Timing Predictability for Resource Sharing Multicore Systems - Challenges and Open Problems

Andreas Schranzhofer Jian-Jia Chen (Speaker)[†] Lothar Thiele
Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland
Email: {schranzhofer,thiele}@tik.ee.ethz.ch
[†] Department of Informatics, Karlsruhe Institute of Technology (KIT), Germany
[†] Email: jian-jia.chen@kit.edu

1 Introduction

In modern computer systems, multiprocessor systems on chip (MPSoCs) and multicore platforms have been widely applied. These commercial off-the-shelf (COTS) products possess significantly increased computational performance in the average case and reduced cost and time-to-market properties that are increasingly appealing for applications in the avionic and automotive industry. Demand for computational resources is growing with the ascend of concepts such as fly-by-wire and assisted driving. However, such concepts rely on guarantees on the worst-case response time (WCRT). In addition, MPSoCs employ shared resources, such as memory or I/O peripherals, for performing communication and data exchange between cores. Multiple processing elements competing for access to a shared resource yields contention and as a result, significantly increased WCRT.

MPSoCs with shared resources have been studied by Schliecker et al. [3] using event models and an iterative approach to estimate the worst-case execution time (WCET). Pellizzoni et al. [1] have studied systems with shared resources and proposed methods to analyze the worst-case delay. In [1], we propose sequentially executing superblocks to constitute tasks. Superblocks are specified by their upper bound on access requests to a shared memory and their maximum required computation time. Superblocks are constructed by execution blocks either by static analysis of a program or by the designers efforts to manually arrange memory accesses at the beginning and the end of the execution blocks. Different arbitration policies on a shared memory (FP, FCFS, RR) are analyzed and the worst-case delay suffered by a task due to the interference on the shared memory is computed, using a dynamic programming approach.

Other works focus on arbitration policies, that eliminate interference. Rosen et al. [2] use Time Division Multiple Access (TDMA) for accessing a bus. A task is modeled as dedicated communication phase at its beginning and end, and an execution phase in between. Static analysis is used to derive all feasible execution traces of a task and to derive the WCET thereof. We propose a task model in [4], where the number of accesses to a shared resource is only known as an upper bound for different time windows. The analysis of the WCRT is based on maximizing the cumulative time a task waits for time slots. Besides the task model with dedicated phases for communication and execution, our algorithm in [4] is also applicable to tasks modeled as a single general phase, without dedicated communication. Following that, we introduce different models of accessing shared resources and models of execution in [5]. As a conclusion, separation of communication and execution is crucial for designing multicore resource sharing systems, but excessive time-triggering increases their WCRT. Moreover, for systems with an adaptive arbiter on the shared resource, i.e., an arbiter that is composed of a static and a dynamic arbitration segment, we show how to derive a WCRT bound by using dynamic programming in [6].

2 Challenges and Open Problems

The problem of deriving the WCRT is hard and as shown in our previous work, so far tight results can only be derived for very restrictive task models, see [4, 5, 6]. In this section, we would like to provide the challenges we have faced for analyzing the worst-case response time and some possible solutions. Deriving the worst-case trace is not trivial when a pending access to the shared resource blocks the execution on the processing element. Consider a multicore platform with multiple tasks competing for a shared resource, and no synchronization among those tasks. Furthermore, consider another task, whose WCRT shall be computed. Then it is not clear, which interference pattern by other tasks results in the WCRT of the task under analysis. Even though the results we have derived in [4, 5, 6] are safe for bounding the worst-case response time, the tightness of the analysis is still questionable.

In the case of static arbitration, when interference is eliminated, this problem is avoided and the worst-case response time is derived by maximizing the stall time in between time slots. In the other case, when arbitration on the shared resource follows a dynamic scheme, deriving the worst-case interference is a major issue. Our work in [1] and the work by Schliecker et al. in [3], approximate the interference as the sum of possible resource accesses by other tasks in a particular time window. This is pessimistic, since these tasks can also be interfered with and therefore the actual interference might be much smaller. So far, a tight algorithm would require to produce all possible interfering traces, which is of exponential complexity. Limiting the number of interfering traces is one potential direction of research to compute tight WCRTs for systems with shared resources and dynamic arbitration policies thereon. This is achieved by assuming not only an upper bound on resource accesses for a task or superblock, but also a lower bound. This way, the number of feasible traces is reduced significantly, since the freedom to postpone resource accesses is restricted. A major issue for this approach is the derivation of an aggregated interference for the task under analysis, which has to be derived from the access patterns of all the other tasks.

Another line of research follows a similar approach as our work in [4, 5], namely the elimination of interference by isolating the accesses by multiple tasks to the shared resource from each other. This can be achieved by using servers to arbitrate the shared resource. Constant Bandwidth Server (CBS) and Total Bandwidth Server (TBS) can be reduced to a TDMA arbitration, in case interference can be assumed to be present at all times. Otherwise, these arbitration policies result in lower WCRTs. Besides applying these server arbitration policies, it is unclear how an optimal server would look like. How a server should be constructed, such that a minimized (optimal) WCRT can be achieved, is an open question. Another issues is the combined optimality criterion for multiple servers, serving multiple tasks. Satisfying real-time guarantees is a condition that has to be satisfied for feasibility, and therefore cannot serve as optimality criterion. This criterion has to consider properties such as WCRT or a combination of the WCRTs for multiple tasks in relation to their respective deadline.

All these approaches consider a given task to processing element allocation. However, taking interference between tasks into consideration during the mapping process, opens another possible direction of research for designing predictable resource sharing multicore systems. As an example, consider a set of tasks that communicate over a shared memory. Distributing these tasks over a large amount of processing elements results in a lot of communication requests over the shared memory. Concentrating these tasks on as few processing elements as possible, on the other hand, reduces the amount of communication over the shared resource. Conclusively the amount of potential interferences is reduced as well.

3 Conclusion

Contention on shared resources results in significantly increased worst-case response time (WCRT) guarantees, which are required for automotive and avionic applications. We show in our work that for general models of execution and general arbitration policies, this problem is hard and upper bounds on the WCRT are not tight. Using more restrictive models of execution, along with a static arbitration on the shared resource, we derive an efficient algorithm to compute a tight bound on the WCRT. We propose new analysis approaches to determine the worst-case interference by enriching the description of tasks with a lower bound on resource access requests. Then we propose the usage of servers to eliminate interference among tasks. Considering interference in the design process, when deciding on the mapping, is a viable way to limit interference and to eventually obtain a predictable resource sharing MPSoC.

References

[1] R. Pellizzoni, A. Schranzhofer, J.-J. Chen, M. Caccamo, and L. Thiele. Worst case delay analysis for memory interference in multicore systems. In *Proceedings of DATE*, pages 741–750, 2010.

[2] J. Rosen, A. Andrei, P. Eles, and Z. Peng. Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip. In *RTSS*, pages 49–60, 2007.

[3] S. Schliecker, M. Negrean, and R. Ernst. Bounding the shared resource load for the performance analysis of multiprocessor systems. In *DATE*, pages 759–764, 2010.

[4] A. Schranzhofer, J.-J. Chen, and L. Thiele. Timing analysis for tdma arbitration in resource sharing systems. In *Proceedings of RTAS*, pages 215–224, 2010.

[5] A. Schranzhofer, R. Pellizzoni, J.-J. Chen, L. Thiele, and M. Caccamo. Worst-case response time analysis of resource access models in multi-core systems. In *Proceedings of DAC*, 2010.

[6] A. Schranzhofer, R. Pellizzoni, J.-J. Chen, L. Thiele, and M. Caccamo. Timing analysis for resource access interference on adaptive resource arbiters. In *Proceedings of RTAS*, April 2011.