Analysis and Optimization of Mixed-Criticality Applications on Partitioned Distributed Architectures

Tamas-Selicean, Domitian; Marinescu, S. O.; Pop, Paul

Published in:
2012 IET System Safety Conference

Link to article, DOI:
10.1049/cp.2012.1504

Publication date:
2012

Citation (APA):
ANALYSIS AND OPTIMIZATION OF MIXED-CRITICALITY APPLICATIONS ON PARTITIONED DISTRIBUTED ARCHITECTURES

D. Tâmas Selicean, S. O. Marinescu and P. Pop
Technical University of Denmark, Department of Informatics, 2800 Kongens Lyngby, Denmark
dota@imm.dtu.dk

Abstract

In this paper we are interested in mixed-criticality applications implemented using distributed heterogeneous architectures, composed of processing elements (PEs) interconnected using the TTEthernet protocol. At the PE-level, we use partitioning, such that each application is allowed to run only within predefined time slots, allocated on each processor. At the communication-level, TTEthernet uses the concepts of virtual links for the separation of mixed-criticality messages. TTEthernet integrates three types of traffic: Time-Triggered (TT) messages, transmitted based on schedule tables, Rate Constrained (RC) messages, transmitted if there are no TT messages, and Best Effort (BE) messages. We assume that applications are scheduled using Static Cyclic Scheduling (SCS) or Fixed-Priority Preemptive Scheduling (FPS). We are interested in analysis and optimization methods and tools, which decide the mapping of tasks to PEs, the sequence and length of the time partitions on each PE and the schedule tables of the SCS tasks and TT messages, such that the applications are schedulable and the response times of FPS tasks and RC messages is minimized. We have proposed a Tabu Search-based meta-heuristic to solve this optimization problem, which has been evaluated using several benchmarks.

1 Introduction

Safety is a property of a system that will not endanger human life or the environment. Safety-Integrity Levels (SILs) capture the required protection against failure when building a safety-critical embedded system, and will dictate the development processes and certification procedures that have to be followed. The current trend is towards integrated architectures, where applications of different safety criticality levels are integrated into the same platform. Such mixed-criticality applications can be integrated onto the same architecture only if there is enough spatial and temporal separation among them.

In this paper we address mixed-criticality hard real-time applications mapped on heterogeneous distributed architectures. We consider that the computation-level temporal and spatial separation mechanisms are provided by “Integrated Modular Avionics” (IMA) [12] through the concept of partitions. Similar separation mechanisms are available in other industries [7], [9]. With IMA, each application runs in a separate partition on the processing elements (PEs) where the application is mapped.

At the communication level, separation is provided by the TTEthernet protocol [2] protocol, which uses the concept of virtual links for the separation of mixed-criticality messages. TTEthernet is a deterministic, synchronized and congestion-free network based on the IEEE 802.3 Ethernet standard and integrates three types of traffic. Time-Triggered (TT) messages, Event-Triggered (ET) messages with bounded end-to-end delay, also called Rate Constrained (RC) messages, and Best-Effort (BE) messages, for which no timing guarantees are given. The TT frames are transmitted based on static schedule tables, and have the highest priority. RC messages are transmitted if there are no TT messages, while BE traffic has the lowest priority.

We assume that applications are scheduled using static-cyclic scheduling (SCS) or fixed-priority scheduling (FPS). Applications scheduled with SCS transmit TT frames, while the ones scheduled with FPS transmit the messages as RC frames. In this paper, we propose a tool for the analysis and optimization of mixed-criticality applications on partitioned architectures. The tool takes as input the model of the application, consisting of tasks and messages, the model of the system platform, and produces that system configuration which guarantees the separation of mixed-criticality applications and their schedulability. The analysis calculates the worst-case response times of FPS tasks and the worst-case transmission times of RC messages, taking into account the particularities of IMA and TTEthernet, respectively. The optimization decides the mapping of tasks to the processing elements (PEs) of the architectures, the sequence and length of the time-partitions on each PE and the schedule tables of the SCS tasks and TT messages, such that the applications are schedulable and, in addition, the response times of FPS tasks and RC messages is minimized.

Researchers have recently started to address the integration of mixed-criticality tasks onto the same platform [4], [14]. In [14], we proposed an optimization approach to determine the mapping of tasks to PEs, the time partitions on each PE
and the schedule tables, such that all applications are schedular and the development costs are minimized. In that work, communications were ignored, and a simple unpartitioned statically scheduled shared bus was used.

2 System Model

The set of all applications in the system is denoted with \( G \). We model an application as a directed, acyclic graph \( G_i(V_i, E_i) \in G \). Each node \( t_j \in V_i \) represents one task. The mapping is denoted by the function \( M: V_i \rightarrow N \), where \( N \) is the set of PEs in the architecture. For each task \( t_j \), we know the worst-case execution time (WCET) \( C_j \) for each PE \( N_j \), where \( t_j \) is considered for mapping. An edge \( e_{jk} \in E_i \) from \( t_j \) to \( t_k \) indicates that the output of \( t_j \) is the input of \( t_k \). Communication between tasks mapped to different PEs is performed by message passing on the TTEthernet network.

Applications are scheduled using SCS or FPS. A deadline \( D_{G_i} \leq T_{G_i} \), where \( T_{G_i} \) is the period of \( G_i \), is imposed on each graph \( G_i \).

An example mixed-criticality system composed of three applications is presented in Fig. 1a. The WCETs of tasks are given in Fig. 1b for two PEs, \( N_1 \) and \( N_2 \). An “x” in the table means that the task is not considered for mapping on the respective PE. The size of the messages is depicted on the graph edges.

2.1 Safety Integrity Levels

During the engineering of a safety-critical system, the hazards are identified and their severity is analyzed, the risks are assessed and the appropriate risk control measures are introduced to reduce the risk to an acceptable level. A Safety-Integrity Level (SIL) captures the required level of risk reduction. SILs are assigned to safety functions, from SIL 4 (most critical) to SIL 0 (non-critical). Functions are decomposed into tasks. We introduce the notation \( SIL: V_i \rightarrow \{SIL \ k \} \), where \( k \in \{0..4\} \), to capture the SIL of a task. The tasks of different SILs have to be separated. Otherwise, for example, a lower-criticality task could write in the code or data area of a higher-criticality task, leading thus to a failure. The same is true for mixed-criticality messages.

2.2 Separation at PE-level

We consider architectures composed of a set \( N \) of PEs connected via TTEthernet. We denote the assignment of tasks to partitions using the function \( \phi: V \rightarrow P \), where \( V \) is the set of tasks in the system and \( P \) is the set of partitions. On a PE \( N_i \), 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 mapped to \( N_i \) and allocated to the partition \( P_j \) are allowed to use \( N_i \). 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 PE. Several MFs are combined together in a system cycle that is repeated periodically, with a period \( T_{cycle} \).

Fig. 2 presents the partitions for 3 applications of different SILs, \( A_1 \), \( A_2 \), and \( A_3 \), implemented on an architecture of 2 PEs, \( N_1 \) and \( N_2 \), with \( T_{MF} = 10 \) and \( T_{cycle} = 2 \times T_{MF} = 20 \).
of a frame \( f_i \) and the given speed of a dataflow link \([v_j, v_k]\) we can determine the transmission duration \( C_{i}^{[v_j, v_k]} \) of \( f_i \) on \([v_j, v_k]\).

### 3 Problem Formulation

The problem we are addressing in this paper is constrained optimization of tasks in EEs and RC frames using the mixed-criticality applications \( A_1, A_2 \) and \( A_3 \) from Fig. 1a, to be implemented on two PEs, \( N_1 \) and \( N_2 \). We have set \( T_{MF} \) to 15 time units and \( T_{cycle} = 2 \times T_{MF} = 30 \). In this example we use a simple static bus.

Let us assume that we have determined the optimal mapping of tasks, in terms of the cost function from Section 4, ignoring the partitioning. Using this optimal mapping, we are interested to obtain the partitions and the schedules, such that, the separations are enforced and the schedule lengths are minimized with the goal of producing a schedulable implementation. This partitioning is presented in Fig. 4a. In this case, although application \( A_3 \) is schedulable, task \( \tau_{15} \) and the second instance
of task $\tau_2$ do not fit into the schedule, and thus applications $A_1$ and $A_2$ are not schedulable.

In this paper we consider that the optimization of mapping and partitioning is done at the same time, and not separately. By deciding simultaneously the mapping and partitioning we have a better chance of obtaining schedulable implementations. Such a solution is depicted in Fig. 4b, where all applications are schedulable. Compared to the solution in Fig. 4a, we have changed the mapping of tasks $\tau_{13}$ and $\tau_{22}$ from $N_1$ to $N_2$ and of task $\tau_2$ from $N_2$ to $N_1$, and we have resized the partition slices and changed the schedule accordingly.

3.2 Optimization of TT Message Schedules

Let us illustrate the TT frame schedule synthesis problem using the setup from Fig. 5, where we have a network composed of three ESes, $ES_1$ to $ES_3$ and a network switch $NS_1$ (see Fig. ??) and three frames, see the table in Fig. ???. The details of the virtual links and frames are in the table and figure. The dataflow links have the same speed. For this example we consider that the RC and TT traffic are integrated using a “timely block” policy, i.e., an RC frame will be delayed if it could block a scheduled TT frame.

We want to determine the TT schedules $S_{TT}$ such that all the TT and RC frames are schedulable. The schedulability of a TT frame $f_i$ is easy to determine: we just have to check the schedules $S_{TT}$. To determine the schedulability of an RC frame $f_j$ we have to compute its worst-case end-to-end delay, from the moment it is sent to the moment it is received. We denote this worst-case delay with $R_{f_j}$. Fig. 5 presents two possible solutions for synthesizing the TT schedules $S_{TT}$. In both cases, Fig. 5a and 5b, we show a Gantt chart, which shows on a timeline from 0 to 600 $\mu$s what happens on the three dataflow links, $[ES_1,NS_1]$, $[ES_2,NS_1]$ and $[NS_1,ES_3]$. Because we are interested in the schedulability of RC frames, for the RC frame $f_1$ we show in both cases (c) and (d) in Fig. 5 the worst-case scenario, i.e., the situation which has generated the largest (worst-case) end-to-end delay. In Fig. 5a the TT schedules are constructed such that the end-to-end delay of TT frames is minimized, i.e., the TT frames arrive at their destination as soon as possible. In this case, the worst-case end-to-end delay of the RC frame $f_1$, namely $R_{f_1}$, is 470 $\mu$s, which is greater than its deadline of 300 $\mu$s, hence $f_1$ is not schedulable. This worst-case for $f_1$ happens for the first frame instance $f_{1,1}$, see Fig. 5a, when $f_{1,1}$ happens to be sent by $ES_1$ at 105 $\mu$s. In this case, as the network implements the timely block integration algorithm, the frame cannot be forwarded by $NS_1$ to $ES_3$ until there is a big enough time interval to transmit the frame without disturbing the scheduled TT frames. We denote these “blocked” time intervals with hatched boxes. The first big enough interval starts only at time 500, right after $f_{2,3}$ is received by $ES_3$, which is too late.

4 Mixed-Criticality System Optimization

The problem presented in the previous section is NP-complete, hence we use a meta-heuristic called Tabu Search (TS) to solve it. Starting from an initial solution, we iteratively run (i) a TS meta-heuristic to optimize the task mapping and the set of partition slices, while considering the schedules for TT frames to be fixed, and (ii) a TS-based TT frames schedule optimization algorithm considering the previously obtained task mapping and task partition slices as fixed.

4.1 Tabu Search

TS [8] is a meta-heuristic optimization, which searches for that solution which minimizes the cost function of an imple-
mentation $\Psi$:

$$\text{Cost}(\Psi) = \begin{cases} c_1 = \sum_{i} \max(0, R_i - D_i) & \text{if } c_1 > 0 \\ c_2 = \sum_{i} \max(R_i - D_i) & \text{if } c_1 = 0 \end{cases} \quad (1)$$

$R_i$ is the response time of the application, while $D_i$ is the deadline of the application. If at least one application is not schedulable, there exists one $R_i$ greater than the deadline $D_i$, and therefore the term $c_1$ will be positive. However, if all the applications are schedulable, this means that 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, since the applications are schedulable, we are interested to minimize response time of the applications.

The worst-case response times for SCS tasks and TT messages are determined based on the schedule tables, while those of FPS tasks and RC messages are calculated using the analysis from Section 4.2.

Tabu Search explores the design space by using design transformations (or “moves”) applied to the current solution in order to generate neighboring solutions. To escape local minima, TS incorporates an adaptive memory (called “Tabu list”), to prevent the search from revisiting previous solutions, thus avoiding cycling.

At the PE level, we use one re-assignment move, which changes the assignment of a task to another partition and four types of moves applied to partition slices: resize, swap, join and split. The re-assignment move re-assigns a task to another, randomly chosen, partition. The partition can be on another PE, thus, implicitly, the re-assignment move will also re-map the task. The resize move, resizes the selected partition slice. The swap move swaps the chosen partition slice with another randomly chosen partition slice. The join move joins two partition slices belonging to the same application, while the split move splits a partition slice into two, and adds the second slice to the end of the MF.

For the TT frames, we use four types of moves: advance, advance predecessors, postpone and postpone successors. The advance move will advance the scheduled send time of a TT frame instance $f_{1,i}$ from a node $v_j$ on a dataflow link $[v_j, v_k]$ to an earlier moment in time. The advance predecessors applied to a frame instance $f_{1,i}[v_j, v_k]$, will advance the scheduled send time for all its predecessors. Similarly, the postpone move will postpone the scheduled send time of a TT frame instance from a node, while postpone successors will postpone the send time for all the successors of that frame instance.

4.2 Analysis of Mixed-Criticality Applications

To determine the schedulability of FPS tasks we use a response-time analysis [5] to calculate the worst-case response time $R_i$ of every FPS task $t_i$, which is compared to its deadline $D_i$. Audsley and Wellings [3] have proposed a schedulability analysis for FPS tasks using temporal partitioning (IMA), 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. In this paper we use the analysis proposed by us in [10] for FPS tasks.

The worst-case end-to-end delay $R_i$ of an RC frame $f_i \in F^{RC}$ sent on a virtual link $v[L_i] = M(f_i)$ is the sum of the worst-case queueing delays $Q^{v[L_i]}_{f_i}$ on each network node (ES or NS) $v_j \in \Psi$ which is the source of a dataflow link $[v_j, v_k] \in v[L_i]$ and the transmission duration $C^{v[L_i]}_{f_i}$ for each dataflow link $[v_j, v_k] \in v[L_i]$ the frame transits. For RC frames, we use the analysis from [13].

5 Experimental Results

Regarding tasks, we were interested to evaluate the proposed optimization strategy (TO) in terms of its ability to find schedulable implementations. Thus, we have used 5 synthetic benchmarks with 4 to 6 mixed-criticality applications mapped on 2 to 6 PEs. Together with TO, Table I also presented the results obtained using a Straightforward Solution for tasks (SST). SST consists of a simple straightforward partitioning scheme which allocated for each application $A_j$ a total time on PE $i$ proportional to the utilization of the tasks of $A_j$ mapped on $N_i$. The partitions $P_j$ thus allocated have the same length and they are distributed with a period equal to the smallest period of a task from $A_j$ mapped on $N_i$. Column 2 in Table I presents the number of tasks. The number of schedulable tasks (out of the total) obtained by our proposed TO strategy is presented in column 5, while column 4 presents the results obtained using SST.

As we can see from “Set 1”, SST which does not perform optimization, is not able to find schedulable implementations. However, by applying our proposed TO approach, we are able to optimize the time partitions such that all applications are schedulable. We have measured the ability of TO to improve SST by using a percentage average increase in the degree of schedulability over all applications, presented in the last column. As we can see there is a dramatic increase over all applications, when using TO. This means that we can potentially implement the applications on a slower (cheaper) architecture.

We have also used 2 real life benchmarks derived from the Embedded Systems Synthesis Benchmarks Suite (E3S) version 0.9 [6], see “Set 2”. We have used the telecom-mocyn and auto-indust-cows benchmarks. The applications are mapped on an architecture of 3 PEs. The results obtained from these real-life benchmarks confirm the results of the synthetic benchmarks.
The results related to messages are presented in Table II. For the synthetic benchmarks, we have used 6 network topologies, and we have randomly generated the parameters for the frames, taking into account the details of the TTEthernet protocol. For all experiments, we have compared our optimization strategy for messages (let us call it TM) with a baseline solution, namely the Straightforward Solution for messages (SSM), which builds the TT schedules with the goal of minimizing the end-to-end response time of the TT frames without considering the RC traffic. The comparison between SSM and TM, $\Delta_{\text{cost}}$, is shown in the last column in the table as a percentage improvement of TM over SSM.

In the sets of experiments labeled “Set 1” and “Set 2” in Table II, we were interested to evaluate the quality of the result obtained with TM as the size of the system increases. Thus, we have used 7 synthetic benchmarks, with the number of network nodes ranging between 16 and 55 nodes. The first set of 4 benchmarks have a load of 50%, and the second set of benchmark have a load of 70%. As we can see, TM is able to significantly improve the cost function over SSM, even as the size of the system increases. We used a time limit of 45 minutes for the first set and 90 minutes for the second set.

Finally, we used one real-life benchmark derived from [11], based on the SAE automotive communication benchmark [1]. In this benchmark we have 18 network nodes (ESes and NSes), and 83 frames (with the parameters) generated based on the messages presented in [11]). Table II contained the results for this benchmark – the last line labeled with “Set 3”. The results obtained for the real-life benchmark confirms the results of the synthetic benchmarks.

6 Conclusions

We have presented an approach to the analysis and optimization of mixed-criticality applications on partitioned architectures. Applications of different criticality levels can be integrated onto the same architecture only if there is enough spatial and temporal separation among them. We have considered IMA for the PE-level Separation and TTEthernet for the communication-level. As the experimental evaluations shows, only by optimizing the implementation of the applications, taking into account the particularities of IMA and TTEthernet, we are able to support the designer in obtaining schedulable implementations.

Acknowledgements

This work has been funded by the Advanced Research & Technology for Embedded Intelligence and Systems (ARTEMIS) within the project ‘RECOMP’, support code 01IS10001A, agreement no. 100202.

References


\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{Set} & \textbf{Test case} & \textbf{ES} & \textbf{NS} & \textbf{Messages} & \textbf{Frame instances} & \textbf{\Delta_{\text{cost}} [\%]} \\
\hline
1 & 11 & 13 & 4 & 80 & 1593 & 2.58 \\
& 12 & 25 & 6 & 88 & 1787 & 24.44 \\
& 13 & 35 & 8 & 103 & 2285 & 20.06 \\
& 14 & 45 & 10 & 165 & 3299 & 11.90 \\
\hline
2 & 21 & 11 & 4 & 115 & 16904 & 9.17 \\
& 22 & 25 & 6 & 179 & 2525 & 20.61 \\
& 23 & 35 & 8 & 154 & 3698 & 39.34 \\
\hline
3 & automotive & 10 & 3 & 170 & 38305 & 50.88 \\
\hline
\end{tabular}
\caption{Experimental results for tasks}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{Set} & \textbf{Tasks} & \textbf{PES} & \textbf{SSM Sched.} & \textbf{IO Sched.} & \textbf{avg. % increase in $\delta$} \\
\hline
1 & 20 & 2 & 10 & All & 832.88 \\
& 26 & 3 & 13 & All & 27.36 \\
& 40 & 4 & 6 & All & 88.41 \\
& 50 & 5 & 10 & All & 73.57 \\
& 62 & 6 & 26 & All & 278.72 \\
\hline
2 & 24 & 3 & 5 & All & 113.95 \\
& 25 & 3 & 7 & All & 61.87 \\
\hline
\end{tabular}
\caption{Experimental results for messages}
\end{table}