On the Scheduling of Real-Time Heterogeneous Multiprocessor Systems-On-a-Chip

Rodrigo Santos
Dep. Ing. Eléctrica y Computadoras, Universidad Nacional del Sur, CONICET
Avda. Alem 1253, Bahía Blanca, Argentina
ierms@criba.edu.ar

and

Jorge Santos
Dep. Ing. Eléctrica y Computadoras, Universidad Nacional del Sur.
Avda. Alem 1253, Bahía Blanca, Argentina.
iesantos@criba.edu.ar

and

Ariel Fernández
Dep. Electrónica, FRBB, Universidad Tecnológica Nacional.
11 de abril 447, Bahía Blanca, Argentina.
arifer@infovia.com.ar

Abstract

In this paper a method for the scheduling of real-time tasks on heterogeneous multiprocessor systems-on-a-chip for embedded applications is presented. It is based on the partition of tasks in subtasks related by precedence and executed in different processors. The processors are linked by a common bus and therefore no delays due to interprocessors network communications are present. An extensive experimental evaluation is presented and the method is compared to other solutions recently proposed.

Keywords: Real-Time Systems, Scheduling, Embedded Systems

Resumen

En este trabajo se presenta un método para la planificación de tareas de tiempo real en sistemas multitarea-multiprocesador integrados para aplicaciones embebidas. Se basa en la partición de subtareas relacionadas por precedencias que ejecutan en distintos procesadores. Los procesadores están comunicados por un canal común y por lo tanto no hay demorás en la transmisión de mensajes asociadas a la red. Se presenta una evaluación experimental comparativa con otras soluciones propuestas recientemente en la literatura.

Palabras Claves: Sistemas de Tiempo Real, Planificación, Sistemas embebidos

1 INTRODUCTION

The development of new chips, often called *system chips* because they contain full systems, has been made possible by increased integrated circuit yields [11]. As a consequence, the last decade has seen important advances in the development of specialized multiprocessor systems-on-a-chip used in embedded applications in cars, trains, space probes, alternative energy sources, networks, mobile computing, appliances, etc. The increased integration and speed of hardware and the low consumption of energy make these kind of devices very attractive. On top of that, they can be reconfigured very easily making the design process more efficient.

Field Programmable Gate Arrays (FPGA) can be used in the implementation of embedded systems since they allow the incorporation of many functions in quite a simple way. However, their use is limited by some inherent characteristics like the cost per unit, the speed of operation, the energy consumption, the interface with the external world, etc. Another problem with FPGA is that their reuse in other applications different from the original one is not as simple as changing the flash ROM of a microprocessor.

Traditional micro-controllers have improved their performance through the use of increased number of pipelines, increased number of stages within the pipeline, bigger and faster cache memories or simply by speeding up their frequencies. The first two possibilities are not very appropriate in certain cases of hard real-time systems because they introduce an important degree of uncertainty in the determination of the worst case execution times. The last one implies more energy consumption, a fact very often not acceptable.

Very Large Instruction Word (VLIW) processors, used for instance by Texas Instruments, are another option for embedded systems. The main idea in this architecture is to process several instructions in parallel. To do this, the processors have specialized ALUs, each one connected to a data bus and a program bus. Some algorithms, *e.g.* Fast Fourier Transform (FFT), are well suited for this kind of parallelism. However, since they are very specialized, more general applications require their combination with general purpose microprocessors. This leads to the integration on a chip of multiple microprocessors linked by a common bus with access to shared RAM, ROM and I/O peripherals. An example of this kind of chips is the SMJ320C80 from Texas instruments. It has a 32-bit RISC processor with four 32 bits Digital Signals Processors (DSP) in parallel [2]. This chip was developed to work as a dedicated DSP in video and sound applications, for example noise cancellation in sonar systems. Another example is the Janus system [9]. It has two processors and it was designed to work in the power train systems of internal combustion engines. Its development was guided by the need to provide an efficient control of the fuel injection to maximize torque, minimize pollution from combustions emissions and reduce fuel consumption.

The challenge to the real-time application designer is how to exploit the increased capabilities of those systems. In this search, the scheduling of sets of real-time tasks plays a central role. The system can be analyzed in a traditional way with tasks assigned to processors following some heuristic restricted by the system constraints [13]. This assignation problem is NP-complete an has no solution in polynomial time. However, in embedded systems, the tasks are usually constrained to execute in the processors in a certain order. In fact, they can be seen as composed of several subtasks related by precedence. These subtasks, can be scheduled in the processors according to different scheduling policies like Earliest Deadline First, Rate Monotonic, Deadline Monotonic and even Round Robin. In general, the execution of subtasks in DSPs is non-preemptable so the DSP looks like a critical section in the sense used in [14]. In [3, 4] Rajkumar introduced the Multiple Priority Ceiling Protocol (MPCP) and the Distributed Priority Ceiling Protocol (DPCP) for the scheduling of tasks with shared resources in generic multiprocessors systems. Because of its generality that does not take into account the particularities of the considered architecture, the method is sometimes quite pessimistic.

In [5] a co-scheduling approach is introduced. Many tasks share multiple resources. The basic idea consists in dividing the tasks in chunks and to assign to them partial deadlines in such a way that the whole system is schedulable. In the paper, the authors propose this mechanism for the scheduling of a processor and its disk controller.

In [1], the authors improve the analysis proposed by Rajkumar by assembling two queues. In one of them, the tasks that use the dedicated processor wait till they have the opportunity to execute. In the other one, tasks that do not use the dedicated processor are enqueued until they gain access to it. They use the Hyperbolic Bound to test the schedulability of the system [6].

In this paper, a method based on the partition of the tasks in subtasks related by precedence is presented. Successive subtasks are executed in different processors. The release times and deadlines are adjusted in such a way that the release of each subtask precedes its deadline and this, in turn, precedes the release of its successor. The Earliest Deadline First policy, EDF, is used for the schedulability of the main processor and the Deadline Monotonic policy, DM, for the auxiliary ones. The method is compartively evaluated against other scheduling methods.

The rest of the paper is organized in the following way. In Section 2 the model of the two processors system is presented. In Section 3 related work is discussed. In Section 4 the Precedences Method is introduced and formalized. In Section 5 the extension for more than two processors is presented. In Section 6 the method is compared to previous solutions and finally in Section 7 conclusions are drawn.

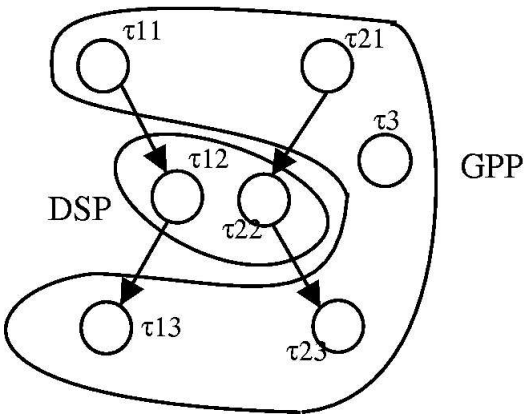

Figure 1: Architecture of the system

2 SYSTEM MODEL

In this paper, an architecture of heterogeneous multiprocessors on-a-chip is considered. The processors share the memories and the I/O devices and are linked by a common bus so that no network delays due to interprocessor communications are present.

Only two processors are considered first. One of them will be a General Purpose Processor (GPP) and the other one a dedicated processor, for instance a Digital Signal processor (DSP). Later this limitation will be relaxed and the general scheduling policy will be stated for more than two processors. Tasks running on the DSP are non-preemptable. A Remote Procedure Call (RPC) paradigm will be used to implement the communications among subtasks running on different processors. The RPC is issued by the GPP. Since its real-time kernel has to prevent a task from making an RPC if the DSP is busy, a waiting queue for the DSP is generated by the kernel to hold the tasks while they wait to execute in the DSP.

Tasks are periodic and independent. $S(m) = \{\tau_i\} = \{C_i, T_i, D_i\}$Denotes the set of tasks τ_i, $i = 1, 2, \ldots, m$.

Each task τ_i can be partitioned in subtasks τ_{ij}. $S^*(m) = \{\tau_{ij}\}$ denotes the set of subtasks τ_{ij}, $i = 1, 2, \ldots, m$, $j = 1, 2, 3$, in the case of a two processors system. The subtasks are ordered by precedence relation, $\tau_{i1} \succ \tau_{i2} \succ \tau_{i3}$. C_{ij} denotes the worst case execution time of the subtask j of task i. If $C_{i2} = 0$, the task is said to be GPP-only, which means that it doesn´t make any RPC to the DSP and can be executed without being blocked by it. If $C_{i2} > 0$, the task is said to be a DSP task.

The utilization factors for the main processor and the auxiliary one are computed in the following way:

$$U_{GPP} = \sum_{i=1}^{m} \frac{C_{i1} + C_{i3}}{T_i} \qquad U_{DSP} = \sum_{i=1}^{m} \frac{C_{i2}}{T_i} \tag{1}$$

In Figure 1 the architecture of a two processors system with three tasks is depicted. Nodes in the graph represent the subtasks, notated τ_{ij}, and the directed arcs represent precedence relations. Two of the tasks are DSP and one is GPP-only

3 PREVIOUS WORK

The problem of scheduling multiprocessors on different chips has been studied in many papers. When tasks constrained by precedence relations are executed in different processors, the communication delays and the schedulability of the interprocessor network, a real-time subsystem itself, have to be taken into consideration. In the case of processors integrated on a chip, this problem does not exist because the different processors sharing the RAM have immediate access to data produced by other processors.

Since the subtask running in the DSP is non preemptable, the processor can be considered as a critical section in the sense use in [14]. The system can be analyzed following the proposal of Rajkumar *et al* with the DPCP in [3, 4]. In this case, the schedulability test is:

$$\forall i = 1, \ldots, m \sum_{j=1}^{i-1} \frac{C_{j1} + C_{j3} + C_{j2}}{T_j} + \frac{C_{i1} + C_{i3} + C_{i2} + B_i}{T_i} \leq i(2^{1/i} - 1) \tag{2}$$

where B_i is computed as:

$$B_i \begin{cases} \max\limits_{p_j<p_i}\{C_{j2}\} + \sum\limits_{p_j>p_i}\left\lceil \frac{T_i}{T_j} \right\rceil C_{j2} & for\ a\ DSP\ task \\ 0 & for\ a\ GPP-only\ task \end{cases} \quad (3)$$

This approximation is not convenient because the execution time in the DSP affects every task, even those that do not use the DSP. The bound is therefore quite pessimistic and many feasible systems are deemed to be non-schedulable by this method.

In [1], the proposal of Rajkumar is improved by assembling two scheduling queues, one for the DSP tasks and the other one for the GPP-only ones. The requests to use the DSP are implemented by means of a blocking primitive that suspends the task in a waiting queue. The waiting task with higher priority is activated by the DSP once it finishes the execution of the current task. GPP-only tasks run independently of the DSP whenever they are ready and have priority enough to access the GPP. In this way, only the tasks that use the auxiliary processor can be blocked by lower priority tasks that use it too.

The blocking time of a task can be computed considering the longest execution time of the lower priority tasks plus the execution of all the higher priority ones that may run in one period of the task under consideration. So B_i is computed as:

$$B_i \begin{cases} C_{i2} + \max\limits_{p_j<p_i}\{C_{j2}\} + \sum\limits_{p_j>p_i}\left\lceil \frac{T_i}{T_j} \right\rceil C_{j2} & for\ a\ DSP\ task \\ 0 & for\ a\ GPP-only\ task \end{cases} \quad (4)$$

Once the blocking time for each task is calculated the schedulability can be tested with the traditional bound of Liu and Layland [7], by means of the Hyperbolic Bound (HB) [6] or by means of the Response Time Analysis (RTA) [10].

This approximation improves the possibilities of differentiating feasible from unfeasible systems because the blocking on the DSP is considered only for tasks that use it. It still has the limitation that, although it is a two- processors system, the test is done as if only one processor were available.

In [12] two scheduling methods for the synchronization of processes in distributed systems are presented. The time that a process is waiting for a response from another process is called *external blocking*. A scheduling algorithm based on the RTA divides the tasks in two independent parts, before and after the external blocking. While the first part has a release time equal to 0, the second one has a release jitter which is equal to the external blocking. Although the proposition provides an interesting way to analyse the feasibility of the system, the schedulability of the second processor is not discussed.

4 THE PRECEDENCES METHOD

The Precedences Method (PM) introduced in this paper to analyze the feasibility of the system divides each task in subtasks ordered by precedence relations. A successor subtask cannot be executed until data produced by its predecessor is available. In what follows it will be assumed that this takes place when the predecessor finishes its execution. The release times and deadlines of each subtask are selected in such a way that the precedence related subtasks can be analyzed as independent ones. In order to do that, for each τ_{ij} a release time r_{ij} and a deadline d_{ij} must be defined in such a way that:

$$r_i = r_{ij} < d_{ij} \leq r_{i(j+1)} < d_{i(j+1)} \leq r_{i(j+2)} < d_{i(j+2)},\ d_{i(j+2)} = D_i \leq T_i \quad (5)$$

It is said then that the system is coherent. The first and the third subtasks execute in the GPP while the second one executes in the DSP. Thus, the schedulability test has to be done for each processor. The EDF policy is used in the GPP and the DM policy in the DSP. As in [1], a DSP task that cannot access the DSP is blocked and waits in a special queue until it has priority enough (shortest relative deadline) to gain access. When the task that is actually running on the DSP finishes its execution, it awakes the highest priority task present in the waiting queue. Since the scheduling in this auxiliary processor is non-preemptable, it is necessary to bound the amount of time that a task may remain in the waiting queue.

The scheduling of the DSP can be established by means of the following:

Theorem 1: A set of periodic, independent, non-preemptable tasks $S(m)$ is DM schedulable if and only if:

$$\forall i, \qquad D_i \geq least\ t | t = C_i + \max_{h=i+1,\ldots,n}\{C_h\} + \sum_{j=1}^{i-1} C_j \left\lceil \frac{t}{T_j} \right\rceil \quad (6)$$

Proof: Each task meets its time constraint if and only if before its deadline it finds time enough to execute itself (first term), to withstand the blocking of tasks of lesser priority (second term) and to give way to task of higher priority (third term).□

To apply Theorem 1 to the determination of the feasibility of the auxiliary processor, it is necessary to establish an appropriate deadline for each subtask. Since in the main processor the tasks are executed following EDF, the latest moment in which a subtask running on the DSP can finish its execution and still leave enough time for the last subtask of the task to end before its deadline is given by:

$$d_{i2} = D_i - C_{i3} \tag{7}$$

Once the deadlines are established for the subtasks that run on the DSP, they are ordered by increasing values and the schedulability test following Theorem 1 is made. If the DSP is not schedulable, the system is not schedulable. If the DSP is schedulable, it is necessary to test the feasibility of the GPP.

In the schedulability analysis of the GPP, the precedence relations and the different releases and deadlines of the subtasks must be taken into account. In [8] a sufficient condition for the schedulability by EDF in systems where the tasks releases and deadlines are not synchronous with their periods is given: $S(m)$ is schedulable by EDF if $\forall h = 1, \ldots, m$, $\forall g = 1, \ldots, m$, such that $r_h \leq r_g$, $d_h \leq d_g$,

$$\sum_{r_k \leq r_g, d_k \leq d_h} C_k \leq d_h - r_g \tag{8}$$

Based on this result it is possible to study the feasibility of the GPP. It is necessary to determine for each one of the subtasks running on it, the release times, $r_{i,1}$ and $r_{i,3}$, and the deadlines, d_{i1} and d_{i3}. Establishing the deadline of the last subtask is simple because it has to be equal to the deadline of the task.

For the first subtask, the deadline is computed from the scheduling condition of the second subtask. In fact the deadline for the first subtask is the latest instant at which the second part has to be ready for execution in order to meet its deadline.

$$\forall i = 1, 2, ..., m \;\; d_{i1} = d_{i2} - least \quad t | t = C_{i2} + \max_{h=i+1,\ldots,m} \{C_{h2}\} + \sum_{j=1}^{i-1} C_{j2} \left\lceil \frac{t}{T_j} \right\rceil \tag{9}$$

For the release times of each subtask the earlier instant at which it may be ready for execution is chosen. In the case of the first subtask, it is the release time of the task. r_{i2} and r_{i3} are computed in the following way:

$$\forall i = 1, \ldots, m \;\; \forall j = 2, 3 \qquad r_{ij} = r_{i(j-1)} + C_{i(j-1)} \tag{10}$$

Once the release times and deadlines are established, the schedulability conditions can be tested. if all the subsets are schedulable the set $S(m)$ is schedulable.

The complexity of the algorithm is $O(mT_m)$ for the auxiliary processor and $O(p^2)$for the main one, where p is the number of tasks after the transformation.

5 EXPERIMENTAL EVALUATION

In this section the results of extensive simulations made to compare the Precedences Method introduced in the previous section with approaches proposed by other authors are presented. In the first set of simulations, the same specifications established in [1], although limiting the number of tasks to 10, were used, namely :

- Tasks´ periods were chosen randomly between 10 and 1000.
- Execution times for the tasks were selected in such a way that the total utilization factor of the system (GPP+DSP) fall in the interval (0.01, 0.99].
- DSP tasks represented, on the average, 80% of the total.
- The execution time of the task in the DSP was selected to be between 10% and 80% of the total execution time of the task.

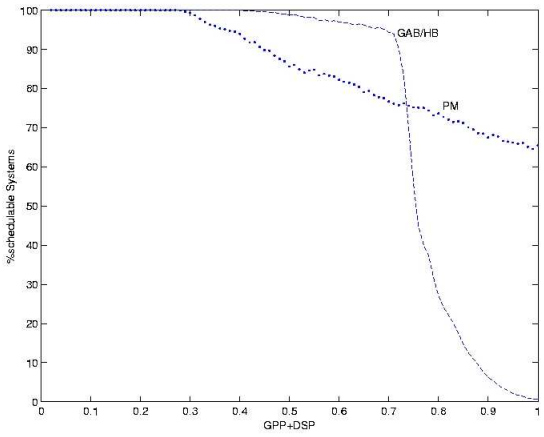

Figure 2: Percentage difference of schedulable sets by each methods

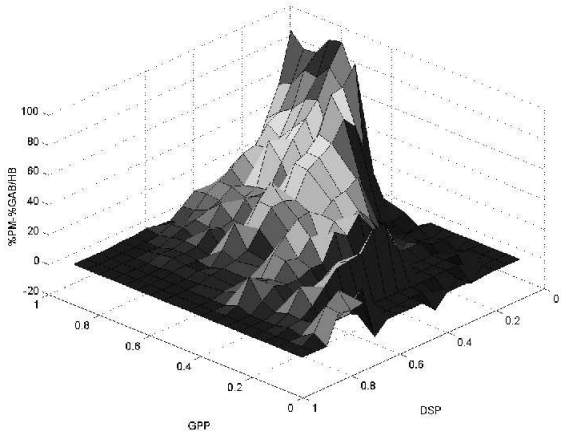

Figure 3: Difference between percentages of systems schedulable by PM and GAB/HB.

With this conditions 10^6 sets of tasks were generated and their schedulability tested by the method proposed in [1] by Gai, Abeni and Buttazzo using the Hyperbolic Bound (GAB/HB), and by the Precedences Method (PM). The percentages of the total number of sets that each method detect as schedulable were determined and their difference plotted *vs* the total utilization factor in Figure 2.

As can be seen up to an utilization factor of 0.3, the two methods produce similar results. In the interval [0.3, 0.75] GAB/HB works better but from 0.75 on, PM outperforms it.

However, the previous results were obtained under the rather restricting condition that the total utilization factor is, at most, 1, disregarding the fact that two processors on the chip allow higher total utilization factors. Thus, a second set of simulations were prepared varying both GPP and DSP utilization factors between 0.01 and 0.99. For each pair (U_{GPP}, U_{DSP}), one hundred systems were randomly generated with periods selected between 10 and 1000. In Figure 3, the difference between the percentage of systems detected as schedulable by each method was plotted for each pair.

As can be seen, as the GPP´s utilization factor approaches 1 and the utilization factor in the DSP is close to 0, the Precedences Method detects a higher number of schedulable systems. This is simply explained by the fact that GAB/HB works with a fixed priorities policy which is not good at high utilization factors, whereas PM uses EDF. As the DSP´s utilization factor is incremented, the difference is reduced to approximately 25% when the DSP is around 0.3. As both utilization factors decrease towards 0, the difference between the two methods is not significant.

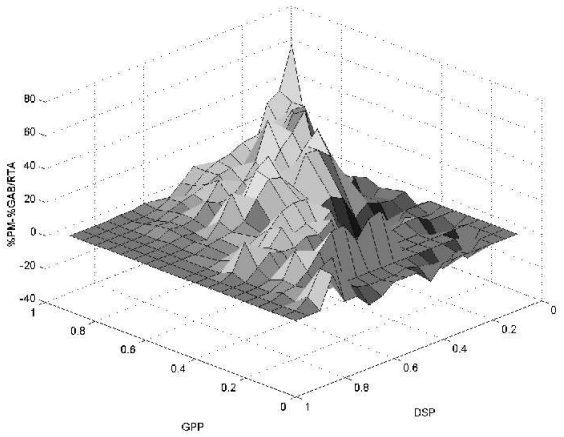

Figure 4: Difference between percentages of systems scheduled with PM and GAB/RTA

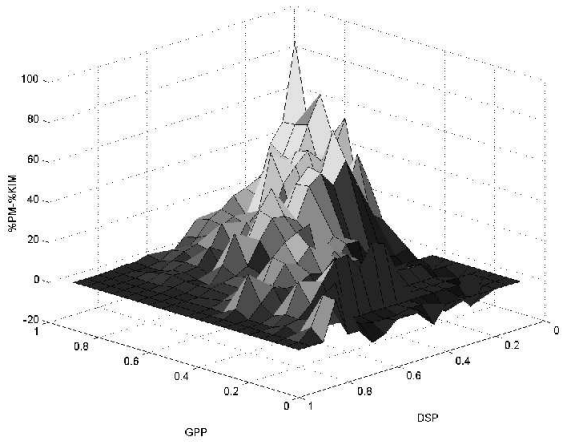

Figure 5: Difference between percentages of systems scheduled with PM and Kim´s method.

In Figure 4 the comparison is made between the PM and GAB using RTA instead of HB. Since it gives a necessary and sufficient condition for the schedulability of the system, it is natural that the difference between the two approaches is lower than before. However, in high CPU utilization factors there is still a clear advantage in favour of the Precedences approach.

Figure 5 shows the difference between the percentage of systems schedulable by PM and by the first method proposed by Kim. In this case, the release jitter was considered to be equal to the B term computed as in [1]. As can be seen, the result is quite similar to the one obtained with GAB/ RTA. As previously, there is a clear cut in favour of PM whenever the utilization factor is high.

6 GENERALIZATION TO MULTIPLE PROCESSORS ON A CHIP

In this section the generalization of the previous results is described. The method proposed up to this point is capable of analyzing the schedulability of two processors on a chip. However, it is possible that more than two processors are integrated on a chip (*e.g.* the Texas Instruments SMJ320C80) and that the tasks execute in them on a predefined order. In such cases it is quite straightforward to generalize the method. The algorithm for doing so is presented in Algorithm 1.

Algorithm 1 Schedulability Analysis for more than 2 processors

1. Divide each task in subtasks and allocate them to the proper processor.
2. Deadlines to the different subtasks are computed as follows:
   (a) The last subtask has the deadline of the task.
   (b) If the subtask´s successor is allocated to a non-preemptable processor then:
   $$\forall i,j \qquad d_{ij} = d_{ij+1} - least \quad t|t = C_{ij+1} + \max_{h=i+1,\ldots,n}\{C_{h2}\} + \sum_{g=1}^{i-1} C_g \left\lceil \frac{t}{T_g} \right\rceil$$
   (c) If the subtask´s successor is allocated to a preemptable processor then:
   $\forall i,j \qquad d_{ij} = d_{ij+1} - C_{ij+1}$
   (d) In the case that a subtask has more than one successor, deadlines are computed following cases (b) and (c) and the minimum is chosen.
3. Release times of the different tasks are computed as follows:
   (a) The first subtask has release time equal to the task release time.
   (b) If the subtask has a predecessor:
   $\forall i,j \qquad r_{ij} = r_{ij-1} + C_{ij-1}$
   (c) If the subtask has more than one predecessor, the greatest release time is chosen from the possibilities computed following (b).
4. To test the schedulability of the system, each processor has to pass the scheduling test. In the case of a non-preemptable processor, Theorem 1 should be applied. In the case of a preemptable processor condition 8 should be used.

7 CONCLUSIONS

The new trend in embedded systems with multiprocessors on a chip requires the use of new scheduling paradigms. The Precedences Method presented in this paper allows a simple implementation of high utilization factor in the different processors.

Earliest Deadline First and Non Preemptable Deadline Monotonic scheduling policies for the different processors are simple to implement. The computation of the release times and deadlines for each subtask is quite straightforward and the generalization to more than two processors is quite natural.

PM was simulated and tested against other methods proposed in the literature for the case of two processors. The simulations show a clear advantage of PM when the GPP utilization factor is high. Although the Kim´s and GAB/RTA methods work quite well in all the range of utilization factors, the paradigm proposed here outperforms them for utilization factors over 0.7.

References

[1] Gai, P., L. Abeni, G. Butazzo. Multiprocessor DSP Scheduling in System-on-a-chip Architectures. *Proc. 14th Euromicro Conference on Real Time Systems*, pp. 231-238, Viena, 2002.

[2] Texas Instruments. Military semiconductor products fact sheet SM320C80/SMJ20C80/5962-9679101 SGYV006C. Agosto 2000.

[3] Rajkumar, R., L. Sha, J. P. Lehoczky. Real time synchronization protocols for multiprocessors. *Proc. IEEE Real Time Systems Symposium*, 1988.

[4] Rajkumar, R. Synchronization in Real Time Systems: A priority inheritance approach. *Kluwer Academic Publishers*, 1991.

[5] Saewong, S., R. Rajkumar. Cooperative scheduling of multiple resources. *Proc. IEEE Real-Time Systems Symposium*, Diciembre 1999.

[6] Bini, E., G. Buttazzo, G. Buttazzo. A hyperbolic bound for the rate monotonic algorithm. *Proc. 13th Euromicro Conference on Real-Time Systems*, junio, 2001.

[7] C. L. Liu & J. W. Layland, Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal ACM*, 20 (1), 1973, 46-61.

[8] Chetto, H., M. Silly-Chetto, T. Bouchentouf. Dynamic scheduling of real-time tasks under precedence constraints. *The Journal of Real-Time Systems*, 2, 181-194, 1990.

[9] Ferrari, A., S. Garue, M. Peri, S. Pezzini, L. Valsecchi, F. Andreatta, W. Nesci. The design and implementation of a dual-core platform for power-train systems. *Convergence 2000*, Detroit, Octubre 2000.

[10] Wellings, A., M. Richardson, A. Burns, N. Audsley, K. Tindell. Applying new scheduling theory to static priority preemptive scheduling. Report RTRS 1921129, Dep. of Computer Science, University of York.

[11] Gupta, R. Driving research in system-chip design technology. *Computer*, 36, 7, 95-97, July 2003.

[12] Kim, I-G., K-H. Choi, S-K. Park, D-Y. Kim, M-P. Hong. Real-time scheduling of tasks that contain the external blocking intervals", 2nd International Workshop on Real-Time Computing Systems and Applications, pp. 54-59, October 25 - 27, 1995 Tokyo, Japan.

[13] Santos J., E. Ferro, J. Orozco, R. Cayssials. A heuristic approach to the multitask-multiprocessor assignment problem using the empty-slots method and rate monotonic scheduling. *Real Time Systems*, 13, pp. 177-199, 1997.

[14] Sha L., R. Rajkumar, J. P. Lehoczky. Priority inheritance protocols: an approach to real-time synchronization. *Transactions on Computers*, 39, 9, pp. 1175-1185, 1990.