Virtual Execution Platforms for Mixed Time-Criticality Applications: Demonstrating the CompSoC Platform and Design Flow

Benny Akesson[1], Martijn Koedam[2], Anca Molnos[3], Ashkan Beyranvand Nejad[3], Andrew Nelson[3], Sander Stuijk[2], and Kees Goossens[2]

[1] CISTER-ISEP Research Unit, [2] Eindhoven University of Technology, [3] Delft University of Technology

Systems-on-Chip (SoC) complexity increases as a growing number of applications are integrated and executed on sophisticated multi-processor systems that strike a balance between performance, cost, power consumption and flexibility [4, 6]. Complexity is further increased by an increasing number of concurrently executing applications, which result in a large number of possible *use-cases*. The applications have *mixed time-criticality*, which is a mix between firm, soft, and no *real-time requirements*. Firm real-time requirements must always be satisfied to avoid unacceptable output quality loss. Occasional failures to meet soft requirements can be tolerated. Non-real-time applications do not have well-defined timing requirements, but must be responsive.

Verification of real-time requirements is traditionally performed using formal timing analysis, simulation, or a combination of the two. Firm real-time applications demand rigorous formal analysis, since their requirements must always be met. In contrast, soft real-time applications are often verified by simulation for a large set of inputs, as they are often dynamic by nature and difficult to verify by formal methods in a cost-effective manner. A key challenge with verification is that platform resources, such as processors, interconnect, and memories, are shared between applications to reduce cost. This results in *interference* between applications, making their temporal behaviors inter-dependent. This causes three problems with respect to integration and verification. Firstly, accurate system-level simulation and several approaches to timing analysis become infeasible, because of the *state-space explosion* resulting from the many use-cases, application inputs, and resources states. Secondly, use-case verification becomes a *circular process* that must be repeated if an application is added, removed, or modified [3]. Thirdly, verification of a use-case cannot begin until all applications it comprises are available. The verification process hence depends on the availability of all applications, which may be developed by independent software vendors. Together, these problems contribute to making the integration and verification process a dominant part of SoC development, both in terms of time and money [3].

The CompSoC platform [1, 2] addresses these problems by executing each application in an independent virtual platform. It also uses the SDF^3 design flow [5] that automatically analyzes firm real-time applications and maps them to resources in a virtual platform, according to their resource and timing requirements. The CompSoC virtualization technology relies on two complexity-reducing concepts: *predictability* and *composability*, detailed as follows.

This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by the EU ARTEMIS JU funding, within the RECOMP project, ref. ARTEMIS/0202/2009, JU grant nr. 100202.

All software and hardware components are designed to make the virtual platforms predictable, which means that all platform and application interference is bounded. This makes the virtual platforms *virtualized in terms of performance bounds*, such as upper bounds on latency or lower bounds on throughput. This enables compositional verification of firm real-time applications using formal performance analysis frameworks, such as data-flow analysis [5].

Composable virtual platforms are completely isolated and cannot affect each other's temporal behaviors by even a single clock cycle. They are hence *virtualized in terms of actual performance*, in particular actual execution time, and actual energy and power usage. This is intuitively achieved by using delays to enforce worst-case behavior of all software and hardware components [1]. Each application is given a fixed virtual execution platform consisting of multiple processors, memories, and interconnect, as well as a virtual battery (energy/power) budget on the processor. This enables applications to be designed, verified, and executed in isolation. This alleviates the verification problem of simulation-based approaches in three ways: 1) verification becomes a non-circular process, 2) the verification process can be incremental and start as soon as the first application is available, 3) the time required by simulations is reduced, since only a single application in its virtual platform has to be simulated.

Predictability and composability are hence complementary concepts that both solve important parts of the design and verification problem for mixed time-criticality systems, and provide a complete solution when combined.

Our contribution consists of two demonstrators. First, we demonstrate the SDF^3 design flow that maps an H.263 decoder to an FPGA instance of the CompSoC platform, while meeting its firm real-time requirements. Second, on the same instance of CompSoC, we demonstrate composable execution for two distributed applications sharing processors, interconnect, and memories, without interference in terms of execution time or energy consumption. Measurements on the FPGA implementation show that the applications in both demonstrators meet their real-time requirements.

References

[1] B. Akesson *et al.* Composability and predictability for independent application development, verification, and execution. In *Multiprocessor System-on-Chip — Hardware Design and Tool Integration*, chapter 2. Springer, 2010.

[2] A. Hansson *et al.* CoMPSoC: A template for composable and predictable multi-processor system on chips. 14(1), 2009.

[3] H. Kopetz *et al.* The time-triggered architecture. 91(1), 2003.

[4] STMicroelectronics and CEA. Platform 2012: A Many-core programmable accelerator for Ultra-Efficient Embedded Computing in Nanometer Technology, 2010. White paper.

[5] S. Stuijk *et al.* SDF^3: SDF For Free. In *Proc. ACSD*, 2006.

[6] C. van Berkel. Multi-core for Mobile Phones. In *Proc. DATE*, 2009.