Timing Analysis of Mixed-Criticality Hard Real-Time Applications Implemented on Distributed Partitioned Architectures

Sorin Ovidiu Marinescu, Domiţian Tămaş–Selicean, Vlad Acretoaie, Paul Pop
Technical University of Denmark, Kongens Lyngby, Denmark
dota@imm.dtu.dk

Abstract

In this paper we are interested in the timing analysis of mixed-criticality embedded real-time applications mapped on distributed heterogeneous architectures. Mixed-criticality tasks can be integrated onto the same architecture only if there is enough spatial and temporal separation among them. We consider that the separation is provided by partitioning, such that applications run in separate partitions, and each partition is allocated several time slots on a processor. Each partition can have its own scheduling policy. We are interested to determine the worst-case response times of tasks scheduled in partitions using fixed-priority preemptive scheduling. We have extended the state-of-the-art algorithms for schedulability analysis to take into account the partitions. The proposed algorithm has been evaluated using several synthetic and real-life benchmarks.

1 Introduction

The current trend in mixed-criticality systems is towards "partitioned architectures", where several safety-critical functions, of different criticalities, are integrated into the same platform. Safety-Integrity Levels (SILs) capture the criticality level, and will dictate the development and certification procedures that have to be followed. In avionics, the proposed partitioning solution is based on "Integrated Modular Avionics" (IMA) [13], which allows the integration of mixed-criticality functions as long as there is enough spatial and temporal partitioning [13].

There are two basic approaches for handling hard real-time applications [7]. In the Event-Triggered (ET) approach, activities are initiated whenever a particular event is noted. In the Time-Triggered (TT) approach, activities are initiated at predetermined points in time. There has been a long debate in the real-time and embedded systems communities concerning the advantages of each approach, and the consensus is that the right approach depends on the particularities of the application [8]. Hence, in this paper, we assume that applications can be scheduled either using a TT approach, e.g., static-cyclic scheduling (SCS) or an ET approach, such as fixed-priority preemptive scheduling (FPS).

There is a large amount of research on hard real-time systems [7]. Researchers have addressed systems with mixed time-criticality requirements, showing how TT/ET tasks [10] can be integrated onto the same platform. Researchers have also started to address the integration of mixed safety-criticality tasks onto the same architecture [2, 14]. In [14], we have proposed an optimization approach to determine the mapping of tasks to PEs, the assignment of tasks to partitions, the sequence and size of the time slots on each PE and the schedule tables, such that all the applications are schedulable and the development costs are minimized.

In this paper we are interested in the timing analysis of applications which are running in partitions that use FPS. There is a large amount of research on scheduling and schedulability analysis [7, 3]. In this paper we use a response-time analysis [3] to calculate the worst-case response time R_i of every task τ_i, which is then compared to its deadline D_i. The basic response-time analysis presented in [3] has been extended over the years [5]. For example, the state-of-the-art analysis from [9] considers arbitrary arrival times and deadlines, offsets and synchronous inter-task communication (where a receiving task has to wait for the input of the sender task).

Audsley and Wellings [1] have proposed a schedulability analysis for FPS tasks in the context of temporal partitioning (let us denote this analysis with "SA", from Schedulability Analysis), which, when analyzing a FPS task in a certain partition, considers the other time-partitions as higher priority tasks. This analysis assumes that the deadlines are smaller or equal to the periods, that the tasks are independent, and that the start times of partition slices within a major frame are periodic. Pop et al. [10] have proposed a schedulability analysis for ET tasks, which extends the schedulability analysis in [9] to consider the influence of the TT tasks on the worst-case response times of the ET tasks. Such an analysis can be extended to consider that the TT tasks are "partitions" which can interfere with the ET tasks.

In this paper we have decided to extend the WCDOPS+ algorithm from [12], which extends the Worst Case Dynamic Offsets with Priority Schemes (WCDOPS) schedulaiblity analysis algorithm from [9]. Our proposed analysis, denoted with "SA+", takes into account the influence of time-partitions on the schedulability of the FPS tasks, and does not assume that the partition slices have to be periodic, as in [1]. Section 5 presents WCDOPS+, and our approach is presented in Section 5.1. The proposed schedulability analysis algorithm has been evaluated using several synthetic and real-life benchmarks, and has been compared to the analysis from [1].

2 Application Model

We consider a set of mixed-criticality applications. Each application has a SIL-level, from SIL_4 (most critical) to SIL_0 (non-critical) and is developed according to the certification requirements for the particular SIL. This section presents the application model of applications scheduled with FPS. For the full model the reader is referred to [14].

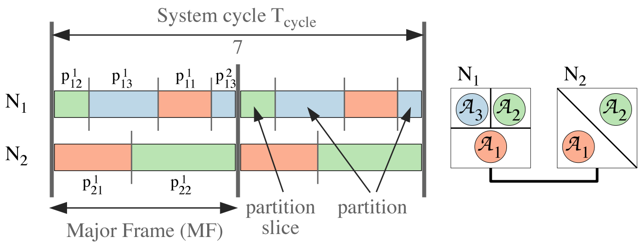


Figure 1: Partitioned architecture

We consider that the mapping of tasks to processing elements (PEs) and the assignment of tasks to partitions is given, and can be determined using our approach from [14]. We model an application $\mathcal{A}$ as a directed, acyclic graph $\Gamma(\mathcal{V}, \mathcal{E})$. Each node $\tau_i \in \mathcal{V}$ represents one task. An edge $e_{ij} \in \mathcal{E}$ represents a precedence relationship between τ_i and τ_j, and indicates that τ_i must complete its execution before τ_j. Each task τ_i is characterized by a worst-case execution time (WCET) C_i (on the PE that is assigned to for execution), a best-case execution time C_i^{min}, and in case the task uses shared resources, a maximum blocking time B_i. Additionally, τ_i has an unique priority denoted by $prio(\tau_i)$, an offset Φ_i and a maximum release time jitter J_i. Thus, considering that the graph Γ to which τ_i belongs is triggered by an external event arriving at t_0, τ_i arrives at time $t_0 + \Phi_i$ and is released after an additional maximum delay of J_i. For each application we have a deadline $D_{\mathcal{A}}$ and a period $T_{\mathcal{A}}$.

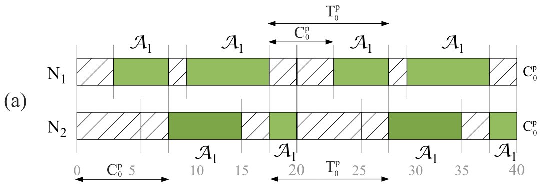
3 System Model

We consider architectures composed of a set $\mathcal{N}$ of PEs connected by a broadcast communication channel. We assume that the hardware and software architecture implements a temporal- and space-partitioning scheme similar to IMA [14].

Each application $\mathcal{A}_i$ is allowed to execute only within its defined partition P_j. Each partition can use its own scheduling policy. On a processing element N_i, a partition P_j is defined as the sequence P_{ij} of k partition slices p^k_{ij}, $k \geq 1$. A partition slice p^k_{ij} is a predetermined time interval in which the tasks of application $\mathcal{A}_j$ mapped to N_i are allowed to use the PE. All the slices on a processor are grouped within a Major Frame (MF), that is repeated periodically. The period T_{MF} of the major frame is given by the designer and is the same on each node. Several MFs are combined together in a system cycle that is repeated periodically, with a period T_{cycle}.

In Fig. 1 we have 3 applications, $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$, implemented on 2 PEs, N_1 and N_2, with $T_{MF} = 10$ and $T_{cycle} = 20$. The tasks of $\mathcal{A}_1$, for example, can execute only in partition P_1 on PE N_1, composed of the partition slice $p^1_{1,1}$, and in partition P_1 on PE N_2, composed of the slice $p^1_{2,1}$. The sequence and length of the partition slices in a MF are the same (on a given PE), but the contents of the slices can differ. An application can extend its execution over several MFs.

For simplicity, in this paper we ignore the communication. However, researchers have shown how realistic bus protocols, e.g., FlexRay [11], can be integrated into the analysis.



(b)

		C_i	T_i	PE	R_i^{SA}	R_i^{SA+}
$\mathcal{A}_1$	τ_1	2	15	N_1	12	7
	τ_2	8	40	N_1	20	19
	τ_3	3	30	N_1	30	27
	τ_4	3	30	N_2	19	14
	τ_5	4	40	N_2	50	18
	τ_6	8	100	N_1	120	59

Figure 2: Motivational example

4 Motivational Example

Let us illustrate the importance of accurately taking the partitions into account during the analysis using the example in Fig. 2. We consider a system with 2 PEs and 6 tasks. The partition table is given in Fig. 2a, where two consecutive MFs with T_{MF} = 20 are presented. The details of the application are given in Fig. 2b (the tasks are sorted according to their priorities–highest on the top–and the deadlines are equal to the periods).

The mapping and the partitioning are given, and we assume they are derived with our approach from [14]. The hashed partition slices represent partitions on which the application $\mathcal{A}_1$ is not allowed to execute. The partition slices corresponding to the application $\mathcal{A}_1$ are coloured in green.

We are interested to determine the worst-case response times of the tasks in Fig. 2b considering the partitions in Fig. 2a. We have compared the SA analysis from [1] with our proposed analysis, SA+. To facilitate the comparison, we consider the assumptions from [1], i.e., deadlines are equal to the periods, and we ignore the dependencies. The SA results are presented in column 5 in Fig. 2b, and the SA+ results are in column 6. As we can see, SA is much more pessimistic (the application is considered unschedulable) compared to our proposed analysis, SA+.

The pessimism of SA comes from its limiting assumptions that the partition slices have to be periodic within a MF. This assumption is not true in practice, but it simplifies the analysis. When analyzing the tasks in a partition P_j on a PE N_i, SA merges all the other partition slices into a "higher priority task" with WCET C^p_0 (the length of the other slices) and period T^p_0. In order to apply SA in the general context of Fig. 2a, we have to consider C^p_0 as the longest time-interval of continuous partition slices $\notin P_j$ on N_i, and T^p_0 as the shortest inter-arrival time of these intervals. These values are calculated for each PE. For example, the values of C^p_0 and T^p_0 for N_1 and N_2 in Fig. 2a are as depicted in the figure. Our proposed analysis, SA+, does not assume that the partition slices have to be periodic, and thus reduces the pessimism of SA by accurately taking into account the exact position and size of the partition slices.

5 Response Time Analysis

WCDOPS+ [12] is an algorithm that performs worst-case response time analysis on fixed priority scheduled tasks disposed in tree-shaped transactions taking into consideration the precedence constraints between them, which later was extended to consider graphs [6].

The WCDOPS+ response time analysis for a certain task τ_{ab} is based on finding the contributions from each transaction in the system to a busy period of τ_{ab}. The busy period (also called busy window) of τ_{ab} is defined as the longest interval of time during which tasks with priority greater or equal than τ_{ab} are executed continuously [5].

When studying the contribution of a transaction Γ_i to the worst case response time of a task τ_{ab}, it may be useful to somehow group the tasks of Γ_i and treat each group as if it were a single task. The creation of these groups of tasks is accomplished by defining the concepts of H sections and H segments. Two tasks of a transaction Γ_i belong to the same H section for the analysis of τ_{ab} if they belong to $hp_i(\tau_{ab})$ and there is no intermediate task in the transaction that belongs to $lp_i(ab)$. Similarly, two tasks of a transaction Γ_i belong to the same H segment for the analysis of τ_{ab} if they belong to $hp_i(\tau_{ab})$ and there is no intermediate task in the transaction that does not belong to $hp_i(ab)$. The set of tasks $hp_i(ab)$ represents the tasks belonging to a transaction Γ_i that are executed on the same PE as τ_{ab} and have a priority greater or equal than τ_{ab}. The set $lp_i(\tau_{ab})$ contains tasks belonging to Γ_i and executed on the same PE as τ_{ab} that have lower priorities than τ_{ab}.

H segments are more restrictive than H sections in the sense that if two tasks belong to the same H section with respect to a task τ_{ab}, then they *may* belong to the same τ_{ab} busy period, while if two tasks belong to the same H segment with respect to a task τ_{ab}, then they *must* belong to the same τ_{ab} busy period.

WCDOPS+ analyzes separately the contributions to the τ_{ab} busy period: first, the contributions made by all the transactions to which the task τ_{ab} doesn't belong to; secondly, the contribution made by the transaction Γ_a of which τ_{ab} is a part of. For each transaction, two types of contributions are considered: a non-blocking interference (W_i) and a blocking interference (WB_i). Since, as shown in [12] only one blocking H segment can contribute to the busy period, WCDOPS+ allows this contribution to the transaction with the maximum interference increase ($\Delta W = WB_i$ - W_i).

The worst-case response time of an instance of τ_{ab} with the index p_{ab} is determined by WCDOPS+ based on its completion time, $w_{abc}(p_{ab})$, which is composed of the following: the maximum blocking time from lower priority tasks, a blocking interference and a non-blocking interference from the transactions in the system. The blocking interference is expressed through the interference increase which has to be maximized for the calculation of worst case completion time. Since there can only be one blocking segment executing in a busy period, the interference increase is chosen to be the maximum from the interference increases calculated separately:

$$\Delta W^*_{ac}(\tau_{ab}, w, p_{ab}) = MAX(\Delta W_{ac}(\tau_{ab}, w, p_{ab}), \\ \Delta W^*_i(\tau_{ab}, w, p_{ab}, \tau_{ac})) \quad (1)$$

The non blocking interference is obtained by summing up the non blocking interferences from all transactions in the system, so the completion time $w_{abc}(p_{ab})$ is

$$w_{abc}(p_{ab}) = B_{ab} + W_{ac}(\tau_{ab}, w, p_{ab}) + \\ \sum_{\forall i \neq a} W^*_i(\tau_{ab}, w, \tau_{ac}) + \Delta W^*_{ac}(\tau_{ab}, w, p_{ab}) \quad (2)$$

This equation is solved iteratively and because the instance p_{ab} of task τ_{ab} arrives at $\varphi_{abc} + (p_{ab} - 1)T_a$, its response time is [12]:

$$R^w_{abc}(p_{ab}) = w_{abc}(p_{ab}) - \varphi_{abc} - (p_{ab} - 1)T_a + \phi_{ab} \quad (3)$$

The worst-case response time R^w_{ab} for the task τ_{ab} is the maximum value of the result in Eq. 3, considering all the critical instants initiated by higher priority tasks and by τ_{ab} and also all the job instances.

5.1 Extending WCDOPS+ with Partitioning

We have extended WCDOPS+ to take into account the partitions by using the concepts of $availability$ and $demand$, inspired by the approach presented in [10]. Informally, the $availability$ associated to a task τ_{ij} during a time interval t, denoted as $A_{ij}(t)$, is equal to the processor time that is not used by other partitions during t. The *demand* for a task τ_{ij} during a time interval t, denoted as $H_{ij}(t)$, is equal to the sum of the processor times required by τ_{ij} and all higher priority tasks mapped to the same processor during t.

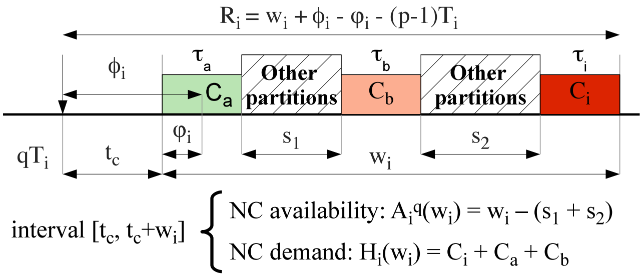

R_i = w_i + ϕ_i - φ_i - (p-1)T_i
ϕ_i
τ_a
C_a
Other partitions
τ_b
C_b
Other partitions
τ_i
C_i
φ_i
s_1
s_2
qT_i
t_c
w_i
interval [t_c, t_c+w_i]
NC availability: A_i^q(w_i) = w_i – (s_1 + s_2)
NC demand: H_i(w_i) = C_i + C_a + C_b

Figure 3: Availability and demand

In more general terms, the demand of a task scheduled in a partition P_k is equal to the length of its busy period when there wouldn't be any time partitions considered or when P_k would be the only partition on the processor. Fig. 3 shows that the demand of task τ_i during the busy window w_i is equal to the sum of the worst-case execution time of the higher priority tasks C_a and C_b and the worst-case execution time of the task in question, C_i. In the WCDOPS+ analysis, the length of a τ_{ab} busy period is called τ_{ab}'s completion time and is expressed in Eq. 2. Thus, the demand $H_{abc}(p_{ab})$ of an instance p_{ab} of a task τ_{ab} during a busy period initiated by a task τ_{ac} is equal to p_{ab}'s completion time.

$$H_{abc}(p_{ab}) = w_{abc}(p_{ab}) \quad (4)$$

The availability associated to an instance p_{ab} of a task τ_{ab}, scheduled in a partition P_k, is the processing time available during $w_{ab}(p_{ab})$ for P_k. Because of the time partitioning scheme and because task τ_i can execute only during its own partition P_k, the availability is calculated by subtracting from $w_{ab}(p_{ab})$ the time reserved for the "other" partitions.

In Fig. 3, the availability is shown as what is left after subtracting from w_i the durations of the other partitions, s_1 and s_2.

As a consequence of considering the partitioning scheme, the completion time $w_{abc}(p_{ab})$ of an instance p_{ab} of task τ_{ab} is replaced by an $extended\ completion\ time\ e_{abc}(p_{ab})$, computed according to Algorithm 1. The purpose of this algorithm is to increase a task's completion time until the availability is at least as large as the demand during this time interval. The algorithm starts by initializing the extended completion time of p_{ab} with p_{ab}'s original completion time and the availability and demand with 0 (lines 10–12). It then proceeds to iteratively re-compute the availability and demand until the availability is greater or equal to the demand (lines 13–19). At each iteration, the extended completion time of p_{ab} is increased with the difference between the current values of the availability and demand (lines 16–18). Finally, the obtained extended completion time is returned (line 20).

Algorithm 1 EXTENDED_COMPLETION_TIME

```
1:  Inputs:
2:  $\tau_{ab}$ - a task;
3:  $p_{ab}$ - an instance of $\tau_{ab}$;
4:  $\tau_{ac}$ - a task starting a $p_{ab}$ busy period;
5:  $w_{abc}$ - completion time of $p_{ab}$;
6:  Outputs:
7:  $e_{abc}$ - extended completion time of $p_{ab}$;
8:
9:  begin
10: $e_{abc} \leftarrow w_{abc}$;
11: $demand \leftarrow 0$;
12: $availability \leftarrow 0$;
13: repeat
14:     $demand \leftarrow COMPUTE\_DEMAND(\tau_{ab}, p_{ab}, \tau_{ac}, e_{abc})$;
15:     $availability \leftarrow COMPUTE\_AVAILABILI - TY(w_{abc})$;
16:     if $demand > availability$ then
17:         $e_{abc} \leftarrow e_{abc} + demand - availability$;
18:     end if
19: until $availability \geq demand$
20: return $e_{abc}$;
21: end
```

6 Experimental Evaluation and Conclusions

Table 1 presents our experimental evaluation. We have used seven synthetic benchmarks and one real-life example. The synthetic benchmarks were generated similar to [14], and the number of PEs and number of tasks in the FPS applications are presented in columns 2 and 3, respectively, in Table 1. The real-life case study is derived from the "automotive" benchmark in the E3S suite [4]. The partition tables and the mapping have been generated using the "Initial Solution" approach from [14]. We have run both SA and SA+ on these tasks sets. Columns 4 and 5 present the number of tasks found schedulable, i.e., $R_i \leq D_i$, using SA and SA+, respectively. To show the reduction in the pessimism of SA+ over SA, the last column in the table represents the percentage reduction of worst-case response times obtained with SA+ compared to SA averaged over all tasks.

As we can see from these results, accurately taking into account the partitions during the analysis can significantly reduce the pessimism of the results.

Table 1: Experimental results

Test Case	PEs	Tasks	SA	SA+	% reduction
1	2	4	3	4	59.04
2	3	7	5	7	27.95
3	3	10	6	10	46.82
4	4	16	12	15	41.66
5	4	19	17	19	24.91
6	5	22	20	22	56.67
7	5	25	21	25	30.23
automotive	3	5	4	4	34.18

References

[1] N. Audsley and A. Wellings. Analysing APEX applications. In *Real-Time Systems Symp.*, pages 39 –44, 1996.

[2] S. K. Baruah, A. Burns, and R. I. Davis. Response-Time Analysis for Mixed Criticality Systems. *Proceedings of the Real-Time Systems Symposium*, pages 34–43, 2011.

[3] G. C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.

[4] R. Dick. Embedded system synthesis benchmarks suite. http://ziyang.eecs.umich.edu/d̃ickrp/e3s/.

[5] C. Fidge. Real-time schedulability tests for preemptive multitasking. *REAL-TIME SYSTEMS*, 14(1):61–93, 1998.

[6] J. P. Kany and S. H. Madsen. Design optimisation of fault-tolerant event-triggered embedded systems. Master's thesis, Dept. of Informatics and Mathematical Modelling, Technical University of Denmark, 2007.

[7] H. Kopetz. *Real-Time Systems-Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[8] H. Lonn and J. Axelsson. A comparison of fixed-priority and static cyclic scheduling for distributed automotive control applications. In *Proceedings of the Euromicro Conference on Real-Time Systems*, pages 142–149. IEEE, 1999.

[9] J. Palencia and M. Harbour. Exploiting precedence relations in the schedulability analysis of distributed real-time systems. *Proceedings - Real-Time Systems Symposium*, pages 328–339, 1999.

[10] T. Pop, P. Pop, P. Eles, and Z. Peng. Analysis and optimisation of hierarchically scheduled multiprocessor embedded systems. *International Journal of Parallel Programming*, 36(1):37–67, 2008.

[11] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei. Timing analysis of the FlexRay communication protocol. *Real-Time Systems*, 39(1-3):205–235, 2008.

[12] O. Redell. Analysis of tree-shaped transactions in distributed real-time systems. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, pages 239–248, Washington, DC, USA, 2004. IEEE Computer Society.

[13] J. Rushby. Partitioning for avionics architectures: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Also to be issued by the FAA.

[14] D. Tămaş-Selicean and P. Pop. Design Optimization of Mixed-Criticality Real-Time Applications on Cost-Constrained Partitioned Architectures. In *Proceedings of the Real-Time Systems Symposium*, pages 24–33, 2011.