Periodic real-time scheduling: from latency-based model to deadline-based model

Liliana Cucu[*], *Yves Sorel*

INRIA Rocquencourt,
BP 105 - 78153 Le Chesnay Cedex, France
liliana.cucu@inria.fr, yves.sorel@inria.fr

Abstract

We evoke our non-preemptive model of real-time systems with strict periodicity constraints. LC When deadlines are defined in the scheduling problem of these systems, we give an algorithm deciding which is the most important operation to be scheduled, LC according to the deadlines of their successors. We show that the scheduling algorithm given for systems with precedence and latency constraints solves the latter problem using the previous algorithm. We also evoke the classical non-preemptive deadline-based model of real-time systems with release times, periodicity constraints and deadlines. We give an algorithm which transforms an instance of the classical scheduling problem of these systems in an instance of the scheduling problem using our model with precedence and latency constraints. We show that the scheduling algorithm given for systems with precedence and latency constraints solves the latter problem using the previous algorithm.
Keywords: scheduling, real-time, latency, deadline, periodicity.

1 Introduction and existing results

Real-time systems are first of all reactive systems, that is, to each event arriving in the system through sensors corresponds an output event produced through actuators to the environment in reaction to this input event. This behavior is, usually, described using periodic systems of operations. In the classical deadline-based model given in [2], an operation becomes available at its release time which is repeated periodically and its start time is greater or equal to its release time. The operation must be scheduled before its deadline which is relative to the release time. The formal definition of this model is given in section 2.

When the real-time applications are based on signal, image and control processing algorithms [3], strict periodicity constraints are defined. In this case, an operation is always available and the notion of release time has no meaning. The start times are periodic and there is no deadline.

In our model given in [15] beside the strict periodicity constraints, two other types of constraints may be imposed on these systems. Firstly, because some operations can produce data consumed by other operation(s) as it is the case when the application fonctionalities are specified with a tool like Simulink, precedence constraints are imposed onto these operations. Secondly, for example in the case of brake control in a car, when the driver presses the brake pedal, the wheels must stop

[*]Corresponding author

before a fixed delay, corresponding to the execution of control laws relating "pedal-press" event and "wheels-stop" event. We call this constraint a *latency* constraint. The word "latency" was already employed by the real-time community. For example, Hagmann [4] calls latency the necessary time to obtain data when reading a hard disk or Aggarwal [5] the necessary time of communicating data from memory to a processor. More theoretical definitions were given by Goddard [6] in the case of data-flow graphs or by Van Beek [7] in the case of repeated precedence constraints. These definitions are limited to particular cases and can not be employed to describe the elapsed time between any two operations related by precedence constraints. Obviously if a latency constraint should describe this time then this constraint is "end-to-end" constraint and the interest in these constraints was very well justified by Gerber [8]. Also, the examples given by Gerber are treating only "end-to-end" constraints between sensors and actuators.

Our paper generalizes the results concerning the end-to-end constraints [8] [9] [10], allowing to have several latency constraints which are not imposed only between sensors and actuators, but also between the operations which are necessary to obtain output events from input events. For example, it is possible to specify the brake system of a car (Figure 1) by a graph where $PRESS$ represents a sensor giving the information which indicates that the driver presses the brake pedal, $SPEED$ represents a sensor giving the wheel speed, $DEAC$ represents the deactivation of the speed regulator, $ESTIM$ represents the estimation of the pressure on the brake pedal, ABS represents the anti-locking brake system (ABS) and $BRAKE$ represents the actuator giving the pressure on the brakes wheels. For example, we may need to impose a delay between $PRESS$ and $BRAKE$, and another delay between $DEAC$ and $BRAKE$.

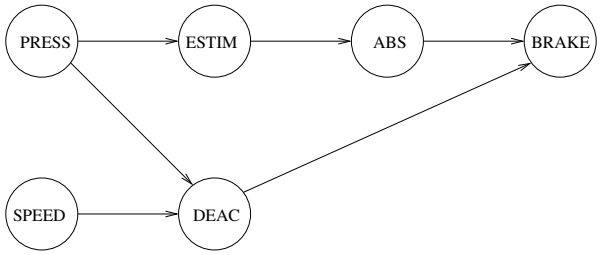

Figure 1: Example of brake control

The formal definition of our model is given in section 2.

The results on non-preemptive deadline-based of real-time systems with release times, periodicity constraints and deadlines are poor [11]. Giving the results we have in our non-preemptive problem, this paper presents a connection between the two models allowing to use our problem in order to propose new results for the classical problem.

This paper is composed of five sections. The following section introduces the two models. The section 3 gives an algorithm indicating between available operations which is the most important according to the deadlines of their successors. It is followed by the section 4 which proposes an algorithm transforming an instance of a scheduling problem using the classical model into an instance of a scheduling problem using our model. The paper ends with section 5 which concludes and gives hints for future research.

2 Two real-time models

Before presenting the two models, we give the definition of a schedule and of the schedulability of a system of operations, where by system of operations we understand a set of operations with computation times and different constraints.

Definition 1 *For a system of operations, a schedule S is a total order on the set of operations associating to each operation A a start time s_A which is the instant when the operation is executed. We denote by $\mathcal{S}$ the set of the possible schedules of a system of operations. An operation with the start time known is called scheduled.*

Definition 2 *A system of operations is schedulable if there is at least one schedule satisfying the constraints of the system.*

We start by presenting the classical model of operations given by Liu and Layland (see Figure 2) in the non-preemptive case. An operation A becomes available at its release time s_A. The release time is periodically repeated with a period T_A. So if the first release of an operation is r_0 then the i^{th} release time of the operation will be $r_A + (i-1)T_A$. Also the computation time C_A is known.

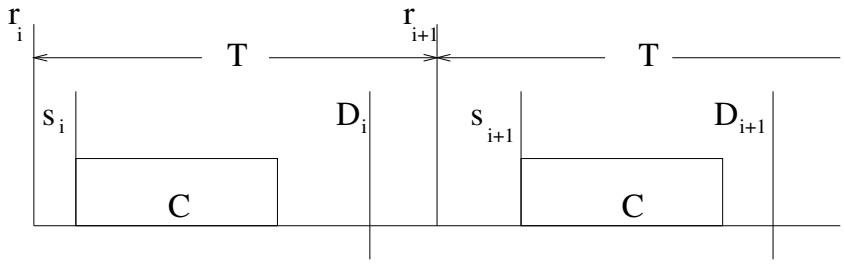

Figure 2: Model proposed by Liu and Layland

As said in the first section, in our model, the operations are always available and the start time s_A of an operation A is repeated periodically with a period T_A (see Figure 3). The computation time C_A is known.

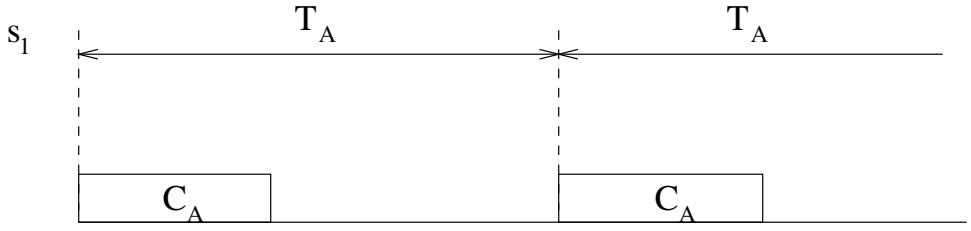

Figure 3: Model of strictly periodic operations without release time

In the classical model the instant when an operation becomes available is always known (release time), but not the instant when it is scheduled. Knowing the start time of the first repetition of an operation does not imply knowing the start times of all the other repetitions of the operation.

In our model the instant when an operation becomes available and the instant when it is scheduled are unknown only for the first repetition of the operation. Once the first start time of an

operation is known, the start times of all the other repetitions are also known and the release times of these repetitions are useless.

The difference between the two models is depicted by comparing them for some operation A in Figure 4. The first schedule is given for the classical model and the second one is given for our model.

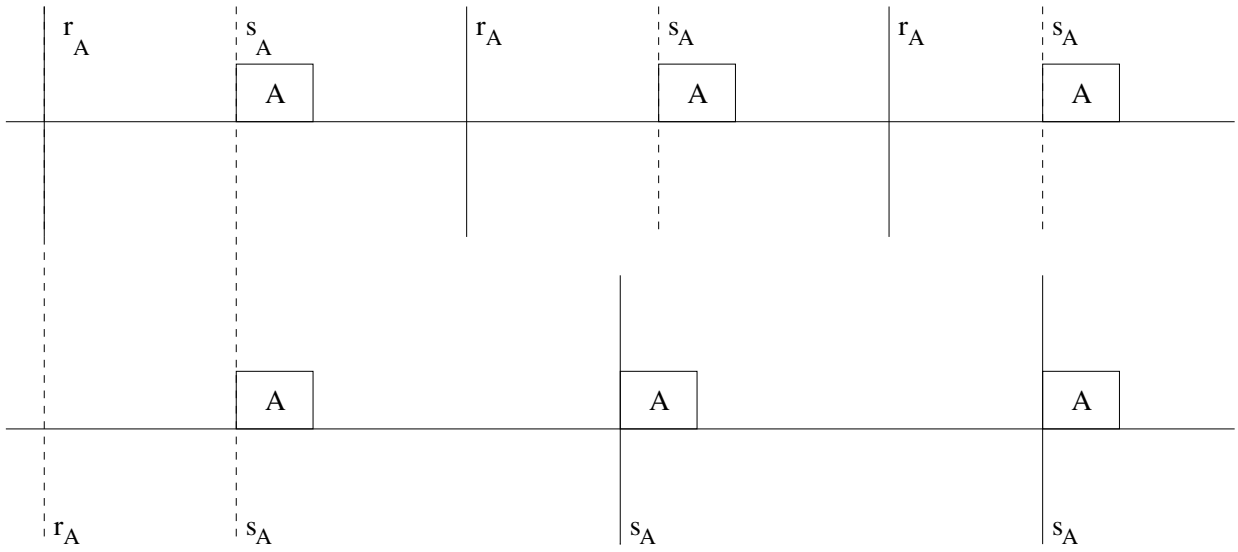

Figure 4: Difference between the classical model (above) and our model

Our model allows to impose two other constraints: the precedence constraints and the latency constraints. We use a graph-based model to specify the precedences between operations through an algorithm graph. The graph is a directed graph $G = (V, E)$ where V is the set of operations and $E \subseteq V \times V$ the set of edges which represents the precedence constraints possibly due to data transfer between operations. Therefore, the directed pair of operations $(A, B) \in E$ means that B must be scheduled, only if A was already scheduled. When precedence constraints are defined, we say that an operation is available when all its predecessors are already scheduled [12].

The edges of the graph define a partial order on the set of operations. The operations which do not belong to an edge define a "potential parallelism" [13]. Moreover, because the graph describes a real-time system, which is reactive, this latter graph has a *pattern* infinitely repeated [14] which induces an infinite repetition of all operations (see Figure 5).

For the sake of simplicity and because we use only directed graphs, from now on we will use the term path instead of directed path.

We denote by $\mathcal{P}$ the set of all paths of graph $\mathcal{G}$ and we say that $P(A, B) \in \mathcal{P}$, if there is at least a path from A to B. If $\nexists P(A, B) \in \mathcal{P}$, then there is no path from A to B. Moreover, we denote by $\mathcal{D}(A) = \{B \in V \text{ such that } P(A, B) \in \mathcal{P}\}$.

Definition 3 *[15] On a pair of operations (A, B) with $\exists P(A, B) \in \mathcal{P}$ and A and B belonging to the same pattern, we say that a latency constraint $L(A, B)$ is imposed if the operations are scheduled such that $s_B + C_B - s_A \leq L_{AB}$, $L_{AB} \in \mathbb{N}^+$. Let $\mathcal{L}$ be the set of all latency constraints imposed for a system.*

Without any loss of generality we assume that all characteristics, i.e. the computation times, the periods and the latency constraints, are defined as multiples of a clock tick τ (time is discrete).

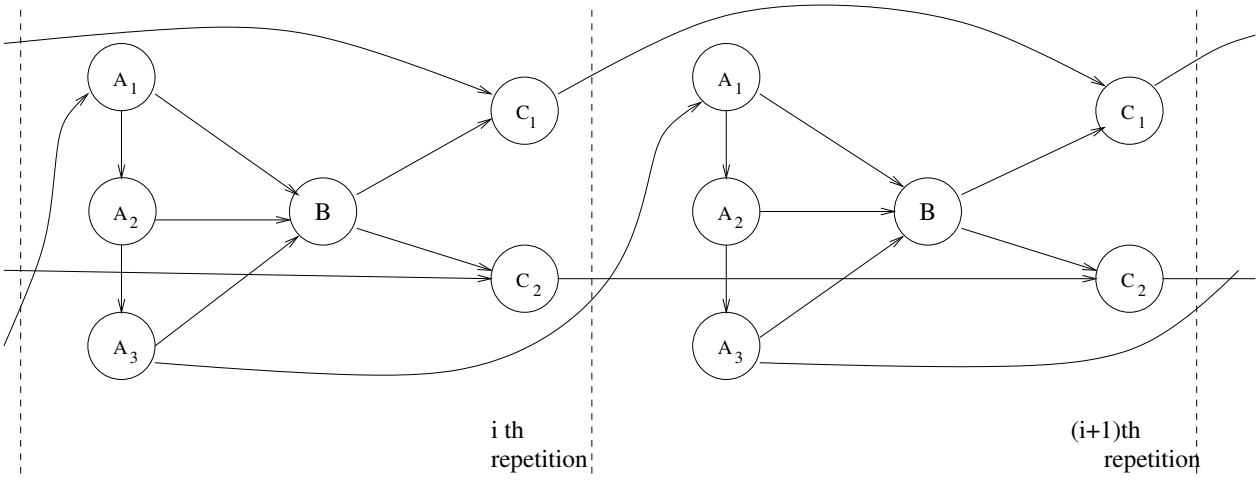

Figure 5: Graph describing a real-time system

Afterward the integer values of the given characteristics are implicitly multiplied by τ.

3 Deadline-marking algorithm

In this section, we present a theorem proving that a latency constraint is a deadline. Thus we may solve the non-preemptive problem of systems with precedence, strict periodicity and deadlines. This problem is a case of our problem with the latency constraints replaced by deadlines. The results of this section may be applied for both uniprocessor and multiprocessor cases.

The following theorem gives the relationship between the deadlines and the latency constraints.

Theorem 1 *A latency constraint $L(A,B)$ is a deadline for the operation B, once the operation A is scheduled.*

Proof We prove the theorem by double implication.

First, we demonstrate that a latency constraint $L(A,B)$ of a pair $(A,B) \in \mathcal{L}$ may be expressed as the deadline of operation B, once A is scheduled. Because $(A,B) \in \mathcal{L}$, then we have $s_B \leq L_{AB} + s_A - C_B$. If we denote by D_B the expression $L_{AB} + s_A - C_B$, we obtain

$$s_B \leq D_B \tag{1}$$

Once A is scheduled, s_A is known and D_B becomes known, also. Because s_A is calculated from the beginning of the schedule then the equation (1) defines a deadline for B.

Second, we express the deadline of an operation B as a latency constraint $L(\cdot, B)$. Because B has a deadline D_B, then $s_B \leq D_B$. We denote by A the first operation which was scheduled, then $s_A = 0$. Because the deadline is defined from the start of the schedule, we obtain that $s_B - s_A \leq D_B$. Moreover, because all the operations without predecessors may be the first scheduled operation, we have $s_B - s_C \leq D_B, \forall C \in V$ with $Prec(C) = \emptyset$ and $B \in \mathcal{D}(C)$, where $Prec(C)$ is the set of predecessors of operation C. So, in order to satisfy the deadline of B, we define several latency constraints $L_{CB} = D_B$ for each C such that $Prec(C) = \emptyset$ and $B \in \mathcal{D}(C)$. The theorem is proved $\square$

Due to this theorem, we may use the scheduling algorithm given in [15] for systems with precedence, strict periodicity and latency constraints to schedule systems with precedence constraints,

strict periodicity constraints and deadlines. This algorithm uses a latency-marking algorithm. The Theorem 1 allows to modify this latter marking algorithm in order to take into account the deadlines. In this case the function *mark* of the operation A must be equal to the $min_{B\in\mathcal{D}(A)}\{D_B\}$. These latter marks are obtained using the following algorithm, called deadline-marking algorithm:

Algorithm 1

Initialization: $\mathcal{W} = \bigcup_{A\in V \text{ and } Suc(A)=\emptyset} Prec(A)$ is the working-set. If A has a deadline D_A, then $mark(A) = D_A$, otherwise $mark(A) = \infty$.

Step 1: For $A \in \mathcal{W}, mark(A) = min(mark(A), min_{B\in Suc(A)}\{mark(B)\})$ and $\mathcal{W} = \mathcal{W}\backslash\{A\}$. We add to $\mathcal{W}$ the operations for which all the predecessors has been removed from $\mathcal{W}$.

Step 2: Repeat Step 1 until $\mathcal{W} = \emptyset$.

Remark 1 *Recursively, each operation inherits the deadline of its successors. At the end of this latter algorithm, the mark of an operation may be equal to either its inherited deadline or its initial deadline. Therefore, the algorithm does not modify the marks of the operations without successors. This algorithm is, also, applied to the pattern.*

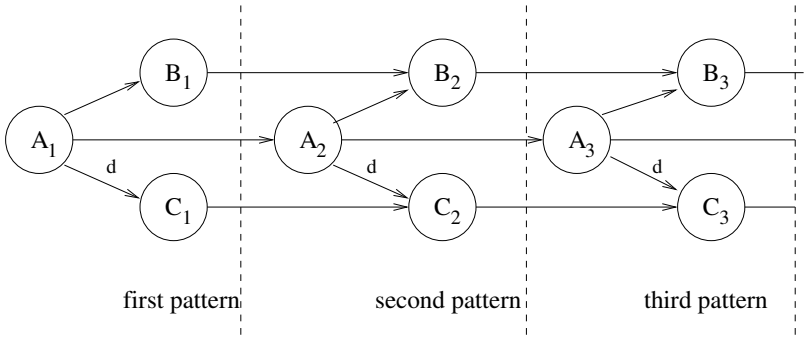

Figure 6: Graph of example 1

Example 1 *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph given in Figure 6 with the deadline D_{C_1}. In Figure 6, the letter d on the edge from A_1 to C_1 implies that the precedence constraint is due to data transfer. All the operations will have the mark D_{C_1} except C_2. The algorithm is given in table 1, where by $m(A)$ we understand $mark(A)$.*

	$m(A_1)$	$m(A_2)$	$m(A_3)$	$m(B)$	$m(C_1)$	$m(C_2)$
Initialization	∞	∞	∞	∞	D_{C_1}	∞
D_{C_1}	D_{C_1}	D_{C_1}	D_{C_1}	D_{C_1}	D_{C_1}	D_{C_1}

Table 1: Marks obtained using the marking algorithm

4 Transformation from problem with deadlines to problem with latency constraints

In this section, we give an algorithm transforming an instance of a scheduling problem using the classical model into an instance of a scheduling problem using our model. We prove that the two problems are equivalent from the complexity point of view. Thus, the classical problem may be solved in the non-preemptive case using the results for our problem. The results of this section may be applied for both uniprocessor and multiprocessor cases.

We consider a system with n operations and for each operation A, we have the computation time c_A, the release time r_A and the deadline D_A. The operation A is periodic with the period T_A. We transform this system using the following algorithm:

Algorithm 2

Step 1: for each operation A we add a fictive operation A' with the computation time equal to 0.

Step 2: we add a precedence constraint between A' and A. We obtain the graph of precedence constraints with $2n$ operations and n edges.

Step 3: we impose latency constraints for each (A', A) with $L_{A'A} = D_A$.

Step 4: we impose strict periodicity constraints for each operation A' with T'_A equal to the period of operation A.

Example 2 *This example illustrates the Algorithm 2. For a system of three operations A, B and C with no precedence constraints, we obtain the graph given in Figure 7.*

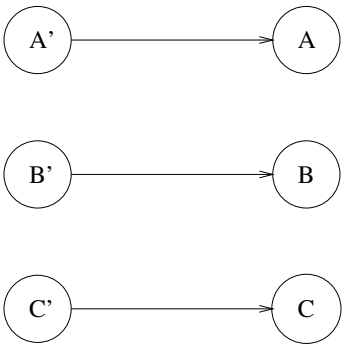

Figure 7: Graph obtained using the Algorithm 2

We denote by PRE the decision problem associated to the scheduling problem of periodic systems with deadlines. We denote by $PPLc$ the decision problem associated to the scheduling problem built from PRE using the Algorithm 2.

Theorem 2 *The decision problem PRE has, as solution, the answer "yes" if and only if the decision problem $PPLc$ has, as solution, the answer "yes".*

Proof We prove the theorem by double implication.

For the decision problem PRE each operation A has a release time r_A which gives the instant when the operation becomes available, a deadline D_A defined from the release time, a period T_A and a computation time C_A.

First, we suppose that the problem PRE has the answer "yes" and we prove the decision problem $PPLc$ has also the answer "yes". We denote by S one of the schedules which satisfies the constraints of the scheduling problem PRE. This implies that for each operation A we know the start time $s_A \in S$ such that $r_A \leq s_A \leq D_A$.

From the schedule S, we build a schedule S' for the problem $PPLc$. Each operation A' associated to an operation A of an instance of the problem PRE has the start time of its first repetition $s'_{A'_1}$ equal to r_A. Each operation A of an instance of the problem $PPLc$ has the start time equal to the start time of the operation A in the schedule S. The Figure 8 illustrates this construction. The operation A' with the computation time equal to 0 is marked with non-continuous line on the figure. Consequently, because the deadlines are satisfied in the schedule S then all the latency constraints are satisfied in the schedule S'. Because the operations A' has the computation times equal to 0 and their start time periodically repeated due to the periodicity of release times in the schedule S, then the periodicity constraints are satisfied in the schedule S.

Second, we build similarly a schedule for the problem PRE from a schedule of the problem $PPLc$. Consequently,there is at least a schedule for the first problem if and only if there is a schedule for the problem built using the Algorithm2. The theorem is proved ⊡

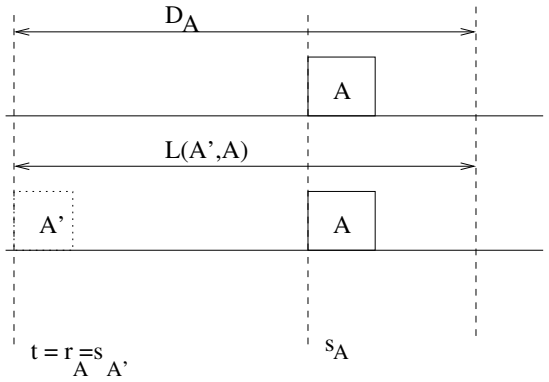

Figure 8: Construction of a schedule for $PPLc$ from a schedule for PRE

We expressed the scheduling problem with release times, periodicity constraints and deadlines using our model.

5 Conclusion and further research

We compare two real-time systems models in the non-preemptive case. We first give an algorithm which allows us to solve a particular case of our problem, the scheduling problem with precedence, strict periodicity constraints and deadlines. Moreover we give an algorithm which transforms an instance of the classical problem into an instance of our problem. This allows to particularize our scheduling algorithm to the classical problem.

We want to extend these results to the non-preemptive in the multiprocessor case, where the results for the classical problem are also poor.

References

[1] D Harel and A Pnueli. On the development of reactive systems. In K. R. Apt, editor, *Logics and Models of Concurrent Systems*. Springer Verlag, New York, 1985.

[2] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 1973.

[3] J.H.M. Korst, E.H.L. Aarts, and J.K. Lenstra. Scheduling periodic tasks. *INFORMS Journal on Computing 8*, 1996.

[4] R.B. Hagmann. Low latency logging. Technical report, Palo Alto Research Center, 1991.

[5] A. Aggarwal and A.K. Chandra. On communication latency in pram computations. Technical report, IBM Research Division, 1989.

[6] S. Goddard. *Constraints on data-flow*. PhD thesis, University of North Carolina, 2000.

[7] P. van Beek and K. Wilken. Fast optimal instruction scheduling for single-issue processors with arbitrary latencies. *Springer-Verlag*, 2001.

[8] R. Gerber, S. Hong, and M. Saksena. Guaranteeing real-time requirements with resource-based calibration of periodic processes. *IEEE Transactions on Software Engineering(21)7*, 1995.

[9] D.-I. Kang, Gerber R., and M. Saksena. Performance-based design of distributed real-time systems. *Proceedings of IEEE Real-Time Technology and Applications*, 1997.

[10] R. Gerber, W. Pugh, and M. Saksena. Parametric dispatching of hard real-time tasks. *IEEE Transactions on Computers(44)3*, 1995.

[11] K. Jeffay, D.F. Stanat, and C.U. Martel. On non-preemptive scheduling of periodic and sporadic tasks. *IEEE*, 1991.

[12] L. Cucu and Y. Sorel. Schedulability condition for systems with precedence and periodicity constraints without preemption. In *Proceedings of Real-time and Embedded Systems*, 2003.

[13] Y Sorel. Massively parallel computing systems with real time constraints, the "algorithm architecture adequation" methodology. In *Proceedings of Massively Parallel Computing Systems, Italy*, 1994.

[14] T. Grandpierre, C. Lavarenne, and Y. Sorel. Optimized rapid prototyping for real time embedded heterogeneous multiprocessors. *Codes'99 7th International Workshop on Hardware/Software Co-Design*, 1999.

[15] L. Cucu, R. Kocik, and Y. Sorel. Real-time scheduling for systems with precedence, periodicity and latency constraints. In *Proceedings of Real-time and Embedded Systems*, 2002.