Optimization of Partitioned Architectures to Support Soft Real-Time Applications

Tamas-Selicean, Domitian; Pop, Paul

Published in:
Proceedings of 2014 IEEE 20th Pacific Rim International Symposium on Dependable Computing (PRDC)

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

Optimization of Partitioned Architectures to Support Soft Real-Time Applications

Domiţian Tămaş–Selicean and Paul Pop
DTU Compute Technical University of Denmark
Kongens Lyngby, 2800 Denmark

I. Introduction

Partitioned architectures (PAs) allow the safe integration of applications of different criticality levels on the same platform, reducing the development, verification and integration costs. PAs rely on partitioning mechanisms at the platform level to ensure temporal and spatial separation between applications of different criticality levels. With PAs, each application is running in its own partition. Spatial partitioning protects the private data or devices of an application in a partition from being tampered with, by another application. Temporal partitioning ensures that an applications access to shared resources is not affected by applications in other partitions.

PAs have been successfully used in several industries, including automotive and avionics. For example, in the avionics area, platform level separation mechanisms are described in the ARINC 653 software specification, also called Integrated Modular Avionics [3]. Recently, the European Space Agency (ESA) and the National Aeronautics and Space Administration (NASA) have also shown interest in PAs, as a way to "manage the growth of mission function implemented in the on-board software" [8], and as intermediate step to introducing multi-core processors in spacecraft computers [7].

In [6], we have addressed the optimization of PAs for *hard* real-time applications, focusing on finding schedulable implementations that minimize the development and certification costs. In this paper we are not interested in the issue of cost minimization, but in supporting *soft* real-time applications that share the same PA with critical *hard* real-time applications. The advantage of a PA is that it allows the integration of mixed-criticality applications, including non-critical and soft real-time applications, onto the same platform. Our proposed optimization approach determines an implementation such that all hard real-time applications are schedulable and the quality of service of the soft real-time tasks is maximized.

II. System Model

On a processing element (PE) N_i, a partition P_j is defined as the sequence P_{ij} of partition slices. A partition slice 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 not yet known and will be decided by our optimization approach. 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 $\Gamma = \Gamma_H \cup \Gamma_S$, where Γ_H is the subset of hard real-time applications (HRT), and Γ_S is the subset of soft real-time applications (SRT). The applications can be of different criticality levels. We model an application as a directed, acyclic graph, where a node represents one task. An edge indicates a communication. 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. We consider this mapping as given by the designer. For each task τ_i we know the worst-case execution time (WCET) C_i on the PE where it mapped. Furthermore, the assignment of tasks to partitions as fixed. The applications can be scheduled using either fixed-priority scheduling (FPS) or static cyclic scheduling (SCS). A deadline $D_i \leq T_i$ is imposed on each task graph $\mathcal{A}_i$ for SCS applications, and on each τ_i for FPS tasks, where T_i is the period of the application/task.

Unlike for HRT applications, missing a deadline will not lead to system failure for SRT applications: they will continue functioning, but with a degraded service. For each SRT application A_j, we use a quality of service (QoS) function $QoS(A_j) \in [0,1]$. This function is specific for each application and is given by the designer.

III. Problem Formulation

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

IV. Partitioned Architecture Optimization

Next, we describe the proposed "Partitioned Architecture Optimization" (PAO) strategy. We have modified and extended our Tabu Search (TS) approach from [6] to solve the problem formulated in the previous section. TS [2] is a meta-heuristic optimization that searches for the solution that minimizes the *cost function*. The exploration of the design space is done by applying design transformations (moves) to the current solution. To escape local minima, and to prevent the search from revisiting solutions, TS uses an adaptive memory (called "tabu list").

PAO uses four types of moves applied to partition slices: *resize*, *swap*, *join* and *split*. These moves are applied to a randomly selected partition slice on each PE. PAO also uses a *resize MF* move that increases or decreases the T_{MF}, proportionally adjusting the partition slice sizes.

We define the cost function as:

$$Cost(\psi) = \begin{cases} c_1 = \sum_{\mathcal{A}_i \in \Gamma_H} \max(0, R_i - D_i) & \text{if} c_1 > 0 \\ c_2 = -\sum_{\mathcal{A}_j \in \Gamma_S} QoS(A_j) & \text{if} c_1 = 0 \end{cases} \quad (1)$$

If at least one HRT application $\mathcal{A}_i$ from the set Γ_H is not schedulable, there exists one R_i greater than the deadline D_i, and therefore the term c_1 will be positive (c_1 drives the search towards schedulable solutions). If all the applications in Γ_H are schedulable, then each R_i is smaller than D_i, and the term c_1 = 0. In this case, we use c_2 as the cost function: once the HRT applications are schedulable, we are interested to maximize the QoS for the SRT applications.

The alternative solutions provided by PAO are evaluated using a List Scheduling-based heuristic to determine the schedule tables for each SCS application. The worst-case response times for the FPS tasks are determined using a Response Time Analysis that we modified to take into account partitions [4].

V. Case Study

We have evaluated our PAO strategy using an aerospace case study, with two mixed-criticality applications running on a partitioned PE: the mixed-critical Mars Pathfinder Mission [1] (MESUR), and the non-critical controller for the Compositional Infrared Imaging Spectrometer [5] (CIRIS), a Fourier Transform Infrared Spectrometer.

The MESUR tasks are HRT (scheduled with FPS), mixed-critical, with 4 high-criticality (MHC) and 3 low-criticality (MLC) tasks (see Fig. 1a). The CIRIS application is non-critical, SRT scheduled with SCS. The task set is shown Fig. 1b. CIRIS acquires 160 interferograms, which it processes using Fast Fourier Transform (FFT) tasks (fft_i tasks in Fig. 1b). The avg_j, dc_j, cal_j and avg tasks in Fig. 1b are post-processing tasks. A detailed description of the task set can be found in [5]. We have shown in [5] how the number of the acquired and processed interferograms affects the signal to noise performance of the instrument, which is a measure of the QoS. We define the QoS function for CIRIS as the:

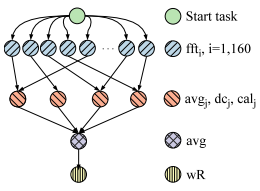
$$QoS(CIRIS) = \frac{executed\ FFT\ tasks}{total\ FFT\ tasks} \quad (2)$$

Thus, a QoS of 1 means that all 160 FFT tasks are executed. A QoS of 0 means that none executed.

We have run our proposed PAO strategy on this case study, and we have obtained the solution depicted in Fig. 2d, where all the hard tasks are schedulable and the QoS for the soft tasks is maximized (QoS=1, corresponding to a high quality signal). Fig 2 presents the partition tables as Gantt charts, with the green partitions corresponding to CIRIS, red partitions

Set	Task name	Priority	Parameters	
			C_i	T_i
1	Bus scheduling	7	25	125
	Data distribution	6	25	125
	Control task	5	25	250
	Radio task	4	25	250
2	Camera task	3	25	250
	Measure task	2	50	5000
	Meteo task	1	75	5000

(a) MESUR task set



(b) CIRIS task graph

Fig. 1: Case study applications

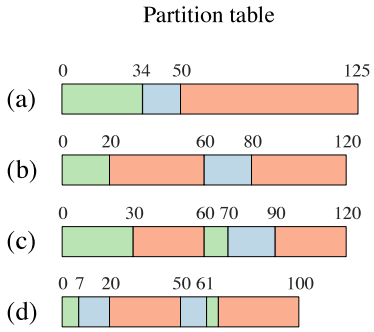


Fig. 2: Partition table configurations

to MHC tasks and blue partitions to MLC tasks. Next to each partition table configuration we present for CIRIS the number of processed interferograms and the resulting QoS, and the number of schedulable MESUR tasks. The figure also shows the initial solution (Fig. 2a) and two intermediate solutions visited during the search. In Fig. 2a all MESUR tasks are schedulable, but CIRIS processes only 4 out of 5 interferograms. In the solution in Fig. 2b CIRIS successfully executes 134 FFT tasks, increasing the QoS to 0.84. In Fig. 2c we are able to increase the QoS to 1 at the expense of 2 MHC tasks missing their deadlines.

This case study shows that it is important to carefully optimize PAs to support soft real-time applications and at the same time meet the stringer constraints of the critical hard real-time applications.

VI. Conclusion

In this paper we have proposed a new Tabu Search-based design optimization strategy for mixed-criticality systems implementing hard and soft real-time applications on the same platform. Our proposed strategy determines an implementation such that all hard real-time applications are schedulable and the quality of service of the soft real-time tasks is maximized. We have evaluated our strategy using an aerospace case study.

References

[1] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri. *Scheduling in Real-Time Systems*. John Wiley & Sons, LTD, 2002.

[2] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[3] J. Rushby. Partitioning for avionics architectures: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999.

[4] D. Tamas-Selicean and P. Pop. Optimization of time-partitions for mixed-criticality real-time distributed embedded systems. *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 1–10, 2011.

[5] D. Tămaş-Selicean, D. Keymeulen, D. Berisford, R. Carlson, K. Hand, P. Pop, W. Wadsworth, and R. Levy. Fourier transform spectrometer controller for partitioned architectures. In *Proc. of the Aerospace Conference*, pages 1–11, 2013.

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

[7] J. Windsor, K. Eckstein, P. Mendham, and T. Pareaud. Time and space partitioning security components for spacecraft flight software. In *Proc. of the Digital Avionics Systems Conference*, pages 8A5–1–8A5–14, 2011.

[8] J. Windsor and K. Hjortnaes. Time and space partitioning in spacecraft avionics. *Proc. of Intl. Conf. on Space Mission Challenged for Information Technology*, pages 13–20, 2009.