Cyber-Physical Systems in Production Engineering

Designing safe, predictable, and high performance embedded platforms for next generation Cyber-Physical Systems

■ *Modern Cyber-Physical Systems (CPS) are the next generation of engineered systems in which computing, communication, and control technologies are tightly integrated. Applications include system automation, Internet of Things (IoT), smart buildings, smart manufacturing, smart cities, digital agriculture, robotics, and autonomous vehicles.*

The Chair of Cyber-Physical Systems in Production Engineering was founded in September 2018. In 2019 Research activities of the Chair focused on two main topics: sandboxing controllers for Cyber-Physical Systems, and predictable high performance computing with heterogeneous SoC multicore platforms. Other research activities are focusing on the secure and safe integration of machine learning algorithms with digital controllers for CPS and development of flexible resource management policies for a broad range of CPS systems. In 2019, Prof. Caccamo was in the editorial board of IEEE Transactions on Computers. Members of the chair were involved in the peer review process of several international conferences/journals in real-time embedded systems and CPS, including ECRTS'19, RTSS'19, EMSOFT'19, RTCSA'19, IROS'19, RTNS'19, ICCPS'20, RTAS'20, DATE'20, ACM TECS, and IEEE TC.

Sandboxing Controllers for Stochastic Cyber-Physical Systems

State-of-the-art Cyber-Physical Systems (CPS) are widely used in various kinds of applications, such as automotive, aviation, manufacturing plants and so on, and they are expected to accomplish complex missions. Therefore, complex, high performance but unverified controllers (e.g., deep neural network or black-box controllers from third parties) are applied to complete these complex missions, which makes it increasingly challenging to ensure the safety of CPS.

In this project, we exploit the idea of the sandbox from the community of computer security to design a novel architecture for sandboxing unverified controllers that uses a **Safe**ty Advisor and a Super**visor** (Safe-visor). On one hand, Safety Advisor only focuses on the safety of the system, which should be treated as a fallback in case the unverified controllers are trying to perform some harmful actions. On the other hand, the unverified controller is designed for functionality, i.e. it is expected to realize some tasks which are much more complicated than purely keeping the system safe. To ensure a specific level of safety probability of the physical system, the supervisor specifies verifiable safety rules for the unverified controller to follow. The architecture of the safe-visor is illustrated in Figure 1.

The key idea of this architecture is that the control inputs of the controller fed to the system are checked and can only be accepted when they are not disobeying the safety rules defined in the sandboxing mechanism. The execution of this architecture can be concluded as the following: During the execution of the Safe-visor, the Safety Advisor is responsible for providing advisory input for the Supervisor based on the current state of the physical system, which seeks to maximize the safety probability. Meanwhile, the Supervisor checks the input given by the unverified controller according to the safety rules. Input from the unverified controller would only be accepted when it follows the rule; otherwise, the Supervisor would accept the advisory input from the Safety Advisor to maximize the safety probability of the physical system.

In general, the Safe-visor architecture can be used to sandbox any types of unverified controllers at run-time in order to guarantee the safety of the physical system. By sandboxing the unverified controller, we are able to exploit its advantages for realizing complex tasks while preventing the system from being threatened by its harmful behaviour, if any.

In our current research, the proposed architecture is applied to two case studies, one for temperature control

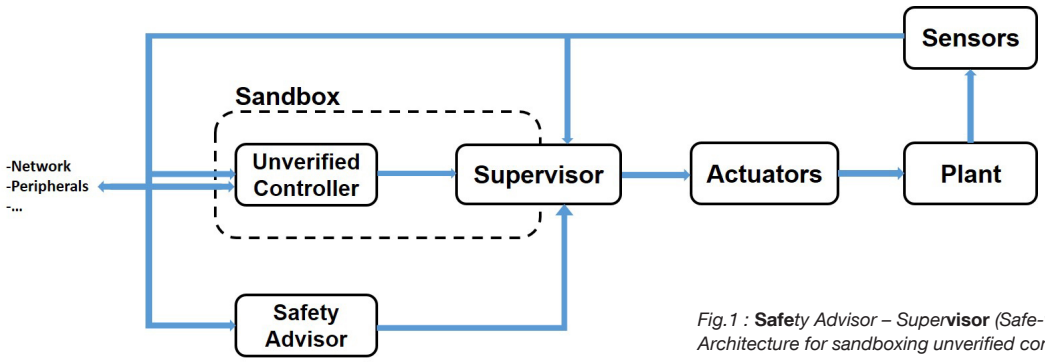

Fig.1 : **Safe***ty Advisor –* Super**visor** *(Safe-visor) Architecture for sandboxing unverified controller*

in smart buildings, the other for traffic flow control in intelligent traffic control systems. We simulated each scenario with a given safety specification, and the expected safety probability in both cases were guaranteed. In the first case, the temperature of the room in the building is expected to be kept between 19 to 21 degrees. We synthesized a safety advisor and a supervisor which guarantee that the probability for fulfilling the safety specification can be higher than 99% in 6 hours. In the second case, the density of traffic on a road segment, which contains a cell with 2 entries and 1 exit, is expected to be kept between 0 to 20 cars within each time unit. We synthesized a safety advisor and a supervisor which guarantee that the probability for fulfilling the safety specification should be higher that 99.95% for more than 12 hours.

Real-Time Scratchpad-Centric OS with Three-Phase Execution Model

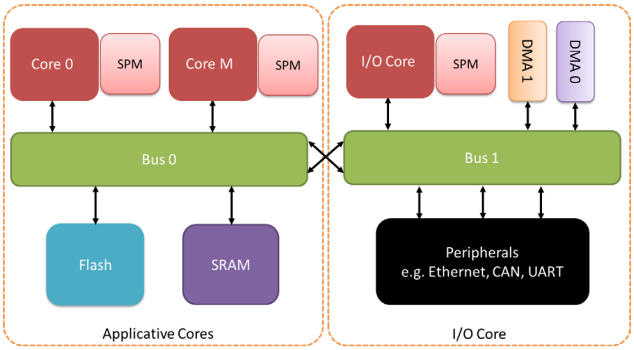

Fig. 2: Multicore architecture satisfying our hardware assumptions

Multi-core processors have replaced single-core systems in almost every segment of the industry. Unfortunately, their increased complexity often causes a loss of temporal predictability which represents a key requirement for hard real-time cyber-physical systems.

This work focuses on the design and development of a predictable Operating System (OS) for hard real-time systems, namely SPM-centric OS. It assumes that each core in the system features a block of private scratchpad memory (see Figure 2). Scratchpad memory is a high-speed internal memory used for temporary data storage. Moreover, it assumes that the size of each per-core scratchpad memory is big enough to fully contain the memory footprint of any two tasks in the system. Such requirements might be over-restrictive for general purpose applications, but are reasonable when implementing factory automation tasks and software controllers commonly deployed on Cyber-Physical Systems.

In the proposed design, before tasks can be executed from scratchpad (SPM) memory, their code and data need to be transferred from main memory. Thus, we adopt a task model that is composed of three phases: a load phase, an execution phase and an unload phase. First, during the load phase, the code and data image for the activated task is copied from main memory to the SPM. Next, during the execution phase, the loaded task executes on the CPU by relying on in-scratchpad data. Finally, the portion of data that has been modified and needs to remain persistent across subsequent activations of the task is written back to main memory during the unload phase. The memory operations are conducted by a Direct Memory Access (DMA) engine to free up the CPU. It is important to highlight that the proposed execution model and resulting OS design are beneficial for tasks that perform a large number of memory accesses over a relatively small memory footprint — i.e. data-intensive applications. This is because the main advantage of the proposed model is that the overhead of accessing the global (slower) memory is encountered only once using the DMA. This can be further reduced by pipelining workload execution and memory accesses. After the process of loading a task's code and data into a local, faster memory (scratchpad) is completed, the task executes without suffering any contention on data/instruction accesses.

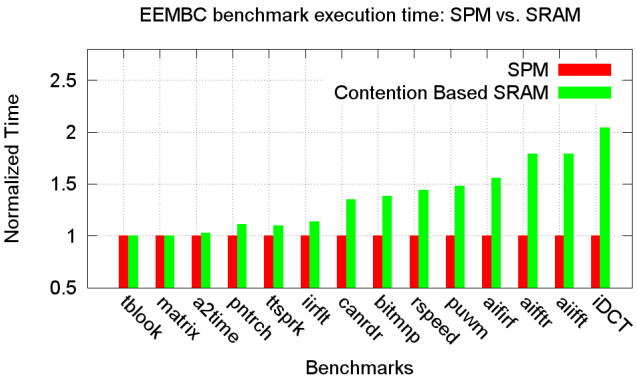

Fig. 3: Experimental execution time for EEMBC benchmarks

Moreover, an I/O subsystem design is proposed that enforces a complete separation between task execution and the asynchronous activity of I/O peripherals: this goal is achieved by offering to application tasks a synchronous view of I/O data. The proposed SPM-centric OS was implemented using Evidence Erika Enterprise. Erika Enterprise is an open-source RTOS that is compliant with the AUTOSAR (Automotive Open System Architecture) standard. The behavior of EEMBC benchmarks, a standard set of benchmarks for CPSs, was investigated to evaluate the performance of the scratchpad-centric OS. The normalized execution times for all the considered

benchmarks are reported in Figure 3. From the results, we note that computation intensive benchmarks do not benefit from SPM-based execution. Conversely, for memory intensive benchmarks SPM-based execution determines substantial speed-ups (up to 2.1x). For the schedulability evaluation of the proposed approach, we compared our system against the contention-based system, in which cores use caches but are left unregulated when accessing main memory.
Figure 4 shows the result of the schedulability analysis when using the proposed SPM-centric OS versus a contention based SRAM system. The figure shows the results in terms of proportion of schedulable task sets for both approaches. The results show that the schedulability of the system increases significantly when the proposed SPM-centric approach is used. Hence, the described

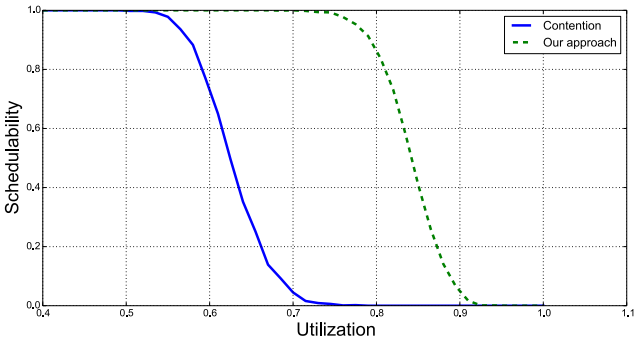

Fig. 4: Schedulability with SPM-based and traditional scheduling models

SPM-centric OS not only improves the predictability of task execution, but it also improves task set schedulability by hiding the main memory access latency, especially for memory intensive applications.

Segment Streaming for the Three-Phase Execution Model

The previously discussed three-phase execution model (load-execute-unload) for tasks can effectively reduce the contention on the shared resources in a multicore processor. In such an execution model tasks are segmented and executed over multiple intervals. A variety of previous works, including the SPM-Centric OS (discussed in the previous section), have shown that, by pipelining memory accesses (unload-load) and workload execution, system schedulability can be improved by hiding the memory transfer time of each task. However, previously the execution of consecutive segments of the same task back-to-back was not allowed.
Recently, we have proposed a new streaming model that allows overlapping (pipelining) the memory (load/unload) and execution phases of consecutive segments of the same task. The high-level idea is to automatically break a large task into multiple smaller segments using an augmented version of the LLVM compiler. The compiler further distinguishes between **terminal segments** and **streaming segments** of the compiled code. Streaming segments have an internal code structure such that the current task segment can be executed while the data and code for the next segment of the same task are loaded in parallel. Parallelizing the execution and memory load/unload within the segments of the same task allows to significantly increase the schedulability and real-time performance of a modern computing platform for CPS.

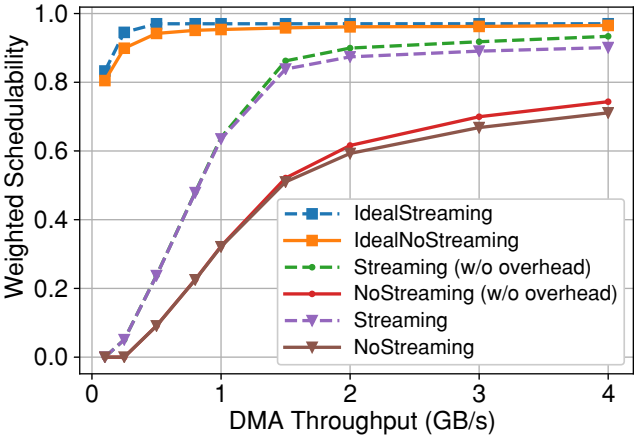

Fig. 5: Weighted Schedulability vs DMA throughput (SPM size = 64 kB)

In order to evaluate the proposed streaming model, a set of multiple benchmark suites (UTDSP, TACLeBench, and CortexSuite) were considered. For evaluation, we considered Xilinx UltraScale+ ZCU102 MPSoC platform running Evidence Erika Enterprise OS. Figure 5 shows the weighted schedulability (y-axis) vs the DMA throughput (x-axis) for SPM size of 64 kB. The streaming execution model largely outperforms the non-streaming three-phase model for different DMA throughputs. In fact, the streaming model results in an improvement up to 16% for the weighted schedulability compared to the non streaming model.

Cyber-Physical Systems in Production Engineering

Prof. Dr. Marco Caccamo

Contact
www.mw.tum.de/cps
mcaccamo@tum.de
Phone +49.89.289.55170

Management
Prof. Dr. Marco Caccamo, Director

Administrative Staff
Anke Harisch, Secretary

Research Scientists
Dr. Giovani Gracioli
Dr. Tomasz Kloda
Dr. Ohchul Kwon
Mirco Theile, M.Sc.
Bingzhuo Zhong, M.Sc.
Micaela Verucchi, M.Sc.
Gero Schwäricke , M.Sc.
Or Dantsker, M.Sc.
Denis Hoornaert, M.Sc.
Jiyang Chen, M.Sc.
Jorge Luis Martinez Garcia, M.Sc.
Alessandro Tesi, M.Sc.
Jayati Singh (MS student)
Richard Nai (MS student)
Andrei Goncharov (MS student)

Research Focus
- Safety-critical cyber-physical systems
- Real-time systems
- Scheduling and schedulability analysis
- Secure and safe integration of machine learning with CPS

Competences
- System-level programming
- Embedded system software design
- Hardware and software integration on FPGAs
- Real-time operating systems
- Real-time reachability analysis

Infrastructure
- 3 DOF helicopter
- Embedded and FPGA multi-core development platforms
- High-performance server
- Linear inverted pendulum
- Fused filament fabrication, dual-head 3D printer
- F1/10 autonomous cars

Lectures
- Advanced Seminar on Safe Cyber-Physical Systems
- PhD-Seminar on Real-Time Cyber-Physical Systems
- Concepts and Software Design for Cyber-Physical Systems
- Tutorial Concepts and Software Design for Cyber-Physical Systems
- Simplex: Fault-Tolerant Control Strategy for Real-Time Cyber-Physical Systems-Laboratory

Collaborations
- University of Illinois at Urbana-Champaign, USA
- Boston University, USA
- University of Colorado Boulder, USA
- University of Waterloo, Canada
- University of Modena, Italy
- Federal University of Santa Catarina, Brazil

Selected Publications 2019

- M.R. Soliman, G. Gracioli, R. Tabish, R. Pellizzoni, and M. Caccamo “Segment Streaming for the Three-Phase Execution Model: Design and Implementation”, Proceedings of the IEEE Real-Time Systems Symposium, Hong Kong, China, December 2019.
- B. Zhong, M. Zamani, and M. Caccamo, “Sandboxing Controllers for Stochastic Cyber-Physical Systems”, Proceedings of the International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS), Amsterdam, Netherlands, August 2019.
- (Best presentation award) O. Dantsker, S. Imtiaz, S. Yu, and M. Caccamo, “Electric Propulsion System Optimization for a Long-Endurance Solar-Powered Unmanned Aircraft”, in AIAA/IEEE Electric Aircraft Technologies Symposium, Indianapolis, IN, USA, August 2019.
- R. Tabish, R. Mancuso, S. Wasly, R. Pellizzoni, and M. Caccamo, “A Real-time Scratchpad-centric OS with Predictable Inter/intra-core Communication for Multi-core Embedded Systems”, Real-Time Systems Journal, on-line, September 2019.
- (Outstanding paper) G. Gracioli, R. Tabish, R. Mancuso, R. Mirosanlou, R. Pellizzoni, and M. Caccamo, “Designing Mixed Criticality Applications on Modern Heterogeneous MPSoC Platforms”, Proceedings of the 31st Euromicro Conference on Real-Time Systems (ECRTS), Stuttgart, Germany, July 2019.