Prediction Performance Method for Dynamic Task Scheduling, case study: the OLLAF Architecture

Ismail Ktata, Fakhreddine Ghaffari, Bertrand Granado, Mohamed Abid

Prediction Performance Method for Dynamic Task Scheduling, case study: the OLLAF Architecture

Ismail Ktata[1,2], Fakhreddine Ghaffari[1], Bertrand Granado[1] and Mohamed Abid[2]
[1]*ETIS Laboratory, CNRS UMR8051, University of Cergy-Pontoise, ENSEA, 6 avenue du Ponceau F95014 Cergy-Pontoise, France*
[2]*Computer & Embedded Systems Laboratory (CES), University of Sfax, National School of Engineers of Sfax (ENIS), B.P.W. 3038 Sfax, Tunisia*
[1]Email: {firstname.name}@ensea.fr
[2]Email: {firstname.name}@enis.rnu.tn

Abstract

Actual dynamic applications, executed on real-time systems, have the tendency to be built on dynamically reconfigurable hardware devices. These applications require high performance and flexibility towards user and environment needs. To perform these application requirements, efficient mechanisms to manage hardware device must exist. In this paper we target OLLAF as a dynamically reconfigurable architecture which is designed to support complex and flexible applications. In order to deal with all of the dynamic aspects of such systems, we describe a predictive scheduling allowing an early estimation of our application dynamicity. A vision system of a mobile robot and an application of 3D synthesis images were served to validate the presented scheduling approach.

Keywords: Dynamically Reconfigurable Architecture, OLLAF, uncertainty variation, predictive scheduling.

I. Introduction

Embedded systems are presented around us in automobiles, robots, planes, satellites, industrial control systems, etc. They offer enormous opportunities for novel functionalities but, at the same time, system complexity is increasing and the actual methods and tools are immature, or have not yet been adapted to. In addition, the today applications have a dynamic behavior which is managed only in software as actual hardware targets are still static. To overcome these problems, designers tend to use dynamically reconfigurable architectures (DRA) that are well suited to deal with the dynamism of applications and allow better compromise between cost, flexibility and performance [1]. Such architectures increase the complexity of the applications design. This complexity could be abstracted at run time by providing an operating system that abstracts the lower level of the system [2]. This operating system has to be able to respond rapidly to events. In this paper, we are interested in the scheduling of applications that could be executed on dynamically reconfigurable architectures and in particular in predictive scheduling. The reason why prediction is becoming an important objective is the recent focus on large systems that can be dynamic and where uncertainty in terms of execution time or resource usage can be very important.

The remaining of the paper is organized as follows. In Section II we introduce a brief review about the context and problematic revealed. Several works dealing with scheduling approaches for flexible and dynamic systems are detailed in Section III. The new proposed approach of predictable scheduling technique is described in Section IV. In Section V we present an example of our method applied to a vision system of a mobile robot. Finally, conclusion and future works are given in Section VI.

II. Problem Definition

Our goal is an efficient management of dynamically reconfigurable architectures; more precisely, as case study, we target the OLLAF architecture. OLLAF, as presented in [2], is an original FGDRA specifically designed to enhance the efficiency of OS services necessary to manage such architecture. From the global view (figure 1), OLLAF has a reconfigurable logic core organized in columns. Each column can be reconfigured separately and offer the same set of services. A task uses an integer number of columns and can be moved from one column to another without any change on the configuration data [2]. Each column provides a hardware configuration manager (HCM) and a local cache memory (LCM). Configurations have to be placed in advance in the local cache memory. In the first prototype, those memories

can store 3 configurations and 3 contexts. The associated service running on the micro-processor will thus need to take into account and prefetch task configuration on the columns where it might most probably be placed. The reconfigurable logic core uses a double memory plan. With this topology, the context of a task can be shifted in while the previous task is still running and shifted out while the next one is already running. The effective task switching overhead is then taken down to one clock cycle. In [2], authors showed some cases studies which demonstrate that the OLLAF architecture can perform a greater efficiency than the one performed using a traditional commercial FPGA. Our intervention is then to make a dynamic and predictable scheduling to better manage a dynamic real time application executed on such architecture. Being able to predict in most cases the future task that will run in a particular column will permit to take even better advantage of the context and configuration management scheme proposed in OLLAF.

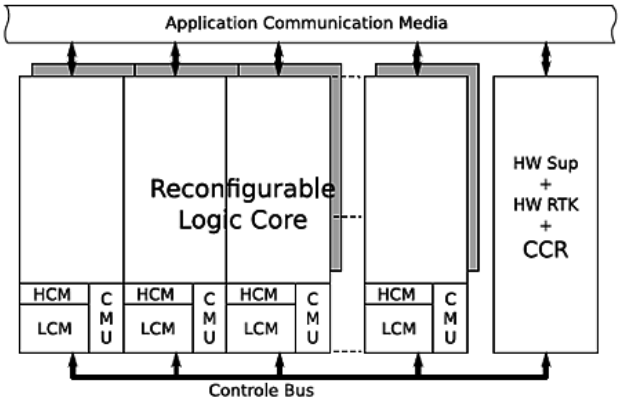

Figure 1. Global view of OLLAF architecture

Uncertainty in scheduling may arise from many sources [3]. In our research, we will consider three cases:

- Tasks execution time uncertainty:

Variability in the execution time requirement of real time tasks is common in actual real time systems. For real time tasks, meeting deadlines is important. Traditional scheduling algorithms schedule real time tasks assuming that each task requires the worst case execution time (WCET). This is a pessimistic approach since tasks are rarely executed via their WCET [4].

- Dynamical creation of multiple instances of a task:

As hardware resources are limited in embedded systems, redundant copies are considered only for software tasks [5]. With its hierarchical memories, OLLAF architecture has overcome this problem.

- Resources failure:

The goal in architectural reconfiguration is to modify the hardware topology by reallocating resources. In some situations, some resources become unavailable either due to a fault or due to reallocation to a higher priority task, or due to a shutdown in order to minimize the power usage. There is a need to predict such failure so that the system could react and failure does not disrupt the application process.

III. Related Work

To realize an efficient predictive schedule of an application, an operating system needs to know the behavior of this application, in particular the part where the dynamicity can be efficiently exploited on a DRA.

The different items of a scheduling problem are the tasks, the constraints, the resources and the objective function. Tasks execution must be programmed to optimize a specific objective with the consideration of several criteria. Many resolution strategies have been proposed in literature [6]. Usually these methods assume that execution times can be modeled with deterministic values. They use predictive schedule that gives an explicit idea of what should be done. Unfortunately, in real environments, the probability of a pre-computed schedule to be executed exactly as planned is low [7]. This is because of, not only variations, but also because of a lot of data that are only previsions or estimations. It is then necessary to deal with uncertainty or flexibility in the process data. Hence, a significant reformulation of the problem and the solving methods are needed in order to facilitate the incorporation of this uncertainty and imprecision in scheduling [8]. In general, there are two main approaches dealing with uncertainty in a scheduling environment according to phases in which uncertainties are taken into account [3]. Proactive scheduling approach aims to build a robust baseline schedule that is protected as much as possible against disruptions during schedule execution. It takes into account uncertainties only in design phase (off-line). Hence, it constructs predictive schedule based on statistical and estimated values for all parameters, thus implicitly assuming that this schedule will be executed exactly as planned. However, this could become infeasible during the execution due to the dynamic environment, where unexpected events continually occur [3]. Instead of anticipating future uncertainties, reactive scheduling takes decisions in real-time when some unexpected events occur. A reference deterministic scheduling, determined off-line, is sometimes used and re-optimized. In general, reactive methods may be more appropriate for high degrees of uncertainty, or when information about the uncertainty is not available. A combination of the advantages of both precedent approaches is called proactive-reactive scheduling. This hybrid method implies a combination of a proactive strategy for generating a protected baseline schedule with a reactive strategy to resolve the schedule infeasibilities caused by the disturbances that occur during schedule execution. Hence, this scheduling/rescheduling method permits to take into account

uncertainties all over the execution process and ensures better performance [9] [10]. For rescheduling, the literature provided two main strategies: schedule repair and complete rescheduling. The first strategy is most used as it takes less time and preserves the system stability [11].

In our case, we are dealing with hardware tasks scheduling for real time applications. We are concerned with soft real-time systems, such as telecommunication and multimedia, where the goal is typically to meet some Quality of Service (QoS) requirements. When developing the real time systems, it is often required to predict tasks properties of the application, especially the timing property [12]. Usually, scheduling algorithms are based on the worst-case execution time (WCET) of a task. They assume that the WCET is known a priori. Unfortunately, in general, the worst-case input is not known and depends on some factors such as input data and available resources [13]. In general, scheduling algorithms focus on offline WCET estimation to use it at runtime. However, those investigations on WCET estimation are not optimal because, in most cases, real time tasks are not executing via their worst case execution times. Therefore, the improvements of efficient execution time analysis are still limited and the resources utilization is extremely high [14]. Instead of scheduling based on the worst case execution time as an upper boundary time elapsed for a task to execute, we prefer to get advantage of that characteristic and attribute, for each task, a dynamic online estimated value of execution time. The predictive algorithm will permit to schedule the different hardware tasks (T_i) and allocate its needed cells of the hardware DRA based on online estimated values.

IV. Proposed Approach

As we mentioned in section III, to realize a predictive scheduling we must exhibit the dynamic characteristics of an application. We consider three cases of dynamicity in applications:

(a) The number of tasks is not fixed. It may change from iteration to another.

(b) The tasks execution time may change too.

(c) The number of needed resources for tasks execution is variable. In addition, the number of available resources may decrease after a failure occurs.

For these cases, the goal is to develop a robust scheduling method that is little sensible to data uncertainties and variations between theory and practice.

Our first work in [15] was to define a new model to represent dynamic applications features we considered. This model will be the input of the scheduler. At design time, the scheduler is able to distinct between two parts of the application: permanent branches (containing tasks that will be executed during the whole application runtime) and hazardous ones. Obviously, scheduler can take into account all dynamic features of the application presented in the model. Permanent tasks will be sorted respecting their precedence constraints (topological order). Its configuration data will be prefetched, as much as possible, on the columns making a maximum use of the whole of the architecture. This is to ensure minimizing the number of configuration data transfers and tasks relocation, and thus the time of saving and restoring process. From the model, the prediction service can identify tasks whose execution time is variable as well as hazardous tasks with possible multiple instances. At the design time, the execution times will be defined as WCET values and dynamic features will not be taken in the first period.

At run-time, there are three dynamic online features:

- Firstly, the graph precedence which is already presented by the graph but it can be modified as some nodes (i.e. tasks) may not be executed in some period.
- Secondly, the new changed parameters of some tasks. This includes execution time and instance number.
- Finally, the resource availability which may change from one period to another depending on the tasks requirements and the fact that a resources failure may occur or not.

After a task's execution and for uncertain parameters, the prediction technique will compute predicted values to be taken for the next execution. Tasks are executed at the earliest possible time. For many tasks activated at the same time, priority will be assigned according to Least Laxity First (LLF) policy. The LLF permits a dynamic priority assumption depending on the laxity which depends on the execution time (L= (D-R)-C); where D is the deadline, R is the ready time and C is the execution time. If there is equality between some tasks, task which has maximum execution time will have priority to be launched before the others.

The scheduler will proceed taking into account those features. It has to prefetch configurations data of new activated tasks in the columns that are available for executing and to find the possible way to integrate them in the current schedule without effect on performance. This schedule must rely on rapid algorithms based on a simple scheduling technique so that it can perform online execution with no overhead. In addition, on the reconfigurable device, resource failure may occur which will affect the resource availability. Thus, for some tasks, execution may occur or not depending on the available resources compared with needed ones. A variable, which is initialized as the total number of resources, will indicate the amount of remaining resources. From the ready list, LLF algorithm determines the tasks that can be executed on the reconfigurable device. Tasks with the higher priorities will be placed first until the area of device is fully occupied. If there are not enough hardware resources available, the Last Recently Used (LRU) strategy is used to

select which tasks configuration data will be removed. Therefore, for a ready task, scheduler will compare its needed resources with the variable reflecting the available ones. If availability is satisfied, the configuration will be fetched on target resources. Otherwise, last recently executed tasks will free their columns until the needed resources will be satisfied. In the case that all executed resources have released their resources and the available resources still cannot fit the needed ones, the tasks in question will be delayed and scheduler goes to the next one. During run-time, and for dynamic parameters, historic values are stored and prediction is made for the next task s period. The choice will depend on the applications requirements and the accuracy of estimating methods. Prediction tested methods will be presented in next section.

V. Experimental Results

To validate our approach, we should make use of dynamic applications presenting uncertainty information. In [16] and [17] two image processing applications are represented. First one is a visual system embedded in a mobile robot, while the other is 3D synthesis images. In the first application, robot moves around and learns its environment to identify keypoints in the landscape. The keypoints correspond to the image filtered by difference of Gaussians. This application is dynamic in the sense that the number of keypoints is not known a priori and depends on the visual scene and on the robot movement if it is slow or fast or if it is stopped. For the application of 3D synthesis images, an object can be generated using various polygons number and different shade algorithm (Gouraud, flat, Phong) so with different visual qualities. For instance, in the application of 3D synthesis images, an object can be generated using various polygons number and different shade algorithm (Gouraud, flat, Phong) so with different visual qualities. Figure 2 shows time execution model based on off-line measures for the 3D application using flat shading method.

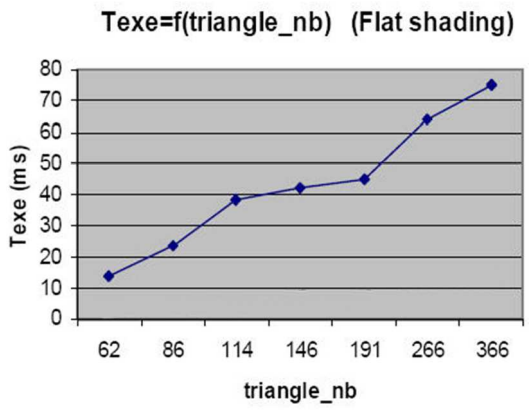

Figure 2. Texe/triangle_nb variation for flat shading

In this section we will more focus on the vision system of a mobile robot as we dispose more experimental data of it. An experiment has been done with a pure software version of this application executed on a single microprocessor. As a consequent, we can divide application tasks in three groups:

- Intensive data-flow computation tasks that execute in a constant time,
- Tasks whose execution number is correlated with the number of interest points,
- Tasks with unpredictable execution time (depending on the images features).

The figure 3 shows our proposed model for the robotic vision application. We can notice the presence of permanent branch (squared nodes) which represents permanent tasks that will be executed in all cases or modes (exp: T1, T14, T19). Tasks with unpredictable execution time are indicated by the asterisk (exp: T9, T11, T20). Another dynamic feature of tasks which execution number depends on the number of keypoints is indicated by the use of resource factor n (exp: T9, T10, T30).

For the prediction techniques of uncertain tasks features, we have studied the task of keypoints search. The provided measured execution times on representative samples are presented in the figure 4. As we can see, the values distribution is random. The elapsed time of keypoints search in an image depends on the number of keypoints that this image contains. So the execution time is correlated to the founded number of keypoints. For the fact this number is also random variable as shown in figure 5. The figure 5 shows the number of founded keypoints in the corresponding image over a sequence of more than 5000 images.

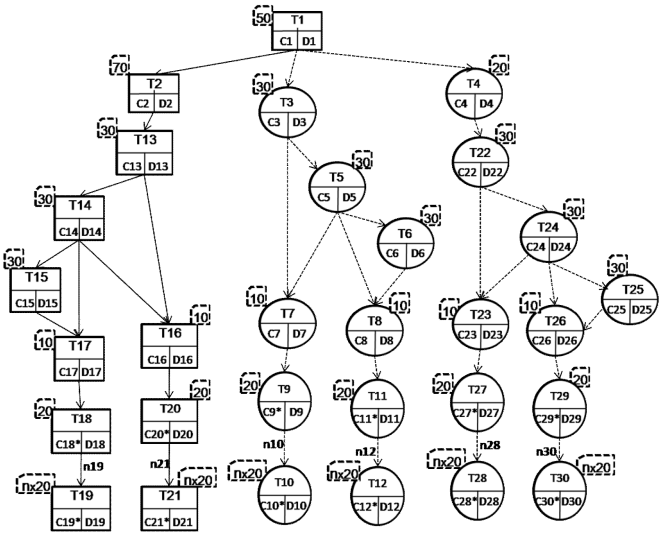

Figure 3. Our proposed model for the robotic vision application

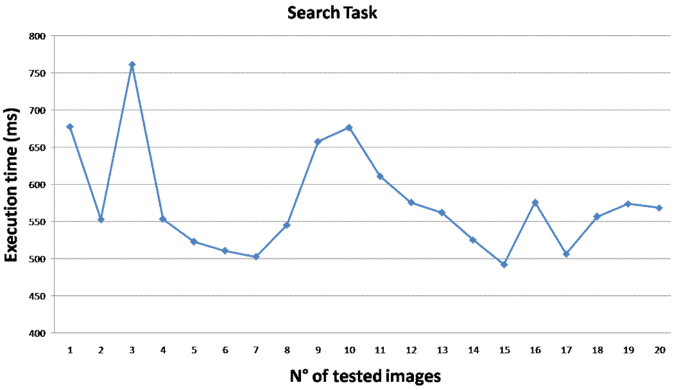

Figure 4. Execution time measurement of search task

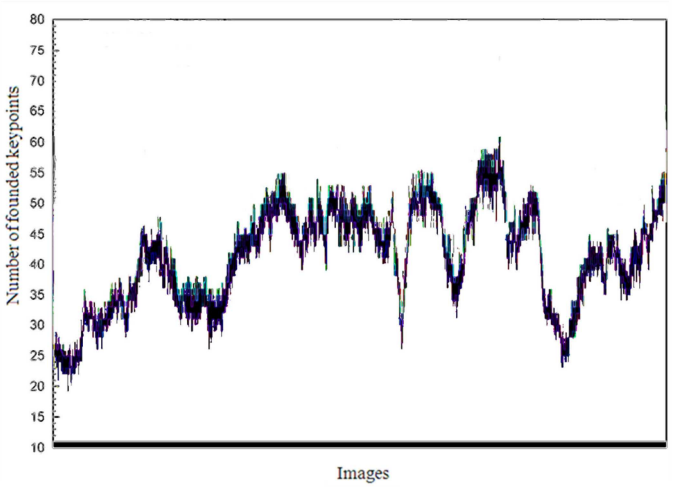

Figure 5. Number of detected keypoints in image sequence

We have compared several techniques to predict the execution time of the search keypoints task. Comparison is based on computing the error in estimation defined as:

$$E = \sum_i \frac{Real\ execution\ time(t_i) - Estimated\ execution\ time(t_i)}{Real\ execution\ time(t_i)}\ x\ 100$$

where t_i is a runtime instant and i={0,..n}.

Tested methods make statistical measurement based on a specific amount of previous succeeded instances of the same task that have been obtained by on-line monitoring. We do not use the whole historic as this need more memory size and is not useful in the prediction of such random values. The first method (a) uses the mean value of the three last real execution times. After making some successive estimations, the predicted values become almost constant and do not follow the graph evolution. The second method (b) uses the mean of the three last values multiplied by different weights. Weights are determined experimentally based on tested data to minimize estimation error. As in nonparametric regression techniques, the higher weights are assigned to the values closed to the actual parameter and lower weights to those more distant. The last two methods are more pessimistic and make overestimation of previous values: one (c) takes the maximum of the last three real values and the other (d) takes the last value increased by 5%. We use estimation error rate as a metric to compare those methods. Firstly, we notice that all methods lead, for the twenty tested images, to an overestimation of the execution time. The higher error rate is for the techniques (a) and (c). For (d), and even with a 1% increase of the last measure value, the error is still greater than the one of (b). Hence, the least obtained error percentage is the one related to the method using weighted average of last values (b). It provides a good prediction for the task execution time distribution. The mean error between estimated and simulated results is about 0.4% for low and medium frequency scales and about 2.45% for high frequency scale. Those results show that the technique based on a weighted average of the execution time values works with an accuracy of almost 98%. Even for keypoints number estimation (figure 5); the mean estimation error of the same technique is 0.406% for the whole 5000 images.

Figure 6 shows the error rate functions of the prediction of the keypoints number corresponding to figure 5. The presented function corresponds to the prediction method which uses the maximum value of last measures. The errors were calculated of over 1000 successive images. The mean error (indicated by a horizon line) is -6.5%. This technique presents significant advantage with a relative acceptable mean overestimation of 6.5% and less than 21% of the whole predicted values were underestimated. The purpose is that prediction be almost near real values and guarantee better compromise between requested QoS and efficient resources management. For this visual system application the latter estimation can assist the prediction engine for better scheduling analysis of real-time tasks and so better exploitation of the DRA like OLLAF.

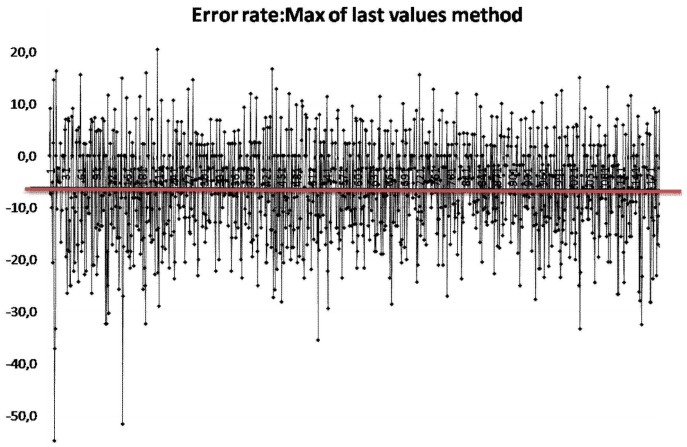

Figure 6. Prediction error function of the number of detected keypoints

VI. Conclusion

In this work, we aim at performing predictive scheduling of hardware tasks on OLLAF as a FGDRA. The proposed approach takes into account the hardware tasks uncertain characteristics, the available resources in the target device and the quality of service of the application. We have presented the scheduling method with the use of a prediction method enabling better adaptation of the architecture to the environment variations. To validate the proposed scheduling method, we have used a case of an image-processing application of a visual system embedded in a mobile robot. Such application shows dynamically variable characteristics that are unpredictable without an entire a priori knowledge on the environment. We have demonstrated that with our modeling we can realize an efficient predictive scheduling on a robot vision application with a mean error of 6.5%. Future works will consist in integrating our scheduling approach among the services of an RTOS taking into account the new possibilities offered by OLLAF. We plan also to employ preemptive scheduling policies by using the mechanisms of migration and reallocation of tasks configuration and context data proposed by OLLAF.

VII. References

[1] J. Noguera, R.M. Badia, "Multitasking on reconfigurable architectures: Microarchitecture support and dynamic scheduling", ACM Transactions on Embedded Computing Systems, Volume 3, Issue 2 (May 2004) pp. 385-406.

[2] S. Garcia, B. Granado, "OLLAF: a Fine Grained Dynamically Reconfigurable Architecture for OS Support", EURASIP Journal on Embedded Systems, October 2009.

[3] A. J. Davenport and J. C. Beck, "A Survey of Techniques for Scheduling with Uncertainty", accessible on-line at http://tidel.mie.utoronto.ca/publications.php on February 2006, 2000.

[4] Abhishek Singh, Kevin Jeffay, "Co-Scheduling Variable Execution Time Requirement Real-Time Tasks and Non Real-Time Tasks", In Proceedings of the 19th Euromicro Conference on Real-Time Systems (ECRTS): 191-200, July 2007,Washington DC.

[5] Alain Girault, Hamoudi Kalla, and Yves Sorel, "A scheduling heuristics for distributed real-time embedded systems tolerant to processor and communication media failures", In International Journal of Production Research, 42(14):2877 2898, July 2004.

[6] P. Brucker & S. Knust, "Complex Scheduling", Springer Berlin Heidelberg, 2006.

[7] V. T'kindt and J.-C. Billaut, "Multicriteria Scheduling: Theory, Models and Algorithms", Springer-Verlag (Heidelberg), second edition (2006).

[8] N. González, R. Vela Camino, I. González Rodríguez, "Comparative Study of Meta-heuristics for Solving Flow Shop Scheduling Problem Under Fuzziness", Second International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2007, Spain, June 18-21, 2007, Proceedings Part I : 548-557.

[9] H.Aytug, M.A.Lawley, K.McKay, S.Mohan, R.Uzsoy, "Executing production schedules in the face of uncertainties: A review and some future directions", European Journal of Operational Research 161, 2005, p86-110.

[10] W.Herroelen and R.Leus, "Project scheduling under uncertainty: Survey and research potentials", European Journal of Operational Research, Vol. 165(2) (2005) 289--306.

[11] G.E.Vieira, J.W.Hermann, and E.Lin, "Rescheduling manufacturing systems: a framework of strategies, policies and methods", Journal of Scheduling, 6 (1), 36-92 (2003).

[12] Jack Ganssle. Really Real-Time Systems. In Proceedings of the Embedded Systems Conference San Fransisco (ESC SF) 2001, April 2001.

[13] Wilhelm R. et al, "The worst case execution time problem overview of methods and survey of tools", ACM Transactions on Embedded Computing Systems (TECS), Vol. 7, No. 3, Article 36 (April 2008), ISSN:1539-9087.

[14] E. Yu-Shing Hu, Andy J. Wellings, G. Bernat. "A Novel Gain Time Reclaiming Framework Integrating WCET Analysis for Object-Oriented Real-Time Systems". Proceedings of the 2nd International Workshop on Worst-Case Execution Time Analysis WCET-2002.

[15] I.Ktata, F.Ghaffari, B.Granado and M.Abid, "Novel Approach for Modeling Very Dynamic and Flexible Real Time Application", 5th International Workshop on Reconfigurable Communication-centric Systems on Chip 2010 "ReCoSoC"10, May 17-19, 2010, Karlsruhe, Germany.

[16] F. Verdier, B. Miramond, M. Maillard, E. Huck, and T. Lefebvre, "Using High-Level RTOS Models for HW/SW Embedded Architecture Exploration: Case Study on Mobile Robotic Vision", EURASIP Journal on Embedded Systems Volume 2008 (2008), Article ID 349465.

[17] K.Loukil, N.Ben Amor, M.Abid, "Self adaptive reconfigurable system based on middleware cross layer adaptation model", 6th International Multi-Conference on Systems Signals and Devices, March 2009, Djerba-Tunisia.