XtratuM hypervisor redesign for LEON4 multicore processor*

E. Carrascosa
Instituto de Automática e Informática Industrial
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022
Valencia, Spain
ecarrascosa@ai2.upv.es

J. Coronel
FentISS
CPI, 9B Building, Office 3
Camino de Vera s/n, 46022
Valencia, Spain
jcoronel@fentiss.com

M. Masmano
Instituto de Automática e Informática Industrial
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022
Valencia, Spain
mmasmano@ai2.upv.es

P. Balbastre
Instituto de Automática e Informática Industrial
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022
Valencia, Spain
patricia@ai2.upv.es

A. Crespo
Instituto de Automática e Informática Industrial
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022
Valencia, Spain
acrespo@ai2.upv.es

ABSTRACT

XtratuM is an open source hypervisor that uses para-virtualization techniques designed to comply with safety critical real-time requirements. Several projects aimed to define a reference architecture for space on-board systems have adopted XtratuM as a virtualization layer in order to enforce the strong spatial and temporal isolation between software components that is required on real-time airborne systems.

Given the shift in the general trend in processor development towards multicore processors, the European Space Agency (ESA) is commissioning a set of studies in order to evaluate their suitability for their use on the space market. This paper focuses on the porting of XtratuM to the LEON4 multicore processor, in the frame of a ESA study that pursues to assess its fitness for its use in future space missions.

Keywords

Hypervisor, para-virtualization, partitioned systems, RTOS, multicore.

1. INTRODUCTION

For the last decade, the European space sector has developed an interest in partitioned software architectures based on the Integrated Modular Avionics (IMA) design concept as a mean to address security and safety issues. Partitioned architectures isolate software components into independent partitions whose execution shall not interfere with that of other partitions, preserving temporal and spatial isolation. Virtualization has been proven an effective method to fulfill those temporal and spatial isolation requirements, which led the European Space Agency (ESA) to adopt XtratuM [8, 6, 7], a hypervisor for real-time embedded systems specifically designed to meet safety critical requirements, as the reference virtualization layer.

At the same time, there has been an increasing trend towards the use of multicore processors in embedded computing due to their improved performance and low impact on critical aspects of embedded systems, as power consumption or heath dissipation. These traits, together with a better trade-off between performance and cost, weight reduction, and the potential to exploit parallel execution, have arisen an interest in the use of multicore technology in the space sector.

In this context, the ESA launched the IMA-SP [5] initiative, aimed to promote the adaptation of the IMA concept to the space domain. The project defined a partitioned reference architecture for space on-board software based on the ARINC-653 avionics standard [1, 2], with XtratuM as the virtualization layer. However, the IMA-SP project was focused on monocore processors, and a series of projects on multicore assessment for its use on space applications were subsequently launched. After the development of the LEON4 multicore processor [4], the project covered by the present paper [9] promoted the porting of the XtratuM hypervisor to this new target and evaluated its suitability for partitioned systems.

*This work has been partially funded by the Multipartes project (FP7-287702), HISEM (Prometeo 2009/02) and HI-PARTES (TIN2011-28567-C03-03).

VtRES 2013 Taipei, Taiwan

2. XTRATUM

XtratuM is a bare-metal hypervisor intended for embedded real-time systems that uses para-virtualization techniques to mimic hardware behaviour as closely as possible. Attending to its purpose, it has been designed using the ARINC-653 standard principles as a basis to achieve temporal and spatial partitioning on safety critical applications.

2.1 Interface

In order to offer basic hardware virtualization support as well as specific high-level services based on ARINC-653 to the partitions, XtratuM provides a set of hypercalls that transfer control to the hypervisor. This implies that partitions shall have hypervisor-specific code to access those services and may need to be adapted so as to interact with the hypervisor instead of with the underlying hardware. Some of the hypercalls are restricted to a special type of partitions (system partitions) that are allowed to manage and monitor the state of the system and other partitions.

```
// Hardware services
void XM_sparc_set_psr (xm_u32_t psr);
void XM_sparc_flush_cache (void);
void XM_sparc_flush_regwin (void);
// High-level services
xm_s32_t XM_create_queuing_port (
    char *portName, xm_u32_t maxNoMsgs,
    xm_u32_t maxMsgSize, xm_u32_t direction);
xm_s32_t XM_hm_status (xm_HmStatus_t
    *hmStatusPtr);
xm_s32_t XM_trace_event (xm_u32_t bitmask,
    xmTraceEvent_t *event);
```

Listing 1: ***Example of XtratuM service interface.***

2.2 Static Resource Allocation

The allocation of the available hardware resources to partitions takes place statically via a configuration file. This allocation shall be performed according to the needs of each partition regarding memory areas, scheduling, communication ports, etc. The configuration file also specifies the board resources, the configuration of the virtualized devices, the set of memory regions allocated to the hypervisor and the scheduling plan.

```
<SystemDescription version="1.0.0">
  [...]
  <ProcessorTable>
    <Processor id="0" frequency="80Mhz">
      <CyclicPlanTable>
        <Plan id="0" majorFrame="2000ms">
          <Slot id="0" start="0ms"
           duration="1000ms" partitionId="0"/>
          <Slot id="1" start="1000ms"
           duration="1000ms" partitionId="1"/>
        </Plan>
      </CyclicPlanTable>
    </Processor>
  </ProcessorTable>
  [...]
  <PartitionTable>
    <Partition id="0" name="Partition1"
     flags="system" console="Uart">
      <PhysicalMemoryAreas>
        <Area start="0x40180000" size="256KB"
         mappedAt="0x40000000"/>
      </PhysicalMemoryAreas>
    </Partition>
    <Partition id="1" name="Partition2"
     flags="system" console="Uart">
      <PhysicalMemoryAreas>
        <Area start="0x401c0000" size="256KB"/>
      </PhysicalMemoryAreas>
    </Partition>
  </PartitionTable>
</SystemDescription>
```

Listing 2: ***XML configuration file example.***

2.3 Cyclic Scheduler

In order to ensure a strong temporal isolation, XtratuM scheduler implements a cyclic scheduling policy according to the ARINC-653 specification. This policy assumes previous knowledge of the time allocation to each partition, which is specified on a cyclic plan that is statically defined during the design phase. A cyclic plan consists in a major time frame (MAF) which is periodically repeated. Inside the MAF, time is divided in slots with a established starting time and duration that are allocated to a given partition. XtratuM assigns the processor to the corresponding partition within each time slot, ensuring that the partition gets only the specified amount of processor time. In the case of partitions where several tasks execute concurrently, the partition is in charge of internally implementing its own scheduling algorithm in a transparent way to the hypervisor (hierarchical scheduling). This policy provides a deterministic behaviour while minimizing scheduling overhead at run-time.

2.4 Trap and interrupt management

A processor trap implements an asynchronous transfer of control to the system as a mechanism to handle hardware interrupts, software traps and processor exceptions. XtratuM extends the concept of processor traps with a new range of additional interrupts (extended interrupts) to indicate to partitions the ocurrence of XtratuM specific events.

Regarding interrupts, XtratuM leaves to the partitions the management of non-critical devices and manages only those hardware interrupts belonging to those hardware devices able to jeopardize the isolation. Hardware interrupts can be allocated only to one partition, that then has the capacity to mask or unmask the interrupt line via specific hypercalls.

2.5 Communication Mechanisms

XtratuM provides robust message passing based mechanisms for inter-partition communication (between two partitions or between a partition and the hypervisor). For this purpose, the hypervisor makes available to the partitions a series of services based on ARINC-653 defined queuing and sampling ports. On its side, the hypervisor implements channels that act as a logical path between source and destination ports, and it is also responsible for encapsulating and transporting the messages.

2.6 Health Monitor

XtratuM provides a Health Monitor as a mechanism to detect and manage unexpected events. The Health Monitor aims to identify those faults that can not be handled at the scope where they take place, and properly manage those events in order to minimize the consequences for the whole system. A set of predefined actions is provided to deal with an error according to its nature as soon as it is detected,

being its behaviour statically configured through the configuration file. After the execution of the handling action, a Health Monitor notification message can be issued and logged to be accessed by a system partition, which later can perform a more detailed error handling.

3. PORTING TO LEON4

3.1 Target Hardware

Aeroflex Gaisler has developed, in conjunction with the ESA, the Next Generation Microprocessor (NGMP) prototype, a multicore processor to be evaluated for its use in the future space missions of the agency that is the target of this study. The NMPG is a quad-core 32-bit LEON4 (Sparc V8 cores) running at 50 MhHz with 4x4Kb instruction and data Level-1 caches, a shared 256 KB Level-2 cache, MMU, IOMMU and two shared FPUs.

3.2 Software Architecture

The porting of XtratuM to a multicore processor has been performed following a Symmetric Multi Processing (SMP) software architecture approach, where a single OS manages all the hardware resources. Nevertheless, this approach is not suitable for its straight adoption, since it poses the undesired restriction of the use of a single OS for all the applications. The use of XtratuM offers a more complete solution consisting on a SMP hypervisor layer that enforces the time and space partitioning of the hardware resources, and provides virtualization services to the guest applications that are able, when needed, to run their own OS on a virtual processor.

3.3 XtratuM on Multicore Systems

XtratuM was originally developed for x86 monocore systems, and later ported to LEON2 and LEON3 processors. Therefore, the adaptation of XtratuM to a multicore LEON4 processor has required a more extensive redesign in order to add the necessary capabilities to manage several processors, as well as the other hardware resources. A SMP approach creates the need for each processor to be able to use XtratuM in a concurrent way. Consequently, this requirement has led to implement a fine-grained synchronization mechanism that grants exclusive access to the critical sections of XtratuM, such as spin-locks protecting shared internal data structures against race conditions. Additionally, the partition model has been adjusted in order to allow the use of multicore partitions, through the incorporation of the concept of virtual CPU. The more relevant aspects of the adaptation of XtratuM to LEON4 are detailed next.

3.3.1 VCPUs

A virtual CPU is an abstraction of a hardware CPU that models its behaviour. However, a virtual CPU can be equally allocated to any of the existing cores. XtratuM provides as many virtual CPUs as hardware CPUs are on the system. Virtual CPUs management is analogous to the real hardware behaviour: at the instant when the partition starts its execution a single vCPU is active, and it is responsibility of the partition to initialize the remaining virtual CPUs. To this end, XtratuM has been extended with new hypercalls that allow the partitions to handle virtual CPUs operation.

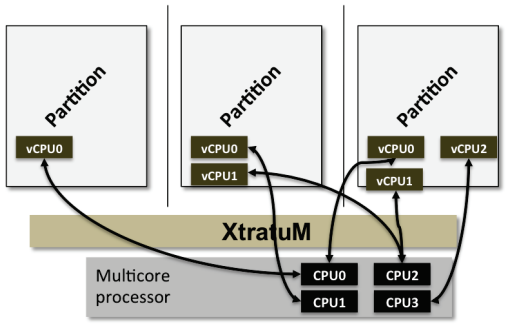

Figure 1: Hypervisor SMP software architecture

Figure 1 depicts a possible vCPU to real processor binding scenario.

3.3.2 IOMMU

A specific feature of the LEON4 processor is the IOMMU. Its inclusion in the LEON4 design was imposed by the ESA as a way to guarantee spatial isolation for I/O devices access. In absence of this mechanism, the DMA device may be used to bypass memory isolation between partitions. Therefore, one of the project requirements was the assessment of this component and its inclusion in the XtratuM redesign in order to provide IOMMU support.

The IOMMU tables are statically defined through the configuration file.

3.3.3 Scheduling policies

In the present multicore approach, each CPU holds its own cyclic scheduler, defining individual scheduling plans for each core. Although the use of a cyclic scheduling policy for partitioned systems is optimal from the temporal isolation point of view, it also represents a technical hurdle for asynchronous interrupt handling, given that an interrupt allocated to one partition may stay pending until the partition is scheduled again. In order to overcome this issue, XtratuM has also included a fixed priority scheduling policy that can be adopted instead, enabling the coexistence of both scheduling policies. However, there can be an only scheduling policy assigned to a given physical CPU. In the case of multicore partitions, threads can be executed under different scheduling policies on different processors, allowing to perform faster I/O communications. Since a plan switch must occur at the end of a MAF, an imposed restriction is that there is an identical MAF for all the cores running under a cyclic scheduling policy.

4. PERFORMANCE EVALUATION

The performance testing has been conceived aiming to address three different aspects of the implementation: the effect of the hypervisor layer, the impact of the partition context switch (PCS) and the influence of the multicore shared hardware resources (memory and FPU) on the execution. Thus, the performance of XtratuM porting to the LEON4 multicore processor will be assessed through a series of tests designed to capture the overheads introduced by the hypervisor under different loads.

Table 1: Native vs. Partition performance

Benchmark	Iterations	Native (μ-sec)	Partitioned (μ-sec)	Performance loss
Dhrystone	100000	2006149	2006315	0,008%
CoreMark	1200	15436453	15604193	1,087%

Table 2: CoreMark execution with different slot durations

Slot duration	No Slots	Time (s)	Perf. loss	CoreMark/MHz
30 sec	1	15,604194		1,538048
1000 ms	16	15,606468	0,0146%	1,537808
500 ms	32	15,608796	0,0295%	1,537497
100 ms	157	15,627475	0,1492%	1,535085
10 ms	1592	15,839812	1,5100%	1,507709

Two well-known standard benchmarks have been used to perform the evaluation: Dhrystone [10] and CoreMark ® [3]. Dhrystone is a synthetic benchmark intended to be representative of system programming. It is mainly addressed to evaluate integer operations. One of the drawbacks of this benchmark is that the operations are focused on the basic CPU working and do not perform an intensive use of the stack. CoreMark is a simple benchmark that is specifically designed to test the functionality of a core. It uses basic data structures and algorithms common to practically any application. One of the advantages of this benchmark with respect to Dhrystone is the overflow of the processor stack (register window), allowing to analyze more accurately the impact of the hypervisor layer.

These benchmarks will be run both as bare-metal applications and as partitions running on top of XtratuM, and the frequency of context switches and the number of partitions executing concurrently will be increased progressively in order to measure the performance of the hypervisor at several different load points. The number of iterations selected for each benchmark is a test-dependant parameter slightly superior to the minimum number of iterations needed in order for the test to be valid. A minimal porting regarding clock access and output has been needed in order to execute these benchmarks as XtratuM partitions.

4.1 Native versus partition based applications

This test aims to evaluate the performance loss due to presence of the hypervisor. The goal is to compare both benchmarks running on the native hardware using a bare implementation against the same benchmarks running as a partition on top of the virtualisation layer. The partitioned benchmarks are executed under the hypervisor cyclic scheduling. The slot duration is larger than 30 seconds in order to complete the execution in one slot and avoid the effect of the partition context switch in the measurement. Table 1 shows the results obtained for both benchmarks.

These results show very low performance loss in the case of the Dhrystone benchmark. This is due to the fact that its operation does not require hypervisor services. On the other hand, the CoreMark benchmark has an effect on the stack management. XtratuM provides a plain stack to the partitions and is in charge of the window management. In that case, this test raises 2235 window overflow and underflow traps that are handled by XtratuM. Each time a trap is raised, XtratuM is executed and saves or restores the register window. The support needed for these operations corresponds with about a 1% of the CPU.

4.2 Partition context switch impact

To evaluate the impact of the partition context switch on the partition performance, a CoreMark benchmark partition has been executed on top of the virtualization layer with different execution slot durations. Table 2 shows the different scenarios considered and the achieved results. The first row defines a slot duration of 30 seconds, which is large enough to complete the benchmark in one slot. This value is used as reference value in the comparison with the subsequent scenarios. The second row defines a slot duration of 1 second, which means that the benchmark will be completed in 16 slots. The time required to complete the execution is then compared with the reference value. The observed performance loss is attributed to the partition context switch at the end of each slot. The following rows detail the results for 500, 100 and 10 milliseconds each.

These results allow to estimate the cost of the PCS. Taking into account that the difference perceived is due to the number of context switches, the PCS can be estimated to be in the range of 149 to 151 microseconds.

4.3 Multicore shared resources impact

In a first evaluation, the CoreMark benchmark has been executed at the same time in several cores. Table 3 shows the results when the benchmark is executed in 1, 2 and 3 cores simultaneously. The slot duration is 1 second for all cases.

The results show the direct influence of the number of cores in the performance loss. This impact is almost negligible, and could be explained by the presence of a 128-bit system bus, which allows concurrent access by the cores.

5. CONCLUSIONS

The NGMP prototype is the resulting product of ESA research concerning future multicore processor utilization in space missions. The analysis of the feasibility of its use to that end comes, thus, as a necessary step. As a reference high-criticality systems virtualization layer, XtratuM has been a natural choice to assess this point. XtratuM porting to the LEON4 processor has provided insight on the challenges of multicore use on partitioned architectures.

In this process, XtratuM has been adapted to support SMP

Table 3: CoreMark execution with different slot durations

Core id	Time (s)	Perf. loss
Core 0	15,606468	
Core 0	15,609504	0,0195%
Core 1	15,609513	0,0195%
Core 0	15,611749	0,0338%
Core 1	15,611685	0,0334%
Core 2	15,611690	0,0335%

hardware architectures. This has implied changes in the XtratuM design internals concerning multiple CPUs management, code critical sections protection through spin-locks, timers and intra-/inter-processor interrupt handling. Additionally, attending to the LEON4 specific features XtratuM has been provided with IOMMU support.

The monocore partition model has been revisited as well in order to support SMP partitions by the inclusion of the concept of virtual CPU, with the consequent API modifications. The scheduler has been extended with fixed priority scheduling policy support to take advantage of the availability of additional processors.

Regarding the board evaluation, it has been observed that LEON4 design overcomes some of the limitations of classical SMP hardware architectures, such as the memory access bottleneck, by implementing a 128-bit bus width enabling simultaneous access by all the cores to the L2 cache. Although not intrinsic to SMP, LEON4 implements an IOMMU device in order to address I/O devices spatial isolation concerns.

As further work, a more extensive evaluation of LEON4 performance remains to be carried out. Research on the use of different software architecture approaches (Asymmetric Multi Processing) could provide useful information about comparative advantages towards the use of a SMP model.

6. REFERENCES

[1] Airlines Electronic Eng. Committee (AEEC). Avionics application software standard interface (ARINC-653). Part 1 - Required services. 2006.

[2] Airlines Electronic Eng. Committee (AEEC). Avionics application software standard interface (ARINC-653). Part 1 - Required services. 2006.

[3] Embedded Microprocessor Benchmark Consortium (EEMBC). CoreMark Benchmark. http://www.coremark.org.

[4] European Space Agency (ESA). The ESA Next Generation Microprocessor (NGMP). http://microelectronics.esa.int/ngmp.

[5] European Space Agency Project. Integrated Modular Avionics for Space (IMA-SP), 2010-2012.

[6] M. Masmano, I. Ripoll, A. Crespo, and J. Metge. XtratuM: a hypervisor for safety critical embedded systems. In *11th Real-Time Linux Workshop*, Dresden (Germany), 2009.

[7] M. Masmano, I. Ripoll, A. Crespo, J. Metge, and P. Arberet. XtratuM: An open source hypervisor for TSP embedded systems in aerospace. In *DASIA 2009. DAta Systems In Aerospace.*, Istanbul (Turkey), 2009.

[8] M. Masmano, I. Ripoll, S. Peiró, and A. Crespo. XtratuM for LEON3: An open source hypervisor for high integrity systems. In *ERTS2 2010. European Conference on Embedded Real Time Software and Systems.*, Toulouse (France), 2010.

[9] M. Patte, V. Lefftz, M. Zulianello, A. Crespo, M. Masmano, and J. Coronel. System impact of distributed multicore systems. In *DASIA 2012. DAta Systems In Aerospace.*, Dubrovnik (Croatia), May 2012.

[10] R. Weicker. DHRYSTONE: A synthetic systems programming benchmark. *Commun. ACM*, 27(10):1013–1030, 1984.