System Impact of Distributed Multicore Systems

ADCSS – MCPSA 2011

Mathieu PATTE – Alfons CRESPO – Vincent Lefftz

ESTEC Contract 4200023100

All the space you need

ASTRIUM
AN EADS COMPANY

Outline

- Study context
- Commercial embedded multi core processors
- Multi core for space use scenarios
  - Integrated payload controller / data processing
  - Extended IMA
  - Dynamic scheduling
- Xtratum port to NGMP
- NGMP Assessment

Study Context

- System Impact of Distributed Multi core Systems objectives:
  - Multi core processor use scenarios
  - Port of an hypervisor on the NGMP
  - NGMP assessment
- Industrial organization:
  - Astrium (Prime): system analysis, hypervisor specification and use case implementation
  - Universitat Politecnica de Valencia: hypervisor port to NGMP (Xtratum), hypervision technology expertise

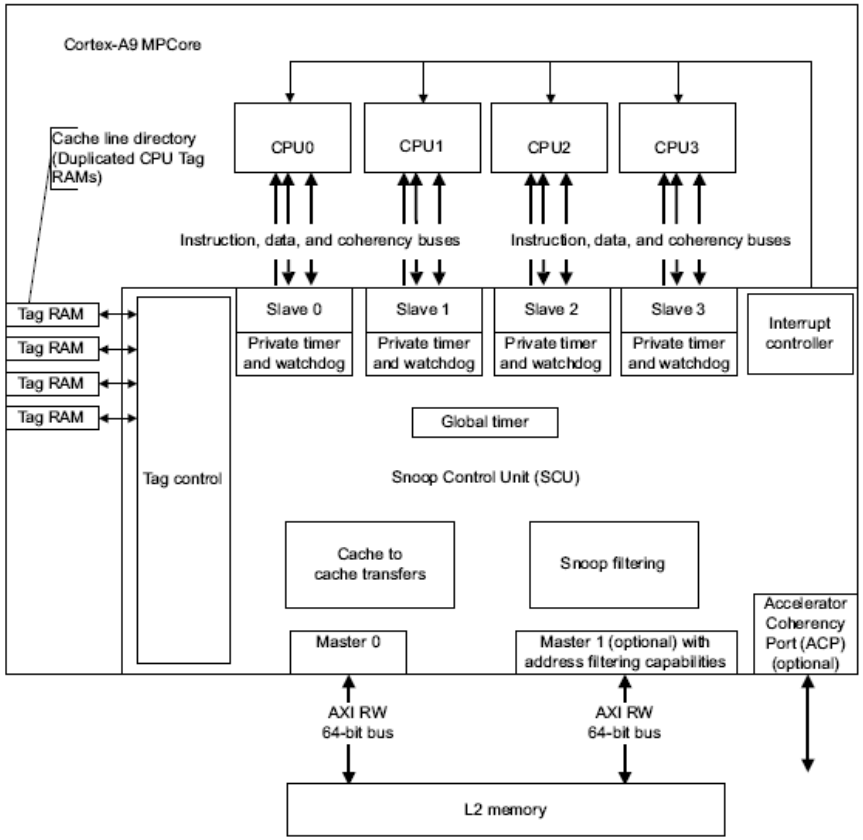
Outline

- Study context
- **Commercial embedded multi core processors**
- Multi core for space use scenarios
  - Integrated payload controller / data processing
  - Extended IMA
  - Dynamic scheduling
- Xtratum port to NGMP
- NGMP Assessment

Existing commercial embedded multi core processors: state of the art

- Driven by mobile applications (smartphones, tablet, portable gaming console)
- Players:
  - CortexA9 MP (Apple A5, samsung exynos)
  - STEricsson Nomadik
  - Qualcomm Snapdragon
- Commonalities:
  - Homogeneous + specialized units
  - ARM based
  - Tightly coupled memory
  - Multilayer AHB / AXI

Existing commercial embedded multi core processors: example



- Cortex-A9 MPCore
- *Copyright © 2008-2011 ARM. All rights reserved.*

Existing commercial embedded multi core processors: programming model

- SMP OS with APIs:
  - PThread
  - OpenMP
  - MPI class of APIs
- Parallelization principles
  - Data parallelism: same code, different bits of data
  - Task parallelism: different code, same data

Outline

- Study context
- Commercial embedded multi core processors
- **Multi core for space use scenarios**
  - **Integrated payload controller / data processing**
  - **Extended IMA**
  - **Dynamic scheduling**
- Xtratum port to NGMP
- NGMP Assessment

Space domain needs for more capable processors

- Integrated Modular Avionics: centralization of processing tasks on a single computer
  - STR processing
  - GPS processing
- 100-1000 MIPS processor building block for payload processing
- → NGMP is the solution currently being investigated

Multi core use scenarios

- Integrated payload controller / data processing
  - Execution parallelism of controller and data processing
  - Efficient implementation for small payload
- Extended IMA application
  - STR and GPS processing in the central OBC
  - Partitioning of the CSW
  - Stringent I/O requirements

Integrated payload controller / data processing: hypervisor

- Hypervisor (Xtratum) for T&SP partitioning of controller and data processing tasks:
  - Hypervisor allows using a guest OS for the controller and close to bare metal for processing
- Need for SMP partitions:
  - Typically, one core for controller, three cores for data processing

Integrated payload controller / data processing: task parallelism

- Task parallelism is better suited than data parallelism for data processing on multi core
  - Explicit control of core execution: allows minimizing hardware concurrency
  - Smaller code fits in each core's L1 caches
  - Code rework to build SW pipeline

Extended IMA application

- Multiple partitions : STR, GPS, FDIR, AOCS, PAYLOAD ….
  - Hypervisor allows using different RTOS
  - Concurrent development and validation, IF HARDWARE COUPLING is low.
  - Indeterminism is dealt with by keeping huge performance margins.
- Processing requirements are low, constraints are on I/O latencies and throughput → use of DYNAMIC SCHEDULING

Dynamic scheduling: I/O facts

- I/Os are mostly not deterministic, eg:
  - Flow controlled UARTs
  - TM link bandwidth allocation mechanisms
  - Asynchronous TC arrival
- Therefore I/O operations are often IRQ driven

Dynamic scheduling: time partitioning and IRQ handling on mono core

- Time partitioning → IRQ handling latencies
  - In order not to break time partitioning, an asynchronous IRQ can only be processed during one of the execution slots of the destination partition
- IRQ handling latencies → performance loss or high hardware cost
  - To reduce IRQ handling latencies, it's possible to schedule partition that handles IRQ more often, but performances are degraded due to numerous context switches
  - Or I/Os must be more autonomous: large buffers, micro-controllers

Dynamic scheduling: IRQ handling on multi core partitioned systems

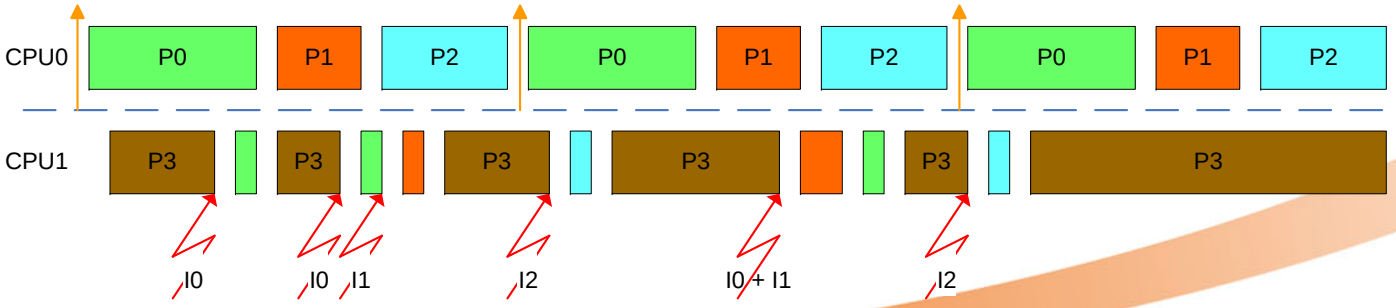
- 1st possibility: I/O management partition running 100% on a dedicated core, problems:
  - Dedicating one complete core for IRQ results in performance loss of the overall system (especially on dual core)
  - IPC communications are needed to transfer data to the final user partition
  - Impossible to dedicate a given I/O to one partition, partitions are coupled via the I/O partition
- 2nd possibility: Dynamic scheduling on the core dedicated for IRQ handlers execution

Dynamic scheduling: principles

- As opposed to fixed cyclic scheduling:
  - Partitions scheduling is modified dynamically at runtime upon occurrence of external events (IRQ)
- Scheduling policy is chosen per core:
  - 3 cores using fixed cyclic scheduling, 1 core using dynamic scheduling

Dynamic scheduling: use case

- Partition execution is split
  - Normal processing on cores using fix cyclic scheduling, IRQ handler execution on core using dynamic scheduling
- Background partition on core using dynamic scheduling
  - The dynamic scheduler executes the background partition when there isn't any IRQ handler to execute



Dynamic scheduling: benefits

- IRQ handling latency minimized
  - Allows lowering constraints on I/O architecture and scheduling plan
- Efficient use of multi core processor
  - A background partition can be executed between IRQs
  - Efficient use of a additional core on strongly hardware coupled multi core processors
- Efficient I/O operations
  - No IPC needed, zero copy communication
  - Allows for I/O direct allocation to partitions

Dynamic scheduling: impacts

- Concurrent execution of IRQ handler and partition
  - Synchronization between IRQ handler task and other partition's tasks is achieved using hypervisor IPC or SMP OS primitives.
- Time partitioning may be impacted
  - The dynamic scheduler might not be able to enforce strict time partitioning, however the intrinsic hardware coupling of a multi core processor doesn't allow for strict time partitioning neither.

Dynamic scheduling: other uses…

- FDIR mechanisms
  - Fixed cyclic scheduling plan doesn't allow for quick reaction to error conditions, dynamic scheduling can allow reducing fault mitigation latencies.
- Enabler for advanced CSW partitioning

Outline

- Study context
- Commercial embedded multi core processors
- Multi core for space use scenarios
  - Integrated payload controller / data processing
  - Extended IMA
  - Dynamic scheduling
- **Xtratum port to NGMP**
- **NGMP Assessment**

NGMP Assessment: IOMMU

- IOMMU allows direct allocation of I/O to partitions
  - Partitions can program DMA transfers in their virtual address space
- IOMMU ensures space partitioning
  - But not time partitioning! Rogue or failing partitions can “illuminate” the bus.

NGMP Assessment: FLUSH instruction

- FLUSH = unprivileged instruction which flush both caches even when they are frozen
  - This breaks partitioning and forbids the possibility to prevent one partition to update the cache contents!
- SPARCv8 ISA doesn't require to flush the caches and moreover the FLUSH instruction is optional
- Some RTOS (RTEMS) are issuing flush instruction after installing trap handlers

NGMP assessment: hardware coupling

- Hardware coupling: concurrency of access to the same resources
- Hardware coupling is an intrinsic property of multi core processors!
- Too much hardware coupling prevents concurrent development and validation
  - Partitions A and B are validated separately, if the hardware coupling is too important, at integration time the execution timings of A and B will change!

NGMP assessment: hardware coupling

- Actual evaluation not yet performed
- Matters of concerns:
  - L1 caches size and write through policy: locality of space applications software is low
  - Shared AHB bus: round robin arbiter doesn't take into account the duration of the accesses
  - Single port L2 cache: the benefit of a cache hit for one core may be annulled by the fact it has to wait for a previous core to complete an external memory access.
  - Single memory controller: dual memory controller deemed necessary for simpler architecture like the SCOC3

Conclusion

- Multi core execution parallelism is an enabler for some applications
- SMP partitions are needed to efficiently use all the cores
- IOMMU and dynamic scheduling should allow for efficient I/O handling mechanisms
- Hardware coupling can be a show stopper if too important

Way forward

- Evaluation of the performance of Xtratum port to the NGMP

- Use case implementation to benchmark I/O handling mechanisms

Q&A