Lessons Learned from the EU Project T-CREST

Martin Schoeberl

Department of Applied Mathematics and Computer Science
Technical University of Denmark
Email: masca@dtu.dk

***Abstract*—A three year EU project, such a T-CREST, with partners from all over Europe and with backgrounds from different domains is a challenging endeavor. Successful execution of such a project depends on more factors than simply performing excellent research.**

Within the three-year project T-CREST eight partners from academia and industry developed and evaluated a time-predictable multi-core processor with an accompanying compiler and a worst-case execution time analysis tool. The tight cooperation of the partners and the shared vision of the need of new computer architectures for future real-time systems enabled the successful completion of the T-CREST project.

The T-CREST platform is now available, with most components in open source, to be used for future real-time systems and as a platform for further research.

I. Introduction

For future real-time systems we need high-performance but also time-predictable computing platforms [1]. Time predictability means that the platform allows static analysis of the worst-case execution time (WCET) of individual tasks. The partners of the T-CREST (Time-predictable Multi-Core Architecture for Embedded Systems) project developed and researched novel solutions for time-predictable multi-core architectures that are optimized for the WCET instead of the average-case execution time. Optimizing for the WCET was the main driving force for all time-predictable resources (processors, interconnect, memory arbiter, and memory controller) and tools (compiler, WCET analysis).

We used two industrial use cases to evaluate the T-CREST platform. With an avionics application we showed that tasks executing on different cores do not interfere. From an railway application we showed that the WCET of a signal processing application can be reduced when distributing the tasks over several cores and using the network-on-chip for task communication.

The T-CREST project is the result of a collaborative research and development project executed by eight partners from academia and industry over three years. The European Unions 7th Framework Programme funded T-CREST under grant agreement no. 288008: Time-predictable Multi-Core Architecture for Embedded Systems.

This paper reflects on the three years of research, development, cooperation, and coordination between partners spread over whole Europe. It provides lessons learned from this medium sized project that included hardware and software design. The paper gives suggestions for future projects of similar structure. The main lesson learned is that, especially with very different partners from all over Europe, integration is key. Early integration of the different project components is the single most important aspect for the success of such a project. And working in open-source with public available (and visible) repositories greatly simplifies integration and cooperation.

The result of T-CREST is a (mostly) open-source hardware platform with a compiler and a static WCET analysis tool [2]. Most of the hardware design has been done from scratch. The compiler and WCET analysis tool research and development was based on mature products.

T-CREST was the joint effort of 8 partners from industry and academia. The Open Group (TOG) was the project coordinator and being the driving force on getting the deliverables finished in time. The Technical University of Denmark (DTU) was the technical coordinator of T-CREST. DTU worked on the development of the time-predictable processor Patmos and the time-predictable network-on-chip (NoC) named Argo. The Technical University of Eindhoven (TUE) brought their expertise on memory controllers and NoC to the project. TUE extended their memory controller Predator for the T-CREST platform. The University of York (UoY), with their expertise in real-time systems, worked on a time-predictable memory hierarchy consisting of scratchpad memories and a memory tree towards the shared memory controller. The Technical University of Vienna (TUV) extended the LLVM compiler to support the Patmos processor and developed compiler optimizations for the WCET, including generation of single-path programs. AbsInt Angewandte Informatik (AbsInt) extended their static WCET analysis tool aiT to support the Patmos instruction set and the special time-predictable method and stack caches of Patmos. GMV and Intecs, our two industrial use case providers, developed the requirements for the project and adapted two use cases from the avionics and railway domain for the multi-core platform T-CREST.

This paper is organized in 8 sections: The following section presents the T-CREST platform. Section III discusses the integration work within T-CREST, the most challenging part of the project. Section IV discusses how developing in open-source simplifies integration and how working in open-source may be challenging for PhD students. Section V presents small projects that have already been using the T-CREST platform. Section VI describes the first uptake of T-CREST in teaching. Section VII presents thoughts on how to keep T-CREST active and alive. Section VIII concludes.

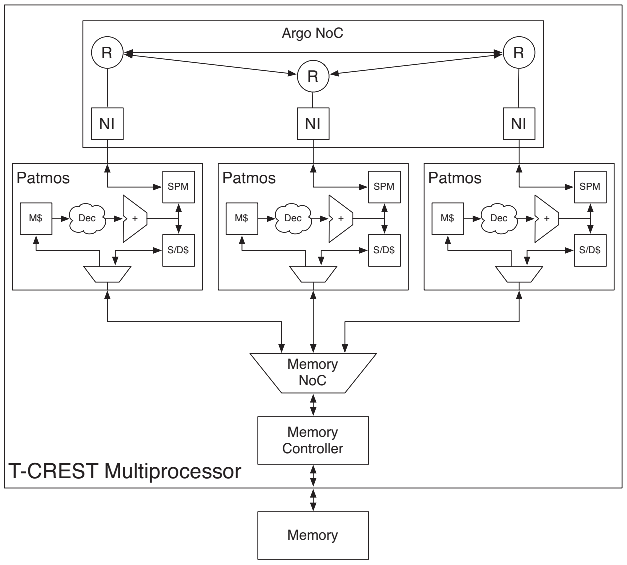

Fig. 1. The T-CREST multiprocessor consisting of Patmos processor nodes that are connected via the Argo network-on-chip for message passing communication and a memory network-on-chip to a memory controller for shared memory access.

II. The T-CREST Platform and Work Packages

Figure 1 shows the hardware of the T-CREST platform, a chip multiprocessor. The individual processors are called Patmos [3]. Patmos is intended to be a time-predictable processor. Patmos is a statically scheduled dual issue pipeline. Patmos contains two specialized caches: the stack cache [4] and the method cache [5]. Patmos implements full predication to support single-path programming [6].

The Patmos processors are connected to two different NoCs: (1) the Argo NoC for core-to-core message passing and (2) a memory NoC for core-to-memory communication. The Argo NoC [7] supports push based communication between processor local scratchpad memories. The NoC uses a static time-division multiplexing schedule to be time-predictable and therefore avoids dynamic arbitration, buffering, and flow control. For the memory NoC we developed two solutions: (1) a memory tree with prefetching [8] and (2) a distributed time-division multiplexing memory NoC [9]. The memory NoC connects all processors to a single real-time memory controller [10]. To attack the growing pressure on the main memory on-chip scratchpad memories have been explored within the project [11].

On the software side the LLVM compiler [12] has been adapted for Patmos [13]. This adaption includes the backend for the Patmos instruction set as well as support for the method cache. For functions that are too large to fit into the method cache the compiler splits functions into smaller ones [14]. The compiler also converts standard C programs into single-path programs [15] to support task execution without timing variability.

The WCET analysis tool aiT [16] from AbsInt has been adapted and extended to support the Patmos instruction set,

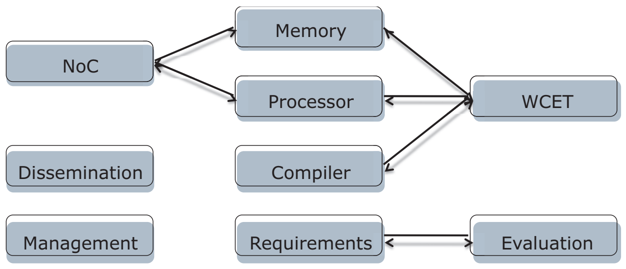

Fig. 2. The T-CREST work packages and their interaction.

support dual issue processors, and extended with a stack cache and a method cache analysis. The compiler and the WCET analysis tool are tightly integrated via the `platin` toolkit to support each other. `platin` translates program information, e.g., branch targets for switch statements, to annotations for aiT. aiT delivers back information on the WCET path to the compiler to guide the optimization along the WCET path.

The T-CREST platform was evaluated with applications from two domains: avionics and railway. The three real-world avionic applications were: (1) an Airlines Operational Centre, (2) a Crew Alerting System, and (3) an I/O Partition. The railway application was the GSM-R Integrity Detection System that has been adapted to the multi-core platform T-CREST. GSM-R is a railway specific version of the commercial GSM standard.

Figure 2 shows the 9 work packages (WP) of T-CREST and their main interaction. WP 1, Requirements Analysis, defined the industrial requirements for a time-predictable platform and was mainly driven by the user partners GMV and Intecs. WP 2, Processor, covers development work on the time-predictable processor Patmos with the main responsibility at DTU. WP 3, Network on Chip, covers the design and development of a time-predictable core-to-core on-chip communication network. WP 3 was a joint effort between DTU and TUE. WP 4, Memory Hierarchy, covers the memory NoC, organization of local scratchpad memories, and the memory controller. WP 4 was a joint effort of UoY and TUE. WP 5, Compiler, covers the adaption of the LLVM compiler to Patmos and research WCET driven compilation. WP 5 was a joint effort between TUV and DTU. WP 6, Code-Level WCET Analysis, covers the adaption of aiT for the Patmos processor and interaction with the compiler. AbsInt was responsible for WP 6. WP 7, Integration and Evaluation, covers the adaption of two use cases to T-CREST and evaluation of the platform. WP 7 was a join effort between GMV and Intecs with considerable contributions of all partners. WP 8, Dissemination, covers the project web site,[1] scientific publications, and organization of workshops. WP 9, Management, covers consortium management, progress tracking, and reporting. WP 9 was lead by TOG with support from the technical lead by DTU. From Figure 2 it can be seen that WCET analysis played a central role in the project as it interacts with the hardware and the compiler.

[1]http://www.t-crest.org/

III. Integration

A main challenge in an EU project that includes partners located all over Europe is the integration of the individual components to provide a complete platform. The effort of this integration shall not be underestimated.

A. Compiler and Processor Coordination

An important first step for the compiler processor integration was the development of a software simulator of the Patmos processor [17]. This software simulator simplified compiler development and testing. The simulator is a plain C++ program. Therefore, a compiler developer need no access to *real* hardware.

The simulator then served as gold standard for the processor development. Test cases have been executed on the software simulator and the hardware emulator and the register file of the processor compared for each clock cycle. As any data will at some point pass the register file, any difference between the simulator and the hardware will be detected by this tests.

For the compiler and the Patmos processor a build bot has been setup at TU Vienna for regression tests. The build bot is triggered by commits to the GitHub repository and on a nightly basis to rebuild and test the compiler and the processor.

However, as convenient as a software simulator is, there is the danger of overusing it. Simulators provide an abstraction of the real world, in our case the real hardware of the processor. As the simulator has been written before any hardware was available, there are differences in the processor model, the pipeline, and the memory subsystem. Therefore, we argue to use the hardware in an FPGA as much as possible for evaluation.

B. Early Integration

The original plan in the T-CREST project was to have the main integration done in the last project year. However, our EC project officer pushed for starting integration earlier. Therefore, we were able to show a first version of the T-CREST platform in the year two EC review meeting.

We are glad that we have been pushed to start integration early. Integration work consumed more resources than we expected. When it comes to debugging of a combination of components from different project partners a lot of time is spent on communication and coordination between partners.

The lesson learned is that it is too easy to underestimate integration work. In future projects we will budget more resources and a dedicated work package for integration. Furthermore, it is of primary importance to start integration as early as possible, best from the project start.

C. Who Owns the Top-Level?

One important question in a hardware design is who is the *owner* of the top-level component. Crosscutting issues such as clock domains and reset sequences should be reflected in the top-level entity of the hardware design. The *owner* of that entity shall be responsible for those issues.

Within T-CREST project we missed to assign a clear responsibility for the top-level entity. The resulting consequences have been: no clear understanding of the different clock domains, unclear requirements of component initialization sequences, and issues with clock domain crossings. This lead to long and troublesome integration and debugging work at the final hardware integration phase.

The lesson learned is that in a future hardware project a clear owner of the top-level component including the clock generation, the clock domain crossing, and the reset sequencing needs to be agreed on.

D. Documentation

The path of documentation in research is in publishing the research results in papers. More details of the individual T-CREST components are available in the EC project deliverables, most of them public available.[2] Furthermore, an open-source project such a T-CREST also publishes the source code.

However, the gap between source code and scientific papers is huge. And individual deliverables represent only snapshots of work packages in different phases of the project. Therefore, we collected the relevant results and added introduction section into the Patmos handbook [18], This handbook is a work-in-progress and will be update when components are changed.

Besides project internal mailing list for general project discussions we also setup a public mailing list[3] to also enable interaction with developers from outside the T-CREST project.

IV. The Open-Source Approach

Providing the source of a project from the start on in open source and publicly visible is an interesting and probably bold step. Many projects, EU projects or just single PhD projects, only publish the results and do not provide access to the source.

However, I think this is wrong for projects financed by the society. First, if the society has financed the research than the society shall also receive the results. Second, maybe more important, providing the sources of the developments enables independent verification of the experimental results. Verification of the results by other researches is common practice in other research fields, e.g., in physics, but very uncommon in computer science. This is a shame, especially because reproduction of results is usually relatively cheap and easy in computer science when the sources are available. Many experiments can be run on a researcher's laptop.

Many researchers are very open to share their work and the sources, but only when the source is polished enough. And this is the main catch: code produced by researchers to explore ideas is usually not polished, not well documented, and probably not written in good software engineering style. However, as this code is research code and not production code this is a valid approach. We shall accept this quality of code from us and from other researchers and make it available to others.

[2]http://www.t-crest.org/page/results

[3]https://groups.yahoo.com/group/patmos-processor/

Another interesting issue on being in open-source during the research work is the exposition of PhD work early in the public. It means that all paths explored by the student, including the dead ends, are in principle visible to the whole world. This might stress the student to an extent that he will not explore risky but maybe interesting directions. In my experience the practical work often involves a mixture of part of the code being in the public main repository of the project, but some of the code being only available on the student's computer. The later is often simply lost when the student finished her PhD thesis.

Furthermore, working with a public repository for a research project means that code is public before a paper that uses this code has been published. There is a strong fear of researchers that this situation can lead to stealing of ideas. Those ideas are somehow encoded in the public code. However, I think this is a non-issue for two reasons: (1) the code of a medium sized project is complex enough that it takes time to understand it and to find the new, unpublished ideas in it; (2) when the idea is published in the code than a repository with version control shows the date of the publishing of the original idea and makes it defendable.

In summary my opinion is that research work shall be done in open-source as much as possible. Source code is part of computer science research projects and shall be considered as part of the publication process. Open-source projects simplify cooperation and collaboration and also simplify building upon earlier research.

V. Further Projects

T-CREST being open-source helped to attract research and development outside of the EC funded project. Furthermore, the entry level to get started with T-CREST is quite low. For simulation of T-CREST a Linux or Mac based laptop with only free software serves as a full development environment. For exploring T-CREST in an FPGA, just a low-cost FPGA board, such as the terasic DE2-155, and the free Quartus version from Altera are needed.

Research in operating systems was not included in the T-CREST project. Nevertheless, two projects explored porting of real-time operating systems (RTOS) to Patmos: RTEMS and TiCOS.

GMV ported RTEMS to provide a run-time environment for the avionics application. RTEMS is an open-source RTOS that is very popular for space missions.

T-CREST is also a good platform for student projects and master thesis. Marco Ziccardi ported the TiCOS operating system [19], which is a fork of the POK RTOS [20], to the Patmos processor [21]. TiCOS is a time-predictable operating system that supports the ARINC 653 standard. In the current port TiCOS supports only a single processor. We plan to extend TiCOS to support several cores and to map ARINC communication primitives to the NoC of T-CREST.

Luca Pezzarossa explored the connection of hardware accelerators to T-CREST during his master thesis [22]. The main challenge is to keep the access to the hardware accelerators time-predictable. Using the time-division multiplexing Argo NoC provides a time-predictable communication channel between the processors and the accelerator [23].

VI. T-CREST in Teaching

TU Vienna offers a course in WCET analysis.[4] Originally the course was using the aiT WCET analysis tool with the LEON processor. However, when the compiler and aiT had been adapted to support Patmos and a software simulator of Patmos was available the course switched to Patmos. This change allowed extending the WCET analysis course with topics on compiler support for single-path code [15].

The COST action TACLe (Timing Analysis on Code-Level) organized a summer school on principles, needs and challenges of timing analysis at code level.[5] Besides WCET analysis and compiler support for timing analysis one topic of the summer school was time-predictable computer architecture. Within this topic the T-CREST platform was presented and the students used the Patmos simulator and hardware emulator in the lab to explore time-predictable architecture.

The exercises from this hardware lab have been extended at DTU to include the Argo NoC. We use T-CREST in an FPGA platform for labs in the advanced computer architecture course.[6]

For self-study, e.g., to get started on a master project with T-CREST, the exercises and the Patmos handbook [18] are available online.[7]

VII. Organizing Future Work

We consider T-CREST as a platform to be used for real-time systems, but also a platform for future research work. With all components and tools in place it is easy to extend in diverse directions. What remains, as a challenge, is to keep the code base alive and having resources to fix errors or integrate new research work.

A. *Keeping a Research Project Alive*

The issue with mainly university and research driven projects is that there is a very high fluctuation in the developers. A PhD student is three to four years involved in the project. It will take time for the student to learn the code base and being able to contribute to the project. The research work of the student might be too experimental to be integrated in the master branch and the last year of the PhD work will be dominated by the actual thesis writing. Therefore, in the best case a PhD student can serve for the project and contribute to the master code base for about two years of her PhD study. To keep a project alive just with PhD students will require a continuous pipeline of funding and research projects.

A better solution for continuous development, and as a basis for future work, would be permanent employees being

[4]http://ti.tuwien.ac.at/cps/teaching/courses/wcet

[5]http://www.tacle.eu/index.php/activities/summer-schools-forums/2014-venice

[6]http://www2.imm.dtu.dk/courses/02211/

[7]http://patmos.compute.dtu.dk/

responsible for the project. For this position we would need an engineer and not a researcher. This person would also be in the best place to sort out which student project results are worth keeping in the main line of development or which results are just temporal sidetracks of the project. However, with the current organization of a university, at least at DTU, there is no place and money for such a position.

Nevertheless, we intend to keep the T-CREST project alive and use it for future research work and research projects. Although the name and acronym have been the result of a EC funded project, and each new project needs a new name, we will keep the name as a kind of branding alive.

B. Next Research Directions

Next research steps are to extend T-CREST to a distributed system. We intend to extend a T-CREST processing node with a time-predictable Ethernet controller. The main candidate for real-time Ethernet is TTEthernet [24], the real-time networking technology developed by TTTech. With a time-predictable computing platform and time-predictable networking solution we are confident to provide a distributed real-time system where we can guarantee end-to-end latencies.

Another direction of research within the multi-core platform is the model of communication. Current multi-core processors use shared memory for communication. The core-to-core communication on the chip is performed via the cache coherence protocol. This has two disadvantages: (1) it does not scale, as the cache coherence mechanism is a bottleneck; (2) it is hardly time-predictable. Therefore, we need to focus on different, scalable, and time-predictable forms of on-chip communications [25].

As shared external memory is the bottleneck of a multi-core system, we need to keep communication on-chip as much as possible. Besides using the NoC of push-based communication we will explore different forms of on-chip memories and the movement of data between those memories or the movement of *the memories* between the processing cores.

We will explore the notion of distributed shared on-chip memory. This distributed memory will be mapped into the global address space. Access to that memory via load and store instructions will be translated to NoC request and reply packets.

A shared scratchpad memory is somehow similar in structure to shared off-chip memory, but on-chip and no cache coherence protocol is needed, as this memory is already on-chip. It can be used to share bulk data. It needs a time-predictable (TDM based) arbitration.

A version of shared scratchpad memory is the shared scratchpad memory with ownership. The idea is that several scratchpad memories are shared, but there is a notion of an owner. Only the current owner is allowed to write (and read) to that memory. To communicate, ownership is transferred to a different core. This mechanism fits well for communicating bulk data and provides short and time-predictable latencies for memory accesses to an *owned* memory.

A further direction of future research based in T-CREST is the extension of the platform to support hardware accelerators with timing constraints [23].

C. Further and Future Research Projects

At DTU we supplement T-CREST with the research project RTEMP (Hard Real-Time Embedded Multiprocessor Platform) that is funded by The Danish Council for Independent Research – Technology and Production Sciences (FTP). The RTEMP project runs from April 2014 to July 2016. The project funds research at DTU and has Danfoss Power Electronics as an industrial partner/user. The RTEMP project uses the T-CREST platform and supplements T-CREST by focusing on developing a time-predictable multi-core platform that is specifically optimized for implementation in FPGA technology. The intention is to use FPGAs for real products and not only for prototyping.

For EU-wide research we have organized a consortium for a project proposal within the ECSEL Joint Undertaking. An ECSEL project is more industry driven than FP7 or H2020 EU projects and will provide a path for T-CREST to find its way into industrial use. The new consortium includes several former T-CREST partners to provide continuity within the T-CREST research and development and several new industrial partners to support innovation and bring T-CREST into use in different real-time application domains.

VIII. Conclusion

In summary, the T-CREST project was a very successful EU project leading to a time-predictable multi-core processor and supporting compiler and WCET analysis tools. Most artifacts of T-CREST are available in open source and therefore provide a low entrance path towards future research.

The main lesson learned in this project, which included eight partners from all over Europe and from different domains, is that early integration is the key to successfully complete such a project within three years. Using public repositories and an open-source license simplified cooperation and integration of components.

Acknowledgment

I would like to thank all partners from the T-CREST project for the deep research interaction and joyful discussions during project meetings: Sahar Abbaspour, Benny Akesson, Neil Audsley, Florian Brandner, Raffaele Capasso, Christoph Cullmann, Christian Ferdinand, Jamie Garside, Gernot Gebhard, Kees Goossens, Sven Goossens, Scott Hansen, Reinhold Heckmann, Stefan Hepp, Benedikt Huber, Guido Ioele, Alexander Jordan, Evangelia Kasapaki, Jens Knoop, Yonghui Li, Daniel Prokesch, Wolfgang Puffitsch, Peter Puschner, André Rocha, Cláudio Silva, Jens Sparsø, Alessandro Tocchi, and Jack Whitham.

This work was partially funded under the European Union's 7th Framework Programme under grant agreement no. 288008: Time-predictable Multi-Core Architecture for Embedded Systems (T-CREST) and the EU COST Action IC1202: Timing Analysis on Code Level (TACLe).

Source Access

Most of the T-CREST project results are available in open source and are hosted at GitHub: https://github.com/t-crest

References

[1] M. Schoeberl, "Time-predictable computer architecture," *EURASIP Journal on Embedded Systems*, vol. vol. 2009, Article ID 758480, p. 17 pages, 2009.

[2] M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann, S. Hepp, B. Huber, A. Jordan, E. Kasapaki, J. Knoop, Y. Li, D. Prokesch, W. Puffitsch, P. Puschner, A. Rocha, C. Silva, J. Sparsø, and A. Tocchi, "T-CREST: Time-predictable multi-core architecture for embedded systems," *Journal of Systems Architecture*, vol. 61, no. 9, pp. 449–471, 2015.

[3] M. Schoeberl, P. Schleuniger, W. Puffitsch, F. Brandner, C. W. Probst, S. Karlsson, and T. Thorn, "Towards a time-predictable dual-issue microprocessor: The Patmos approach," in *First Workshop on Bringing Theory to Practice: Predictability and Performance in Embedded Systems (PPES 2011)*, Grenoble, France, March 2011, pp. 11–20.

[4] S. Abbaspour, F. Brandner, and M. Schoeberl, "A time-predictable stack cache," in *Proceedings of the 9th Workshop on Software Technologies for Embedded and Ubiquitous Systems*, 2013.

[5] P. Degasperi, S. Hepp, W. Puffitsch, and M. Schoeberl, "A method cache for Patmos," in *Proceedings of the 17th IEEE Symposium on Object/Component/Service-oriented Real-time Distributed Computing (ISORC 2014)*. Reno, Nevada, USA: IEEE, June 2014, pp. 100–108.

[6] P. Puschner and A. Burns, "Writing temporally predictable code," in *Proceedings of the The Seventh IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2002)*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 85–94.

[7] E. Kasapaki, M. Schoeberl, R. B. Sørensen, C. T. Müller, K. Goossens, and J. Sparsø, "Argo: A real-time network-on-chip architecture with an efficient GALS implementation," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, 2015.

[8] J. Garside and N. C. Audsley, "Prefetching across a shared memory tree within a network-on-chip architecture," in *System on Chip (SoC), 2013 International Symposium on*, Oct 2013, pp. 1–4.

[9] M. Schoeberl, D. V. Chong, W. Puffitsch, and J. Sparsø, "A time-predictable memory network-on-chip," in *Proceedings of the 14th International Workshop on Worst-Case Execution Time Analysis (WCET 2014)*, Madrid, Spain, July 2014, pp. 53–62.

[10] S. Goossens, J. Kuijsten, B. Akesson, and K. Goossens, "A reconfigurable real-time SDRAM controller for mixed time-criticality systems," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2013 International Conference on*, Sept 2013, pp. 1–10.

[11] J. Whitham, R. Davis, N. Audsley, S. Altmeyer, and C. Maiza, "Investigation of scratchpad memory for preemptive multitasking," in *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, Dec 2012, pp. 3–13.

[12] C. Lattner and V. S. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *International Symposium on Code Generation and Optimization (CGO'04)*. IEEE Computer Society, 2004, pp. 75–88.

[13] P. Puschner, D. Prokesch, B. Huber, J. Knoop, S. Hepp, and G. Gebhard, "The T-CREST approach of compiler and WCET-analysis integration," in *9th Workshop on Software Technologies for Future Embedded and Ubiquitious Systems (SEUS 2013)*, 2013, pp. 33–40.

[14] S. Hepp and F. Brandner, "Splitting functions into single-entry regions," in *Proceedings of the 2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, ser. CASES '14. New York, NY, USA: ACM, 2014, pp. 17:1–17:10.

[15] D. Prokesch, S. Hepp, and P. Puschner, "A generator for time-predictable code," in *Proceedings of the 17th IEEE Symposium on Real-time Distributed Computing (ISORC 2015)*. Aukland, New Zealand: IEEE, April 2015.

[16] R. Heckmann and C. Ferdinand, "Worst-case execution time prediction by static program analysis," AbsInt Angewandte Informatik GmbH, Tech. Rep., [Online, last accessed November 2013].

[17] DTU, "D 2.1 software simulator of patmos," T-CREST: http://www.t-crest.org/page/results, Tech. Rep., 2012.

[18] M. Schoeberl, F. Brandner, S. Hepp, W. Puffitsch, and D. Prokesch, "Patmos reference handbook," Tech. Rep., 2014.

[19] A. Baldovin, E. Mezzetti, and T. Vardanega, "A time-composable operating system," in *12th International Workshop on Worst-Case Execution Time Analysis, WCET 2012, July 10, 2012, Pisa, Italy*, ser. OASICS, T. Vardanega, Ed., vol. 23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012, pp. 69–80.

[20] J. Delange and L. Lec, "POK, an ARINC653-compliant operating system released under the BSD license," in *13th Real-Time Linux Workshop*, vol. 10, 2011.

[21] M. Ziccardi, M. Schoeberl, and T. Vardanega, "A time-composable operating system for the Patmos processor," in *The 30th ACM/SIGAPP Symposium On Applied Computing, Embedded Systems Track*. Salamanca, Spain.: ACM Press, April 13–17 2015.

[22] L. Pezzarossa, "Hardware Accelerators in Network-on-Chip Based Multi-Core Platforms," Master's thesis, Technical University of Denmark, Dept. of Applied Mathematics and Computer Science, 2014.

[23] L. Pezzarossa, R. B. Sørensen, M. Schoeberl, and J. Sparsø, "Interfacing hardware accelerators to a time-division multiplexing network-on-chip," in *Proc. of the 1st Nordic Circuits and Systems Conference (NORCAS 2015)*. Oslo, Norway: IEEE, October 2015.

[24] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (TTE) design," in *ISORC '05: Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 22–33.

[25] M. Schoeberl, R. B. Sørensen, and J. Sparsø, "Models of communication for multicore processors," in *Proceedings of the 11th Workshop on Software Technologies for Embedded and Ubiquitous Systems (SEUS 2015)*. Aukland, New Zealand: IEEE, April 2015, pp. 44–51.