A Hypervisor Architecture for Low-Power Real-Time Embedded Systems

Peio Onaindia*, Tomaso Poggi*, Mikel Azkarate-askatsua*, Kim Grüttner†, Maher Fakih†, Salvador Peiró‡, and Patricia Balbastre‡

*Embedded Systems Group - IK4-IKERLAN, Mondragón, Spain Email: {ponaindia, tpoggi, mazkarateaskasua}@ikerlan.es

†OFFIS Institute for Information Technology, Oldenburg, Germany Email: {gruettner,fakih}@offis.de

‡Fent Innovative Software Solutions, Valencia, Spain Email: {speiro,pbalbastre}@fentiss.com

***Abstract*—This paper presents a hypervisor architecture tailored to low-power real-time applications. This architecture extends the capability of a hypervisor by providing power management techniques and power monitoring services. An implementation based on an existing hypervisor XtratuM that runs over the ARM of a Zynq-7000 SoC device is proposed as a proof of concept. Measurement results show that the extended hypervisor can obtain information on the power consumption and reduce it.**

***Index Terms*—real-time, hypervisor, low-power, Zynq, ARM**

I. Introduction

The integration of real-time applications is becoming fundamental in the development of complex embedded systems. In some domains, this integration is often required to satisfy non-functional requirements related to cost, weight or power consumption [1], [2]. Multicore processors using hypervisors as software execution environments are one possible solution for many systems which require significant computational power from the underlying platform [3]. An Hypervisor (also known as virtual machine monitor, VMM) is a system software that allows running several independent execution environments in a single computer introducing a very low overhead. Therefore, hypervisors are a promising technology even though they have to be adapted and customised to the requirements of the target application [4]. When a hypervisor is designed for real-time embedded systems, the main issues that have to be considered are: temporal and spatial isolation, basic resource virtualisation (clock and timers, interrupts, memory, CPU time, serial I/O), real-time scheduling policy, deterministic hypervisor system calls, efficient inter-partition communication, efficient context switch, low overhead and low footprint.

At the same time, the increasing interest in integrating applications on the same processor, led to a branch of new design concepts like the ones recommended by the ARINC-653 standard [5] in avionics domain or the AUTOSAR standard in automotive domain. Nowadays hypervisors are adapting these design concepts from different standards to allow their usage in different domains [6].

In this paper, we propose an extended hypervisor to tackle the needs of the power management in a heterogeneous multicore platform. The extensions require relevant hypervisor design modifications that impact the scheduling policies, reconfiguration management and hardware mechanisms. The virtualisation layer proposed, integrates monitor mechanisms to deal with power, energy and temperature measurements. The monitoring service implements power and temperature measurement techniques in the hardware. The extended hypervisor provides two ways of reacting to changes in the monitored variables: i) automatic reactions of the platform based on reconfiguration of precomputed modes; ii) by providing notifications to user applications that can implement the desired countermeasures. Moreover, the proposed virtualisation layer, enables power, energy and temperature management services, while taking into consideration at the same time, certification, real-time and time/space partitioning constraints. The virtualisation layer extends the state of the art towards a new hypervisor architecture that implements power management and monitoring services and permits the integration of scheduling policies to deal with power aware scheduling.

The rest of the paper is organised as follows. Section II presents the state of the art of the hypervisors focusing on power management extensions; Section III gives a short overview about low-power services supported by nowadays hardware architectures; Section IV presents the proposed extension of the system software architecture and its corresponding realisation on top of a specific hypervisor: the XtratuM hypervisor; Section VI introduces the test setup to evaluate the proposed architecture and shows the obtained measurement results.

II. Related and Preliminary Work

Several commercial RTOS products conform to the ARINC-653 standard and also offer commercial virtualisation products for safety-critical systems, many by adapting existing RTOS products. In [7], a survey on real-time issues in virtualisation for embedded systems is presented where several hypervisors for real-time are detailed. For example, LynuxWorks implements a virtualisation layer to host their LynxOS product, called the LynxSecure hypervisor that supports MILS and ARINC-653. WindRiver provides a hypervisor product for

both its VxWorks MILS and VxWorks 653 platforms, including support for multicore. The RT-Xen project implements a compositional and hierarchical scheduling architecture based on fixed-priority scheduling within Xen, and extensions of compositional scheduling framework and periodic server design for fixed-priority scheduling. L4 is a representative microkernel operating system, extended to be a Type-1 virtualisation architecture. XtratuM is a Type-1 hypervisor targeting safety critical avionics embedded systems. There are also adaptations to provide real-time capabilities to KVM. KVM is a Type-2 virtualisation solution that uses Linux as the host OS.

Regarding hypervisors with low-power features, only few works can be mentioned addressing these features. Xen hypervisor provides two hypercalls, which are platform hypercall XENPF_change_freq and XENPF_getidletime, to assist the domain0 kernel to get the system status and also change the CPU frequency [8]. In [9] and [10] a performance comparison in terms of power consumption of KVM and Xen is presented but in the framework of cloud data centres. In the same framework, [11] explores how to integrate power management mechanisms and policies with virtualisation technologies. However, and to the best of our knowledge, low-power management in real-time embedded hypervisors is not a topic addressed in the literature.

A. The XtratuM Hypervisor

The XtratuM hypervisor [12], [4] is a bare-metal hypervisor for embedded real-time systems that has been used in several EU projects and it is currently being used on several space missions. Initially developed for LEON processors, it has been adapted to other platforms such as ARM Cortex R4/R5, A9 and PowerPC. In the FP7 EU MultiPARTES project [13], XtratuM was adapted to be used in multicore heterogeneous platforms. The FP7 EU DREAMS project [14] extended XtratuM to work on multiple multicore nodes connected through NoC (Network on Chip) and TTEthernet networks.

XtratuM was specifically designed for embedded real-time systems that makes use of para-virtualisation techniques to closely emulate hardware behaviour. The design of XtratuM has been guided by the ARINC-653 avionics standard, which defines a software specification for safety critical real-time operating systems regarding time and space partitioning (TSP). Some of the main traits of XtratuM are:

- XtratuM replaces conflicting processor instructions with hypervisor-specific code (hypercalls) that provide a set of high-level services based on ARINC-653 to the partitions. Software running on XtratuM is hypervisor-aware, and may need to be adapted to yield control to the hypervisor by means of the hypercalls in order to interact with the underlying hardware.
- XtratuM manages those hardware interrupts that are susceptible to compromise isolation, leaving to partitions the management of non-critical devices. Hardware interrupts can be allocated to a given partition. Additionally, XtratuM provides a series of extended interrupts intended to inform partitions of XtratuM specific events.
- According to the ARINC-653 standard, XtratuM implements a cyclic scheduling policy. A cyclic scheduling plan statically defines processor time allocation to each partition, enforcing time isolation and providing deterministic behaviour. Partitions are able to individually implement an internal scheduling algorithm.
- The allocation of the available hardware resources (memory areas, scheduling, communication ports, etc.) to partitions, as well as the configuration of the virtualized devices and the specification of the scheduling plan, are statically defined via a configuration file.
- XtratuM provides robust message passing based mechanisms for inter-partition communication based on ARINC-653 defined queuing and sampling ports. The hypervisor implements a logical path between source and destination ports (channels) and is responsible for encapsulating and transporting the messages.
- In order to detect and manage unexpected events, a Health Monitor is provided as a mechanism to properly deal with faults that can not be handled at the scope where they take place. The Health Monitor defines a series of preconfigured actions aimed to contain faults and minimise their impact on the system that are executed as soon as an error is detected.

III. Low-power Techniques (LPT)

This section summarises power management techniques (based on [2]) that can be used to provide a hypervisor, in particular XtratuM, with low-power features. The techniques are presented independently from a specific hardware platform (the applicable LPT on the Xilinx Zynq platform will be introduced in Section VI). In addition, monitoring techniques enabling to monitor system status are presented.

A. Overview of power management techniques

Especially for battery-operated embedded devices, energy saving is of vital impact. In addition, applying power management techniques reduces heat dissipation which in turn increases the long-term availability and reduces cooling equipment costs. Furthermore, with the help of LPT, resource usages can be also optimised (for e.g. by shutting down resource when not used) leading to an overall cost reduction. The state-of-the-art encompasses a broad spectrum of LPT which we will briefly review in the following (for a detailed survey c.f. [15]).

The supply voltage is an important factor for both dynamic (decreases quadratically by voltage decrease) and static power (decreases linearly). Due to that, several techniques exist which manipulate the supply voltage and threshold voltage dynamically or statically to reduce the voltage swing of switching transistors and the total number of switching transistors in the design. For e.g. *different supply voltages (Multi-Voltage)* can be used for different components in combination with level shifters. DVFS [16] is another technique where a power manager controls different power modes, consisting of a pre-defined set of supply voltage and clock frequency tuples. The idea here is to find the ideal combination of the

clock frequency and supply voltage for achieving lower power consumption while still fulfilling real-time requirements.

Instead of scaling down the supply voltage, it can be switched off completely (*power gating*) if the switched-off parts are not used over a longer period of time. Points of consideration are the high energy costs and delays for shut down and start up, the need for isolation cells and state retention registers. Alternatively, clock gating disables the clock for complete system blocks or selectively suspends clocking. It requires less effort than power gating, but only controls the dynamic power consumption, while power gating also attacks the static leakage power, that may have a considerable impact on the overall power consumption.

Table I summarises the main LPT with targeted power source and possible disadvantages.

Power management methods can be of static or dynamic nature. *Dynamic Power Management (DPM)* (also called Power Mode Management: PMM) [17] reduces energy consumption by the utilisation of different low-power modes (e.g., idle, sleep, stand-by) which are realised through combination of LPT supported by the underlying hardware. In every mode, different energy budgets and response times are needed. The intelligent management of transitions between different modes is done at run-time based on system state.

Other LPT include *micro-architectural techniques* for specific components of the MPSoC. One proposal here is to use small architectures (e.g. with scratch-pad memory instead of caches) targeting a less static power consumption. Other micro-architectural techniques utilise run-time parameters (e.g. workload) to apply dynamic reconfiguration of specific components for saving energy. Examples are (c.f. [15]): selectively clock gated caches, effective cache reconfiguration, memory compression or usage of appropriate cores (GPUs, FPGA, ASICs, DSPs, etc.).

B. Monitoring Techniques

The basic idea of the monitoring framework is to monitor the current system status to be able to change its behaviour according to the evaluated system status. The monitoring devices can be divided in three categories:

- *I2C devices*: many voltage regulators and temperature sensors provide information about rails and measurement points through a digital connection, mostly an I2C bus or a PMBus.
- *On-chip firmware devices*: they are particular circuits that can be implemented inside an FPGA to monitor the temperature at a location specified by the user. For instance, Ring Oscillators (RO) use the temperature-delay relationship of CMOS devices to infer the temperature by measuring the frequency of resonance phenomena [18], [19]. As an alternative to ROs, it is worth to mention sensors based on Flip-Flop-Metastability [20], even if they are not widely used.
- *Analogue-to-Digital Converters*: they can be used to acquire information on physical variables (temperature, voltage, current, etc.). For instance the XADC present inside each Zynq device is set up to monitor the voltage level on several power rails as a default factory configuration. Other rails or current measuring can be added at design phase by properly connecting the XADC input with the rails.

IV. Proposed architecture

In this section a safety-oriented power aware architecture for hypervisors is proposed making use of the low-power and monitoring services. Even though the final implementation is based on XtratuM hypervisor (see Section II-A), the architecture has been defined with enough abstraction level so that it can be implemented regardless of the platform or the underlying hypervisor. The suggested architecture, shown in the diagram in Figure 1, is composed by the elements explained in the remainder of this section.

A. Extended Hypervisor (DynamicLPT):

The basic component is an extended version of a hypervisor that integrates a subset of LPT, named as "DynamicLPT". These techniques take advantage of the dynamic slack time of the running application i.e. the idle time since a partition completed a periodic task until the next start of the task itself. On one hand these techniques allows the hypervisor to implement scheduling plans that take advantage of the dynamic behaviour of the running application. For example, the system frequency can be automatically reduced, or a core suspended, if a slack time is detected in a running task. On the other hand, they cannot affect safety, temporal and spatial isolation, i.e. the overall system reliability has preference with respect to the power reduction.

By configuring the dynamic LPTs it is possible to tune the power consumption at partition level, for instance by assigning a different frequency to each partition.

B. Static Low-Power Block (StaticLPT):

This software component placed outside hypervisor system software includes the static LPT (peripherals suspension or voltage scaling), that do not affect the hypervisor behaviour by interfering in the temporal and spatial isolation of the partitions. For instance, voltage scaling LPT is implemented in "SLP_StaticLPT" because, usually, it implies a few milliseconds delay that would break the temporal isolation. This timing overhead is introduced by the communication interface (for e.g. PMBus interface) used to communicate with the external power management devices as in the case of the Zynq MPSoC.

The static LPTs implemented in this block can be used to lower the power consumption of the overall application. For example it could be possible to scale the voltage of the CPU or disable some peripheral.

C. Power Services Interface (PSI):

This is the API that is used by the application developers to enhance their software with power aware services. This interface provides the monitoring services and the entry point to the use of the DynamicLPT integrated in the hypervisor. For

Table I
OVERVIEW OF LOW-POWER TECHNIQUES AND THEIR IMPLICATIONS

Low-Power Technique	*Targeted Power Source*	*Disadvantage(s)*
Voltage Scaling	Dynamic and Static	• Limited to manufacturing retention voltage • Increasing application execution time
DVFS	Dynamic and Static	• Limited to retention voltage/frequency • Finding lowest possible operating frequency and supply voltage is not easy • Increasing application execution time
Clock gating	Dynamic	• Ignores static leakage power
Power gating	Static	• High energy costs and delay for shut down and start up phases
Microarchitectural Optimisations	Static	• Ignores dynamic leakage power • High costs

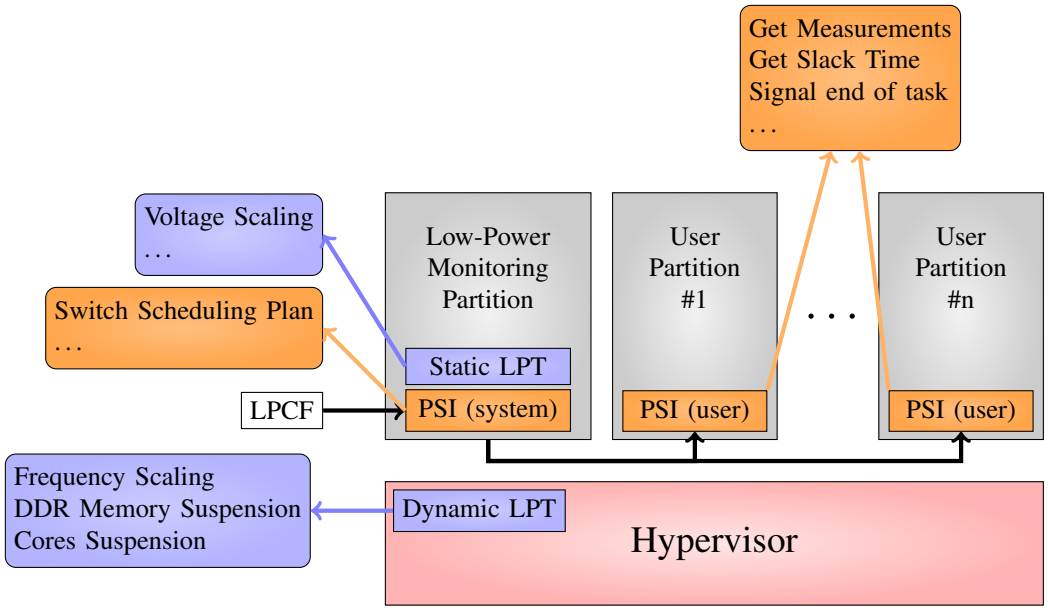

Figure 1. Schematic view of the proposed architecture and its extensions.

instance, thanks to functions like GET_CHANNELS_INFO and TASK_FINISHED the partitions will be able to receive information about power consumption and enable the hypervisor to suspend the core during partitions idle time respectively.

The behaviour of the PSI is defined by a *Low-Power Configuration File (LPCF)*. This file will be used to statically configure the extension of the hypervisor to provide power aware services. Indeed, it allows switching between static schedules and applying different LPTs depending on the schedules.

D. Low-Power Monitoring Partition (Monitoring Partition):

This special partition runs on the hypervisor with special system rights. It is one of the main components to extend the hypervisor with power awareness. Its main functionality is to monitor the power consumption by accessing the sensors and by providing the measurement results to the rest of the partitions via the mentioned PSI. The second functionality is to apply the static LPT via SLP. The power management services will be centralised only on this partition to assure that other partitions do not interfere between each other by accidentally applying LPT in critical circumstances. This decision implies that monitor partition should have application specific information to know when to apply which power reduction services.

V. ARCHITECTURE IMPLEMENTATION

In this section we describe the hardware platform and software configuration used to test the proposed architecture and LPT.

A. The hardware platform: Xilinx Zynq-7000 SoC

The XtratuM hypervisor has been implemented on the ARM of a Zynq-7000 device, although it can be implemented over other processors. This type of DSP is manufactured by Xilinx and include:

- A Processing System (PS), i.e. a two-core ARM microprocessor that runs XtratuM and the host applications.
- Input/output (I/O) peripherals interfaces (for instance, SPI, Ethernet, I2C).

Table II
POWER SUPPLY RAILS AND RECOMMENDED VOLTAGE LEVELS FOR THE ZYNQ-7000 SOC.

Type	Name	Nom. Voltage [V]	Description
PS	VCCPINT	1.0	Internal logic PS
PS	VCCPAUX	1.8	I/O buffer pre-driver
PS	VTTDDR	0.75	DDR memory interface
PS	VCCMIO	1.8–3.3	I/O peripherals
PS	VCCPLL	1.8	Three PLL clocks
PL	VCCINT	1.0	Internal logic PL
PL	VCCAUX	1.8	I/O buffer pre-driver
PL	VCCO	1.8–3.3	I/O buffers drivers
PL	VCCBRAM	1.0	PL block RAM
PL	VCCAUX_IO	1.8–2.0	PL auxiliary I/O circuits
XADC	VCCADC	1.8	ADC analogue power

- Programmable Logic (PL), that is basically a FPGA embedded into the device.
- Internal monitoring devices, in particular the XADC, an analogue-to-digital converter with an interface to the ARM cores.

Although only the first two elements are actually required for the implementation of the architecture of XtratuM, the Zynq platform has been chosen since it offers a base for wider research topics that are out of the scope of this paper. [1]. Moreover, the presence of the XADC and the PL allows the direct measure of voltages and temperatures inside the Zynq SoC itself. Another interesting point in the selection of the Zynq SoC is the availability of off-the-shelf test boards that already includes on-board memories, I/O connectors, measurement devices and other general purpose circuitry. In particular, the Xilinx ZC702 [22] has been used as a testbench for the proposed architecture.

The PS and PL power supplies are overall independent, however the PS power supply must be present whenever the PL power supply is active. The PS includes an independent power supply for the DDR I/O and two independent voltage banks for I/O peripherals. Moreover a separated power rail is dedicated to the XADC. The power supply rails are summarised in Table II.

B. Power and temperature control with the Xilinx Zynq-7000 SoC

The control and measure of power consumption of Zynq SoC instances via external devices is an active development area, see e.g. [23]. Indeed, power controllers and sequencers can be used to control the voltage levels and the power on sequence of different power rails. These devices are key components of any power control scheme, since they allow both to monitor the state (voltage, current, temperature) of voltage regulators and to undertake regulating actions. They are intelligent devices that host a microcontroller that runs the control and sequencing algorithms. Internal parameters can be accessed and modified by a serial bus connection, usually based on the PMBus standard.

One example of design employing these devices is the ZC702 evaluation board [22], that mounts 3 UCD9248 digital power controllers by Texas Instruments (TI) to monitor 5 switching voltage regulators, each of them with two channels, for a total of 10 power rails (sample rate of 5kHz). The information provided by these devices can be accessed by employing the PMBus interface of the device, connected to the I2C port of the Zynq SoC through a TI PCA9548 1 to 8 channels I2C bus switch. This hardware configuration is tailored for online measuring and controlling.

A similar solution can be found in the ZC706 Xilinx evaluation board [24], that mounts a Zynq SoC too. In this case, the board hosts a UCD90120A power supply sequencer and monitor together with a LMZ31500 and a LMZ31700 voltage regulators.

Temperature measurements can be performed both by accessing external devices on the I2C bus (e.g. UCD9248, UCD90120A) or by resorting to internal resources like the Xilinx ADC (XADC) or Ring Oscillators implemented in the PL. The XADC is an ADC embedded into all Zynq devices that can be used for monitoring applications on the Zynq SoC with a sampling rate of 1 MSPS Million samples per second. The XADC has 16 input multiplexed channels. Five of them are dedicated to measuring internal voltages and temperature: VCCINT, VCCAUX, VCCBRAM, VCCDDR and the device temperature can be always monitored in any Zynq device. Each of these sensors can be configured to hold minimum/maximal thresholds which when violated during runtime alarm signals can be issued.

C. Architecture implementation: XtratuM on a Zynq platform

The XtratuM hypervisor has been modified to instantiate the architecture proposed in Section III. The mapping has been straightforward: the PSI has been renamed as extended Dreams Abstraction Layer (xDRAL) and the Low-Power Monitoring Partition as Monitoring Partition. XtratuM already supported ARM Cortex A9 processor so it was decided to use the Processing System of a Zynq-7000 SoC.

XtratuM has been extended with the monitoring services that collect information from the external devices of the ZC702 evaluation board and the previously mentioned XADC. The LPT integrated have been those ones supported by ZC702 Xilinx board. Frequency scaling, DDR suspension and core suspension (power gating) are the "dynamicLPT" implemented; the ones that affect the hypervisor behaviour and take advantage of slack time. Due to the shut down and start up time required, voltage scaling and programmable logic power gating have been included as "SLP_StaticLPT".

Moreover, a tool, called Xoncrete, has been implemented to ease the inclusion of Power Mode Management. It creates automatically different scheduling plans, six low-power profiles, based on the concept of frequency scaling and the input provided by the designer regarding worst case execution time and criticality of the partition. Criticality definition is

[1]Notice that this work has been founded by the European project SAFE-POWER, that aims at the development of a concept hardware-software architecture that relies on both the ARM cores and the PL [21], [2]

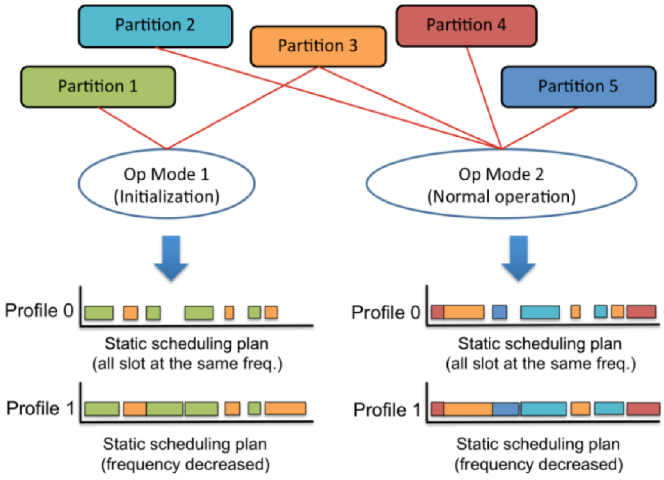

Figure 2. Scheduling example with 5 partitions, 2 Operational Modes and 2 profiles.

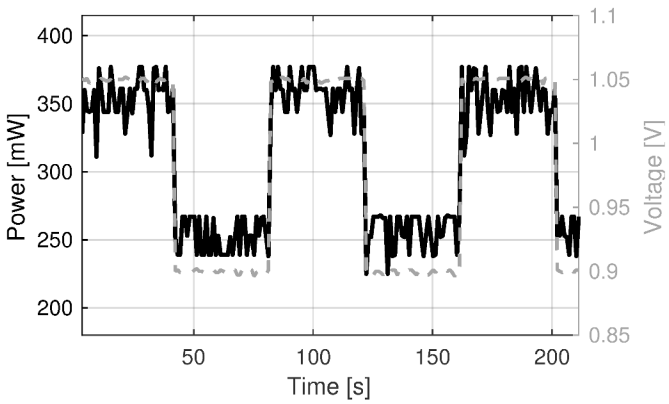

Figure 3. Experimental measurement obtained by applying *voltage scaling*. This figure shows the evolution of the power consumption and of the power rail voltage of the ARM cores over time (rail VCCPINT in Table II). The ARM power consumption is drawn as a black continuous line; the voltage as a grey dashed line.

required because in some cases the energy saving will be done at the expense of a performance loss like halting a partition. Based on the criteria defined during architecture description, the changes between different low-power profiles or schedules precomputed during design phase is launched by the Monitoring Partition. Figure 2 shows a simple example in which an additional scheduling plan (Profile 1) is generated for each operational mode. This new plan executes partitions at different frequencies so it saves energy with respect to the original plan (Profile 0) in which all partitions run at the highest frequency.

VI. Experimental results

The goal of this section is the presentation of a case-study based on a version of XtratuM extended with low-power features and implemented on a Xilinx Zynq-7000 device. This case-study demonstrates the monitoring capabilities and some of the LPT implemented in the hypervisor.

A. Test application

An evaluation framework has been constructed to perform a feasibility study of the architecture described above. This framework consists of a test application that will be used to show how the main low power services can be applied within the extended hypervisor.

The test application consists of the following 3 partitions:

1) *Monitoring Partition:* it collects power and temperature information and applies static LPT.
2) *Debug Partition:* it allows to collect data by sending through the UART power, voltage and temperature information.
3) *Benchmark Partition:* it runs a benchmark algorithm (floating point matrix multiplications) that consumes power.

The first two partitions run on core 0 of the ARM Dualcore, the last one on core 1. The partitions are executed periodically with a period of 1s. The *Monitoring Partition* and the *Debug Partition* have a time slot of 900ms and 100ms respectively; whereas the *Benchmark Partition*, executed in parallel on core 1, takes the whole 1s time slot.

The application is executed according to the following scheduling, composed by 3 operational modes:

1) *Initialisation:* The Monitoring Partition is the only active partition; it initialises the peripherals.
2) *Normal:* All the three partitions run at 400MHz (maximum allowed frequency) and at nominal power voltage level.
3) *Low-power:* The application enters a low-power mode, where one of the following LPT is applied: static frequency scaling, static voltage scaling, or dynamic core suspension.

The schedule starts in the *Initialisation* and goes to *Normal* after 300ms. Then it starts to switch back and forth from *Normal* to *Low-power* every 40s, that is a time large enough to collect meaningful data to plot.

B. Measurement results

For compactness only the results of voltage scaling and frequency scaling, as static LPT, and of core suspension, as dynamic LPT, are reported in this paper. The performed tests consist in applying one of this techniques at a time by switching between plans of the XtratuM schedule.

In the first test case, the voltage is changed from 1.05V (*Normal* plan) to 0.9V (*Low-power* plan). The voltage switch is controlled by the Monitoring Partition that calls the function SLP_SCALE_VOLTAGE(...). Figure 3 shows the results of the voltage scaling example. The mean power consumption changes from 355mW at 1.05V to 254mW at 0.9V (approximately 28% power reduction).

In the second test, frequency scaling is applied during the low-power plan in the schedule. All the three partitions run at 32MHz (minimum allowed frequency). The frequency switch

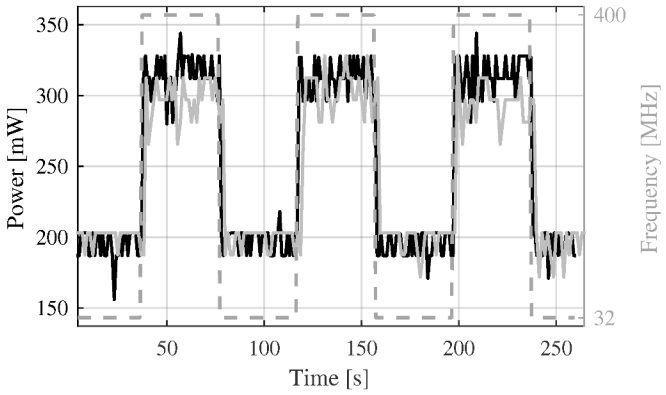

Figure 4. Experimental measurement obtained by applying *frequency scaling*. This figure shows the evolution of the power consumption and of the frequency of the ARM cores over time (rail VCCPINT in Table II). The ARM power consumption is drawn as a black continuous line; the frequency as a grey dashed line. The grey continuous line is the ARM power consumption measured by using an external PMBus adapter connected to a PC that runs a data logger, thus bypassing the Monitoring Partition. Notice that the grey and black lines represent data acquired in two different measurement runs, necessary to avoid conflicts on the PMBus.

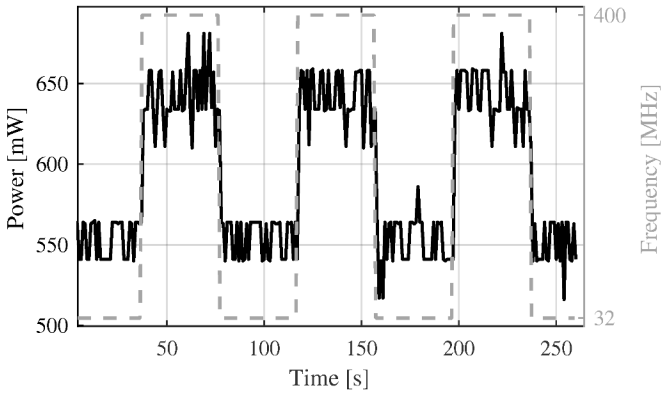

Figure 5. Experimental measurement obtained by applying *frequency scaling*. This figure shows the evolution of the power consumption of the DDR and of the frequency of the ARM cores over time (rail VTTDDR in Table II). The DDR power consumption is drawn as a black continuous line; the ARM frequency as a grey dashed line.

is controlled by the Monitoring Partition that asks XtratuM to change the scheduling plan. Figure 4 shows the results of the frequency scaling example. The mean power consumption drops down from 320mW at 400MHz to 210mW at 32MHz (approximately 33% power reduction).

Notice that in this case, due to the lower number of accesses, also the power of the DDR varies considerably, as can be seen in Figure 5. The mean power consumption of the DDR changes from 640mW at 400MHz to 550mW at 32MHz (approximately 14% power less).

It is important to remark that the performances of the Monitoring Partition vary with the frequency. Indeed, it is not possible to vary the frequency of a single core on a Zynq device. Thus, since both cores at once are affected by

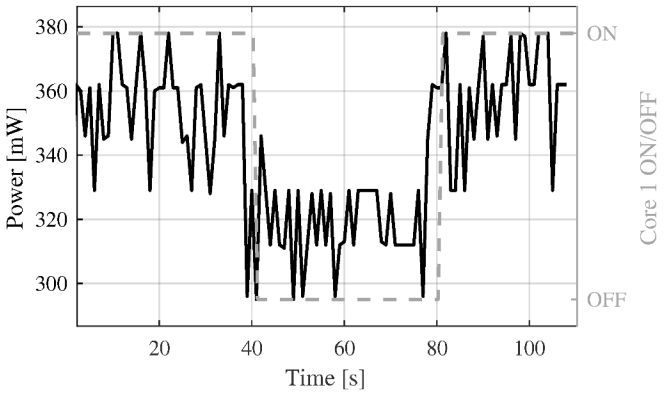

Figure 6. Experimental measurement obtained by applying *core suspension*.

frequency scaling, the time performance of the Monitoring Partition varies with the frequency. It takes 40ms to retrieve the temperature, voltage and power information from all the measurements points on the ZC702 test boards at 400MHz. This time increase to $\approx$ 482ms when the frequency is set to 32MHz. Anyway, it is possible to configure the Monitoring Partition at compilation time in order to monitor just a limited subset of the available measurement points. For instance, if only the power on the VCCPINT rail is considered, the measurement time drops to 2ms at 400MHz.

As regards the accuracy the data acquired by the Monitoring Partition has been compared to the measurements performed by an external data logger. Figure 4 shows that the power measured by the monitoring partition is slightly greater than the one obtained with the data logger (black and grey lines respectively). This is due to the extra overhead required to run the monitoring services on the Zynq, that were disabled during the external measurement runs.

Finally, core suspension has been selected as an example of dynamic LPT. In this test the benchmark partition do not perform any operation in the *Normal* scheduling plan and it is switched off completely during the *Low-power* plan. Figure 6 shows the results of the core suspension example. The mean power consumption drops down from 357mW to 317mW when core 1 is suspended (approximately 11% power reduction).

Finally, it is worth to point out that the partition execution time is not altered by the applied LPTs. Indeed, the mean period has been measured to be 1s, with a standard deviation of 2µs, that is in the order of precision of the time monitoring function provided by XtratuM. This indicates that the partition schedule is not affected by the proposed LPT and extended architecture.

VII. Conclusion and Future Work

As a result of the evaluation done with XtratuM it can be concluded that the architecture defined for power aware hypervisors is deployable without disrupting normal behaviour. Both monitoring and power management services depend on

the platform and hardware used, but the architecture could be applied even for power services that have not been taken into account in this work. The experimental results showed that all the power techniques reduce power consumption and that monitoring services based on external devices are heavily time consuming.

The use of the extended XtratuM within very different real use cases (avionic and railway) will show the cross domain capabilities and the effectiveness of the defined Low Power Hypervisor architecture. The robustness should be also tested, but a certification authority has positively assessed the safety concept of a real railway application where the proposed architecture was used [25].

As a plan for future development, once the architecture will be included in a real application, it would be interesting to evaluate power saving and timing overheads quantitatively. Another research line, could be implementing LPT also in other parts of the Zynq SoC like clock gating softcores based on information shared by a network on chip. The same concept could be used more globally on a network of different SoC communicated by a field bus.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 687902 (SAFEPOWER).

REFERENCES

[1] I. Agirre, M. Azkarate-Askasua, A. Larrucea, J. Perez, T. Vardanega, and F. J. Cazorla, *Automotive safety concept definition for mixed-criticality integration on a COTS multicore*, ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 9923 LNCS. [Online]. Available: www.scopus.com

[2] M. Fakih, A. Lenz, M. Azkarate-Askasua, J. Coronel, A. Crespo, S. Davidmann, J. C. Diaz Garcia, N. G. Romero, K. Grüttner, S. Schreiner, R. Seyyedi, R. Obermaisser, A. Maleki, J. Öberg, M. T. Mohammadat, J. Pérez-Cerrolaza, I. Sander, and I. Söderquist, "Safepower project: Architecture for safe and power-efficient mixed-criticality systems," *Microprocessors and Microsystems*, vol. 52, pp. 89–105, 2017. [Online]. Available: www.scopus.com

[3] P. P. Gelsinger, "Microprocessors for the new millennium: Challenges, opportunities, and new frontiers," in *2001 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. ISSCC (Cat. No.01CH37177)*, Feb 2001, pp. 22–25.

[4] A. Crespo, I. Ripoll, M. Masmano, and P. Arberet, "Xtratum: an open source hypervisor for TSP embedded systems in aerospace," 2011.

[5] S. Avramenko, S. Esposito, M. Violante, M. Sozzi, M. Traversone, M. Binello, and M. Terrone, "An hybrid architecture for consolidating mixed criticality applications on multicore systems," in *2015 IEEE 21st International On-Line Testing Symposium (IOLTS)*, July 2015, pp. 26–29.

[6] D. Juergens, D. Reinhardt, R. Schneider, and G. Hofstetter, "Implementing mixed criticality software integration on multicore - a cost model and the lessons learned," in *SAE 2015 World Congress & Exhibition*, vol. 2015-01-0266, 2015.

[7] Z. Gu and Q. Zhao, "A state-of-the-art survey on real-time issues in embedded systems virtualization," *Journal of Software Engineering and Applications*, vol. 5, pp. 277–290, 01 2012.

[8] , Inc., "Xen Power Management," https://wiki.xenproject.org/wiki/Xen_power_management, 2018.

[9] R. Morabito, "Power consumption of virtualization technologies: an empirical investigation," *CoRR*, vol. abs/1511.01232, 2015. [Online]. Available: http://arxiv.org/abs/1511.01232

[10] C. Jiang, D. Ou, Y. Wang, X. You, J. Zhang, J. Wan, B. Luo, and W. Shi, "Energy efficiency comparison of hypervisors," in *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, Nov 2016, pp. 1–8.

[11] R. Nathuji and K. Schwan, "Virtualpower: Coordinated power management in virtualized enterprise systems," in *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, ser. SOSP '07. New York, NY, USA: ACM, 2007, pp. 265–278. [Online]. Available: http://doi.acm.org/10.1145/1294261.1294287

[12] M. Masmano, I. Ripoll, A. Crespo, and J. J. Metge, "Xtratum: a hypervisor for safety critical embedded systems," in *In: Proceedings of the 11th Real-Time Linux Workshop*, 2009.

[13] S. Trujillo, A. Crespo, and A. Alonso, "Multipartes: Multicore virtualization for mixed-criticality systems," in *2013 Euromicro Conference on Digital System Design*, Sept 2013, pp. 260–265.

[14] A. Crespo, M. Masmano, J. Coronel, S. Peiró, P. Balbastre, and J. Simó, "Multicore partitioned systems based on hypervisor," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 12 293 – 12 298, 2014, 19th IFAC World Congress. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016435706

[15] A.-C. Orgerie, M. D. d. Assuncao, and L. Lefevre, "A survey on techniques for improving the energy efficiency of large-scale distributed systems," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 47:1–47:31, Mar. 2014.

[16] J. O. Coronel and J. E. Simó, "High performance dynamic voltage/frequency scaling algorithm for real-time dynamic load management," *Journal of Systems and Software*, vol. 85, no. 4, pp. 906–919, 2012.

[17] C. S. Stangaciu, M. V. Micea, and V. I. Cretu, "Energy efficiency in real-time systems: A brief overview," in *Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium on*. IEEE, 2013, pp. 275–280.

[18] P. Chen, M. C. Shie, Z. Y. Zheng, Z. F. Zheng, and C. Y. Chu, "A fully digital time-domain smart temperature sensor realized with 140 fpga logic elements," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 12, pp. 2661–2668, Dec 2007.

[19] C. . Lefebvre, J. L. Montero, and L. Rubio, "Implementation of a fast relative digital temperature sensor to achieve thermal protection in zynq soc technology," *Microelectronics Reliability*, 2017, article in Press. [Online]. Available: www.scopus.com

[20] G. Tarawneh, T. Mak, and A. Yakovlev, "Intra-chip physical parameter sensor for fpgas using flip-flop metastability," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2012, pp. 373–379.

[21] A. Lenz, M. A. . Blazquez, J. Coronel, A. Crespo, S. Davidmann, J. C. D. Garcia, N. G. Romero, K. Gruttner, R. Obermaisser, J. Oberg, J. Perez, I. Sander, and I. Soderquist, "Safepower project: Architecture for safe and power-efficient mixed-criticality systems," in *Proceedings - 19th Euromicro Conference on Digital System Design, DSD 2016*, 2016, pp. 294–300. [Online]. Available: www.scopus.com

[22] *Xilinx Zynq-7000 All Programmable SoC ZC702 Evaluation Kit*, Xilinx. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html

[23] A. F. Beldachi and J. L. Nunez-Yanez, "Accurate power control and monitoring in zynq boards," in *Conference Digest - 24th International Conference on Field Programmable Logic and Applications, FPL 2014*, 2014, cited By :3. [Online]. Available: www.scopus.com

[24] *Xilinx Zynq-7000 All Programmable SoC ZC706 Evaluation Kit*, Xilinx. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html

[25] A. Bilbao, I. Yarza, J. Montero, M. Azkarate-Askasua, and N. Gonzalez, "A railway safety and security concept for low-power mixed-criticality systems," 2017, pp. 59–64, cited By 0. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041199218&doi=10.1109%2fINDIN.2017.8104747&partnerID=40&md5=61b07e8824034b4a8d1992a3e03b10a5