Task Mapping and Partition Allocation for Mixed-Criticality Real-Time Systems

Domiţian Tămaş–Selicean and Paul Pop
DTU Informatics, Technical University of Denmark
Kongens Lyngby, Denmark

Abstract—**In this paper we address the mapping of mixed-criticality hard real-time applications on distributed embedded architectures. We assume that the architecture provides both spatial and temporal partitioning, thus enforcing enough separation between applications. With temporal partitioning, each application runs in a separate partition, and each partition is allocated several time slots on the processors where the application is mapped. The sequence of time slots for all the applications on a processor are grouped within a Major Frame, which is repeated periodically. We assume that the applications are scheduled using static-cyclic scheduling. We are interested to determine the task mapping to processors, and the sequence and size of the time slots within the Major Frame on each processor, such that the applications are schedulable. We have proposed a Tabu Search-based approach to solve this optimization problem. The proposed algorithm has been evaluated using several synthetic and real-life benchmarks.**

I. Introduction

The current trend is towards "integrated 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 processes and certification procedures that have to be followed. In avionics, the proposed integration solution is based on "Integrated Modular Avionics" (IMA) [9], which allows the integration of mixed-criticality functions as long as there is enough *spatial* and *temporal* partitioning [9].

There is a large amount of research on hard real-time systems [2], [6]. However, there is little research work on the integration of mixed-criticality applications onto the same platform. Lee et al. [7] consider an IMA-based system where all tasks are scheduled using fixed-priority preemptive scheduling (FPS). There are several works where mixed-criticality tasks are addressed, but researchers assume that tasks of different criticality share the same processor with little or no separation (i.e., there is no spatial-partitioning) [1], [3].

In this paper we consider distributed embedded systems composed of heterogeneous processing elements (PEs), that implement hard real-time applications with different SILs, including non-critical functions. We consider that the hardware and software architecture provides both spatial and temporal partitioning. A detailed discussion about partitioning is available in [9]. We assume that the applications are scheduled using static-cyclic scheduling (SCS).

In [10] we have proposed a Simulated Annealing-based approach to the optimization of time partitions, considering that the mapping of tasks to processing elements is fixed. In this paper, we are interested to determine the mapping of tasks to PEs, the sequence and length of the time partitions on each PE such that the applications are schedulable. We have proposed a Tabu Search-based approach to solve this optimization problem.

II. System Model

On a processing element N_i, a partition P_j is defined as the sequence P_{ij} of k partition slices p_{ij}^k, $k \geq 1$. A partition slice p_{ij}^k 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}. Within a T_{cycle}, the sequence and length of the partition slices are the same across MFs (on a given PE), but the contents of the slices can differ.

The set of all applications in the system is denoted with Γ. The applications can be of different criticality levels, and all the tasks of an application have the same SIL. We model an application as a directed, acyclic graph, where a node represents one task. An edge e_{ij} indicates a synchronous communication: task τ_j waits for the output of τ_i. Communication between tasks mapped to different PEs is performed by message passing over the bus. The message sizes s_{m_i} of each message m_i are known. A deadline $D_{\mathcal{G}_i} \leq T_{\mathcal{G}_i}$, where $T_{\mathcal{G}_i}$ is the period of $\mathcal{G}_i$, is imposed on each task graph $\mathcal{G}_i$.

The mapping of tasks to processors is denoted by the function $M : \mathcal{V}_i \rightarrow \mathcal{N}$, where $\mathcal{N}$ is the set of PEs in the architecture. This mapping is not yet known, and will be decided by our optimization approach. For each task τ_i we know the worst-case execution time (WCET) on the processing elements where it is considered for mapping.

III. Problem Formulation

The problem can be formulated as: given a set Γ of applications, an architecture consisting of a set $\mathcal{N}$ of processing elements, the size of the major frame T_{MF} and the application cycle T_{cycle}, we are interested to find an implementation Ψ such that all applications meet their deadlines. Deriving an implementation Ψ means deciding on the mapping M of tasks to PEs, the set $\mathcal{P}$ of partition slices on each PE, the assignment of tasks to partitions and the schedule $\mathcal{S}$ for all the tasks.

The problem is illustrated in Fig. 1a where we have applications $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ to be implemented on two PEs, N_1 and N_2. The size of the messages is specified on the edges. The WCET for each task on the two PEs is in the table in Fig. 1a, with "*x*" meaning the task cannot be mapped to that PE. We have T_{MF}=15 and T_{cycle}=30 ms.

Fig. 1b presents the optimal mapping (see the definition of the cost function in section IV) of tasks to PEs, in case the system would *not* be partitioned. The optimal time-partitioning result, considering the mapping obtained separately from Fig. 1b, which does not consider time partitions, is depicted in Fig. 1c. As we can see, this is an unsuccessful implementation where $\mathcal{A}_1$ and $\mathcal{A}_2$ miss their deadlines.

In order to successfully schedule all the applications, mapping has to be determined at the same time with the optimization of time partitions. The optimal mapping *and* partitioning solution is depicted in Fig. 1d. All the applications are schedulable.

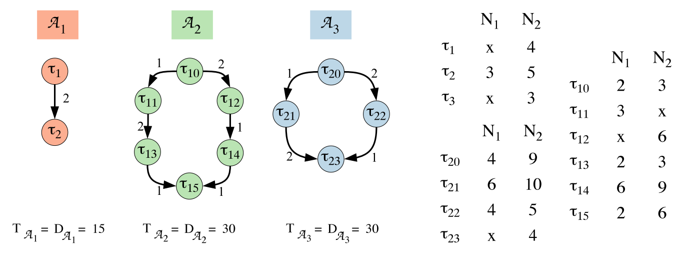
Moreover, the partitions assigned to applications $\mathcal{A}_2$ and $\mathcal{A}_3$ have extra unused time. Such slack can be used for future upgrades.

IV. Mapping and Time-Partition Optimization

The problem presented in the previous section is NP-complete, hence, we use a Tabu Search (TS) meta-heuristic to determine the task mapping M and the set of partition slices $\mathcal{P}$ such that the applications are schedulable and the unused partition space is maximized. TS [5] is a meta-heuristic optimization, which searches for that solution which minimized the *cost function*. We define the cost function as the "degree of schedulability" [8], which is a sum, for all applications, of the difference between the end-to-end completion time of the application and its deadline.

The neighboring solutions of the current solution are generated using design transformations (or "moves"). Thus, we apply *re-mapping* moves by randomly selecting a task from each application and changing its mapping to a randomly chosen PE. Also, there are four types of moves applied to partition slices: *resize*, *swap*, *join* and *split*. The moves are applied to a randomly selected partition slice on each PE.

Each alternative provided by TS (a mapping M and a partition set $\mathcal{P}$) is evaluated using a List Scheduling-based heuristic to determine the schedule tables $\mathcal{S}$ for each application, which we have modified [10] to take into account the time partitions.

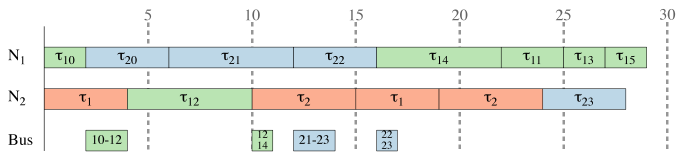

$\mathcal{A}_1$
$\mathcal{A}_2$
$\mathcal{A}_3$
$T_{\mathcal{A}_1} = D_{\mathcal{A}_1} = 15$
$T_{\mathcal{A}_2} = D_{\mathcal{A}_2} = 30$
$T_{\mathcal{A}_3} = D_{\mathcal{A}_3} = 30$

	N_1	N_2
τ_1	x	4
τ_2	3	5
τ_3	x	3

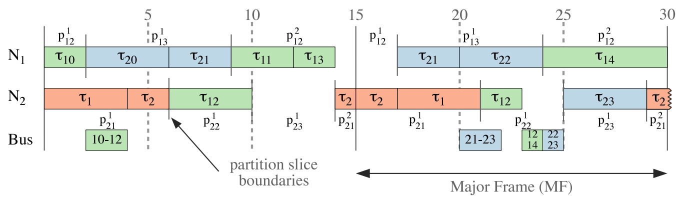
	N_1	N_2
τ_{20}	4	9
τ_{21}	6	10
τ_{22}	4	5
τ_{23}	x	4

	N_1	N_2
τ_{10}	2	3
τ_{11}	3	x
τ_{12}	x	6
τ_{13}	2	3
τ_{14}	6	9
τ_{15}	2	6

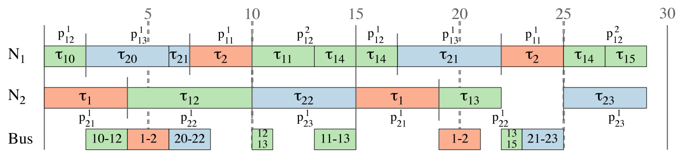
(a) Application model



(b) Optimal mapping, no partitioning



(c) Optimizing partitioning using the mapping from Fig. 1b – not schedulable. τ_{15} and τ_2 do not fit in the schedule.



(d) Optimizing mapping and partitioning simultaneously – all applications schedulable. We remapped τ_{13} and τ_{22} from N_1 to N_2 and τ_2 from N_2 to N_1 and resized the partition slices.

Fig. 1: Motivational example

TABLE I: Experimental results for benchmarks

Test Case	Apps	Tasks	PE	MO+TPO	MTPO	% increase
1	3	15	2	0	3	261.54
2	3	20	3	2	3	223.81
3	4	34	4	2	3	78.13
4	4	40	5	2	4	153.66
5	5	53	6	4	5	3116.67
consumer	2	12	3	2	2	19.88
networking	4	13	3	4	4	55.52
telecom	9	30	3	3	9	100.01

V. Experimental Evaluation

For the evaluation of our proposed algorithm we used 5 synthetic benchmarks and 3 real life case studies from the Embedded Systems Synthesis Benchmarks Suite (E3S) [4].

In the first set of experiments we were interested to evaluate the proposed MTPO strategy in terms of its ability to find schedulable implementations. We used 5 synthetic benchmarks (lines 2–6 in Table I). Columns 2, 3 and 4 in the table present the number of applications, tasks and PEs, respectively. Table I also presents the results obtained using another approach, MO+TPO, where we first optimize the mapping (MO), but without considering partitioning, and using this mapping we perform a time partition optimization (TPO). The *number* of schedulable applications obtained by MO+TPO strategy is presented in column 5, while column 6 presents the results obtained using our proposed MTPO. We have used a time limit of 4 hours for all experiments.

As we can see, MO+TPO which does not perform simultaneous mapping and partition optimization, is not able to find schedulable implementations in all the cases. However, with MTPO, we are able to optimize the mapping and the time partitions concurrently, such that all applications are schedulable. We have measured the ability of MTPO to improve over MO+TPO by using a percentage average increase in the degree of schedulability over all applications, presented in the last column. There is a large increase in the degree of schedulability when using MTPO, which means that we can potentially implement the applications on a slower (cheaper) platform.

The results obtained from the real-life benchmarks are presented in lines 7–10, and confirm the results of the synthetic benchmarks.

References

[1] S. Baruah, H. Li, and L. Stougie. Towards the design of certifiable mixed-criticality systems. In *Real-Time and Embedded Technology and Applications Symp.*, pages 13 –22, 2010.
[2] G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, Boston, 1997.
[3] D. de Niz, K. Lakshmanan, and R. Rajkumar. On the scheduling of mixed-criticality real-time task sets. In *Proc. of the Real-Time Systems Symposium*, pages 291–300, 2009.
[4] R. Dick. Embedded system synthesis benchmarks suite. http://ziyang.eecs.umich.edu/ dickrp/e3s/.
[5] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.
[6] H. Kopetz. *Real-Time Systems-Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.
[7] Y.-H. Lee, D. Kim, M. Younis, J. Zhou, and J. McElroy. Resource scheduling in dependable integrated modular avionics. In *Proc. of Dependable Systems and Networks*, pages 14 –23, 2000.
[8] P. Pop, P. Eles, and Z. Peng. *Analysis and Synthesis of Communication-Intensive Heterogenous Real-Time Systems*. Kluwer Acad. Publ., 2004.
[9] J. Rushby. Partitioning for avionics architectures: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999.
[10] D. Tamas-Selicean and P. Pop. Optimization of Time-Partitions for Mixed-Criticality Real-Time Distributed Embedded Systems. In *Proc. of the Workshop AMICS*, 2011.