Real Time Distributed Embedded Systems Performance Optimization using Multi-objective Genetic Algorithms

Fateh Boutekkouk
Department of Mathematics and computer science,
University of Oum El Bouaghi, Algeria
fateh_boutekkouk@yahoo.fr

Chafia Bounabi
Department of Mathematics and computer science,
University of Oum El Bouaghi, Algeria
Bounabi_ch@yahoo.fr

Abstract—**In this paper we present a multi-objective genetic algorithm to optimize the performance of a time-driven real-time distributed embedded system with mixed constraints and a shared bus based on the so-called technique: Dynamic Voltage Scaling (DVS). The three objectives to minimize are the energy consumption, the average response time and the number of tasks missing their deadlines.**

Keywords; Real Time distributed embedded systems; Real Time Schedulingl; DVS; Gentic Algorithms ; Multi-Objective Optimization

I. INTRODUCTION

Embedded Systems (ES) [8] are increasingly present in our daily life. An ES is a system that contains application-specific hardware and software suited to a particular task that is part of a larger system that is not necessarily computer (e.g. electronic, mechanical, etc.). . An ES interacts with the outside world via the sensors / actuators and subjected to strict spatial, temporal and energy constraints. Design of such complex systems faces several challenges and requires collaboration between teams from different disciplines (e.g., software, hardware, system integrator, etc. . .). Indeed, ES are heterogeneous in nature due to the heterogeneity of the applications they implement. They typically combine software components (general purpose processors, Digital Signal processors, etc.). Unlike a hardware implementation, a software implementation has the advantage of providing flexibility (i.e. the possibility of reprogramming), but at the prize of satisfying performance constraints. A distributed embedded system (DES) represents the distributed version of the centralized ES with several controllers (node) independent which are interconnected by a shared bus, a hierarchy of buses or a communications network generally with three layers: the physical layer, the link layer and the application layer. Automobiles, aircraft, submarines, networks on chip (NOC: Network On Chip) are good examples of systems integrating DES. A DES is called real time if it is able to meet its timing constraints. In fact, we can classify the DES depending on time constraints into three classes: hard DES, soft DES and firm DES. Similarly, DES can be classified according to the type of process and/or communications in three families time-driven (periodic), event-driven (Aperiodic) or mixed. In what follows, we are interested in time-driven DES with mixed constraints (hard and soft). Since these systems have independent batteries, their design should minimize energy consumption to prolong the life of these batteries. Managing energy consumption has become a major challenge in such systems. Among the possible solutions we find energy-aware scheduling algorithms. The aim of this work is to apply multi-objective genetic algorithms to optimize the performance of a time-driven DES with mixed constraints and a shared bus topology relying on the so-called technique: Dynamic Voltage Scaling (DVS). We believe that genetic algorithms can find near optimal especially for nonlinear problems which have a vast space research. Although genetic algorithms are very sensitive to some random parameters (e.g. mutation probability), the experience of the user can play a leading role to drive the genetic algorithm towards good solutions. For this reason, we generally prefer to use interactive genetic algorithms. Our paper is organized as follows: the second section reviews briefly the state of the art on scheduling algorithms. The third section presents some energy reduction techniques. The fourth section will shed light on multi-objective optimization using genetic algorithms. In the fifth section, we try to define and model our problem. The sixth section develops our proposed genetic algorithm. The seventh section talks about our implementation with an explanatory example. Finally, we conclude this paper by presenting the contributions of our solution and identifying the perspectives.

II. STATE OF THE ART

Among the techniques that have been used to optimize the performance of real time DES, we find in particular schedulability analysis. In hard real time systems, the objective of this analysis is to ensure compliance with timing constraints. In soft real time systems, the primary objective is to minimize the response time of tasks. In fact, the literature is very rich and classifies the real-time scheduling algorithms according to several criteria such as: uniprocessor/multiprocessor off-line/on-line, preemptible non- preemptible, time-driven/event-driven or mixed, independent/dependent tasks, fixed or variable priority, etc. The schedulability analysis can be extremely difficult for complex systems. This is why the complex system may be transformed into a simpler model whose analysis is known. However, this practice provides only

partial results (e.g. a necessary and sufficient condition for schedulability in uniprocessor becomes a necessary condition for multiprocessors). Recently, a new class of real-time scheduling algorithms that take into account the reduction of energy consumption (energy-aware scheduling algorithms) appear [1, 2, 3, 5, 6, 7, 9]. The problem becomes more complex by adding a new dimension that is energy. Find a good solution that minimizes consumption while meeting timing constraints is a difficult problem. For this reason, researchers resort to optimization heuristics as an alternative to exact solutions. According to the literature, we can state that there are few works that take into account real-time DES with mixed constraints. In addition, most works optimize only one or two goals and do not take into consideration tasks dependencies. Our contribution is therefore to apply multi-objective genetic algorithms to minimize three objectives that are the energy, the average response time of tasks and the number of tasks failing to meet their deadlines for real-time DES with a set of dependent tasks and mixed constraints.

III. ENERGY CONSUMPTION REDUCTION TECHNIQUES

Reducing techniques can be divided into two strategies: hardware and software solutions. The first strategy is working on the technology hardware. For instance reducing the size of the components, limiting the power of the component blocks required for ongoing treatment, limiting the number of state changes in a circuit, using FPGAs and asynchronous electronic circuits. The software also has an important role to minimize energy consumption, for example, optimizing programs executable code and replacing memory operations by register to register ones. There are of course hybrid techniques based on synergy between hardware and software components; for example Dynamic Power Management (DPM) strategy and Dynamic Voltage Scaling (DVS) strategy. Many modern processors can dynamically lower the voltage to reduce energy consumption. Unfortunately, reduction of the supply voltage leads to an increase in the circuit delay. In turn, the propagation delay limits the clock frequency of the microprocessor. Therefore, the benefits of DVS technology can be operated in real-time systems only after careful identification of conditions under which we can safely slow down the processor without missing a deadline.

IV. GENETIC ALGORITHMS AND MULTI-OBJECTIVE OPTIMIZATION

Genetic algorithms are optimization algorithms based on techniques inspired from genetics and natural evolution mechanisms like crossover, mutation, selection, etc. They belong to the class of evolutionary algorithms [4]. However, the principle of multi-objective optimization is different from that of a single objective approach. In a multi-objective optimization problem, there are many objective functions to optimize; each objective function may have a different optimal solution. The purpose of a multi-objective problem is to find a compromise rather than a single solution. When there are multiple objectives, the concept of optimum change and it is better to use the optimal of Pareto. In the context of solving multi-objective problems, there is no accurate and efficient procedure. The NP-hard complexity and the multi-criteria framework of these problems justify the resort to heuristics that sacrifice completeness to gain efficiency. The literature is very rich, but we chose the weighted aggregation method which reduces the multi-objective optimization problem to a problem with a linear combination of the original objectives. It means to transform the multi-objective problem to a mono-objective problem by associating a weight with each objective function and then summing them to obtain a new single objective function. The coefficients are generally selected based on the relative importance of objectives. The formulation of an appropriate fitness function is one of the major problems in a genetic algorithm. Suppose there are n attributes to minimize and m attributes to maximize. These attributes have different scales, values and measures. For this we must normalize them, then we associate to each attribute a weight that reflects its relative importance. The fitness function can be defined as follows: $F = \sum w_i*((q_i-q_{min})/(q_{max}-q_{min})) + \sum w_j*(1-(q_j-q_{min})/(q_{max}-q_{min}))$ such that $w_i(w_j)$ is the weight associated with the attribute to minimize (maximize) with $\sum w_i+\sum w_j=1$.

q_{min} and q_{max} are the minimum and the maximum values of the objective function in the current population.

V. PROBLEM DEFINITION AND MODELING

A. *Problem definition*

We deal with the well-known problem of allocation / scheduling in real-time time-driven (periodic tasks) DES with mixed constraints (tasks with hard and soft constraints). Our goal is to find the best pair: allocation /scheduling of tasks which minimizes the average response time of tasks, the total energy consumption and the number of tasks with soft constraints do not meet their deadlines so that all tasks with hard constraints must meet their deadlines.

B. *Modeling*

A real-time DES consists of two parts: the application and the hardware architecture on which the application runs. In our case, we assume that the application is modeled as a task graph which consists of a set of sub-graphs. Each sub-graph includes a set of nodes (dependent tasks) and is characterized by a period. So the tasks of a sub-graph have the same period. Two dependent tasks are connected by an arc labeled with a name of a message. Each task in the sub-graph is characterized by a name, the constraint type (hard or soft), the period, the precedence priority, the relative deadline and the execution time in terms of clock cycles which can be estimated for the worst case (WCET) if the constraint is hard or the average case (ACET) if the constraint is soft . Each message is characterized by its size in bytes and a priority.

Similarly, the hardware architecture is modeled as an undirected graph where nodes denote the component architecture processors and edges denote the physical links between the processors (there are several topologies for linking processors but we choose the shared bus topology). Each processor has features such as a number to identify it and the different modes of functioning. For each mode, the dynamic energy consumed per cycle and the associated relative frequency and the static energy which is assumed constant. Of

course, we introduced the time and energy overheads to pass from one mode to another. The bus also has its own characteristics such as the throughput (bytes/cycle), the average dynamic energy per cycle and the static energy. The allocation is to assign tasks to processors and messages to the shared bus. Scheduling, however, is to define the order of execution of the tasks(s) and messages on processor(s) and buses. The order of execution is defined by the priority. We assume that scheduling is not preemptible. Note that if two dependent tasks are allocated to the same processor, the message transfer time between the two tasks is considered neglected. For a complete model, we introduced for each task and message, the arrival date, the execution start date and the date of completion. The response time equals to the difference between the completion and arrival dates. Performance at the processor level is computed through a time interval that corresponds to the lowest common multiplier between tasks periods.

VI. OUR ALGORITHM

In this section, we will detail the main components of our multi-objective genetic algorithm.

A. Fitness function

This function allows us to measure the efficiency of the solution. The relevance of potential solutions essentially depends on the formulation of this function. Indeed, whatever its definition, the algorithm converges to an optimum of this function. In this work we minimize three objectives that are the average response time, the energy consumption, and the number of tasks missing their deadlines. The function is defined as follows: $F = \sum w_i*((q_i-q_{min})/(q_{max}-q_{min}))$ *with* $\sum w_i=1$

w_i is the weight associated with attributes to minimize. q_{min} and q_{max} are respectively the minimum and the maximum values of the objective function in the current population .

We have three variables:

$q1 =(1/n)* \sum Rt$ (the average response time), where Rt is the response time = time elapsed between the arrival date and the completion date of the task including transfer time messages.
$q2 = \sum (Edyn+Estat)$ (energy consumed = dynamic energy + static energy).

$Edyn=\sum Ci*Ei+Em$ where C_i is the execution time of task i and E_i is the dynamic energy per cycle of the processor on which the task i runs, *Em* is the energy consumed by messages.
$q3 = comp$ (number of tasks missing their deadlines), where *comp* is an integer variable which is is incremented by 1 when a task exceeds its deadline.

Note that any processor with a number of allocated tasks equals to zero, have to be turned off (static energy equal to zero).

B. Solutions coding

The basic chromosome is an array of genes. Each gene is a composite structure that stores the relevant information about the task including like the task name, the task priority, the number of the processor on which the task runs, and the operating mode of the processor. All this information is extracted from the graph of tasks and the associated architecture. Generally dependent tasks keep their precedence priorities if they are assigned to the same processor. In the other cases, the priorities are set randomly. The initial population of chromosomes is generated randomly, but we must ensure that a task can not be assigned to different processors at the same time. Of course we must also ensure that all precedence priorities are respected and that each message has a different priority to resolve the conflict on the shared bus.

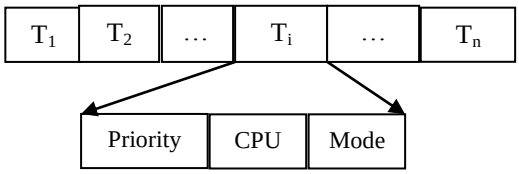
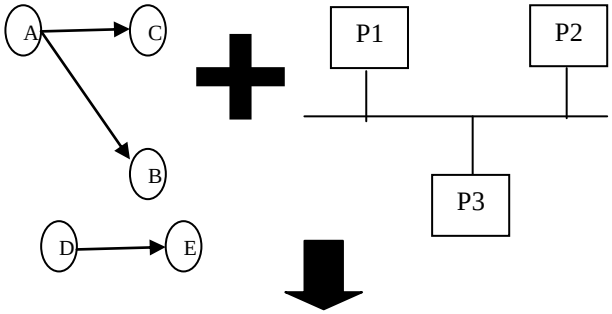


Figure 1. Chromosome coding



A	B	E	D	C
P1	P1	P3	P2	P3
1	2	2	1	1
low	low	low	high	low

Figure 2. Chromosome generation from tasks graph and hardware architecture

C. Selection

We have adopted the so-called elitist selection technique to sort the solutions increasingly depending on their adaptation to the fitness function, then it takes up half of the population that improves the fitness function , that is to say, we choose the N / 2 best solutions for the next generation.

D. Crossover

This operator allows the creation of new individuals from parents, and therefore it allows the exchange of information between chromosomes. Firstly two individuals are randomly selected then the genes are selected randomly from both

parents P1 and P2 according to their adaptations to the fitness function. The number of selected genes is defined as follows:

if β = *fitnessP1/ (fitnessP1+ fitnessP2)* then P1gives β% and P2 gives 100 - β% genes.

E. Mutation

This operator allows the search to escape from local optima by changing the value of a single gene in a chromosome. In our case the change concerns the priority task (change the priority of a task randomly), the mode of CPU (choose high or low mode) and the CPU on which the task is allocated (change the number of CPU). For each mutation (e.g. change the processor) a certain probability (introduced by the user) is associated with. The algorithm generates a random number between 0 and 1. If this number is less than or equal to the probability of mutation then the gene value is changed otherwise we do not change it.

VII. IMPLEMENTATION

Our genetic algorithm is implemented in C + + Builder 9. We defined five classes that are subgraph, task, Message, architecture, processor, and bus classes. We also implemented the operators and the genetic algorithm.

Example

lets assume that we have a graph composed of four (4) tasks with hard constraints and a period equals to 100 clock cycles. The architecture consists of three processors connected by a shared bus with a throughput equals to 400 bytes/cycle, a dynamic energy equals to 2 watts/cycle and a static energy equals to one watt. We assume that the three processors have two modes: the high mode: frequency = 1, the dynamic energy = 3 , the low mode : frequency = 0.1 , dynamic power = 0.5 . Note that all these values are normalized (high frequency corresponds to 1). The task parameters are shown in Table 2. P = period D = deadline, S = the date of arrival, WCET = execution time in the worst case. Table 3 gives messages parameters. .

The weights associated with the three objectives are as follows:
Average response time → w1 = 0.3.
Energy consumption → w2 = 0.6.
Number of tasks missing their deadlines → w3 = 0.1.

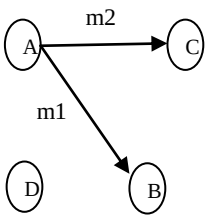


Figure 3. Tasks graph with dependencies

TABLE I TASKS PARAMETERS

Task	Type	P	Priority	WCET	D	S
A	Dure	100	1	13	16	0
B	Dure	100	2	13	30	0
C	Dure	100	3	15	50	0
D	Dure	100	3	1	40	0

TABLE II PROCESSORS PARAMETERS

	Processor Number	Dynamic energy/ frequency	Static energy
P1	0	3 1 0.5 0.1	3
P2	1	3 1 0.5 0.1	2
P3	2	3 1 0.5 0.1	1

TABLE III MESSAGES PARAMETERS

Message	Size (Bytes)	Priority
m1	800	1
m2	1200	1

The probabilities of mutations were initially set to 1. The population size is assumed to be constant (equals to 20). To do this, we always select the half (the top ten solutions) for the crossover. Each crossover between two parent solutions produced two new child solutions. In the end, we merge the ten parents with the ten new children to form a population of twenty (20) chromosomes.

The algorithm stores the four best found solutions. The high processor mode is denoted by 0. The low mode is denoted by 1. Similarly, the hard constraint is denoted by 0 and the soft constraint by 1. Tasks and processors are numbered from 0 (task A) to 3 (task D) and 0 (processor P1), 2 (P3 processor) respectively. The time and consumed energy overheads during transitions from one mode to the other are the same for all processors:

From the high mode to the low mode: time = 1 clock cycle, energy = 1.5 watt.

From the low mode to the high mode: time = 2 clock cycles, energy = 2.5 watt.

Figure 4 shows the result of running the algorithm after 100 iterations. The best solution found has a mean response time equal to 21 cycles, energy consumption equals to 126 watt and the number of tasks missing their deadlines equals to zero. The found solution suggests that tasks 0 (A), 1 (B) and 3 (D) are assigned to processor 2 (P3) with the high mode (0) and Task 2 (B) is assigned to processor 1 (P2) with the high mode (0). We note that the processor 0 (P1) is free.

VIII. CONCLUSION AND PERSPECTIVES

This work proposes the use of multi-objective genetic algorithms to optimize performance in real-time DES with periodic tasks and mixed constraints. The three objectives considered are the average response time, the energy consumption and the number of tasks missing their deadlines. To find a good compromise between time and energy, we adopted the DVS technique. We believe that our algorithm with the user support (for example, he can change the weights, the probability of mutation and the number of solutions to choose from) can find Pareto optimal. As a perspective, we plan to conduct more tests to investigate the influence of weights, the probability of mutation, the rate and the crossover technique on the quality of the solutions and also to incorporate Aperiodic tasks in our algorithm .

REFERENCES

[1] S. Albers and A. Antoniadis, "Race to idle: new algorithms for speed scaling with a sleep state". In Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'12, pp. 1266–1285. SIAM, 2012.

[2] S. Albers and H. Fujiwara, "Energy-efficient algorithms for flow time minimization". *ACM Trans. Algorithms*, 3, November 2007.

[3] H. Aydin, R. Melhem, D. Mosse, and P. Mejıa-Alvarez, "Power-aware scheduling for periodic real-time tasks". In IEEE Transactions on Computers, 53(5):584– 600, May 2004.

[4] P. Calegari, G. Coray, A. Hertz, D.Kobler, and P. Kuonen, "A taxonomy of evolutionary Algorithms in combinatorial optimization". Journal of Heuristics, 5(2):145–158, 1999.

[5] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors". In Proceedings of the International Symposium on Low Power Electronics and Design, pages 197–202, Monterey, CA, August 1998.

[6] C. Scordino and G. Lipari, "Using resource reservation techniques for power-aware scheduling". In Proceedings of the 4th ACM International Conference on Embedded Software, pages 16–25, Pisa, Italy, September 2004.

[7] Y. Shin, K. Choi, and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors". In Proceedings of the International Conference on Computer Aided Design, pages 365–368, November 2000.

[8] W. Wolf, "Computers and Components Principles of Embedded Computing System Design", Morgan Kaufman Publishers, 2000.

[9] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced cpu energy". In Proceedings of the 36th Annual Symposium on Foundations of Computer Science, FOCS '95, pages 374–382, Washington, DC, USA, 1995.

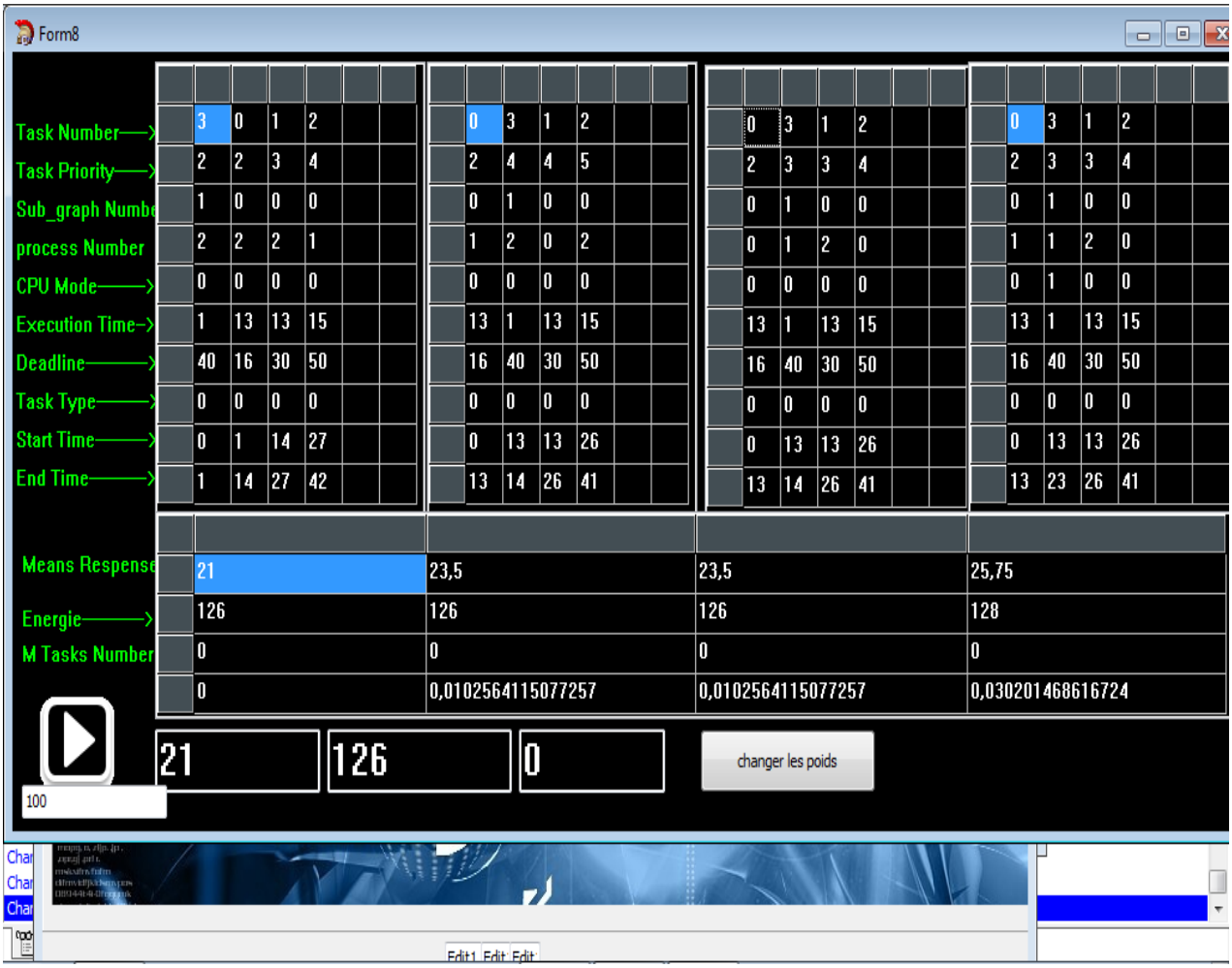


Figure 4. Results of algorithm genetic running