Optimiertes Buffering für zeitgesteuerte automobile Software

Dr. Eugene Yip und Prof. Dr. Gerald Lüttgen, Universität Bamberg
Dr. Michael Deubzer, Timing-Architects Embedded Systems GmbH, Regensburg

14. November 2016

1 State-of-the-Art

Project context. The development of an automotive system involves the integration [19] of many real-time software functionalities, and it is of utmost importance to guarantee strict timing requirements. An important type of requirement, called *end-to-end response time*, specifies the maximum time that the system can take to generate an output to a corresponding input. Such timing requirements are easier to guarantee with time-triggered implementations because they offer better time-predictability than their event-triggered counterparts [17]. However, the recent trend towards multi-core architectures poses significant challenges for the timely transfer of signals and data between processor cores so as not to violate data consistency.

In this light, automotive manufacturers [12, 20] have shown interest in using the Logical Execution Time (LET) programming model [15] for designing time-triggered, multi-core systems. A LET task has a statically defined period and block of time, called the *logical execution time*, during which the task is allowed to execute its computations. The task's inputs are read at the start of the LET, and the task's outputs are written at the end of the LET. Data buffering is needed to preserve the data-flow between tasks [10], especially when the timings of task inputs and outputs do not align. Thus, a significant portion of end-to-end response times may be spent accessing and managing the buffers, and a significant amount of memory may be allocated to the buffers [10]. The restriction of task communications to predefined time points enables time-predicable behaviour across platforms [13], and provides the opportunity to statically optimise task communications.

Despite the increasing interest in the LET-based time-triggered approach, the event-triggered approach remains popular because of its ability to achieve better average-case response times and resource utilisation [17]. To improve the practicality of the time-triggered approach for multi-core systems, this project shall investigate the opportunity to employ modern techniques to optimise the buffering overheads of LET tasks. There is much freedom to do so since conventional event-triggered automotive implementations use periodic tasks that are free to communicate with other tasks at any time during their execution. However, adapting buffering optimisations for periodic tasks to LET tasks is by no means straightforward, because consideration must be given to the fact that LET task communications only occur at predefined time points. Thus, the optimisations must be cognisant of worst-case timings to ensure that the LET timing semantics are always respected at runtime, which can be checked with timing analysis.

Simulation-based timing analysis tools, such as the Timing-Architects Tool Suite [25],

allow run-time behaviours of event-triggered automotive designs to be quickly analysed at reasonable accuracy. For worst-case guarantees, the Symtavision SymTA/S [24] tool performs statistical and scenario-based timing analysis. Both tools specialise in analysing multi-core automotive designs based on the AUTOSAR [2] standard: an AUTOSAR compliant design consists of software components that contain a set of runnables that each implement a specific functionality. An efficient implementation requires the careful mapping of runnables to tasks and to processor cores, while preserving the data-flow. The proposed project focusses on multi-core AUTOSAR-based designs, but the project results are also relevant to avionic and industrial automation systems.

Project objectives. The objective of this project is to develop buffering optimisation algorithms and heuristics that reduce the memory consumption, processor utilisation, and end-to-end response times of time-triggered AUTOSAR designs on multi-core or multi-processor platforms. The algorithms and heuristics select the buffering mechanism to synthesise for each data signal, and the mapping and execution order of runnables on each processor core. The objective is important because it would allow time-triggered implementations to become competitive alternatives to their event-triggered counterparts. Timing-Architects Embedded Systems GmbH will cooperate to apply the developed algorithms and heuristics into a demonstrator tool to enable their evaluation on industrial automotive software.

The remainder of this section reviews the optimisations developed for single and multi-core AUTOSAR designs with periodic tasks. Next, optimisations for LET implementations are reviewed in relation to the project objective. Finally, buffering optimisations for the related synchronous-reactive programming model are highlighted.

Related work on optimising AUTOSAR designs for single-core processors. Studies have investigated the minimisation of end-to-end response times and memory consumption on single-core processors [10, 30, 31]. For end-to-end response times, a common strategy is to reorder the runnable execution orders to allow data to be produced earlier, and to regroup runnables into the same task if they frequently communicate together. Tasks on critical response paths can be reprioritised to reduce time delays caused by task preemptions. There are two main approaches to efficient buffering [30]. The first is the *centralised* approach, where the sending task writes to a global buffer that the reading tasks proactively read from at correct times. Although this approach could be memory efficient, mutual exclusion is required for consistent reading and writing. The second is the *decentralised* approach, where the sending task proactively writes to the local buffers of each receiving task. Alternatively, the write buffer of the sending task can be swapped with the read buffer of the receiving tasks at correct times. Although this approach could be time efficient, more memory would be required.

Related work on optimising AUTOSAR designs for multi-core and multi-processors. Some studies have been performed on minimising the end-to-end response times of automotive software on multi-core processor architectures. Faragardi et al. [7] minimise the overall runnable communication time by finding good mappings of runnables and tasks to processor cores, and finding tasks that can be merged. However, no attempt is made to minimise memory consumption. Different communication times are considered for runnables in different tasks or on different cores. A heuristic based on simulated annealing [8] is used to solve the minimisation problem within a reasonable time frame. Wozniak et al. [26] minimise end-to-end

response times and memory consumption on heterogeneous processors by selecting whether to use a time-consuming or a memory-consuming buffering mechanism for each signal [30]. An exact method using mixed integer linear programming (MILP [4]) and a heuristic based on a genetic algorithm [11] are proposed to solve the minimisation problem. Unlike Faragardi et al. [7], non-uniform memory access times are not considered. Saidi et al. [21] only minimise inter-core communication, while statically balancing the processor loads. Zhang et al. [32] focus on the design-space exploration problem of deploying new software components onto a legacy AUTOSAR system. Interestingly, consideration is given to whether a processor's memory capacity is sufficient for its mapped runnables. However, buffering requirements are not considered and runnables are assumed to have predefined priorities.

Related work on optimising LET programs. LET programs are typically compiled [13] for execution on a virtual *embedded machine*, which makes performance-related optimisations difficult to apply. Resmerita et al. [20] analyse the signals that actually need buffering to preserve the LET semantics, and identify tasks with non-overlapping LETs that can reuse the buffer of a common signal. Farcas et al. [9] observe that a task's output does not need to be sent if it will be overwritten by a fresher output by the time the receiving task starts its LET. Moreover, data does not need to be sent instantaneously, so long as the data arrives by the time the receiving task starts its LET. However, LET programs can suffer from long end-to-end response times because task outputs are only available at the end of their LET. Bradatsch et al. [6] perform response time analysis to find tasks that can have shorter LETs compared to their periods. This results in shorter average-case end-to-end response times, but the worst-case remains unchanged. Kluge et al. [16] propose a LET implementation for a physical time-predictable multi-core platform, but optimisations are not discussed.

Related work on optimising the buffers of synchronous-reactive tasks. In contrast to LET tasks, where outputs are expected after a fixed time delay, the outputs of synchronous-reactive tasks [5] are assumed to be produced instantaneously when inputs arrive. However, in any real implementation, tasks need time to compute their outputs. Thus, buffering is needed to ensure that tasks will read from the correct output instances [18, 23]. Sofronis et al. [23] propose a dynamic buffering mechanism that is memory optimal for the worst-case, in the sense that only the output instances needed for semantics preservation are buffered. The tasks manipulate pointers at run-time to manage their buffers. Buffer memories are reused if they are no longer needed by their respective signals. Natale et al. [18] compute tighter bounds for buffer sizes by determining the maximum number of concurrent readers and for how long the readers need each signal value. Notably, these studies have not considered how a buffering mechanism influences end-to-end response time.

2 Preliminary Work

Our preliminary work includes the development of ForeC [27, 29], a synchronous programming language for multi-cores, and an associated static timing analysis tool [29]. This work has been extended by our ForeMC framework [28] with the development of a buffered communication model and a time-triggered scheduling approach for multi-rate synchronous tasks. Finally, Timing-Architects Embedded Systems GmbH has developed a demonstrator tool for the modelling and simulation of basic time-triggered LET systems.

The ForeC language and static timing analysis. We developed a C-based synchronous language, called ForeC [27, 29], for the deterministic parallel programming of embedded multi-core platforms. Task communication is via shared variables, where tasks read and write to local copies of the shared variables. When the program completes its reaction, the local copies are combined together with programmer-defined functions. The process of copying and combining the shared variables is equivalent to a buffering mechanism. Although optimisations for memory consumption and end-to-end response times were not considered, a precise static timing analysis method [29] was developed to guarantee time-predicable execution. This required a detailed understanding of how task-to-core mappings and variable-to-memory mappings influence a program's execution time.

The ForeMC framework. Our ForeMC framework [28] explores the scheduling of multi-rate synchronous tasks on homogeneous multi-processor platforms for maximum processor utilisation. ForeMC tasks are a special case of LET tasks, because inputs are only read at the start of the period and outputs are only written at the end of the period. However, ForeMC extends LET tasks with bounded task periods to support the notion of *timing criticality*. The challenge is to schedule mixed-criticality tasks on the same platform such that all tasks can complete their computations within their period bounds. In ForeMC, a static schedule is constructed using integer linear programming (ILP) to ensure that the critical tasks always have time to complete their computations. At runtime, processor slack is utilised to dynamically execute the less critical tasks. ForeMC uses *lossless* buffering for multi-rate task communications, where all outputs from a faster task are buffered for a slower receiving task. When the slower task starts its period, it consumes all the buffered values and clears the buffer. The maximum buffer size of a signal is bounded by the period bounds of the communicating tasks. However, no buffering optimisations were proposed.

Timing-Architects Tool Suite. A basic demonstrator tool for simulating LET-based time-triggered systems is being developed by Timing-Architects, in which the University of Bamberg is a consultant. The demonstrator would allow time-triggered AUTOSAR runnables to be manually created and grouped into LET tasks, and manually mapped to an available core. The demonstrator would only be able to simulate the timing behaviour of time-triggered systems under the base period or hyper period scheduling approaches. Timing-Architect's Tool Suite [25] has the capability to optimise the event-triggered design parameters, but investigation is needed to determine their applicability to LET-based systems.

In a separate internal project, Timing-Architects is implementing an algorithm into their Tool Suite that determines whether or not a signal needs buffering to maintain data consistency. However, the algorithm is unable to select buffering mechanisms.

Recent publications

[27] **Eugene Yip**, Alain Girault, Partha S. Roop, and Morteza Biglari-Abhari. The ForeC Synchronous Deterministic Parallel Programming Language for Multicores. In *10th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 297–304, IEEE, September 2016.

[28] **Eugene Yip**, Matthew M. Y. Kuo, Partha S. Roop, and David Broman. Relaxing the Synchronous Approach for Mixed-Criticality Systems. In *20th IEEE Real-Time and*

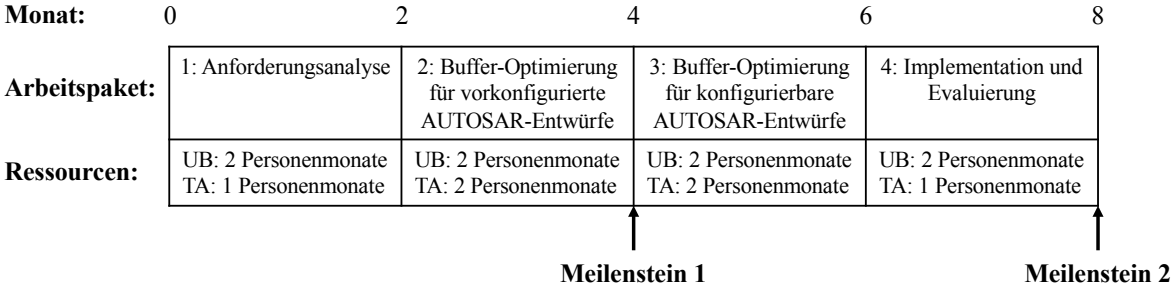

Abbildung 1: Projektübersicht.

Embedded Technology and Application Symposium (RTAS), pages 89–100. IEEE, April 2014.

[**29**] **Eugene Yip**, Partha S. Roop, Morteza Biglari-Abhari, and Alain Girault. Programming and Timing Analysis of Parallel Programs on Multicores. In *13th International Conference on Application of Concurrency to System Design (ACSD)*, pages 160–169. IEEE, July 2013.

3 Scientific and Technical Project Description

The proposed project involves four work packages to develop and evaluate algorithms and heuristics that reduce the memory consumption, processor utilisation, and end-to-end response times of time-triggered designs on multi-core or multi-processor platforms. The project scope is limited to AUTOSAR compliant designs that are based on the LET task model. The work packages are scheduled sequentially as shown in Figure 1, with personnel resources from the University of Bamberg (UB) and Timing-Architects (TA) indicated below each work package. The project has two milestones that coincide with the completions of the second and fourth work packages.

The **first work package** identifies buffering mechanisms that are suited to different time-triggered system configurations. The **second work package** develops an algorithm and heuristics to determine which buffering mechanism to implement for a signal, assuming that the user defines the runnable-to-task mappings, task-to-core mappings, and runnable execution orders; the user controls the optimisation objective by selecting a combination of the following optimisation criteria: processor utilisation, memory consumption, and end-to-end response times. The **third work package** relaxes the assumptions to develop an enhanced algorithm and heuristics to adjust the runnable and task mappings to find better solutions. The **fourth work package** evaluates the performance of the algorithms and heuristics, developed in the second and third work packages, on a collection of benchmarks that are representative of automotive software. The majority of the work will be conducted at the University of Bamberg. Timing-Architects will provide technical feedback and guidance on the practical relevance of the developed solutions from an industrial standpoint, and cooperate to apply the algorithms and heuristics into a demonstrator tool to enable their evaluation on the automotive benchmarks.

Work package 1: Requirements definition. The project begins with a detailed study of the centralised and decentralised buffering mechanisms proposed in the literature. The

mechanisms may have static or dynamic implementations where the allocation of memory and buffering decisions are determined at design time, whereas details are postponed until runtime for dynamic implementations. A detailed categorisation and analysis of the mechanisms shall be performed to identify buffering mechanisms that have complementary time and memory efficiency trade-offs for LET-based tasks. Software and hardware configuration parameters that have significant influence on buffering performance will be identified.

Work package 2: Buffering optimisation for pre-configured AUTOSAR designs. An algorithm and heuristics for selecting good buffering mechanisms, from those identified in the first work package, shall be developed for LET-based AUTOSAR designs. We assume that runnable-to-task mappings, task-to-core mappings, and runnable execution orders are pre-configured by the user. This leaves the following parameters for the algorithm to set: the timing of buffering operations, the memory locations of buffers, the use of centralised or decentralised buffering mechanisms, and the use of static or dynamic implementations.

The optimisation objective is to minimise the processor utilisation, memory consumption, end-to-end response times, or a combination of these as decided by the user. The data-flow between runnables must be preserved during optimisation. The initial algorithm and heuristics will assume a platform with homogeneous multi-core processors, unlimited local and global memories, and constant memory access times. The algorithm will be enhanced by progressively relaxing each assumption and studying their effects on the performance of the buffering mechanisms.

To manage the anticipated large problem space, heuristics such as simulated annealing [8] or a genetic algorithm [11] shall be employed to find good approximate solutions. If the identified task and platform parameters are insufficient for finding good solutions, then the parameters will be reprioritised and reweighed, and algorithm and heuristics will be reformulated and re-evaluated accordingly.

Work package 3: Buffering optimisation for reconfigurable AUTOSAR designs. The algorithm and heuristics of the second work package shall be enhanced with the ability to reconfigure the runnable-to-task mappings, task-to-core mappings, and runnable execution orders of LET-based AUTOSAR designs to find better solutions. For example, it is better to map runnables to the same task if they communicate frequently with each other, so as to avoid unnecessary delays due to inter-task communication. Conversely, it is better to partition runnables to separate cores for shorter end-to-end response times due to parallel execution and to increase task schedulability.

By allowing mappings to be changed, the problem space becomes much larger, and advanced heuristics shall be used to find good solutions within reasonable time frames. A sensitivity study will be performed to identify the important model parameters of problematic AUTOSAR designs. Several heuristics will be employed together to find runnable and task mappings, runnable execution orders, and to select buffering mechanisms. This will be complemented with an iterative process that refines the mappings and the selection of buffering mechanisms.

Work package 4: Implementation and evaluation. The algorithms and heuristics developed in the second and third work packages shall be evaluated on a set of AUTOSAR benchmarks to compare the following metrics of event and time-triggered implementations: memory consumption, processor utilisation, and end-to-end response times. Timing-Architects will cooperate to guide the development of a set of event-triggered AUTOSAR benchmarks,

containing realistic design patterns found in automotive software, from which time-triggered versions will be created. The scalability (e.g., analysis time and memory consumption) and the quality of solutions (e.g., optimality) are also evaluated. In addition, an industrial case study will be performed on the embedded engine system of the FMTV Challenge [3]. Design variations of the case study will be created to include common design patterns that it does not already contain.

To enable the timely evaluation of the algorithms and heuristics developed in the second and third work packages, Timing-Architects will cooperate to apply the devised algorithms and heuristics into a demonstrator tool. The findings of the project are expected to be disseminated as a conference paper, with details being included in a technical report to be made publicly available.

4 Innovation and Goals of the Project

Automotive manufacturers are turning to time-triggered systems for time-predictable input and output behaviour. The LET programming model is seen as a promising approach, but the objective of most LET implementation studies has been on preserving time predictability. The proposed development of algorithms and heuristics for optimising the buffering of LET-based systems would assist automotive manufacturers to search the design space for high performance time-triggered implementations. This reduces development time and costs and allows time-triggered implementations to become practical alternatives to their event-triggered counterparts. The results of this project are not limited to AUTOSAR and are relevant to the aviation or industrial automation domains. For example, the ARINC 653 [1] and IEC 61499 [14] standards for designing aviation and industrial automation systems, respectively, prescribe software architectures that are similar to that of AUTOSAR. Time-triggered implementations are not uncommon [22, 33] for both standards.

One goal of this project is to investigate the circumstances in which static buffering mechanisms for LET programs are preferred over their dynamic counterparts, and vice-versa. Static mechanisms facilitate static analysis because all buffering decisions, such as the location of buffers and the timing of buffering operations, are determined at design time. Moreover, bus traffic due to buffering can be managed by creating a static bus schedule that guarantees the absence of timing failures at runtime. Static mechanisms generally require longer compilation times because all possible execution scenarios must be examined. By contrast, dynamic mechanisms generally have longer runtime overheads because all buffering decisions are made at runtime. However, better decisions are possible because the system's current state is known. Thus, the challenge is to determine when the overhead of a buffering mechanism begins to outweigh its benefits.

Another goal of this project is to propose a clever analysis of automotive software to optimise LET implementations in terms of memory consumption, processor utilisation, and end-to-end response times. None of these optimisations has been considered deeply by the state-of-the-art with respect to the LET programming model, especially in the automotive domain. Although clever buffering mechanisms have been developed for synchronous-reactive programs, the timing constraints of LET programs have to be carefully considered before they can be applied. End-to-end response times could be reduced by modifying LET task parameters, but the effects of runnable ordering and grouping have not been studied. Because most studies have focussed on the implementation of LET programs on the virtual embedded

machine [13], no optimisation strategies have been developed to take inter-core and inter-processor communication costs, and memory sizes and hierarchies into account. Moreover, the utilisation of heterogeneous processors and their effect on task execution times have not been studied for LET programs.

5 Economic Feasibility

Timing-Architects Embedded Sytems GmbH ist auf die Evaluierung und Optimierung von eingebetteten Multicore-Prozessor Echtzeitsystemen spezialisiert und bietet hierzu ein umfassendes Lösungspaket, bestehend aus Softwarewerkzeugen (TA Tool Suite) und Dienstleistungen (Consulting und Training), an. Aktuell sind in der Automobilbranche im Multicore-Kontext ereignisgesteuerte Echtzeitsysteme vorherrschend. Jedoch werden dort aufgrund des wachsenden Funktionsumfangs wie bspw. der Umfelderkennung beim autonomen Fahren, Ansätze mit zeitgesteuerten Systemen untersucht, da deren Zeitverhalten oft besser vorhersagbar ist.

Um die wirtschaftliche Anschlussfähigkeit nicht zu verlieren und weiter als Innovations- und Technologieführer zu bestehen, ist es für Timing-Architects von größter Bedeutung, sich im Bereich zeitgesteuerter Echtzeitsysteme einen technologischen Vorsprung zu erarbeiten. Hierbei spielen Buffering-Mechanismen eine wichtige Rolle, um den Speicherbedarf sowie die Prozessorauslastung auf ein Minimum zu reduzieren und anwendungsspezifisch zu optimieren; diese werden in der Automobilbranche bereits angefragt. Zudem arbeitet Timing-Architects mit Hochdruck daran, auch weitere Anwendungsdomänen wie bspw. Avionik, Automation oder Robotik zu erschließen und diese mit ihren Tools in der Softwareentwicklung zu unterstützen.

Die Ergebnisse des Vorhabens sollen zeitnah — etwa ein Jahr nach Projektabschluss — in die Produktlinie von Timing-Architects einfließen. Damit können die zeit- und ereignisgesteuerten Ansätze untereinander verglichen und dem Kunden somit eine Alternative der Softwareimplementierung zur Verfügung gestellt werden.

6 Patent Situation

Es sind den am Antrag Beteiligten keinerlei Schutzrechte und Patente bekannt, die für das hier beantragte Projekt relevant sind.

Literatur

[1] Aeronautical Radio, Inc. ARINC Specification 653P1-4, 2015.

[2] AUTOSAR. Release 4.2.2. `http://www.autosar.org`, July 2015.

[3] Alessio Balsini, Alessandra Melani, Pasquale Buonocunto, and Marco Di Natale. FMTV 2016: Where is the Actual Challenge? In *7th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, pages 54–59. INRIA, July 2016. (`https://waters2016.inria.fr/files/2011/12/WATERS16-proceedings.pdf`).

[4] M. Benichou, J. M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent. Experiments in Mixed-Integer Linear Programming. *Mathematical Programming*, 1(1):76–94, December 1971.

[5] Albert Benveniste, Paul Caspi, Stephen A. Edwards, Nicolas Halbwachs, Paul Le Guernic, and Robert de Simone. The Synchronous Languages 12 Years Later. *IEEE*, 91(1):64 – 83, January 2003.

[6] Christian Bradatsch, Florian Kluge, and Theo Ungerer. Data Age Diminution in the Logical Execution Time Model. In *29th International Conference on Architecture of Computing Systems (ARCS)*, volume 9637 of *LNCS*, pages 173–184. Springer, March 2016.

[7] Hamid Reza Faragardi, Björn Lisper, Kristian Sandström, and Thomas Nolte. An Efficient Scheduling of AUTOSAR Runnables to Minimize Communication Cost in Multi-Core Systems. In *7th International Symposium on Telecommunications (IST)*, pages 41–48. IEEE, September 2014.

[8] Hamid Reza Faragardi, Reza Shojaee, and Nasser Yazdani. Reliability-Aware Task Allocation in Distributed Computing Systems using Hybrid Simulated Annealing and Tabu Search. In *14th IEEE International Conference on High Performance Computing and Communication and 9th IEEE International Conference on Embedded Software and Systems (HPCC-ICESS)*, pages 1088–1095. IEEE, June 2012.

[9] Emilia Farcas, Claudiu Farcas, Wolfgang Pree, and Josef Templ. Transparent Distribution of Real-time Components Based on Logical Execution Time. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 31–39. ACM, July 2005.

[10] Alberto Ferrari, Marco Di Natale, Giacomo Gentile, Giovanni Reggiani, and Paolo Gai. Time and Memory Tradeoffs in the Implementation of AUTOSAR Components. In *Design, Automation Test in Europe Conference Exhibition*, pages 864–869. European Design and Automation Association, April 2009.

[11] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[12] Julien Hennig, Hermann von Hasseln, Hassan Mohammad, Stefan Resmerita, Stefan Lukesch, and Andreas Naderlinger. Towards Parallelizing Legacy Embedded Control

Software Using the LET Programming Paradigm. In *22nd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 9–12. INRIA, April 2016. (`https://hal.inria.fr/hal-01305183`).

[13] Thomas A. Henzinger and Christoph M. Kirsch. The Embedded Machine: Predictable, Portable Real-time Code. *ACM Transactions on Programming Languages and Systems*, 29(6):33:1–33:29, October 2007.

[14] International Electrotechnical Commission. International Standard IEC 61499, Function Blocks – Part 1: Architecture, November 2012.

[15] Christoph M. Kirsch and Ana Sokolova. The Logical Execution Time Paradigm. In *Advances in Real-Time Systems*, pages 103–120. Springer, 2012.

[16] Florian Kluge, Martin Schoeberl, and Theo Ungerer. Support for the Logical Execution Time Model on a Time-predictable Multicore Processor. In *14th International Workshop on Real-Time Networks (RTN)*, July 2016. (`http://www.jopdesign.com/doc/mossca-pat.pdf`).

[17] Hermann Kopetz. Event-Triggered Versus Time-Triggered Real-Time Systems. In *Operating Systems of the 90s and Beyond*, volume 563 of *LNCS*, pages 87–101. Springer-Verlag, July 1991.

[18] Marco Di Natale, Guoqiang Wang, and Alberto Sangiovanni Vincentelli. Optimizing the Implementation of Communication in Synchronous Reactive Models. In *14th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 169–179. IEEE, April 2008.

[19] Roman Obermaisser, Christian El Salloum, Bernhard Huber, and Hermann Kopetz. From a Federated to an Integrated Automotive Architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(7):956–965, July 2009.

[20] Stefan Resmerita, Andreas Naderlinger, Manuel Huber, Kenneth Butts, and Wolfgang Pree. Applying Real-Time Programming to Legacy Embedded Control Software. In *18th International Symposium on Real-Time Distributed Computing*, pages 1–8. IEEE, April 2015.

[21] Salah Eddine Saidi, Sylvain Cotard, Khaled Chaaban, and Kevin Marteil. An ILP Approach for Mapping AUTOSAR Runnables on Multi-core Architectures. In *Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO)*, pages 6:1–6:8. ACM, January 2015.

[22] Christian Schwab, Marcus Tangermann, Arndt Lüder, Athanasios Kalogeras, and Luca Ferrarini. Mapping of IEC 61499 Function Blocks to Automation Protocols Within the TORERO Approach. In *2nd IEEE International Conference on Industrial Informatics (INDIN)*, pages 149–154. IEEE, June 2004.

[23] Christos Sofronis, Stavros Tripakis, and Paul Caspi. A Memory-Optimal Buffering Protocol for Preservation of Synchronous Semantics Under Preemptive Scheduling. In *6th ACM and IEEE International Conference on Embedded Software (EMSOFT)*, pages 21–33. ACM, October 2006.

[24] Symtavision GmbH. https://www.symtavision.com/products/symtas-traceanalyzer, 2016.

[25] Timing-Architects Embedded Systems GmbH. https://www.timing-architects.com/ta-tool-suite, 2016.

[26] Ernest Wozniak, Asma Mehiaoui, Chokri Mraidha, Sara Tucci-Piergiovanni, and Sébastien Gerard. An Optimization Approach for the Synthesis of AUTOSAR Architectures. In *18th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–10. IEEE, September 2013.

[27] Eugene Yip, Alain Girault, Partha S. Roop, and Morteza Biglari-Abhari. The ForeC Synchronous Deterministic Parallel Programming Language for Multicores. In *10th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pages 297–304. IEEE, September 2016.

[28] Eugene Yip, Matthew M. Y. Kuo, Partha S. Roop, and David Broman. Relaxing the Synchronous Approach for Mixed-Criticality Systems. In *20th IEEE Real-Time and Embedded Technology and Application Symposium (RTAS)*, pages 89–100. IEEE, April 2014.

[29] Eugene Yip, Partha S. Roop, Morteza Biglari-Abhari, and Alain Girault. Programming and Timing Analysis of Parallel Programs on Multicores. In *13th International Conference on Application of Concurrency to System Design (ACSD)*, pages 160–169. IEEE, July 2013.

[30] Haibo Zeng and Marco Di Natale. Efficient Implementation of AUTOSAR Components with Minimal Memory Usage. In *7th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 130–137. IEEE, June 2012.

[31] Haibo Zeng, Marco Di Natale, and Qi Zhu. Minimizing Stack and Communication Memory Usage in Real-Time Embedded Applications. *ACM Transactions on Embedded Computing Systems*, 13(5s):149:1–149:25, July 2014.

[32] Xinhai Zhang, Lei Feng, De-Jiu Chen, and Martin Törngren. Design-Space Reduction for Architectural Optimization of Automotive Embedded Systems. In *12th International Conference on Embedded Software and Systems (ICESS)*, pages 1103–1109. IEEE, August 2015.

[33] Alexander Zuepke, Marc Bommert, and Daniel Lohmann. AUTOBEST: A United AUTOSAR-OS and ARINC 653 Kernel. In *21st IEEE Real-Time and Embedded Technology and Application Symposium (RTAS)*, pages 133–144. IEEE, April 2015.