# Virtual Platform Based Development Environments for Low Power, Mixed Level Safety Critical System

Duncan Graham and Larry Lapides
Imperas Software Ltd.
Oxford, United Kingdom
LarryL@imperas.com

Sören Schreiner and Kim Grüttner
OFFIS - Institute for Information Technology
Oldenburg, Germany

***Abstract*—Critical Real-Time Embedded Systems for industries such as railway, aerospace, automotive and energy face multiple challenges including the growing need to support mixed-criticality applications, power and timing restrictions, and a need to develop and test these complex devices and the accompanying software. The approach adopted by the SAFEPOWER project was to develop a SoC architecture including a NoC, plus the hypervisor to support spatial and/or temporal isolation of the various functional units. Advantages of this hardware/software architecture include the increased isolation provided by using both spatial and temporal isolation and the adaptability of this architecture to changing conditions. This paper discusses the virtual platform methodology employed by SAFEPOWER. Unique tools developed to provide observability into the hypervisor-based system are described, as well as the methods for providing timing and power estimation with sufficient accuracy.**

***Keywords—Virtual Platform, SAFEPOWER, Safety Critical, Low Power***

## I. INTRODUCTION

Critical Real-Time Embedded Systems (CRTES) for industries such as railway, aerospace, automotive and energy generation face multiple challenges including the growing need to support mixed-criticality applications, power and timing restrictions and a need to develop and test these complex devices and the accompanying software. Currently these systems use multiple devices, which enables a divide-and-conquer approach to the specifications but does not reduce the overall complexity of the system, and the multiple devices limit the power reduction that can be achieved.

The obvious hardware approach is to use a single device. Integrating and partitioning mixed-criticality applications on a single target device can save costs and overall power consumption. The question of overall device architecture to support mixed-criticality systems, including both the hardware and software, remains to be answered.

One approach to this is to use time-triggered hypervisor management for separate control of tasks and functions with different levels of criticality. These systems can then both adapt to changing conditions and deliver the required performance with the appropriate level of safety on tasks with varying performance and safety requirements. This is the approach adopted by the SAFEPOWER project [1].

The SAFEPOWER initiative enables the development of mixed-criticality systems with low power in combination with safety and real-time advantages by providing a reference architecture, platforms and tools to facilitate the development, testing and validation of such systems. The SAFEPOWER reference architecture [6, 7] addresses the integration and partitioning of mixed-criticality applications on a single device, with a target of reducing total power consumption by up to 50% compared to a non-integrated multi-chip implementation. Example systems included within the SAFEPOWER project include aircraft flight controls, a quadcopter drone with auto tracking camera and auto flight controls, and a wireless train signal control box.

The SAFEPOWER project chose the Xilinx Zynq SoC FPGA as a hardware platform that was both well-known and flexible for the exploration and development of mixed-criticality systems. While the hardware platform is well-known, methodologies for the development, debug analysis and test of the

This work has been partially supported by the SAFEPOWER project with funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 646531.

individual applications running on hypervisors are not established, nor are the methods for analyzing and optimizing the fully integrated system. Hypervisors provide the separation of applications and tasks necessary for mixed-criticality systems, however, that very separation makes hardware-based software development challenging due to the lack of visibility into and across the hypervisor containers.

A software simulation approach was chosen by the project to provide the needed visibility. To optimize the power efficiency of the end system the software methodology needs to include considerations for both timing and power estimation. Current cycle accurate simulation techniques, while accurate, are considered too slow for software development. Using a design flow based on instruction accurate virtual platforms allows rapid software development and includes the power and timing estimates based on using profiles based on actual hardware observed results.

This paper discusses the virtual platform based methodology implemented by the SAFEPOWER project for developing mixed-criticality systems based on time-triggered hypervisor management. Unique tools specifically developed to enable the virtual platform to provide the needed observability into the hypervisor-based system operation are described, as well as the methods for providing timing and power estimation with sufficient accuracy in the instruction accurate simulation environment. The test results show a useful correlation that allows for software development to include and anticipate the power impact throughout the design cycle.

## II. SAFEPOWER ARCHITECTURE

The architecture of the SAFEPOWER platform (hardware and virtual platform) is shown in Fig. 1. A tile-based architecture was used, with the tiles connected via a time-triggered Network on Chip (NoC). The tiles are managed by a Type 1 hypervisor [2]. This allows for an arbitrary number of user partitions. There is also a special partition on the hypervisor for monitoring and controlling power management services, and the hypervisor includes support for time-triggered task management.

The SAFEPOWER SoC includes both the hypervisor tile and bare metal tiles. The bare metal tiles are connected to the time-triggered NoC, and implement a light version of the hypervisor application interface. The time-triggered behavior is

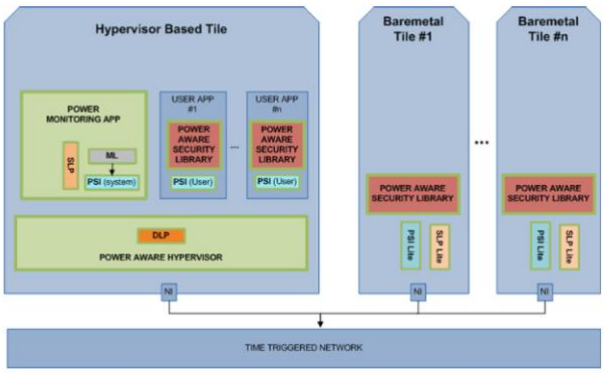


Fig. 1. SAFEPOWER reference architecture [7].

executed based on a pre-computed communication schedule that triggers the message injection times [8].

Architecting the SoC in this manner achieves both the spatial isolation and temporal independence required in safety standards such as IEC-61508. This time-triggered architecture provides deterministic scheduling of software tasks, with Worst Case Execution Time (WCET) analysis supporting the achievement of timing requirements.

### A. *SAFEPOWER Hardware*

The SAFEPOWER project chose to use a Xilinx Zynq 7000 board. The Zynq 7000 device includes an Arm Cortex-A9MPx2 (dual core) Processor Subsystem (PS), plus the Programmable Logic (PL) FPGA fabric. The PL was used to create the NoC communication network and MicroBlaze based tiles (subsystems). Each tile was comprised of the MicroBlaze processor, memory, a NoC communication node and power management logic. Two tiles were typically implemented in the hardware PL, while number of tiles was not limited on the virtual platform. Various I/O formats and protocols are also supported on the Zynq. The platform hierarchy and NoC are shown in Figs. 2 and 3.

There is fixed connectivity between the PS and PL, with address mapped memory and GPIO used.

Key pieces were the power monitoring components for obtaining feedback of the real-time current and voltage values and the voltage and clock control. These allowed the implementation of Dynamic Voltage and Frequency Scaling (DVFS) for dynamically changing the system voltages and the processor clock frequency in order to minimize power consumption.

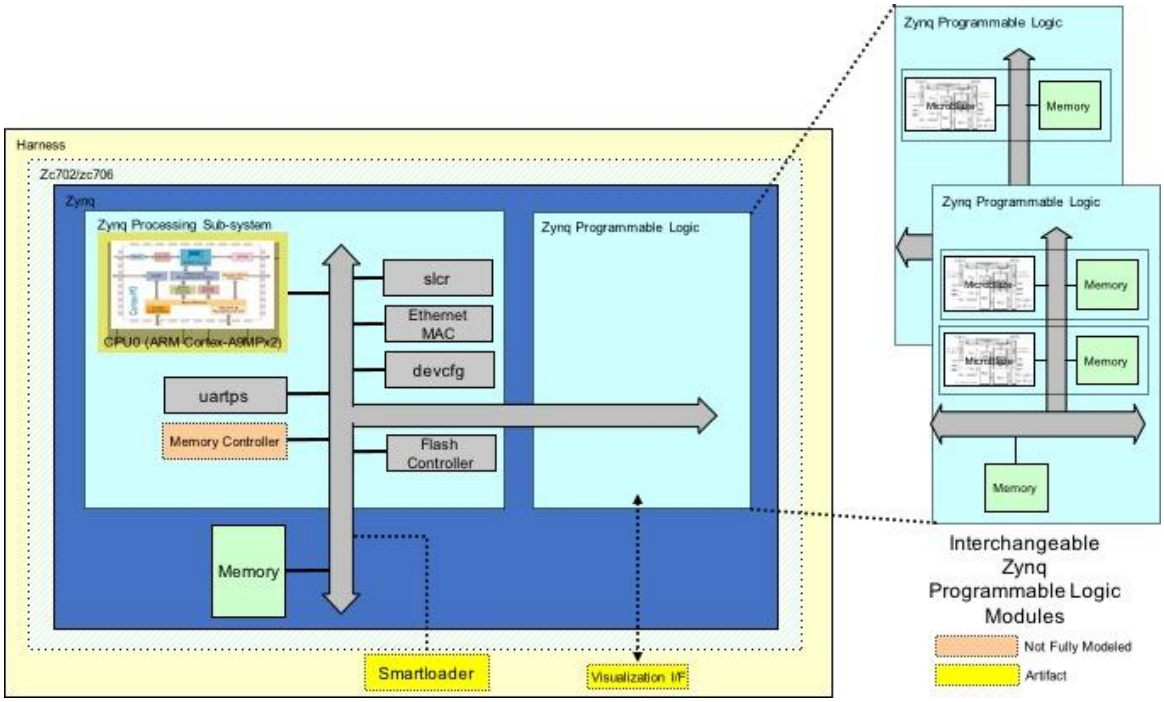


Fig. 2. Platform hierarchy block diagram [7].

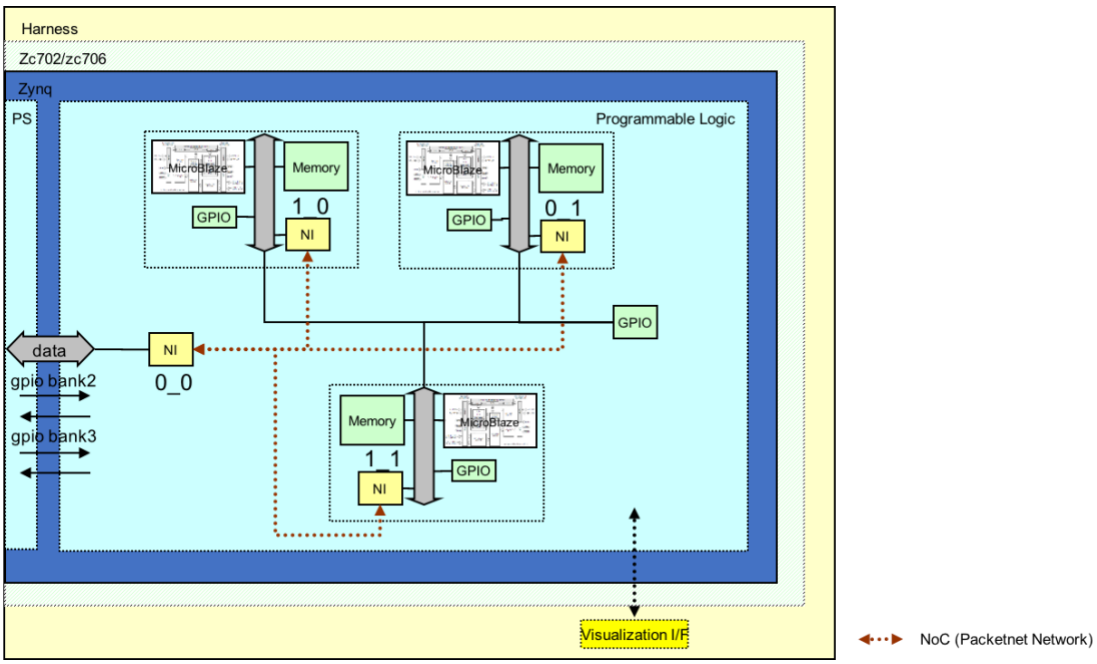


Fig. 3. NoC for the platform [7].

## III. VIRTUAL PLATFORM TECHNOLOGY

### A. *Instruction Accurate Software Simulation*

Just-in-time (JIT) code translating simulators are now widely recognized to be the fastest and most powerful tools for development of instruction accurate virtual platforms. A single-threaded JIT-based simulator is typically capable of delivering simulation performance of billions of simulated instructions per second. Virtual platform simulators operate by dividing time into *quanta*, of fixed or variable size, specified by the platform user. For multiple processor platforms, each processor is simulated in turn for a quantum. When all processors

have finished the quantum, time is advanced and simulation resumes for the first processor in the next quantum.

### B. *Running Software on the Virtual Platform*

One of the key advantages of the instruction accurate virtual platform is that it can execute the same binaries that would execute on the hardware. Software is compiled using the same toolchain and libraries to generate ELF and binary files that can be loaded and executed on the virtual platform. The JIT simulator translates the Arm, MIPS, RISC-V, etc. instruction to a x86 instruction(s) to be executed on the host PC.

During this translation phase, additional x86 instructions can be added to support analysis of the software application executing using a binary interception technique. Because the analysis tools are added during this translation process, the tools are "non-intrusive", in that there is no instrumentation or modification of the source code, and no change to the order of instruction execution for the binary.

## IV. Virtual Platform Based Tools for Software and System Analysis

The virtual platform environment used for this project was the Imperas M*SDK [3] product, utilizing models built using the Open Virtual Platforms (OVP) APIs and model library [4].

### A. *Open Virtual Platforms Models*

Open Virtual Platforms is a website containing the OVP APIs, the OVP model library and a reference simulator, OVPsim. Where there were existing models in the OVP library for components, such as for the Arm Cortex-A9 and Xilinx MicroBlaze processors, those models were used. Otherwise, new models were built, including the network interface and various peripherals. The overall platform model was hierarchical, enabling easy debugging of individual tiles prior to assembly into the complete platform.

The OVP models are open source, distributed under the Apache 2.0 open source license.

The virtual platform models are instruction accurate versions of the actual hardware used.

### B. *MultiProcessor Debugger*

The M*SDK product includes a MultiProcessor Debugger (MPD), which supports debug of heterogeneous multiprocessor platforms, and also supports introspection of peripheral components, all within a single debug session. This introspection enables, for example, the setting of breakpoints on the registers of a peripheral, allowing for the co-debug of peripherals and drivers.

MPD also allows viewing the software at various levels of abstraction, including hypervisor, operating system, drivers, firmware and user applications.

MPD can be run from the command line, or controlled using an Eclipse user interface.

### C. *Verification, Analysis and Profiling Tools*

The M*SDK Verification, Analysis and Profiling (VAP) tools include features such as tracing (instruction, function, variable, register changes, …), code coverage, profiling, memory monitoring, protocol checking and assertion checking. OS-aware tools are also included, enabling OS task tracing and scheduler analysis. Users can also create custom analysis tools using the binary interception ("SlipStreamer") API.

### D. *Timing Estimation*

While the virtual platform is instruction accurate, not cycle accurate, timing estimation can be achieved by annotating the instructions being executed using the binary interception technology [5]. This timing estimation capability certainly has limitations, such as not supporting out-of-order pipelines. However, it can provide estimates to an accuracy of +/- 10%, and supports timing analysis for individual instructions, data-dependent instructions, sequence-dependent instructions, different latencies for different memory regions and delays due to cache hit/miss analysis.

### E. *Power Estimation*

Power estimation is achieved in much the same way as timing estimation, by annotating the instruction stream [9].

## V. Results

Both railway and avionics use cases were tested. The avionics system is shown in Fig. 4. The objectives of the testing were to 1) demonstrate that the software components interact correctly with each other and with the hardware; 2) verify that the entire system complies with the requirements – functional and extra-functional, also when including low power techniques – of the use case specification; and 3)

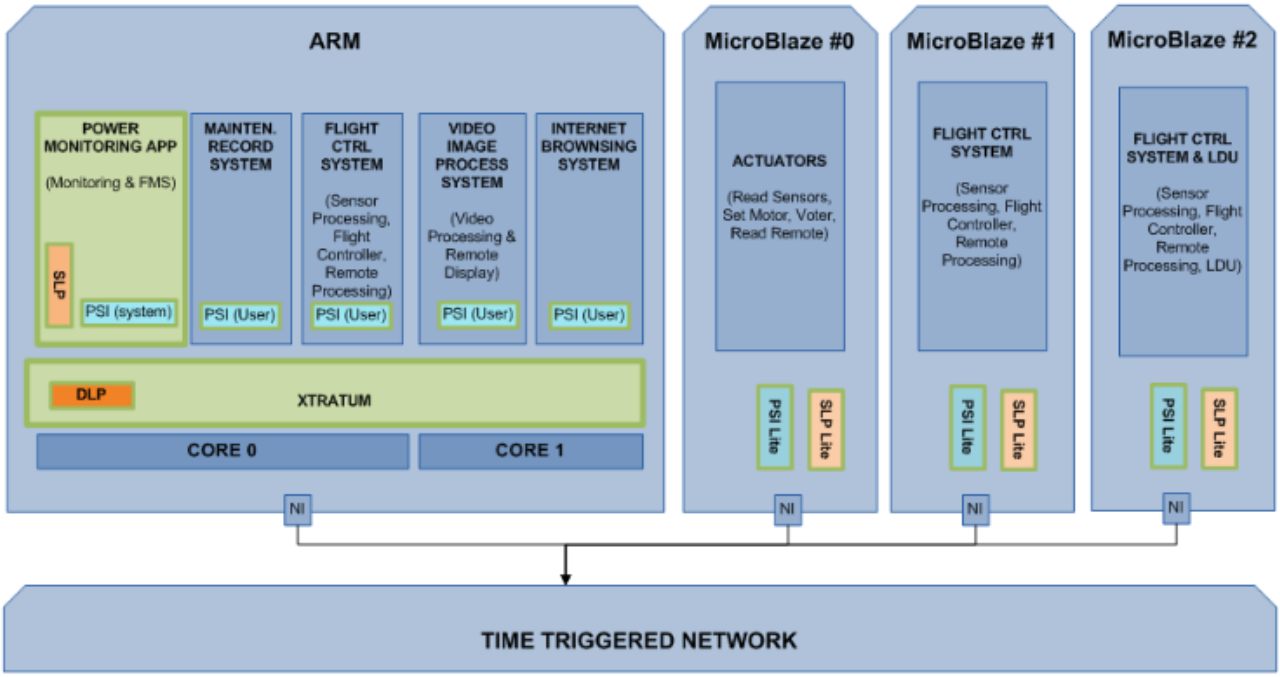


Fig. 4. Avionics use case block diagram [7].

show that power savings can be achieved for safety critical applications in different domains.

Key challenges for the virtual platform environment include the modeling of the DVFS subsystem, fault injection testing and supporting the time-triggered system. These are discussed below.

For the tests that could be run on both the virtual platform and the hardware, the results were equivalent.

### A. *DVFS Subsystem*

Dynamic Voltage and Frequency Scaling (DVFS) is, as the name implies, a dynamic process. The power estimation cannot be done by post-processing the instruction trace; instead, the power estimation has to be done in real time in the simulation environment. Then the DVFS module [10, 11] must do the calculations and make any adjustments, also in real time.

The block diagram of the virtual platform is shown in Fig. 5. The DVFS subsystem is implemented in a binary interception library, with the power model as a module in the intercept library. The power sensors are real data input into the simulation via an I2C peripheral model.

Fig. 6 shows the subsystem at work, with the virtual platform executing Linux at an initial frequency, then calculating the power consumption and changing the operating frequency of the simulated platform by changing the simulation speed of the processor.

### B. *Fault Injection Testing*

Fault injection is a common and necessary testing method for embedded systems in high reliability systems, including the safety critical systems considered in the SAFEPOWER project. The fault injection framework needs to be able to inject different types of faults, and also to inject the faults at different times.

The types of faults supported in the framework include memory corruption, GPIO corruption, faults in the switch partition scheduler, in resetting of the various processors and in uncontrolled interrupts. The implementation was done via an intercept library, with a simple user interface built to help control the fault campaigns. This user interface is shown in Fig. 7.

### C. *Support for Time-Triggered System*

The default scheduler for the simulator is a “round robin” scheduler, which executes a quantum of instructions on one processor, then moves to the next to do the same, and continues in that manner until resuming the instruction execution on the first

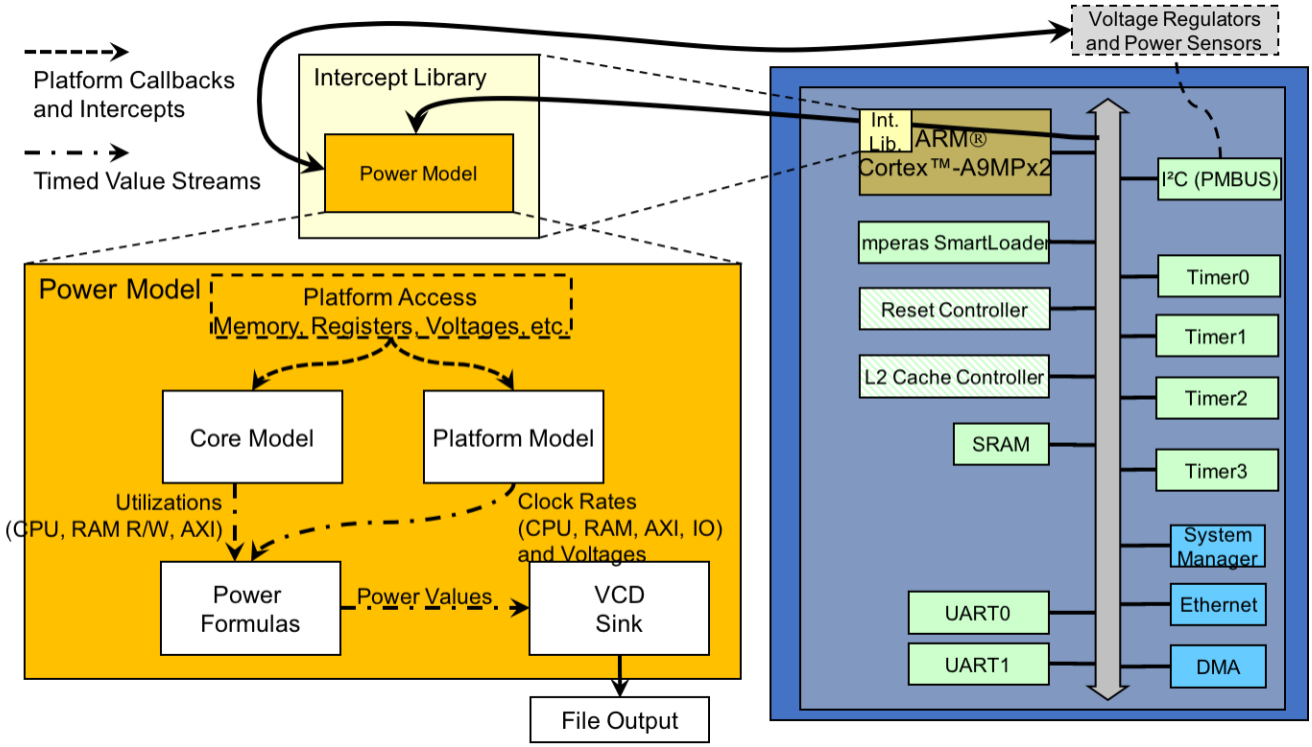


Fig. 5. Block diagram of the DVFS subsystem.

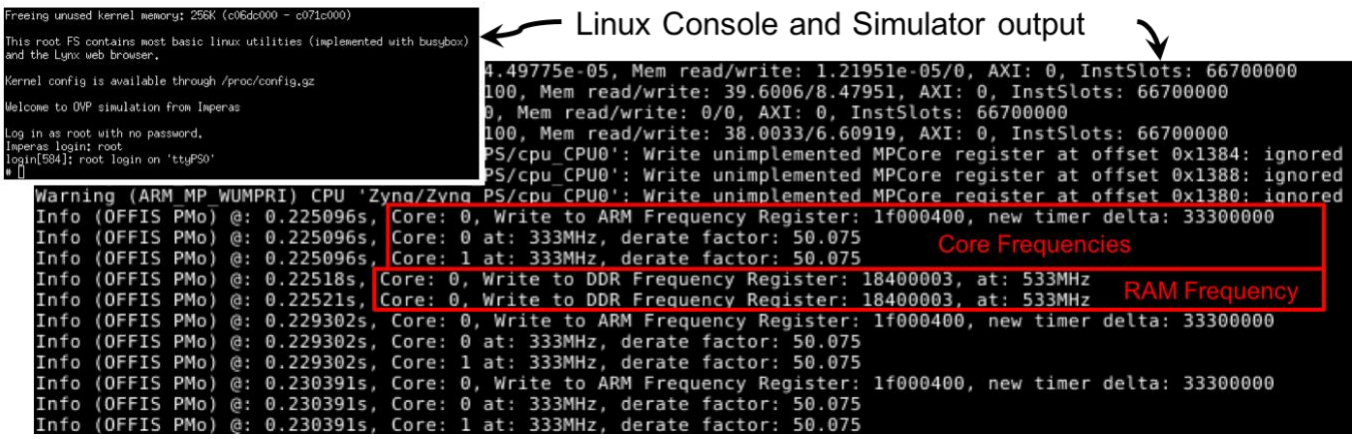


Fig. 6. DVFS subsystem results, showing a reduction in operating frequency to reduce power consumption.

processor. This provides functionally correct results, however, it provides little in the way of timing reference points. The SAFEPOWER system is a time-triggered system, with the synchronization defined statically. These time triggers do not always align with the boundaries of the quanta of instructions being executed.

The solution was to develop a new scheduler for the simulator [12], a time-triggered scheduler, with which applications are executed until the next event. Using this scheduler, execution is scheduled to complete a task, and the scheduler can detect the synchronization points in the code.

The comparison between quantum-based scheduling and time-triggered scheduling is shown in Fig. 8.

## VI. Conclusions

Since the results for virtual platform based testing were equivalent to those for hardware based testing, and since the virtual platform testing used the same

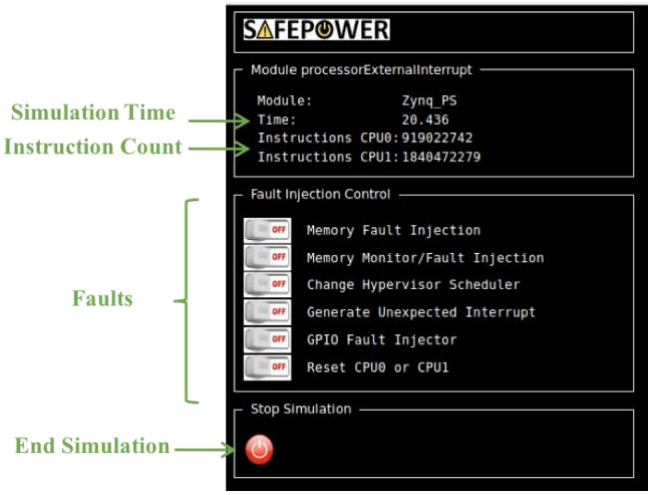


Fig. 7. Fault injection user interface for configuration of fault injection campaigns.

binaries, virtual platform testing can be utilized in the test and certification of safety critical systems.

Additionally, the virtual platform provided benefits over the use of the hardware platform for the development, debug, analysis and verification of software. These benefits included

- Execution control: Simulation is deterministic
- Unified debug environment: Simultaneous debug of all application code executing on all processors in the platform, including access to peripherals
- Non-intrusive analysis tools: Tools such as profiling, code coverage, dynamic assertions, etc. are implemented with no modification or instrumentation of source code
- Power Interface Library, implemented using Imperas SlipStreamer API (binary interception), enabled support for real time power management techniques such as DVFS within the virtual platform environment
- Fault injection: The virtual platform provides visibility and observability, so faults can be injected anywhere in the platform, e.g. memory and processor registers. Fault injection is implemented by an external library, so fault generation can be deterministically controlled.

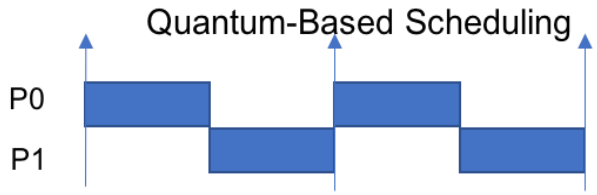


Fig. 8a. Quantum-based scheduling for the virtual platform.

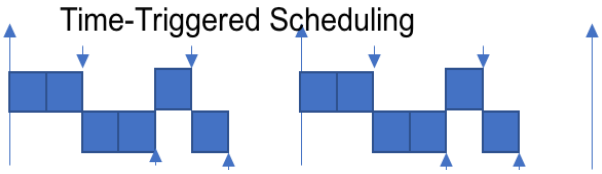


Fig. 8b. Time-triggered scheduling for the virtual platform.

## ACKNOWLEDGMENT

The authors wish to give thanks to all the organizations that participated in SAFEPOWER, both universities and commercial companies.

## REFERENCES

[1] SAFEPOWER project website: https://safepower-project.eu/

[2] T. Poggi et al., "A Hypervisor Architecture for Low-Power Real-Time Embedded Systems," 2018, 21st Euromicro Conference on Digital System Design (DSD), https://doi.org/10.1109/DSD.2018.00054

[3] Imperas M*SDK see http://www.imperas.com/products

[4] Open Virtual Platforms (OVP) Library: http://www.ovpworld.org/library

[5] L. Moore, D. Graham, S. Davidmann, F. Rosa, "Cycle Approximate Simulation of RISC-V Processors", Embedded World 2018. http://www.imperas.com/presentations

[6] M. Fakih et al., "SAFEPOWER project: Architecture for safe and power-efficient mixed-criticality systems", Microprocessors and Microsystems, 2017, https://doi.org/10.1016/j.micpro.2017.05.016

[7] M. Fakih et al., "Experimental Evaluation of SAFEPOWER Architecture for Safe and Power-Efficient Mixed-Criticality Systems", Journal of Low Power Electronics and Applications, Special Issue "Ultra-low Power Embedded Systems", 2019, https://dx.doi.org/10.3390/jlpea9010012

[8] R. Obermaisser et al., "Adaptive Time-Triggered Multi-Core Architecture", Journal of Designs, Special Issue "Challenges and Directions Forward for Dealing with the Complexity of Future Smart CPS)", 2019, https://doi.org/10.3390/designs3010007

[9] R. Görgen, D. Graham, K. Grüttner, L. Lapides, S. Schreiner, "Integrating Power Models into Instruction Accurate Virtual Platforms for ARM-based based MPSoCs", ARM TechCon 2016

[10] S. Schreiner, M. Fakih, K. Grüttner, D. Graham, W. Nebel and S. P. Frasquet, "A Functional Test Framework to Observe MPSoC Power Management Techniques in Virtual Platforms," 2017 20th Euromicro Conference on Digital System Design (DSD), https://doi: 10.1109/DSD.2017.14

[11] S. Schreiner, R. Seyyedi, M. Fakih, K. Grüttner, W. Nebel, “Towards power management verification of time-triggered systems using virtual platforms”, SAMOS '18: Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, 2018, https://doi.org/10.1145/3229631.3235025

[12] R. Seyyedi, S. Schreiner, M. Fakih, K. Grüttner and W. Nebel, "Functional Test Environment for Time-Triggered Control Systems in Complex MPSoCs Using GALI," 2018 21st Euromicro Conference on Digital System Design (DSD), https://doi.org/10.1109/DSD.2018.0001