SYSTEM IMPACT OF DISTRIBUTED MULTICORE SYSTEMS

Mathieu Patte [1], Alfons Crespo[3], Marco Zulianello [2],Vincent Lefftz [1],Miguel Masmano[3], Javier Coronel[3]

[1] Astrium SAS, Toulouse, France

[2] ESA/ESTEC, Noordwijk, The Netherlands

[3] UPV, Valencia, Spain

Abstract Multi-core processors are increasingly being considered to provide the performance required by future safety critical systems. In this line, Aeroflex Gaisler has developed, in conjunction with the European Space Agency, the NGMP, a quad-core processor to be used in the future space missions of the Agency. Unfortunately, its use is not straightforward as it poses various challenges at functional and system level. ESA is currently leading a study that aims at investigating the potential use of the NGMP for space on-board processing application [5]. This paper presents the outputs of this study, which includes an analysis of the use cases of the NGMP, the porting of a bare metal hypervisor to the NGMP, and the benchmarking of the new multi core specific hypervisor features.

Introduction

The market for Real-Time Embedded Systems has experienced an unprecedented growth, and is expected to grow for the foreseeable future [1][2]. Because of the competition on functional value, measured in terms of application services delivered per unit of product, the embedded industry is faced with rising demands for greater performance, increased computing power, and stricter cost-containment. These factors put pressure to deliver increasing performance for future embedded processors, as well as to reduce the number of processing units used in the system [3].

Such high performance requirements could be met by designing more complex processors with longer pipelines, out of order execution, and higher clock frequency. However, applying these solutions to real-time embedded systems design is not feasible, because they could introduce timing anomalies [4] due to their non-deterministic run-time behaviours. In addition, the high energy requirements of such complex processors do not satisfy the low-power constraints and the severe cost limitations of common embedded systems. Multi-cores offer better performance per watt than single-core processors, while maintaining a relatively simple processor design. Moreover, multi-core processors ideally enable co-hosting applications with different requirements (e.g. high data processing demand and stringent time criticality). Co-hosting non-safety and safety critical applications on a common powerful multi-core processor is of paramount importance in the embedded system market, allowing to schedule a higher number of tasks on a single processor so that the hardware utilization is maximized, while cost, size, weight and power requirements are reduced. This is especially important for the space industry where weight reduction is essential. In addition, space applications especially benefit from simpler cores with lower clock frequencies because errors induced from cosmic radiation are reduced.

Even if multi-core processors may offer several benefits to embedded systems, their use is not straightforward. On the one hand, real time embedded systems require guarantees on the timing correctness of the system, providing strong arguments for the most critical ones [11]. On the other hand, it is required to prevent that one application could corrupt the state of other applications; paying special

attention in preventing low-criticality applications to affect the high-criticality ones. This can be accomplished using time and space separation techniques; the application of such techniques to multi core processor is explored in this paper. In addition, the SW architects have to reconsider their designs whit the introduction of multi-core processors, especially in a traditionally conservative domain such as the one of space applications.

The place of multi core within space systems has not been defined yet. Depending on the application targeted, different design features could be required at hardware and/or software level. As a preliminary example, certain payload data processing algorithms could benefit from an efficient parallelisation scheme over multiple cores, while several platform control applications could be implemented on a single multi core processor while being safely segregated.

In this paper we propose different techniques to efficiently harness the processing power of multi core processor, including time and space partitioning techniques for Integrated Modular Avionics applications. We describe the porting process of the Xtratum hypervisor to the NGMP target. We also present in this paper the tests, and their associated results, that have been implemented to measure the performance overheads induced by the use of a hypervisor, as well as the use case developed to benchmark the newly proposed time and space separations techniques against traditional ones.

Harnessing the processing power of multi core processor for space applications

We have identified two kinds of applications that can make use of the increased processing power offered by multi core processors such as the NGMP: data processing applications and Integrated Modular Avionics applications (IMA). For each of these applications families, special techniques must be developed to efficiently use multi core architecture.

For some scientific missions, such as Gaia or Euclid, flexibility and on-board reprogramability of the payload data processing is a requirement. In theses cases, general purpose processors are used to implement the data processing functions.

The commercial embedded world has tackled the issue of the implementing data processing applications on top of the multi core processors by developing two approaches: data parallelism and task parallelism. Using the data parallelism approach, each core of the processor is executing the same processing, input data is split among the cores, and each set of data is processed independently on each core. Using this approach, an existing mono core implementation can be reused on multi core processor: it’s just duplicated over several cores. The drawback of this approach is that the cores are loosely synchronized and are competing for access to the same hardware resources. On the other hand, in the task parallelism approach, each core is executing a different step of the processing; data is flowing from one core to another. Using this approach the developer can precisely control the execution of the cores and can plan hardware resources allocation in order to reduce indeterminism. Given the facts that: scientific data processing applications are most often not the same for one mission to another, that space applications are very sensitive to indeterminism, and finally that there is a strong hardware coupling between the cores of the NGMP; we recommend using the task parallelism approach to implement space data processing applications.

ESA and the European space industry are currently pursuing a roadmap aiming at integrating more and more on-board applications on a single processing platform.

The additional processing power of a multi core processor could allow for integration of even more applications. The two main solutions to integrate several applications on a single multi core processor developed by the industry are the Symmetric Multi Processing and the (SMP) and the Asymmetric Multi Processing (AMP). Using the SMP solution a single Operating System (OS) runs on all the cores, each applications is executed in a separated process of the OS. This solution usually allows for good performance; however the development and qualification of an SMP OS is a complex and costly process. Moreover, all the applications have to use the same OS, which can be a constraint in some cases. In the AMP scheme, an independent OS is executed on each core. This obviously allows for running applications using different OS, but in this case the space partitioning between the applications can be difficult to enforce, as each OS has access to the complete memory map. For these reasons straightforward SMP or AMP solutions are not suited for space IMA applications.

Instead we propose to use a hybrid solution based on a lightweight SMP hypervisor kernel providing virtualization services to guest applications. Each application can run its own OS on its virtual processor. The hypervisor kernel is able to enforce a strict time and space partitioning of the hardware resources. This approach has the advantages of allowing a smooth transition from mono core IMA space applications: indeed hypervisor kernels are part of the current roadmap for IMA. An application developed to run on top of a mono core hypervisor would run transparently on a multi core hypervisor, only the hypervisor kernel needs to be ported and qualified to the multi core processor.

Advanced features for efficient multi core processor hypervisor

We have identified two advanced features of the hypervisor kernel that could allow for a more efficient use of a multi core processor in a time and space partitioning context.

The first one is the possibility to allocate several virtual processing cores to a single partition at the same time. This allows allocating more processing resource to a given partition, for instance a data processing partition. The virtual processing core are exported to the partition as raw processing resources, it's the partition responsibility to implement the necessary mechanisms to handle the concurrency of execution. For that purpose, as explained before in this paper, we propose using task parallelism techniques.

The second advanced feature we have identified aims at facilitating the implementation of Inputs/Outputs tasks in a time partitioned system.

In space systems an I/O task is made of two different operations: the management of the I/O hardware (interrupt servicing, DMA management…) and the actual processing of the I/O data (telecommand processing, command elaboration…). Implementing the I/O hardware management operations in a time partitioned system is challenging: either the scheduling plan has to be carefully designed in order to execute each partition at the required time, or the I/O hardware must be made more autonomous in order to relieve the partition from hardware management.

We have therefore identified the need to be able to schedule the execution of partitions in a dynamic way: as opposed to a fix static cyclic scheduling plan that's usually used to enforce time partitioning, we propose to schedule the execution of the partitions according to the occurrence of I/O events (interrupts) or internal requests (hypercalls). The system integrator

shall be able to use different scheduling policies for the different cores of the processor.

Using the two advanced features described above, we can implement a time partitioned system in which the management of I/O operations does not constrain the scheduling plan or the I/O hardware. Inside each partition, the I/O hardware management tasks are separated from the actual I/O data processing tasks; at least two virtual cores are allocated to each partition. The I/O data processing tasks (which represents the bulk of the processing requirement), are scheduled on one or more cores of the platform using traditional fix cyclic scheduling. The I/O hardware management tasks are scheduled for execution on a core of the platform using a dynamic scheduling policy. Inside a partition, I/O data is passed between the virtual cores using task parallelism techniques.

Using this approach for managing I/O operations in time partitioned systems has several advantages. First it releases the constraints on the scheduling plan, thus reducing the number of required context switches and improving the performance. Second, it allows using any kind of generic hardware. Another solution to solve this problem could have been to move the execution of all I/O hardware management operations inside one single partition which would be scheduled for execution all the time on one core of the platform; however this solution necessitates that all partitions sub contract the management of their I/O hardware to the I/O partition. Using the dynamic scheduling solution, each partition can manage its own hardware without interaction or coupling with the other partitions.

Figure 1 is illustrating en example in which three partitions are scheduled using a cyclic scheduling plan on CPU0 of the platform, while the I/O hardware management tasks of each of these partitions are executed on a different virtual CPU inside the partition's context on a different CPU using a dynamic scheduling policy.

Virtualization layer: XtratuM

XtratuM is a hypervisor for embedded real-time systems that initially was developed for x86 processor and ported to LEON2[12] and LEON3[13]. Starting from this porting, XtratuM has been adapted to be executed on the LEON4 multi core processor. The adaptation design has followed the next criteria:

- A hypervisor is a software layer that offers a virtual machine near the real one. From this point of view, the design has been focused to provide as many virtual CPUs as the hardware provides.
- The hypervisor mimics the hardware behaviour. The hypervisor, as the hardware does, offers one virtual CPU initialised to the partitions and they are responsible of the initialisation of other virtual CPUs when needed.
- Partitions can be mono or multi core. The hypervisor does not force to have

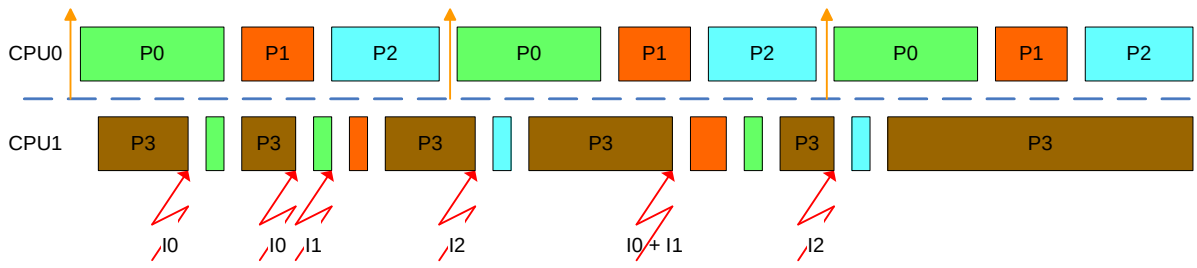

Figure 1 - Multi core partitions and dynamic scheduling

multi core partitions (multi core operating systems). A mono-core partition can be executed without the knowledge of the underlying multi core hardware.

- Separation concerns among virtual and real CPUs. A partition using a virtual CPU can allocate it in any of the real CPUs.
- Real CPUs can be scheduled under different scheduling policies. The system integrator can decide the more appropriated scheduling policy for the real CPUs. Some CPUs can be scheduled under a cyclic scheduling policy and others can be scheduled under a priority-based scheduling.

Next figure shows an example of three partitions using different cores and the mapping between the virtual and real CPUs.

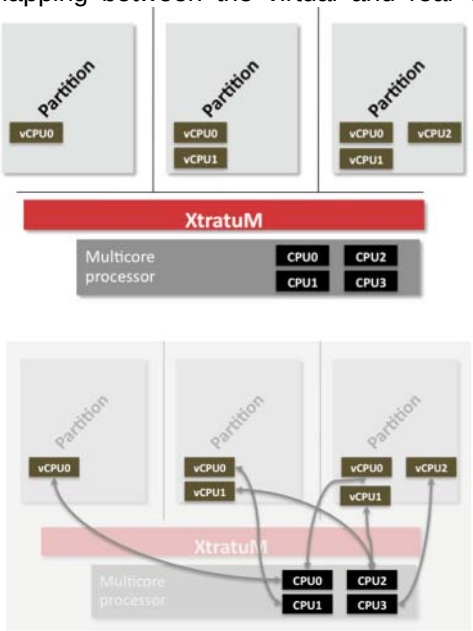

Figure 2 Virtal CPUs mapping

Taking into account these criteria the main adaptation of XtratuM to LEON4 includes the following aspects

- XtratuM offers as many virtual CPUS as the hardware provides. Partitions can be mono or multicore.
- Each virtual CPU in each partition has its own internal state. Services to start, stop, reset, suspend and resume a virtual CPU have been added.
- Interrupt management per virtual CPU
- IOMMU support, the system integrator can allocate an I/O interface to one partition, all the memory accesses performed by this I/O interface will be translated by the IOMMU so that the physical memory area used is solely the partition's one; therefore ensuring space partitioning of the memory accesses performed by the I/O interfaces.

The adaptation also has a strong impact in the XtratuM behaviour. While in previous adaptations, XtratuM was not preemptible, in multi core this approach is not valid. It is required that each CPU can use the hypervisor in a concurrent way. From this point of view, a redesign of the internals has been needed.

Other aspect that has been strongly influenced by the multicore approach is the scheduling policy. Following the ARINC-653 approach, the basic scheduling policy is the cyclic scheduling. However, in order to add flexibility several scheduling policies can be specified in the configuration file. The main assumptions are:

- A scheduling policy can be attached to each physical core.
- Several scheduling policies can coexist
- All cores scheduled under cyclic scheduling share the MAF.
- Other policies are: Fixed priority scheduling.
- A multi-core partition can execute two threads in different cores under different policies.

These features permit to implement a fast IO communications. A multi-core partition can allocate a thread to core under a cyclic scheduling and other thread to a core under

FPS. It allows to react very fast to interrupts and receive/send messages from the same partition context without need of additional delays.

Performance tests of the hypervisor and demonstrator application

The performance overhead induced by the hypervisor with respect to bare metal performance was assessed by running multiple partitions, each running its own copy of the standard CoreMark™ benchmark. The measured performance overhead depends on the frequency of context switches between partitions. In the most demanding scenarios for which the scheduling plan was built to trigger a partition context switch every 10 ms, the performance overhead is lower than 3%.

In order to test the proposed advanced features of the hypervisor, we implemented a demonstrator application. The application is made of two independent partitions. The first partitions is implementing a synthetic data processing task that receives a continuous stream of SpaceWire packets, compute a MD5 digest of the packets and send back the digest over the same SpaceWire link. The second partition is implementing a data management task, which receives a continuous stream of SpaceWire packets, store these packets, and upon reception of a specific packet starts the retransmission of all previously stored packets.

An independent SpaceWire interface is allocated to each partition. Thanks to the support of the IOMMU unit of the NGMP by the Xtratum hypervisor, each partition is able to program DMA transfers to/from its SpaceWire interface in its own virtual memory address space. For each partition, the SpaceWire management tasks are separated from the main partitions task and executed on a different virtual CPU, which is in turn scheduled for execution on a dedicated CPU of the platform using dynamic schedule plan.

Using the embedded SpaceWire router of the NGMP we were able to precisely measure the performance of each partition. Indeed the SpaceWire router allows duplicating the incoming SpaceWire flux to a partition, sending one copy of the flux to monitoring equipment in order to track precisely the input data flux of the partition.

The tests have been run on a FPGA development platform, with the NGMP clocked at 50 MHz. The first results obtained are

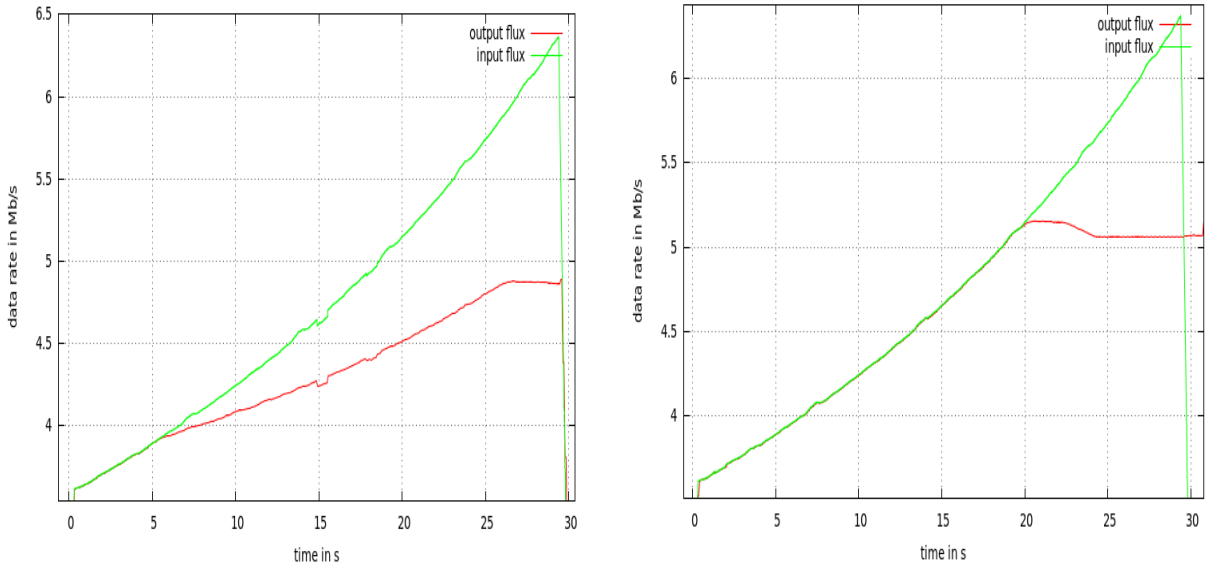

Figure 3 Performance of the synthetic data processing partition without dynamic scheduling (left) and with (right)

depicted on Figure 3. We see that when using advanced hypervisor features we can improve the performance of the synthetic data processing partition by 30%. The combined use of the DMA capabilities of the SpaceWire interface of the NGMP, which can be fully exploited thanks to the IOMMU support, and the smart use of the dynamic scheduling scheme, has allowed us handling input packet frequencies as high as 2.4 KHz.

Conclusion

Efficiently harnessing the processing power of multi core processor for space applications is not an easy task, and will require quite some modifications to existing architecture. In this paper we identified several techniques that go in this direction. The advanced hypervisor features that were defined and developed through the course of this study are a first step. As a next step, task parallelism API should be ported on top of the Xtratum hypervisor so that execution parallelism can be easily and efficiently managed inside partitions.

Reference documents

1] ARC Advisory Group. Process Safety System Worldwide Outlook. Market Analysis and Forecast through, 2012.

[2] P. Clarke, Automotive chip content growing fast, says Gartner (9/6/2010). http://www.eetimes.com/electronics-news/4207377/Automotive-chipcontent-growing-fast.

[3] MERASA, EU-FP7 Project: www.merasa.org.

[4] T. Lundqvist and P. Stenstrom, “Timing anomalies in dynamically scheduled microprocessors,” in RTSS, 1999.

[5] ESA contract 4200023100, System Impact of Distributed Multi-core Systems.

[6] ESA contract: 22279/09/NL/JK, Next generation multipurpose microprocessor.

[7] ESA contract: 4000100764, IMA for space

[8] ESA contract: 22280/09/NL/LvH, Securely Partitioning Spacecraft Computing Resources

[9] J. Andersson, J. Gaisler, and R. Weigand, “Next generation multipurpose microprocessor,” in DASIA, 2010.

[10] NGMP Preliminary Datasheet Version 1.6, August 2011

http://microelectronics.esa.int/ngmp/LEON4-NGMP-DRAFT-1-6.pdf.

[11] ESA contract 4000102623, Multi core OS Benchmarks

[12] A. Crespo, I. Ripoll, M. Masmano, P. Arberet, and J.J. Metge. “XTRATUM: an open source hypervisor for TSP embedded systems in aerospace”. DASIA 2009. DAta Systems In Aerospace., May. Istanbul 2009.

[13] T. Pareaud, L. Planche, D. Mylonas, V. Kolias, N. Pogkas, A. Crespo, I. Ripoll, M. Masmano, J. Windsor and K. Eckstein. “Securely Partitioning Spacecraft Computing Resources:Validation of a Separation Kernel”. DASIA 2011. DAta Systems In Aerospace., May. Malta 2011.