Xoncrete: a scheduling tool for partitioned real-time systems

Vicent Brocal†, Miguel Masmano†, Ismael Ripoll†, Alfons Crespo†, Patricia Balbastre† and Jean-Jacques Metge‡

Instituto de Informática Industrial. Universidad Politécnica de Valencia.
Camino de Vera s/n, 46022 Valencia, Spain
† {vbrocal, mmasmano, iripoll, alfons, patricia}@ai2.upv.es
CNES Centre Spatial de Toulouse, France
‡ jean-jacques.metge@cnes.fr

Abstract: ARINC 653 defines a partitioned framework where the partitions are scheduled according to a predefined cyclic plan and the processes of each partition are scheduled with a fixed priority policy. The timing characteristics defined in ARINC (period and duration) can hardly be used to precisely represent the timing requirements of the applications. We extend the timing model of ARINC 653 to consider deadlines and the periodic behaviour of the individual processes. A novel definition of how to model periodic activities and how this new model is specially useful in an heterogeneous partitioned system is also presented.

The new model and the set of scheduling algorithms have been implemented in a scheduling tool (called Xoncrete) to assist the designer to generate the cyclic plan table. Although founded on solid theoretical results, Xoncrete is not a general purpose tool, it is a tool designed to provide real help to the system designer.

Keywords: Scheduling, System Modelling, Software engineering tools.

1 Introduction

The increment in the processing power of the new processors and the technical advances in platform virtualisation, enable to use virtualisation or partition solutions in the embedded real-time systems as a way to increase the level of security.

An important characteristic that shall be preserved in a partitioned system is the temporal and spatial isolation between partitions. That is, the applications that are executed in a partition shall not have interferences from other partitions. The amount of resources (devices, memory, and processor time) allocated to each partition shall always be the same, independently of the resources requested by other partitions.

In the case of hardware devices or memory space is it relatively easy to guarantee the proper isolation using memory protection (MMU) and IO protection (IOMMU) capabilities available in most processors. Temporal isolation (how the processor is allocated to each partition) deserves more attention.

The ARINC 653 standard defines a partitioned system where the global scheduler (partition scheduling) is a cyclic, and a local scheduler (process scheduler) is preemptive fixed priority. The table driven cyclic scheduler provides a strong temporal isolation among partition and the priority driven scheduler gives a POSIX like execution environment in each partition.

There is a vast amount of research and published papers addressing real-time scheduling problems. Although the general topic is the same: "real-time scheduling", the very large combination of system models (pure periodic, sporadic, with resources, etc.) and the exact problem considered (schedulability analysis, sensitivity analysis, resource sharing, hierarchical scheduling, etc.) makes it difficult to employ those research results in real problems. Moreover, the system models considered by researches and the ones used by system designers not always coincide.

Several tools have been developed to help system designers to perform scheduling analysis. There are two kind of tools:

- General purpose tools (MAST, CHEDDAR, RapidRMA [1,2,3]) that support several scheduling policies (among others, fixed priority and earliest deadline first).
  These tools are basically an implementation of the scheduling analysis techniques published in the literature. The main goal of these tools are to determine whether the given system is schedulable or not. In some cases, they also provide some hints on how to make the system schedulable if it is not.
- Specific scheduling tools (netcar-ecu, SYMTA/S ECU). These tools have been designed considering the exact task[1] and system characteristics of a specific problem (automotive execution model) and to compute the required result. The result is a plan table, where all the resources and tasks are pre-allocated.

[1] We will use either "task" or "process" to refer to an execution activity.
[2] In a partitioned framework, the role of the *integrator* is to allocate resources to each partition and build the complete system.

This paper presents a new tool, Xoncrete, to assist the system integrator[2] to analyse the schedulability of a partitioned system and to create the cyclic scheduling table. Rater than a general purpose scheduling tool, Xoncrete has primarily been designed to meet the ARINC 653 system model. But it also allows to analyse other scheduling policies for processes such as EDF.

Besides the scheduling analysis capabilities, the Xoncrete tool also provides a user friendly interface for capturing and editing all the elements that are part of a partitioned system. It has been specially designed to generate configuration files compatible with XtratuM 2.2. XtratuM [4] is a type 1 hypervisor for safety critical real-time systems. XtratuM is an open source development that is availble for LEON2 and is being ported to LEON3 with/without MMU in the framework of the ESA Project AO5829 "Securely Partitioning Spacecraft Computing Resources" (version 3).

2 Scheduling approaches

There are basically three approaches to schedule and analyse a partitioned system: "aperiodic servers", "Compositional scheduling" and "flat model".

2.1 Aperiodic servers

Each partition is managed as a server (CBS, TBS, IRIS [5], periodic server, etc.) by the global scheduler.

During the 90^{th} a lot of work was done to solve the problem of jointly scheduling tasks with both, hard and soft real-time requirements. The goal was to provide the fastest response time to the soft real-time tasks whilst not jeopardising the timing requirements (deadline) of the hard real-time tasks. An aperiodic server is commonly defined with the tuple (budget, period). The scheduling algorithm is then modified to allocate budget units of time every period to the server. The term "aperiodic" was used to denote that soft tasks (those executed by the server) should not be considered periodic, since no guarantee could be done about how much and when the processor would be assigned to them. Soft real-time tasks are executed by aperiodic servers. An aperiodic server is a process that receives requests to execute aperiodic jobs (a function, a thread or a process, depending on the implementation) and executes them until the budget is exhausted.

One way to implement an aperiodic server is to handle it as if it were a regular periodic task (periodic server). Several other aperiodic server algorithms (sporadic server and deferrable server for fixed priority; CBS and IRIS for dynamic priority) were proposed to improve the responsiveness by improving how and when the budget can be replenished and consumed.

The improvement of the aperiodic servers and a better understanding of the isolation properties of this mechanisms refocused the application of these servers to what was called bandwidth servers or resource reservation protocols, which allows real-time tasks to execute in a dynamic environment under a temporal protection mechanism, so that each server will never exceed a predefined bandwidth, independently of its actual workload requests.

Originally, a FIFO policy was used to schedule the aperiodic jobs executed by a server. **Using a real-time policy inside the aperiodic server results in a two level hierarchical system**: the global scheduler is the EDF or the RM modified to support the selected aperiodic server, and the local scheduler can be RM or EDF. The implementation of most servers except the periodic (and the background server), require the modification of the global scheduling policy to maintain the budget of

Pros & cons.

[+] A vast amount of theoretical results are available.
[−] Although they are mature enough [6].
[−] The global scheduler shall support aperiodic servers.
[−] Complex task models are difficult to analyse and schedule.

2.2 Compositional scheduling

The basic idea is to extend the classical and widely used divide and conquer strategy to the temporal requirements [7]. The complexity of each component is hidden and abstracted through a clean and well defined interface.

One of the goals of the compositional scheduling model is to **avoid performing a global schedulability analysis** that considers the timing requirements of all the tasks in all the task groups. Ideally, each task group can be analysed by itself for schedulability.

The compositional model addresses the problem of guaranteeing the correct temporal operation of the composed system. The following two problems need to be addressed:

Abstraction problem: analyse the timing properties of a component independently. This problem consists on abstracting the set of real-time requirements of a component as a single real-time requirement, called *scheduling interface*. Ideally, the single requirement is satisfied, if and only if, the collective requirements of the component are satised.

Composition problem: compose independently analysed local timing properties into a global timing property. This problem is defined as composing the scheduling interfaces of components as a single real-time requirement.

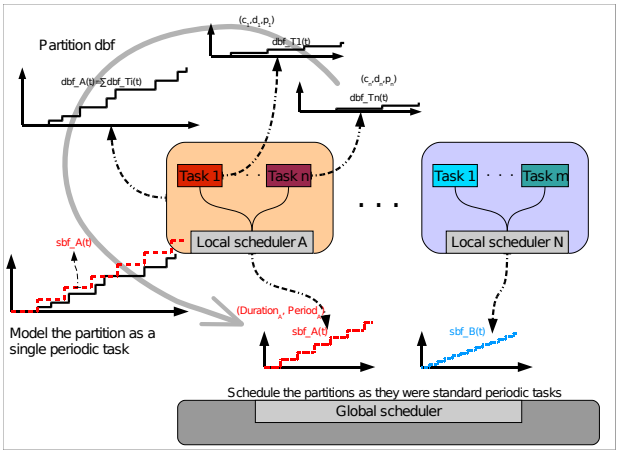

Fig. 1. Compositional system model.

In most approaches, each partition is modelled as a single periodic task (server) characterised by the pair (C_A, P_A), with the meaning that the server is allocated C_A units of execution every P_A units of time. The global scheduler decides when to schedule each partition (server).

Although starting from a completely different problem definition, the compositional scheduling and the aperiodic server models seems to arrive to the same solution.

Pros & cons.

[+] Clean isolation of scheduling concerns between partition developers and system integrator. The partition developers does not have to provide details about its internal operation (task attributes), just the temporal abstract interface of the partition.
[+] Close to the ARINC 653 system model.
[−] The abstract interface is an upper bound of the real needs of the partition, therefore there is a non-negligible utilisation penalty.
[−] Inter-partition resource sharing may difficult to implement and also take into account in the schedulability analysis.

2.3 Flat model

This approach consists of removing the barriers defined by the partitioned system and consider all the systems tasks at once. Then suppose that a single global scheduler is in charge of managing all the tasks, and conduct the corresponding schedulability analysis.

The last step is to adapt the solution back to the partitioned system by grouping (schedule the tasks of the same partition one after the other) the tasks of each partition in order to reduce the number of partition context switches.

Pros & cons.

[+] Dependencies between tasks of different partitions can be analysed and solved.
[+] Mature theory support for this model.
[+] The resulting schedule (or scheduling policy) can be very efficient. Depending on the task model, it may be possible to nd the optimal solution.
[−] A deep knowledge of the timing attributes of all the tasks is needed to do the scheduling analysis.
[−] There is not a clean separation of concerns between partition developers and system integrator, or even among partition developers.
[−] A change of an attribute of a task may change the whole schedule.

3 Tools review

The Autosar system model is not close to the ARINC 653 one, which is the target of this work, for this reason, the automotive related tool will not be considered in this brief review.

3.1 CHEDDAR

Cheddar [2] is an Ada framework which provides services to study the temporal behaviour of real-time applications, in particular if the application meets its temporal constraints.

It is a framework to **design and test custom schedulers**. The tool already implements the most common schedulers (RM, EDF, LLF, fixed priority, Round Robin, etc.).

The systems to be analysed can be described with AADL or a with Cheddar specific language. Therefore, it can used for quick prototyping of real-time schedulers or for educational purposes.

System model The system model is very complete and can capture most of the features of a real system.

A system is defined as:

One or more processors: Each processor has an scheduling policy (and the required scheduling options, if any).
One or more *address spaces*: These address spaces are scheduled by the processor scheduler and contains tasks, buffer or shared resources (an address space is equivalent to a POSIX processes). This element may define a

local scheduling policy in the case of hierarchical scheduling. If no scheduling policy is specified, then the processor scheduler is used to schedule the tasks.

One or more tasks: Tasks are the execution units (threads in the POSIX model). Each task belongs to an address space. Among others, a task has the following attributes: Capacity (WCET), Jitter, Deadline, Period, Priority, Context switch overhead, Start time and Blocking time.

It is possible to model the use of shared resources between tasks. Access to the shared resource can be controlled by an access protocol (PIP, PCP and IPCP). Also, precedence constrains can be specified in the form of:

- absolute precedence between tasks: a task can only start when another ends. Both tasks must have the same period.
- message: express relationships between a sender and a receiver.
- buffer: express relationships between a producer and a consumer of data from a buffer.

A resource is an object that defines the access protocol and the list of tasks that use it. For each task, when the resource is acquired and released is specified.

An example in the user guide explains how to define an ARINC 653 scheduler (Cheddar is a framework to build schedulers). Partitions are periodic activities defined with a duration and a period. Tasks are scheduled with a preemptive fixed priority policy in the partition's time.

3.2 MAST

MAST (Modelling and Analysis Suite for Real-Time Applications) is an open source set of tools for modelling real-time applications and performing timing analysis of those applications.

Initially designed around the RM scheduling policy, it was extended to support also other scheduling paradigms. Hierarchical scheduling has been included in the last version. The MAST analysis tools are listed below:

- Worst-case response time analysis (RTA):
  - Offset Based RTA for fixed priorities
  - Holistic RTA for fixed priorities
  - Offset-Based Optimised RTA for fixed priorities
  - Classic Rate Monotonic RTA for fixed priorities
  - Varying Priorities RTA
  - EDF Mono-processor RTA
  - EDF-Within-Priorities RTA, for hierarchical scheduling
- Calculation of blocking times
  - Single processor
  - Remote blocking for multi-processor
  - Assignment of optimum priority ceilings and preemption levels
- Calculation of Slack Times
  - Transaction Slacks
  - System Slacks
  - Processing Resource Slacks
  - Operation Slacks
- Optimised Priority Assignment Techniques
  - Single-processor
  - HOPA
  - Simulated Annealing

System model The system is as a set of platform resources (processors and networks) and a set of concurrent transactions.

Each *processing resource* (CPU) has a primary *scheduler object*. The scheduler allocates its capacity to the *scheduling servers*. A scheduling server can attend one or more *transactions*. The priority (in the case of fixed priority policy) of the scheduling server can be given explicitly of calculated according to the timing parameters of the transactions allocated to it.

Complex tasks (execution activities) are modelled as transactions. A simple transaction is defined as:

1. an input event, which defines the activation pattern (periodic, sporadic, aperiodic, etc.) of the task. In the case of a periodic task, object defines the period of the task
2. an operation, which represents a piece of code. The main attribute is the WCET of the code.
3. an output event, which is used to define the end of the transaction. The output event is used to define the deadline of the transaction.

It is possible to define complex transactions, composed of several sequential operations. Also, each operation can use one or more protected resources (critical sections).

3.3 Rapid-RMA

Rapid-RMA is an analysis and simulation scheduling tool. Although it has been designed for the fixed priority scheduling methodology, it has a module to perform basic scheduling analysis of cyclic schedulers.

Provides a powerful user friendly interface, and the system model is complete.

System model Rapid-RMA models the system as a set of resources that are used by the tasks. There are three kind of resources: "CPU", "Non-CPU" and "Node".

CPU: Models a processing unit. The most important attributes are: processing rate and context switch time.

Non-CPU: This category contains the following resource types: Memory, DMA, Bus, Database lock, Semaphore, IO port, Service, Method, Sensor, Panel. The user can define new types of resources.
The attributes are: ID, name, type, node (where the resource is located), acquisition time and deacquisition time, list of other nodes that access this resource.

Node: Models the notes of a distributed system.

Each task is a unit that performs work in the system concurrently with other tasks.

Basic task attributes: period, ready time (time when the task can start its first activation), WCET (Amount of work), priority and phase. It is also possible to define for each task:

- a set of non-preempt able subsections,
- intermediate deadlines,
- a list of resources,
- and a list of task dependencies.

It is possible to model both, the "amount of work" and the period of a task as a an statistical distribution (uniform, exponential, or user defined histogram).

4 Xoncrete overview

Xoncrete is a tool to assist the system designer to configure the resources (memory, communication ports, devices, processor time, etc.) allocated to each partition. It has two parts: 1) an resource editor to define the resources allocated to each partition; and 2) a scheduling analysis tool.

Although Xoncrete has been designed considering the syntax and resources managed by XtratuM, it can be easily adapted to the ARINC 653 model. In fact, the scheduling analysis part can be used directly to generate ARINC plans.

4.1 System model

The system model of Xoncrete is a based (simplified version) of the MARTE-UML specification. The elements of the system are:

Mutual exclusion resource: An abstract object used to define mutual exclusion between tasks. It is only characterised by its name.
We will assume the SRP as the access protocol to avoid unbounded priority inversion. The SRP protocol can be used with both, fixed and dynamic priorities.

Task: The elemental execution unit. A task is executed by a partition. A task belongs to a partition. Tasks are characterised by its computation time (WCET) and the set of mutual exclusion resources used. It is assumed that all the mutual exclusion resources are requested (locked) when the task starts and released (unlocked) when the task ends.
Contrarily to the classic periodic task model, where a task T_i is defined as the tuple (C_i, D_i, P_i, O_i), where C_i is the WCET, D_i is the deadline, P_i is the period and O_i is the initial offset; in Xoncrete, a task identifies just the block of code.

ETEF: It stands for End-to-End flow. A sequence of tasks with temporal attributes. The ETEF is the element that defines the periodic behaviour and temporal restrictions of the tasks. The tasks that are part of an ETEF can belong to different partitions, that is, an ETEF is not binded with an specific partition, it is a system wide element.
ETEFs are characterised by period, deadline and initial offset: as well as the sequence of tasks: $(\{T_a, .., T_k\}, D_i, P_i, O_i)$. The sequence of tasks is executed periodically, and the last task of the sequence must complete its execution before the specified deadline. Optionally, each task may have a partial deadline.
A task can appear in one ore more ETEFs. Since a task may be part of several ETEFs, the task may have different timing requirements depending the ETEF which is part of. An ETEF with one task, is equivalent to a single task in the classic periodic model.

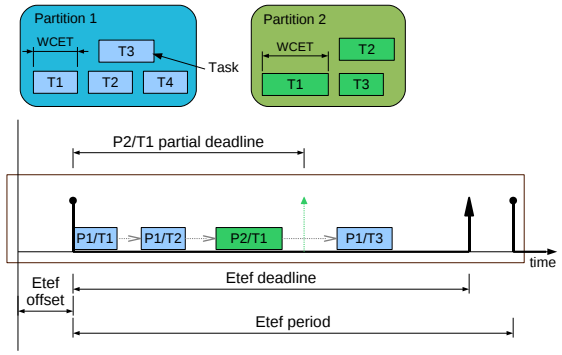

Fig. 2. Diagram of an ETEF.

Partition: A container where the tasks are executed. Task are directly scheduled by the partition itself. The partition defines the scheduling policy used to schedule its tasks.
Partitions are used to define spatial isolation, and where the tasks are executed/scheduled.

Hypervisor: The global scheduler. It schedules partitions.

About the periodic behaviour: As far as we know, the period has been always an "input" parameter to the scheduling analysis. In the classical models, the period is a parameter derived from the discrete regulator of a control system, or other physical constraint of some external actuator or sensor. In some cases, the tasks can operate at different rates without per-

formance loses, or the control system can be adjusted to operate at a given rate.

Therefore, it would be possible to define the period of an ETEF as a range of acceptable periods rather than a single fixed period, and let the analysis tool to select the period that produces a better plan.

The Major Active Frame (MAF) of a cyclic scheduling plan is computed as the least common multiple (LCM) of the periods of all the ETEFs. LCM is very sensitive to the primality between the periods. For example, if the periods are 8 and 17 then the LCM is 120[3]; reducing the period of the second ETEF to 16, then LCM(8,16)=16. The second ETEF is executed at a higher rate, and the LCM is much smaller.

The final MAF length can be greatly reduced if the user allows the periods to be "adjusted" by the Xoncrete tool, even when a very small flexibility is allowed.

The process to work with ranges of periods is:

1. The user defines WCET of each task.
2. The user defines the periodic behaviour of the ETEFs as ranges of valid periods or as fixed periods depending on the kind of application.
3. The tool calculates the period for each ETEF that produces the smallest MAF. The algorithm for finding the set of periods which produces the smallest MAF is a work in progress.
4. The user adjusts (if needed) the algorithms and operation of the tasks to the computed periods.

Typically, MAF length is reduced by making the periods of the tasks/ETEFs to be harmonic (to have large common divisors for the periods). This can be easily done in a single system (non-partitioned), where all the tasks belong to a single application and are developed by a single team. In a partitioned system, each partition may be developed by a different team, using a different execution environment, and with a different time bases. In this scenario, there are more chances of having non harmonic periods, and so the MAF may be too large to be practical. Note that a cyclic scheduler has to store the plan table in main memory.

Besides a smaller MAF, the specification of periods as ranges allows the integrator to adjust the utilisation factor during the integration phase.

5 Scheduling analysis

Once the system parameters and timing requirements hasve been introduced and validated, the system can be analysed and the scheduling plan can be generated.

[3] 8 and 17 are coprime or relatively prime.

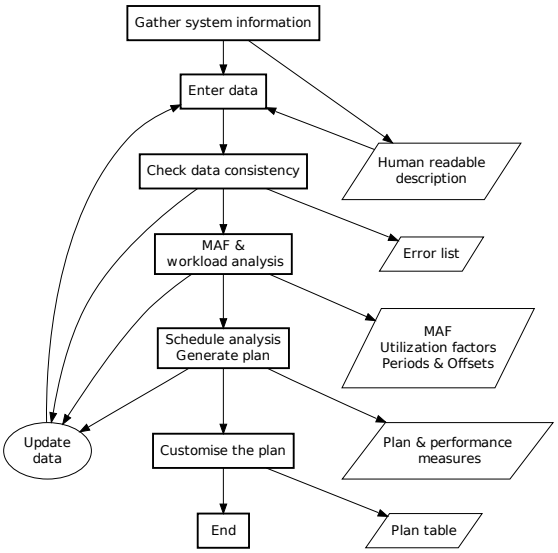

Fig. 3. Xoncrete work flow.

The analysis and plan generation is divided in three steps:

1. Tune ETEF parameters. Compute the MAF, compute the periods, and select the offsets. Figure 4 shows the analysis window, previously to the MAF calculation. The user is allowed to modify the ranges of the periods, while the tool computes the MAF in an iterative process.
   The result from this step is an exact, i.e. fixed periods and fixed offsets, workload (ETEFs) specification which can be used to generate the plan in the next step.
2. Generate schedule. The plan is generated and statistical information jointly with a graphical representation of the plan is presented to the user (figure 5).
   The plan is generated using well known scheduling techniques:
   - EDF as the base scheduling policy [8].
   - Minor modifications to the EDF criteria in order to reduce the number of partition context switches.
   - SRP to access mutual exclusion resources [9].

   The goal is to generate a short cyclic plan (with less slots) which can be used as an ARINC 653 partition plan. The tool also checks if the tasks of each partition can be scheduled by the local scheduler. In the case that the local scheduler is not able to schedule it (because there is not a valid priority assignment) then the plan that makes it schedulable is provided.
3. Tune schedule. The user can make minor local changes to the generated plan.
   This step is still under development.

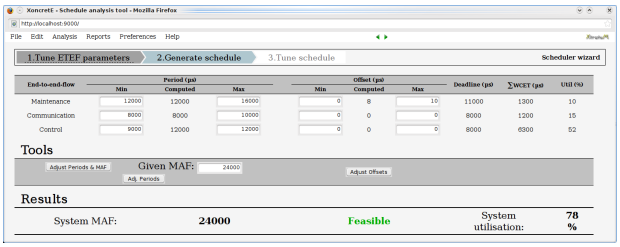

Fig. 4. Analysis: step 1.

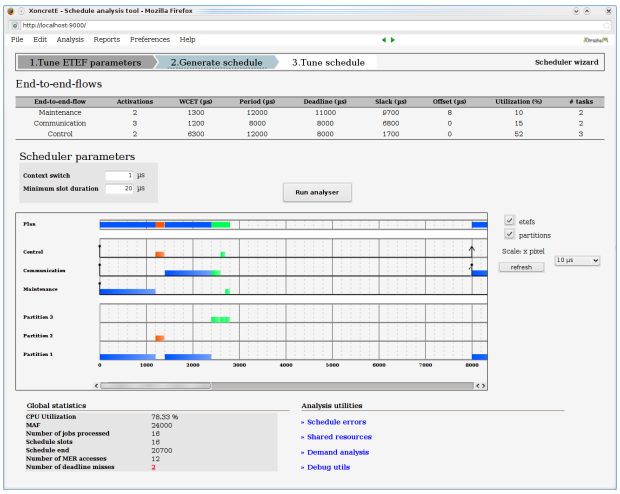

Fig. 5. Analysis: step 2.

6 Conclusion and future work

There is a gap between the research on scheduling algorithms and how these results can be used in real problems. Partitioned systems are widely used in safety critical systems where the applications are both, complex and critical.

Rather than implementing a complex tool (been able to use a wide variety of scheduling policies and complex systems models), Xoncrete is a tool designed to meet the requirements of an specific target: aeronautical and aerospatial systems. There are similar tools in the automotive industry (NETCAR-ECU ,SymTA/S ECU) specially designed to meet the automotive framework: simple tasks, bus centered, muli-processor, etc.

Although ARINC 653 specification defines a partitioned system with two level scheduling policies (cyclic for partitions and fixed priority for processes), it does not give information about how to build the cyclic plan or how to analyse the schedulability of the system.

This paper presents an extension of the ARINC 653 system model (based on the MARTE-UML) which is captures the timing requirements of complex partitioned systems. The model is powerful enough to capture most of the requirements of any real world application in an intuitive way; but also it is possible to build a compact and easy to debug plan.

Xoncrete is not a scheduling analysis algorithm, but an interactive tool to help the user to build the system.

As an on going work, we have developed an algorithm to find the minimum LCM of a set of ranges of periods.

7 Acknowledgement

This work has been partially supported by a CNES contract and an Spanish Government Research Office under grant TIN2008-06766-C03-02/TIN.

8 References

[1] Julio L. Medina, Julio L. Medina Pasaje, Michael Gonzlez Harbour, and Jos M. Drake. Mast real-time view: A graphic uml tool for modeling object-oriented real-time systems. In *In the 22nd IEEE Real-Time Systems Symposium (RTSS01*, pages 245–256, 2001.

[2] F. Singhoff, J. Legrand, L. Nana, and L. Marcé. Cheddar: a flexible real time scheduling framework. *Ada Lett.*, XXIV(4):1–8, 2004.

[3] Tri-Pacific. RapidRMA. www.tripac.com, 2002.

[4] M. Masmano, I. Ripoll, A. Crespo, J.J. Metge, and P. Arberet. Xtratum: An open source hypervisor for TSP embedded systems in aerospace. In *DASIA 2009. DAta Systems In Aerospace.*, May. Istanbul 2009.

[5] Luca Marzario, Giuseppe Lipari, Patricia Balbastre, and Alfons Crespo. Iris: A new reclaiming algorithm for server-based real-time systems. *Real-Time and Embedded Technology and Applications Symposium, IEEE*, 0:211, 2004.

[6] Patricia Balbastre, Ismael Ripoll, and Alfons Crespo. Exact response time analysis of hierarchical fixed-priority scheduling. In *Proceedings of 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, August 2009.

[7] Easwaran Arvind, Lee Insup, Sokolsky Oleg, and Vestal Steve. A compositional framework for avionics (arinc-653) systems. Technical report, Department of Computer & Information Science, University of Pennsylvania. No. MS-CIS-09-04, January 2009.

[8] E. W. Giering, III and T. P. Baker. A tool for the deterministic scheduling of real-time programs implemented as periodic ada tasks. In *SETA2: Proceedings of the second international symposium on Environments and tools for Ada*, pages 54–73, New York, NY, USA, 1994. ACM.

[9] Sanjoy K. Baruah. Resource sharing in edf-scheduled systems: A closer look. In *RTSS '06: Proceedings of the 27th IEEE International Real-Time Systems Symposium*, pages 379–387, Washington, DC, USA, 2006. IEEE Computer Society.

9 Glossary

AADL	Architecture Analysis and Design Language.
CBS	Constant Bandwidth Server
EDF	Earliest Deadline First
ETEF	End-to-End-Flow
IOMMU	Input/Output Memory Management Unit
IPCP	Immediate Priority Ceiling Protocol
IRIS	Idle-time Reclaiming Improved Server
LCM	Least Common Multiple
LLF	Least Laxity First
MAF	MAjor Frame
MAST	Modeling and Analysis Suite for Real-Time Applications
MMU	Memory Management Unit
PCP	Priority Ceiling Protocol
PIP	Priority Inheritance Protocol
RM	Rate Monotonic
RTA	Response Time Analysis
TBS	Total Bandwidth Server
SRP	Stack Resource Protocol
WCET	Worst Case Execution Time.