Self adaptive reconfigurable system based on middleware cross layer adaptation model

Kaïs Loukil, Nader Ben Amor, Mohamed Abid
CES Laboratory
ENIS National Engineering School
Sfax, Tunisia
Email: Kais_loukil@ieee.org

***Abstract*—The emergence of mobile multimedia systems and the diversity of the supported multimedia applications put new challenges for their design. These systems must provide a maximum application quality of service (QoS) in the presence of a dynamically varying environment (e.g. video streaming and multimedia conferencing) and multiple resources constraints (e.g. remaining energy). To respond to the changing resource availability and application demands, a new class of adaptation method is emerged. It combines the adaptation simultaneously upon the different layers related to the target system. This paper presents a framework dedicated for mobile multimedia systems. It supports application QoS under real time and lifetime constraints via coordinated adaptation in the hardware, OS, and application layer. In this framework, we present a new middleware approach based on a global and a local manager. The global manager (GM) is used to handle large and long-term variations whereas the local manager (LM) is used to guarantee the real time constraint. The GM intervenes in three layers but the LM intervenes only in the application layer and OS layer. We have implemented this approach on reconfigurable platform using Altera technology.**

***Index Terms*—Adaptation, middleware, Embedded systems, Quality of Service.**

I. INTRODUCTION

The embedded popular electronic multimedia systems has actually been emerged and used frequently. Their functionalities are increasingly complex and need high performance architecture (a priori dedicated architectures). Due to their mobility, those systems must also function under often difficult conditions: fluctuation of network transmission, limited energy resources, etc which involves the use of programmable or reconfigurable architectures. So, these systems must be, on the one hand, powerful enough to treat the complex multimedia applications. On the other hand, they must be quite flexible to be able to adapt to the external variable environment and to respect the functional stringent constraints (like real time, lifetime, and quality of service). As those constraints are generally antagonistic, the system must find the right compromise between performance and flexibility according to its state (number of running applications, user choices, state of the battery, etc).

Various adaptation techniques were proposed for the respect of the constraints of the system while giving a better quality of service. These techniques can intervene on three different levels: on the application level, operating system level or on the hardware level. As the constraints imposed on the embedded systems are increasingly strong (increase in the number of supported applications and their complexities, limitation of available energy resources, etc), it is necessary to adopt a more global adaptation strategy which combines the previously described adaptations methods. In this context, our work consists of the addition of a middleware layer, which allows the dynamic cross layer adaptation of the system. In this paper, we present a new approach of adaptation dedicated for embedded systems which comprises primarily two managers (global manager and local manager). The global manager can intervene in the three layers in order to answer the great variations of the system constraints whereas the local manager intervenes only in the layers application and operating system. It is of a great importance to mention that the LM is set up to control the respect of the real time constraint of the system. The LM can also question the global manager if it does not manage to find a solution to solve the problem.

Our work has two major contributions. First, we propose the use of the cross layer adaptation approach. The second contribution consists of the design and the real implementation of the cross layer adaptation approach on a reconfigurable system.

This paper is organized as follows: The first part is devoted to the presentation of the state of the art. The second part presents the adaptation model. The third part treats the various algorithms set up for the

implementation of the approach on a real system. We devote the last part of our paper to the validation of the approach and the presentation of some prospects concerning this issue.

II. STATE OF THE ART

Recently, there were many research contributions on the auto-adaptation for the autonomous systems. These adaptations can be applied to the following layers: the architectural layer, the operating system layer (OS) and the application layer. Hereafter, we will show these various techniques, their contributions and their limitations.

A. Adaptation at the architecture level (HW adaptation)

Many HW adaptation techniques were proposed. One of them is the dynamic voltage scaling (DVS). It consists of adjusting the frequency and the power supply of the CPU. It is based on the application workload prediction using heuristic methods [4, 16] or worst cases CPU time estimation [2, 3]. HW adaptation was also applied to reconfigurable platforms. The change of the system architecture is made according to the needs of the application and the environmental constraints. Such method was applied for the partitioning of the system using two techniques: the former uses a heuristic algorithm [13] and the latter uses a genetic algorithm [15].

B. Adaptation at the operating system level(OS adaptation)

Much of researches have dealt with the operating system and middleware layer to provide predictable CPU allocation and adaptation services of the operating system [5, 7]. In [9, 12, 17, 1] the managers of CPU resources, provide performances guarantees in soft real time. Schedulers further adapt the scheduling policy to handle the variations of application runtime [5, 6, 7]. In [1, 11, 19], the authors use a middleware layer to facilitate the adaptation of QoS for the application system which is submitted to the time constraints of execution and to the supplied energy.

C. Adaptation at the application level

Several projects recommend the adaptation at the application layer for different purposes. For example, the authors of [18] explore the technique for adapting the behaviour of the application to the constraints of energy consumption. Mesarina and others discuss in [8] how to reduce the energy for the “decoding MPEG” application using parameter modifications. In [12, 13, 14] two approaches are proposed for the deterioration of the quality of an object 3D to satisfy the constraints of resources and network bandwidth.

D. Cross layer adaptation

Many researches have, recently, been elaborated to profit from the advantages of the previously described techniques and to reduce their restrictions. Most of these researches were based on methods which work on the various layers of the system such as model GRACE-1 [1] and TIMELY [20]. However, these approaches suppose that the material layer is not reconfigurable. The only possible adaptation to this level is on the CPU frequency.

Our model targets a mobile system with adaptive hardware architecture and running a set of parameter tuned applications (which is the case of the most popular multimedia applications). It optimizes the system resources use under constraints of lifetime, real time and quality of service. The suggested model uses adaptation in the three layers hardware, operating system and application. For the hardware layer, adaptation is done using one of the different application implementations called configuration that are previously set up and characterized. Those configurations vary from a pure SW configuration (called config_sw) to a mixed implementation with several HW components (called config_hw). As config_sw involves only CPU resources, it will consume less energy than config_hw but will less performing. So those, the system can choose the adequate configuration according to the constraints and the user preferences. Adaptation at application layer is performed by modifying the application parameters or algorithm according to the imposed constraints. The third adaptation level is made at the operating system layer in order to allocate necessary time CPU for each task (we suppose that each task is an one application).

III. ADAPTATION MODEL

In this section, we present our cross layer adaptation models. We present also the interaction between the various layers. We end the section by the presentation of the different conditions of activation of the adaptation task.

A. Hardware adaptation model

At the hardware layer, an offline step is used to characterize the different configurations (their energy consumption and execution time). Those information’s stored in a database are used on line to choose the most suitable configuration. As the system performance (energy consumption and execution time) may vary from test conditions (made off line) to real conditions (on line) even using the same architectural and application parameter, we may notice a difference between real values and stored database ones. As the system performs frequent real measures, database information can be easily updated.

B. Operating system adaptation Model

The execution of any task requires a number of CPU cycles. In order to respect the real time constraint, we must allocate, for each task, a number of CPU cycles. However, the same task can consume various numbers

of CPU cycles according to its implementation (hardware or software or mixed), the data types, etc. Hence there is a need for allocating, for each task, a well-defined number of cycles according to the applicative and architectural parameters. This time is estimated by using the profiling technique.

C. Application adaptation model

The adaptation at the applicative level is made by changing the parameters and/or the treatment algorithm. This adaptation technique is more and more used since many of multimedia applications such as MPEG, H264, speech coding or 3D image processing have different working modes and functional parameters. 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.

At this stage, an offline study must be also investigated to get different performance model (energy consumption and execution time) according to the application parameters and modes.

The coordination of the adaptation in the three layers offers a "cross layer adaptation model". Specifically, to have a quality of service for a given application which consumes a quite fixed quantity of energy, we need the configuration of adequate architecture in the hardware layer, the allowance of a number of processor cycles for each task in the operating system layer and the suitable applicative parameters in the application layer.

D. Adaptation triggers

We consider in this paper two modes of adaptation activation. The first adaptation mode is related to the execution of the task of the GM. It occurs in four cases. The first one occurs when there is a change of the number of tasks in the system. The second one takes place when there is an unexpected change of the energy level in the battery. The third one is when there is a modification of the user preferences (level of QoS, Lifetime). The last one occurs when there is a request from the LM. The second adaptation mode, is related to the execution of the task of the local manager. This task is activated in the event of deadline missing.

IV. CROSS LAYER ADAPTATION MODEL

This section presents the design of the cross layer adaptation approach. So we shall describe the architecture and the operating mode of this approach.

A. Overview

The proposed adaptation technique is integrated in a framework which coordinates the adaptation in the three layers. The figure 1 presents an overview of this framework. It is primarily made up of a global manager, a local manager, an adapter of task for each task, an OS adapter, an architecture adapter, a battery monitor and a configurations database.

The global manager coordinates between the three layers based on the supplied energy and the user preferences (desired level of the quality of service and lifetime).

The local manager coordinates between the application layer and the operating system in order to guarantee the real time constraint.

The task adaptor makes it possible to adjust the parameters or the algorithm of the task.

The OS adaptor makes it possible to adjust the number of affected CPU cycle for each task.

The architecture adaptor deals with the change of hardware architecture system.

The battery monitor gives an indication of the remaining energy of the battery.

The database of the configurations contains all the possible mixed configurations (hardware or software) for our system.

As mentioned above, the adaptation approach must have a minimum cost. Thus, the challenge is to provide best possible quality of a system while respecting the constraints of the system with a minimum overhead. To address this problem, our approach uses three techniques. The first one is an automatic adaptation when a new task is activated in the system or an existing one is stopped. In the case of a newly added task, the GM configure the hardware layer with adequate architecture, to assign to the application the suitable applicative parameters and to allocate with each task a number of cycles CPU (in the case of a new activated task). In the case of a stopped of finished task, the GL reallocate its architecture resources. The second adaptation technique is the use of estimation technique for lifetime computation. As the battery monitor activity consumes non negligible energy, estimation technique can be used to reduce direct measures (and their relative energy and time costs). When the prediction gives correct results, period of direct measures can be increased. The battery monitor is activated all the periods of time (Pa). The found value is compared with estimated value. If they are close to each other, the system increases the period of direct measures ((Pa+=n*Pa) with n>0). In the opposite case, direct measure period is readjusted. The third technique used to reduce the total cost of the adaptation approach consists of the use of a local manager which allows changing the applicative parameters of a task in the event of a deadline miss. This permit a more local adaptation (restricted to real time constraint) that does not a heavy cost whole system architecture reconfiguration. In case where the local manager does not find a solution for the system it can activate the global manager. In the following section, we give more details on these three techniques.

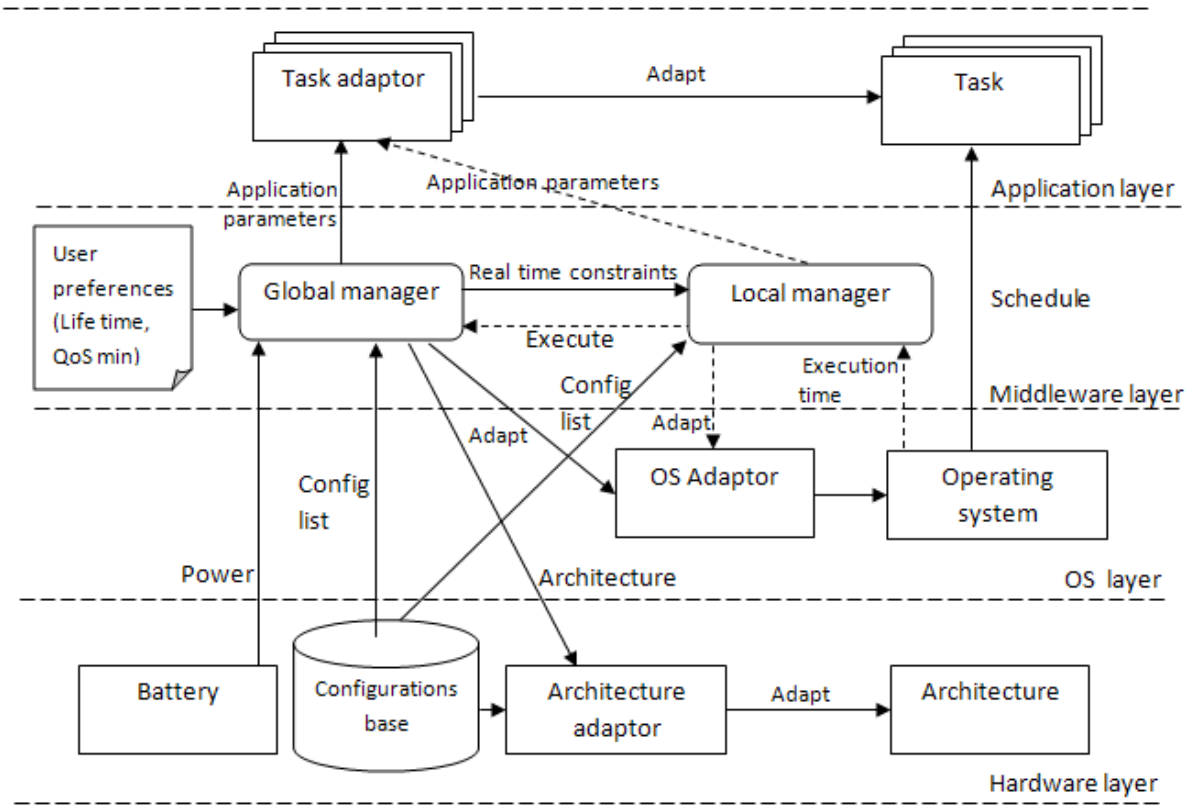

Figure 1. Cross Layer adaptation approach

B. Global manager "GM":

The global manager is activated only in three events: i) when a task appears or leaves the system, ii) when the residual energy level in the battery reaches a breaking value or iii) to answer a request of the local manager. In such a case, the GM coordinates between the three layers (application, OS, hardware) to choose the best system requirements starting from the base of configuration, which makes it possible to provide the best quality of service while respecting the preferences of the user.

The entries for the GM are the user preferences (see figure 2). For the moment, we consider two parameters, the desired lifetime "Lt" and the minimum level of quality of service accepted "LQoSmin". Through these preferences the user seeks to maximize the level of quality of service of the multi-media application running for a fixed given lifetime.

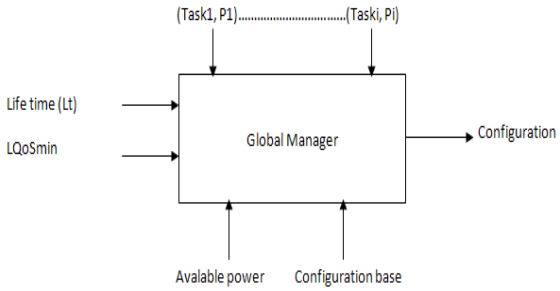

Figure 2. Global Manager

We suppose that the system executes N concurrent task. Each task noted as "Ti", 1<i<n, requests a number of CPU cycles noted as "Nci" of period called "Pi" and consumes an energy "Eci" during one period. We note with "ED" the quantity of energy available in the battery.. The Global Manager seeks to find a configuration that provides a quality of service "Qi" and lifetime "Lt" according to the equation 1 to equation 4.

Maximize Q (Q1, ..., Qn) Equation 1

Under constraint :

$$\begin{cases} Qi >= LQoSmin & \text{Equation 2} \\ \sum_{i=1}^{n} Eci * \frac{LT}{Pi} <= Ed & \text{Equation 3} \\ \sum_{i=1}^{n} \frac{Nci}{Pi} <= 1 & \text{Equation 4} \end{cases}$$

Equation 1 and 2 guarantee to provide to the user the best quality of service which must be higher than the fixed minimum QoS level. Equation 3 guarantees the satisfaction of the lifetime constraint. Equation 4 represents system scheduling constraint. This constraint requires that the execution time of all the tasks divided by period should not exceed 1 since we use the EDF scheduling algorithm. The global manager will have to select the adequate configuration for each task (architecture + applicative parameters) from the configuration database which will have to contain information related to each configuration such as the levels of QoS, the number of required CPU cycles, and the quantity of power consumption for one period. Thus, in order to choose the best configuration of each task, the GM is in front of an Np-complete problem, since it will have to extract all the possible combinations which can answer the already quoted constraints. For example, if it is considered that our system executes three concurrent tasks and we have "N" possible configurations for each task the GM will have to check n^3 solutions to extract all the possible combinations to fulfil the requirements of the system. As the adaptation task must have a very small overhead, the resolution of this problem even with heuristic techniques can have a big impact of the adaptation cost. So, we try to find out another solution simpler to solve this problem. We use a more local method based on energy and execution time budgets, allocated to each task, in order to reduce the search time of the adequate configuration.

The desired Lt is divided in equal periods of a fixed value called Quantum Q which corresponds to the moments of energy measurements. This quantum is a multiple of the hyper period (lowest common multiple of all the periods) system and can vary during the system operation (as it is mentioned above). In each Quantum, there is a budget of energy and CPU cycles (processing time) allocated which should not be exceeded (under penalty of not satisfying the constraints Lt and real time) (see figure 3).

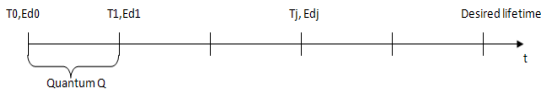

Figure 3. Quantum and energy budget

It should be noted that:

- We treat the case where the system executes several applications during the Quantum Q.
- Each application can be running one or more time during Q.
- The power consumption depends on the number of execution time of the application during Q
- The system has several constraints: Lt, real time and QoS.

1) Allocation of CPU cycles *to each task*

In order to respect the real time constraint, all the tasks must be running at each period. Thus, in the hyper period "h" of the system, the task Ti of period pi will have to be running $ni = \frac{h}{Pi}$ times. Thus, the problem amounts assigning to each task Ti an execution time Tei is given by equation 5 where n is the number of applications

$$\sum_{i=1}^{n} Tei * ni <= h . \quad \text{Equation 5}$$

Generally, we must satisfy the equation6:

$$\sum_{i=1}^{n} \frac{h}{Pi} * Tei <= h \quad \text{Equation 6}$$

In this manner, we guarantee that all the tasks are running in the hyper period of the system witch satisfies the equation3.

The GM is in front of a great number of possible solutions whose constraints need to be respected. The challenge is establish a mechanism which can find a good solution for the system with an acceptable overhead. The solution installation is based on the assignment of a budget of time for each task. Then, we choose the configuration which uses the nearest execution time and which respects the other constraints of system (QoS and LT).

In order to assign a budget of execution time to each task we proceed as follows:

We calculate the maximum total execution time of all the applications Smax as shown in equation 6.

$$Smax = \sum_{i=1}^{n} Temaxi * ni \quad \text{Equation 6}$$

With Temaxi the maximum execution time for a task i among all the possible configurations of the system

If the Smax is lower than the hyper period of the system, we can affirm that all the possible configurations represent a solution for our state. On the opposite case, we calculate the occupation "Oci" of Temaxi in Smax for each task according to equation 7

$$Oci = \frac{Temaxi}{Smax} \quad \text{Equation 7}$$

We apply the found value Oci to the amount of time exceeded:

$$Tdepi = Oci * (Smax - h) \quad \text{Equation 8}$$

The budget of time assigned to the spot Ti is:

$$Tei = Temaxi - \frac{Tdepi}{ni} \quad \text{Equation 9}$$

Otherwise:

$$Tei = Temaxi - \frac{\frac{Temaxi * Smax - h}{Temaxi * ni}}{ni} * (Smax - h) \quad \text{Equation 10}$$

In this way, it is guaranteed that all the tasks can be running by the necessary number of times in a hyper period. However, it was not guaranteed that each task is running by all the periods once. For example if a task which has a little high execution time compared to the other tasks we can maintain the CPU for two periods of another task. To allow this possibility, we use the EDF (Earliest Deadline First) scheduling algorithm. The task which has the smallest execution time will be running initially.

2) Allocation of the energy budgets :

The energy management is done in an interval of time equal to Q. In each quantum Q, there is a budget of energy "Beq" that should be not exceeded. The energy consumption depends on the applications in the course of execution. The budget of energy for a quantum is given by equation11:

$$Beq = \frac{ED}{NQ} \quad \text{Equation 11}$$

Where ED is the remaining energy in the battery and NQ is the number of quantum to reach desired Lt.

We divide the assigned budget with a quantum on all the applications. Since the tasks do not consume the same quantity of energy, we proposed to use a factor of assignment of energy for each task. This factor must be fixed by the designer of the system and reflect its priority for the various tasks execution.

The budget of energy for a task Ti:

$$Bei = Beq * \frac{facteur\ affecter * 100}{\sum facteurs} \quad \text{Equation 12}$$

Since each application will be running several times in a quantum, it is necessary to divide the budget allocated for each application by the number of times that it will be running by ni.

The Budget of energy for each application during one period will be given by equation 13.

$$Bei = \frac{Bei}{ni} \quad \text{Equation 13}$$

Thus, all the data are ready to choose the adequate configuration for our system (acceptable level of QoS provided by the user, the number of cycle CPU as well as the assigned budget of energy to each task).

3) Search of the solution:

The search of the best solution uses the configurations database. It contains all necessary informations such as the applicative and architectural parameters, the worst case execution time, the level of QoS as well as the quantity of power consumption by each configuration. The best solution for each task offers the best QoS while respecting the following constraints:

- Eci<=Bei
- Tei<Btei
- NQoS>NQoSmin

The algorithm proposed allows:

- to eliminate the states which have an unacceptable level of quality by the user (NQoS< NQoSmin)
- to propose values for the execution time of each application.
- To affect the budgets of energy
- To check if the states giving best quality (best effort) for all the tasks respect the user constraints. If yes, we can choose these states directly as a solution for the system, if not we seek in the configurations base a solution for each application whose execution time is closest to the time already calculated with the proviso of respecting the assigned budget of energy.
- To start again a new iteration in order to improve the quality of service of some tasks by gathering the differences between the allocated budgets and the values of the selected configurations.

Once the GM chooses a new state for the system, it will have to send its instructions to the adapters of architecture, OS and application in order to change the system requirements.

C. *Local manager*

Local manager "LM" can be considered as a "watch dog" which permits to detect any overtaking in the expiry time of the tasks. Since the embedded multi-media systems are often subject to hard real time constraints, if one of the tasks misses its deadline, the LM must check if this overtaking influences the system operation (i.e. if all the tasks are running normally throughout the hyper period of the system). If there is no influence, the LM does not intervene and the system continues to function with the same parameters. On the contrary case, initially, the LM will have to check if it has a configuration in the base which can solve the problem without a costly hardware system reconfiguration. If it finds an adequate configuration it sends its instructions to both application and OS adapters to change their parameters. Otherwise, the LM will request to the GM to reconfigure the totality of the system. We give more explanations through a simple example shown in figure 4. We consider a system which runs three consecutive tasks T1, T2, T3 of respective period: 160, 40, and 80. The hyper period of the system is thus 160. Consequently, task T1 is executed once, T2 twice and T3 four times during one hyper period. The starting configuration chosen by the GM has assigned for the three tasks the respective execution times: 50, 10, and 30. We note, according to the execution diagram of these three tasks using EDF scheduling (see figure 4), that:

- During the first hyper period all the tasks are carried out in their expiries.
- During the second hyper period we note that task T1 exceeded the assigned execution time, but without an influence on the execution time of the other tasks. Thus the local manager does not intervene and the system continuous to function with the current parameters.
- At the time of the third hyper period we note that the tasks T1 and T2 have overtaken (noted as Dep in figure 4) the assigned execution time. But in this case, task T1 could not be carried out during the hyper period of the system. At this time the LM must intervene to search a new configuration for the system either or by activating the GM without finishing the execution of task T1 in order not to cause a delay of the complete system.
- The system begins execution again with the new configuration.

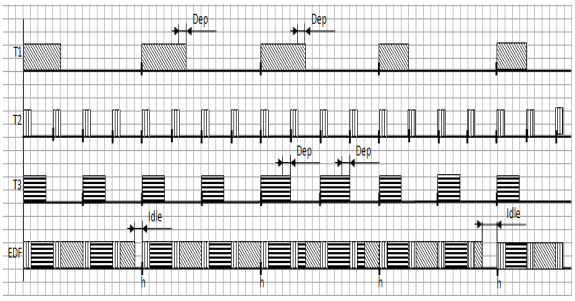

Figure 4. Local manager intervene

V. IMPLEMENTATION

We plan to validate the proposed adaptation model on a reconfigurable platform based on ALTERA FPGA (including NIOSII as a processor soft core).

A. *Configuration Database set up*

We use QUARTUS software as development environment for the HW design and NIOS IDE for software implementation. This environment includes also the real time operating system (RTOS) MicroC_OS-II. This development tool allows us to build a set of heterogeneous configurations around the NIOS processor: purely SW configurations using only the NIOS or other configurations using NIOS II with specific HW functions implemented as internal coprocessors or HW accelerators via the NIOS Avalon extension bus (see figure 5).

HW accelerators can be applied through two methods. The first one is by using specific communication lines called PIO (parallel input output). In this case the accelerator is slave. The second method consists in manually interfacing the accelerator directly on the Avalon bus. In this case the accelerator can be master or slave. The two methods make it possible to accelerate differently the treatment of a task. Master accelerator is quicker since it can reach the memory, read the data, make its treatment and write the result in the memory without direct intervention of the main processor. At the end of computations, the master sends an interruption to the processor.

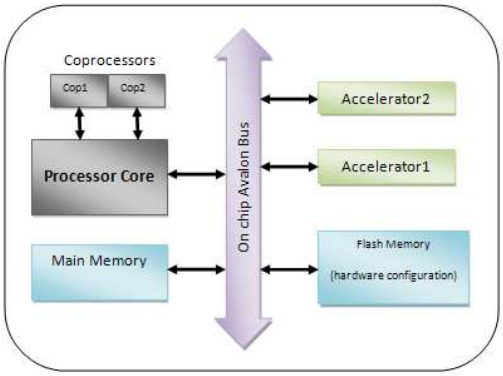

Figure 5. Hardware architecture

The choice of candidate HW task from all application is critical to have an efficient and optimized HW configuration. From a pure software application description, we detect most called tasks using profiling techniques. Further tests are made to select the most suitable tasks for a HW implementation [22].

Unfortunately, contrary to Xilinx, Altera FPGA does not allow dynamic reconfiguration. We cannot change the system hardware architecture during the system running by a dynamic reconfiguration involving a complete modification of the architecture. At present, the change of configuration is made simply by a switch between HW components or using a load of another SW code (in the case of application adaptation).

To set up the configurations base for our system, we make some off-line tests to determine some settings which are used to characterize each state.

Figure 6 and 7 show time execution model based on off-line measures for the 3D application using both flat and Gouraud shading method. NbPoly represent the polygon number of the 3D object. HW implementation correspond to a one HW accelerator

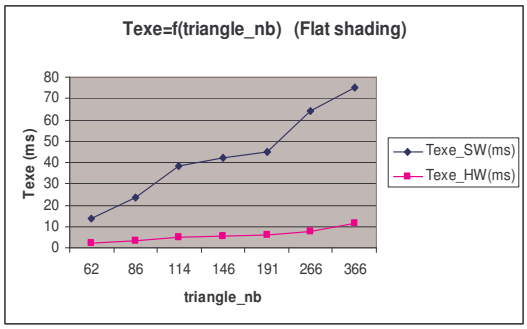

Figure 6. Texe/triangle_nb variation for flat shading

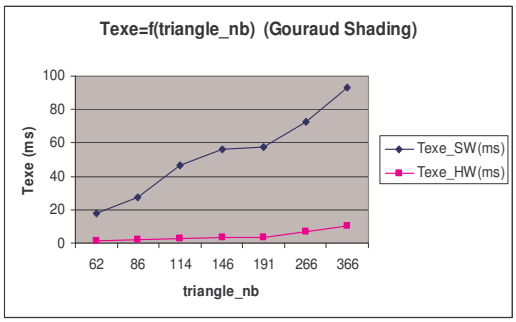

Figure 7. Texe/triangle_nb variation for Gouraud shading

Figure 8 shows the consumed power model based on real measure in the board for both Flat and Gouraud shading method in HW/SW implementation.

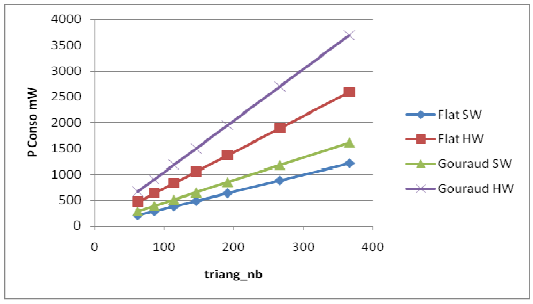

Figure 8. Power/triangle_nb variation for HW/SW Flat/Gouraud shading

For QoS computation, we use a model inspired from [21] to quantify the QoS of a 3D image. This model was modified to add Gouraud influence of the original model. Figure 8 shows this model.

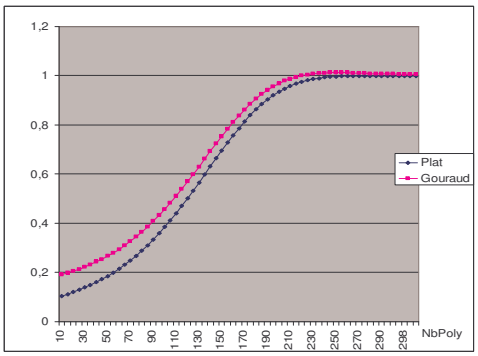

Figure 9. QoS model for the 3D image synthesis

Thus, we can build our configurations base. It contains, for each state, the type of implementation (HW or SW), the number of polygons, the type of the shade algorithm and the execution time.

B. Adaptation module implementation

We integrated the approach in a middleware layer which is a transition layer between the operating system and application layer and which uses components of these two layers.

The implementation of the software part was made by using the C language and the MicroC_OS-II routines. It uses a pre-emptive scheduling with fixed priority which is not adapted for our adaptation model which needs the EDF (earliest deadline first) scheduling. As this RTOS is open source we can easily do necessary modifications.

We created two tasks Local_Manager and Global_Manager which are blocked on standby event. We used a controller piloted by a hardware timer to supervise the execution of the various tasks throughout the hyper period "H" of the system. Whenever a deadline miss event occurs, the controller creates an event to activate the Local_Manager task.

The adaptation approach was tested using a configuration database. Tests are made by a modification of the system constraints (lifetime) and the user preferences.

VI. CONCLUSION

This paper presents a cross layer adaptation approach for the embedded multi-media reconfigurable systems. This approach makes it possible to improve the quality of service of a system while respecting its constraints (energy, real time, QoS) and the user preferences (level of minimum QoS, lifetime). The problem is to provide an approach which allows the co-ordination between the different system layers, hardware, OS and application, in order to maximize system QoS for desired lifetime. This approach present an acceptable overhead which not degrade the performances of the system. With an aim of addressing this problem, we proposed to add to the system a middleware layer which comprises primarily: (1) a global manager who coordinates between the three layers according to the system constraints and the user preferences by choosing the adequate configuration of the system starting from a configuration base; (2) a local manager who allows to guarantee the real time aspect of the system. This last intervenes only in the application and operating system layers.

The implementation of this approach was made through Altera EXCALIBUR development environment. We validated our approach through 3D synthesis images applications. Each time we change the system constraints and we observe the behaviour of our adaptation model. Our future work consists in (1) validating this approach through two applications. For this, we chose the 3D synthesis images application and MPEG II. (2) evaluating the performances and the overhead approach through these applications (3) and validating the same work on a dynamic reconfigurable platform via Xilinx.

REFERENCES

[1] Wanghong Yuana, Klara Nahrstedta, Sarita V. Advea, Douglas L. Jonesb, Robin H. Kravets "Design and Evaluation of a Cross-Layer Adaptation Framework for Mobile Multimedia Systems" Appears in SPIE/ACM Multimedia Computing and Networking Conference (MMCN), 2003

[2] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," in Proc. of 18th Symposium on Operating Systems Principles, Banff, Canada, Oct. 2001.

[3] R. Melhem, N. AbouGhazaleh, H. Aydin, and D. Mosse, "Power management points in power-aware real-time systems," in Power Aware Computing, R. Graybill and R. Melhem, eds., Plenum/Kluwer Publisher, 2002.

[4] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in Proc. of USENIX Symposium on Operating Systems Design and Implementation, 13-23, Nov. 1994.

[5] A. Bavier and L. Peterson, "The power of virtual time for multimedia scheduling," in Proc. of 10th International Workshop for Network and Operating System Support for Digital Audio and Video (NOSSDAV), June 2000.

[6] H. H. Chu and K. Nahrstedt, "CPU service classes for multimedia applications," in Proc. of IEEE Int. Conf. On Multimedia Computing and Systems (ICMCS'99), Florence, Italy, pp. 296–301, June 1999.

[7] S. Banachowski and S. Brandt, "The BEST scheduler for integrated processing of best-effort and soft real-time processes," in Proc. of SPIE Multimedia Computing and Networking Conference, San Jose, CA, Jan. 2002.

[8] M. Mesarina and Y. Turner, "Reduced energy decoding of MPEG streams," in Proc. of SPIE Multimedia Computing and Networking Conference, San Jose, CA, Jan. 2002.

[9] A. Vahdat, A. Lebeck, and C. Ellis, "Every joule is precious: A case for revisiting operating system design for energy efficiency," in Proc. of 9th ACM SIGOPS European Workshop, Kolding, Denmark, Sept. 2000.

[10] B. Li and K. Nahrstedt, "A control-based middleware framework for quality of service adaptations," IEEE J. Select. Areas Commun., 17(9) , pp. 1632–1650, Sept. 1999.

[11] J. Flinn, E. de Lara, M. Satyanarayanan, D. Wallach, and W. Zwaenepoel, "Reducing the energy usage of office applications," in Proc. of Middleware 2001, Heidelberg, Germany, Nov. 2001.

[12] Pham Ngoc, G. Lafruit, J-Y. Mignolet , G. Deconinck, and R. Lauwereins "QOS aware HW/SW partitioning on run-time reconfigurable multimedia platforms" Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms, ERSA'04, June 21-24, 2004, Las Vegas, Nevada, USA. CSREA Press 2004, ISBN 1-932415-42-4

[13] W. Van Raemdonck, G. Lafruit, E.F.M. Steffens, C.M. Otero Pérez, R.J. Bril "Scalable graphics processing in consumer terminals" Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference

[14] N. Pham Ngoc, W. van Raemdonck, G. Lafruit, G. Deconinck, and R. Lauwereins "A QoS framework for interactive 3D applications" Proceedings of the ninth international conference on 3D Web technology 2004

[15] N. Pham Ngoc, G. Lafrui, G. Deconinck, and R. Lauwereins "Terminal QOS management on run-time reconfigurable platforms" Third PA3CT-symposium 22-23 September 2003

[16] Wanghong Yuan, Klara Nahrstedt "A Middleware Framework Coordinating Processor/Power Resource Management for Multimedia Applications" Proc of Globecom 2001

[17] M. Corner, B. Noble, and K. Wasserman, "Fugue: time scales of adaptation in mobile video," in Proc. of SPIE Multimedia Computing and Networking Conference, San Jose, CA, Jan. 2001.

[18] J. Flinn and M. Satyanarayanan, "PowerScope: A tool for profiling the energy usage of mobile applications," in Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications, Feb. 1999.

[19] S. Brandt and G. J. Nutt, "Flexible soft real-time processing in middleware," Real-Time processing in middleware," Real-Time Systems 22(1-2), 2002.

[20] V. Bharghavan, K. Lee, S. Lu, S. Ha, J. Li, and D. Dwyer, "The TIMELY adaptive resource management architecture," IEEE Personal Communications Magazine, 5(4) , Aug. 1998.

[21] Y.Pan, I.Cheng, A Basu., "Quality Metric for Approximating Subjective Evaluation of 3-D Objects", IEEE transactions on multimedia, Vol. 7, N°. 2, avril 2005 p. 269.

[22] N.Ben Amor, Y. Le Moullec, J-Ph Diguet., J-L Philippe, M.Abid, « Design of a multimedia processor based on metrics computation», Special Issue for "Advances in Engineering Software", volume 36 (2005) p 448–458.