in Fig. 3.

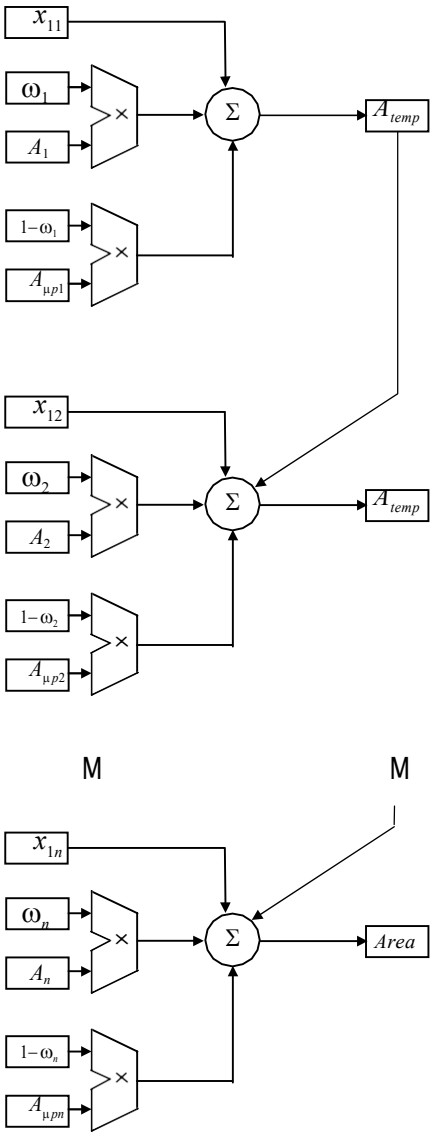

Fig. 3. Task graph with MAC_operation

Where the MAC–element operation is given in Fig. 4.

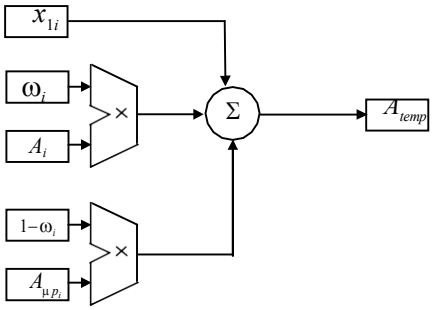

Fig. 4.MAC_element operation

C. *Multi-objective optimization*

1) *Multi-objective cost function*

In this part we define the different objectives to be optimized with the proposed algorithm for the overall system.

According to the referred system in [1], the multi-objective optimization problem is given by (6), where:

- $M = \{M_1, M_2, \ldots, M_n\}$: 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).
- $A_{\mu p}$: 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 .
- *Tcom*: communication time between module *i* and the other modules.

The communication model used in (5) is detailed as follow [6]:

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

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

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

ISSN:2356-5608

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

- 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}.

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

Where $Read\_cycle = Write\_cycle = 3*Clock\_cycle$ [6]

$$T_{com} = \sum_{i=1}^{n-1} \begin{cases} (1-w_i).(1-w_{i+1}).C_i^{ss} \\ +w_i.w_{i+1}.C_i^{hh} \\ +(1-w_i).w_{i+1}.C_i^{sh} \\ +w_i.(1-w_{i+1}).C_i^{hs} \end{cases} \quad (5)$$

$$\begin{cases} Area(x_1, w) = \sum_{i=1}^{n} w_i.A_i + \sum_{i=1}^{n} (1-w_i).A_{\mu p} + (x_1)_i \\ HM_{blocks}(x_2, w) = \sum_{i=1}^{n} w_i.HM_i + \sum_{i=1}^{n} (1-w_i).HM_{\mu p} + (x_2)_i \\ Mem(x_3, w) = \sum_{i=1}^{n} (1-w_i).H_i + (x_3)_i \\ T_{ex}(x_4, w) = \sum_{i=1}^{n-1} \begin{cases} y_i.w_i.w_{i+1}.\max(t_{hi}, t_{hi+1}) \\ +y_i.w_i.(1-w_{i+1}).\max(t_{hi}, t_{si+1}) \\ +y_i.(1-w_i).w_{i+1}.\max(t_{si}, t_{hi+1}) \\ +y_i.(1-y_i).(1-w_{i+1}).\max(t_{si}, t_{si+1}) \\ +(1-y_i).(1-y_{i-1}).w_i.t_{hi} \\ +(1-y_i).(1-y_{i-1}).(1-w_i).t_{si} \end{cases} \\ +(1-y_{n-1}).(w_n.t_{hn} + (1-w_n).t_{sn}) + T_{com} + (x_4)_i \end{cases} \quad (6)$$

2) Constraints

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

$$\begin{cases} Area < S \\ Mem < H \\ HM_{blocks} < HM \\ T_{ex} < T_{alg} \end{cases} \quad (7)$$

Where S, H, and HM are respectively the area, memory size, and number of hardware multiplier available in the FPGA. T_{alg} corresponds to the maximum time delay that satisfies the control performance requirements (bandwidth and stability margin) [1].

D. Genetic algorithm for partitioning

In the following, the goal is to find the optimal Hw/Sw partitioning with lower execution time, area, memory, and hardware multiplier. To do this, the GA was chosen for its efficiency and its recognized performances.

This optimization algorithm aims to find out the optimal solution of partitioning. In each generation, GA classifies the candidate solution taking into account all of the objectives and conserve the best one for the next generation, then the use of operators (selection and mutation) will guarantee the population diversity. This procedure is iterated until finding the best solution. The genetic algorithm is based on chromosomes that will be decoded and converted from binary to continuous values. These values represent the four parameters to be optimized and will be used as the first entry (x_i) of the cost function. The second entry is a vector of n values set to 0 and 1 which represent the partitioning solution.

Each value w_i of the vector is a module implemented in software resources (Microblaze soft core), if it is equal to 0, or in hardware resources (FPGA), if it is equal to 1. Here the vector size corresponds to the modules number (n = 15).

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

The main program is described by a flowchart as shown in Fig. 5 and detailed as follows:

Step 1: Initialization. The initial population is generated randomly in order to get diversity. Individuals (chromosomes) are composed of genes set to 0 or 1.

Step 2: Decoding. The decoding of the initial population composed of binary values (0 and 1) allows the evaluation of the four parameters.

Step 3: Evaluation. The population cost is calculated using the multi-objective function. Then, the costs are sorted.

Step 4: Selection. An N / 2 pairs of individuals in generation k are performed according to the criterion to be optimized. Where N=population size, and k=index of the current generation.

Step 5: Mutation. Mutation operator is applied on selected parent individuals. The mutation rate is set as 0.15.

Step 6: Re-evaluation. The evaluation is recalculated for new population. The new costs are sorted, the best individual is made and new ones are added to form new generation.

Step 7: The previous and next costs are compared. Then, the binary vector corresponding to the best compromise of costs is returned.

ISSN:2356-5608

Step 8: Stop Criteria. If the maximum generations are reached or the load limits are exceeded, stop to export results; otherwise repeat step 4 to 7.

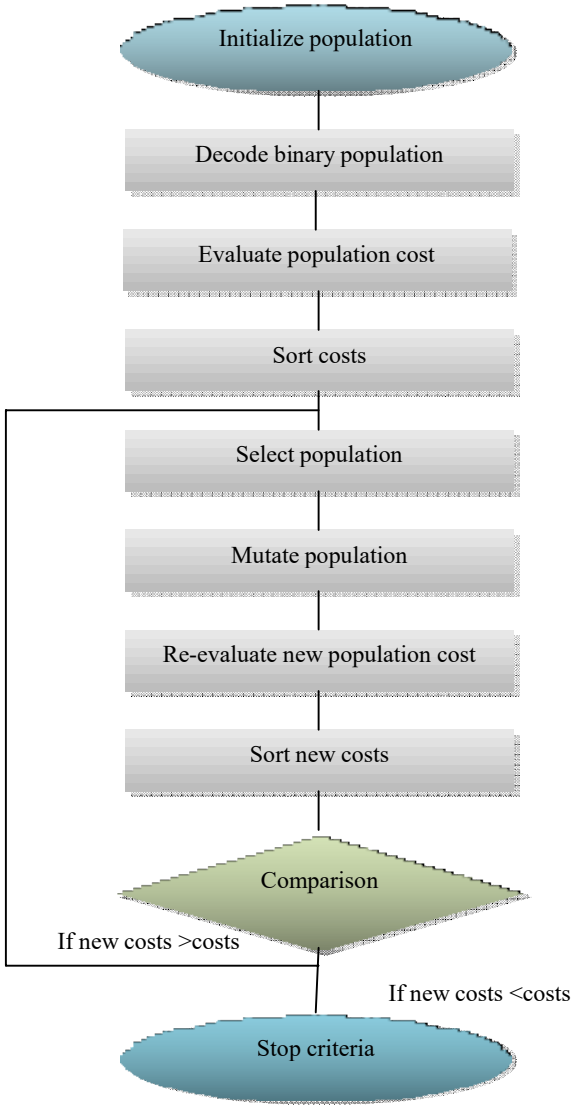

Fig. 5. Main program of proposed algorithm based on GA

IV. Simulation Results and Analysis

In this section, we tested the proper operation of the genetic algorithm to optimize the multi-objective function of partitioning problem that depends on the characteristics of functional modules.

This procedure is iterated until reaching the objectives limits or maximum number of generation. Genetic algorithm converges to the optimal solution. In this case, it can be concluded that we obtained the best solution respecting the different constraints: For area the maximum consumed resources are fixed to 7500 six-input LUTs, the maximum hardware multipliers is set to 288 DSP , for the memory used we don't have to exceed 4752 Kb, and the maximum execution time is set to 3791 clock_cycle. Thus, the designer can choose the appropriate allocation and scheduling for his application.

Our results are evaluated by approximating to that obtained by NSGAII. Table 1 presents a comparison between the two algorithms. According to these results, we got the best compromise of the different costs corresponding to the partitioning solution represented by the binary vector. It can be seen that GA gives better results compared to NSGAII especially the values obtained for the execution time. However, these values are acceptable knowing that the communication time between all the used functional modules is taken into acount.

TABLE1. Comparison between optimization solution of GA and NSGAII

	NSGAII[1]	***GA(our results)***
Binary vector	111111000111111	101001111111111
Area	7134	7287
Hardware Multiplier	33	87
Memory	103	59
Execution time	5768	2543

We point that the best solution doesn't manifest in values represented in Table 1. Indeed, for a fair comparison, we tried to choose the nearest aspect of the solution obtained by NSGAII respecting the same number of implemented modules in hardware and software with different scheduling. However, we can offer better tradeoffs area/execution time obviously for other partitioning solution.

For example, by reducing the number of modules assigned to hardware resources as shown in Table 2, the area value is increased while maintaining an acceptable result for execution time.

TABLE2. Example of optimization solution

	Binary vector	***A***	***HM***	***Memory***	***Tex***
S1	101101101001000	7062	198	112	3654
S2	101101101001001	7101	191	68	3316

V. Conclusion

In this paper, we presented a new method based on Genetic algorithm for the Hw/Sw partitioning problem for SoC FPGA-based sensorless AC drive applications. The aim of this work lies in the optimization of a multi-objective problem on automatic architecture that seeks to find the optimal solution in terms of Hw/Sw partitioning.

The proposed approach is taking into account different constraints that give better compromise area/execution time. A set of tests was provided to verify the results given by our

ISSN:2356-5608

algorithm. These results were found satisfactory following the comparison with those obtained by the NSGAII algorithm.

Other approaches or hybrid algorithms for partitioning problem on the system of multiprocessors will be included in future work.

REFERENCES

[1] I. Bahri, L. Idkhajine, and E. Monmasson, "Hardware/Software Codesign Guidelines for System on Chip FPGA-Based Sensorless AC Drive Applications,"IEEE. Trans. On. Industrial. Informatics, vol. 9, NO. 4, pp. 2165-2176, November 2013.

[2] B. Knerr, M. Holzer, and M. Rupp, "Novel Genome Coding Of Genetic Algorithms for the System Partitioning Problem," Institute of Communications and RF Engineering Vienna University of Technology, Austria, pp. 134-141, 2007 IEEE.

[3] L. Cui,"A Novel Approach to Hardware/Software Partitioning for Reconfigurable Embedded Systems," Journal Of Computers, vol. 7, NO.10, pp. 2518-2525, October 2012.

[4] P. Liu, and Y. Wang, "Integrated Heuristic for Hardware/Software Co-design on Reconfigurable Devices," 13th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 370-375, 2012.

[5] Y. Manai, J. Haggège, M. Benrejeb, "New Approach for Hardware/Software Embedded System Conception Based on the Use of Design Patterns", J. Software Engineering & Applications, pp. 525-535, 2010.

[6] I. Bahri, "Contribution of FPGA-based System-on-Chip controllers for embedded AC drive applications", Universite De Cergy Pantoise, Labortoire SATIE – UCP/UMR 8029, 1 rue d'Eragny, 95031 Neuville sur Oise France, 2011.

[7] P. Arato, S. Juhasz, Z. A. Mann, A. Orban, and D. Papp, "Hardware-software partitioning in embedded system design," Budapest University of Technology and Economics Department of Control Engineering and Information Technology, H-1117 Budapest, Magyar tudosok korutja 2, Hungary, pp. 197-202, 2003 IEEE.

[8] G. Wang, W. Gong, R. Kastner, "A New Approach for Task Level Computational Resource Bi-partitioning", unpublished.

[9] B. Mei, P. Schaumont, S. Vernalde, "A Hardware-Software Partitioning and Scheduling Algorithm for Dynamically Reconfigurable Embedded Systems", unpublished.

[10] Y. Manai, "Contribution à la conception et la synthèse d'architectures de systèmes embarqués utilisant des plates-formes hétérogènes", L'Unité de Recherche LA.R.A Automatique de l'´Ecole Nationale d'Ingénieurs de Tunis, 2009.

[11] M. Imran Ali, "Hardware/Software Partitioning and Scheduling Algorithms for Dynamically Reconfigurable Architectures", COE-572 TERM PAPER FALL, 2003.

[12] B. Knerr, M. Holzer, and M. RuppNovel, "Genome Coding of Genetic Algorithms for the System Partitioning Problem", Institute of Communications and RF Engineering Vienna University of Technology, Austria, pp. 134-141, 2007 IEEE.

ISSN:2356-5608