
# A Memory Arbitration Scheme for Mixed-Criticality Multicore Platforms

Bekim Cilku, Peter Puschner,
TU Wien
**Alfons Crespo,** Javier Coronel and Salvador Peiró
Universitat Politècnica de València

# Outline

- Introduction
- Hypervisor
- Memory arbitration based on TDMA
- Evaluation
- Conclusion

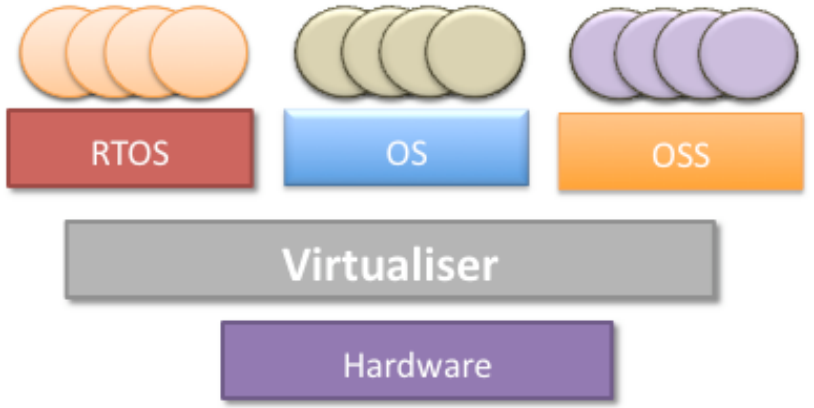
# Mixed-Criticality Systems

- Mixed-criticality systems integrates applications with different levels of criticality on the same computational platform.
- Certification process on multi-core hardware is complex and expensive. One of the aspects that introduces complexity is resource sharing
- More critical activities will require higher certification levels
  - Levels A,B,C,D (DO-178, ECSS,...)
- Applications are, in general, composed by critical and non critical activities.

# Mixed-Criticality Systems

- Partitioned systems permits to split the application in several subsystems (partitions) that facilitates the system decomposition where
  - Each partition can be developed/validated/certified independently

- In MultiPARTES project:
  - Platform for MCS
  - Based on XtratuM hypervisor
  - Multicore



# Mixed-Criticality System with Hypervisor

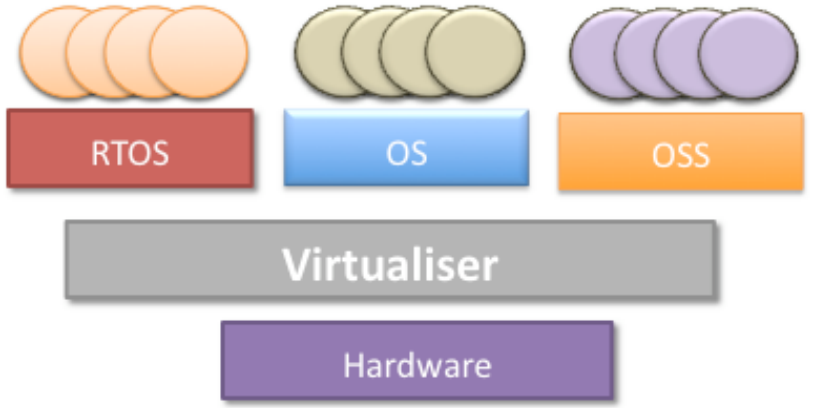
- Using hypervisor with temporal and spatial properties can reduce complexity by isolating virtual partitions on time and space domain.

- Partitioning criteria:
  - Time constrained activities run on critical partitions while non-critical ones on other virtual partitions.

- Benefits:
  - Independent design and development of partitions
  - Partitions can be analyzed and certified in isolation
  - Fewer hardware components
  - Reduces the cost of embedded systems

# Certificability

- Partitions allocate activities of the same criticality level
  - A partition has to be certified according the criticality level and, as consequence, as A,B,C,D
  - OS in a partition has to be certified at the same level than the allocated activities
- Hypervisor has to be certified at the highest criticality level of the partitions



# Platform requirements

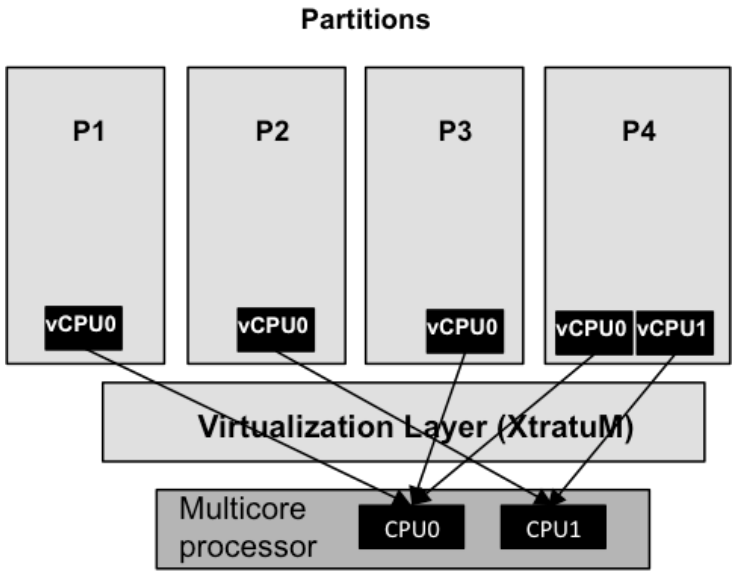
- **Temporal isolation** preserves the timing behavior of applications such that they are not affected from other applications that execute concurrently on the shared platform:
  - The timing requirements of different components can be validated independently.
- **Spatial isolation** protects memory elements of one application so they cannot be accessed by any other application.
- **Fault containment**: failures in one component (partition) must not propagate to cause failures in other components (partitions)
- **Inter-partition communication** permits to communicate partitions through ports and channels (hypervisor)

# Outline

- Introduction
- **Hypervisor**
- Memory arbitration based on TDMA
- Evaluation
- Conclusion

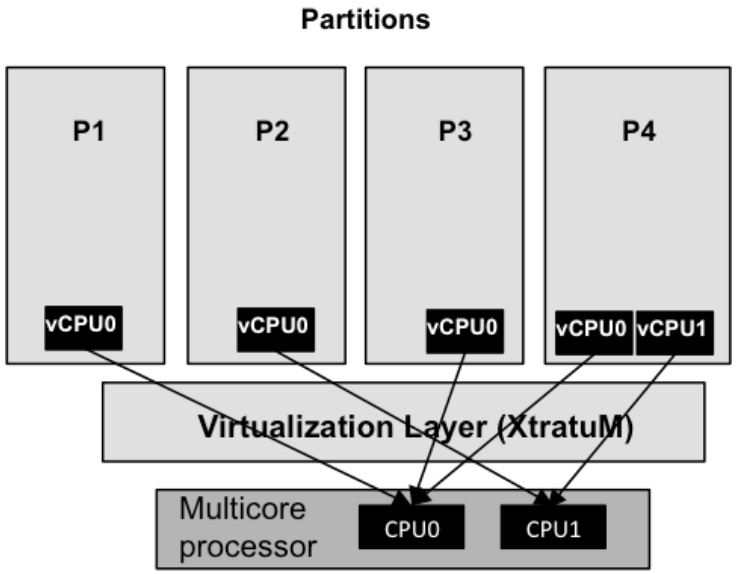
# XtratuM

- MultiPARTES platform is based on XtratuM hypervisor
- XtratuM is a bare-metal hypervisor for embedded real-time systems that provides:
  - Spatial and temporal isolation
  - Fault containment
  - Static allocation of resources
  - Multicore
    - Virtual CPUs
    - AMP and SMP architecture
    - Mono/Multi core partitions



# XtratuM

- Hypervisor schedules partitions
  - It does not know the partition internal (RTOS/tasks)
- Each core can be scheduled under a scheduling policy
  - Cyclic scheduling
  - Priority based Periodic Server
- Scheduling plan is offline generated

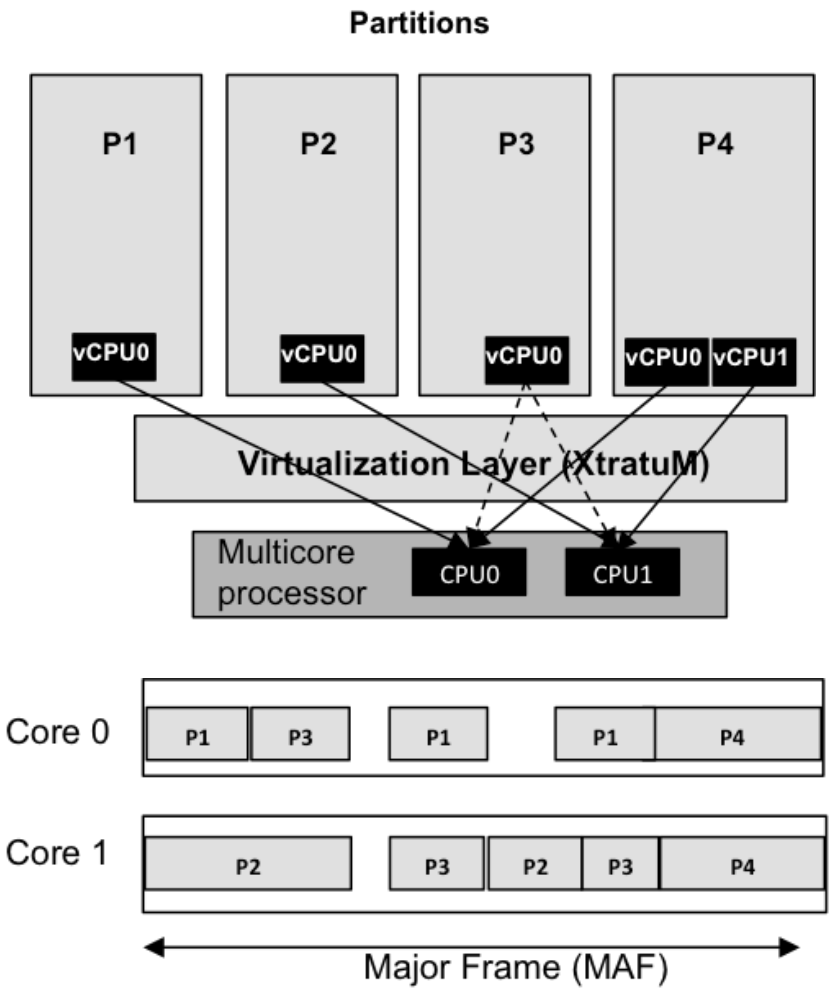



[17] V. Brocal, M. Masmano, I. Ripoll, A. Crespo, and P. Balbastre, "Xoncrete: a scheduling tool for partitioned real-time systems," in Embedded Real-Time Software and Systems, 2010.

# Scheduling

- Configuration file: scheduling example

```
<ProcessorTable>
 <Processor id="0">
  <CyclicPlanTable>
   <Plan id="0" majorFrame="20ms">
    <Slot id="0" start="0ms" duration="3ms" partitionId="1" vCpuId="0"/>
    <Slot id="1" start="3ms" duration="3ms" partitionId="3" vCpuId="0"/>
    <Slot id="2" start="7ms" duration="3ms" partitionId="1" vCpuId="0"/>
    <Slot id="3" start="12ms" duration="3ms" partitionId="1" vCpuId="0"/>
    <Slot id="4" start="15ms" duration="5ms" partitionId="4" vCpuId="0"/>
   </Plan>
  </CyclicPlanTable>
 </Processor>
 <Processor id="1">
  <CyclicPlanTable>
   <Plan id="0" majorFrame="20ms">
    <Slot id="0" start="0ms" duration="6ms" partitionId="2" vCpuId="0"/>
    <Slot id="1" start="7ms" duration="3ms" partitionId="3" vCpuId="0"/>
    <Slot id="2" start="10ms" duration="3ms" partitionId="2" vCpuId="0"/>
    <Slot id="3" start="13ms" duration="2ms" partitionId="3" vCpuId="0"/>
    <Slot id="4" start="15ms" duration="5ms" partitionId="4" vCpuId="1"/>
   </Plan>
  </CyclicPlanTable>
 </Processor>
</ProcessorTable>
```



# Temporal isolation

- Temporal isolation is impacted by
  - **Temporal allocation**:
    - At the same time and same duration (strict allocation)
      - Required for incremental certification
    - Same duration in an interval (soft allocation)
  - **Temporal interference**
    - Interference due to shared resources
- Cyclic scheduling permits to control the exact partition schedule: **strict temporal allocation**
  - At Partition Context Switch, local cache is flushed
- The **integrator** can control the level of overlapping between cores.

*Integrator: person in charge of the system design => configuration file and schedule*

# Temporal interference

- Temporal interference can be:
  - Modelled and considered as margins in the WCET of tasks (COTS solution)
  - Hardware support to avoid interference (specific hardware or custom hardware)

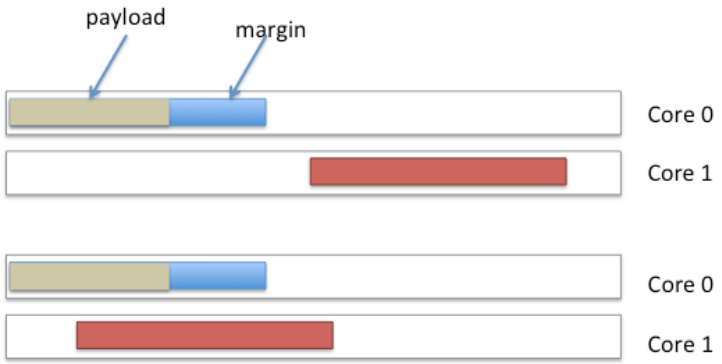
- Modelled:
  - Analyse the effects of the execution in other cores
  - Identify the impact on a payload
  - Integrator defines the margins
  - Schedule generation considers the margins for the partition slots

# Temporal Interference: experimentation

- **Worst Case Execution Time Impact**

$$WCET = WCET_{task} + Interference$$

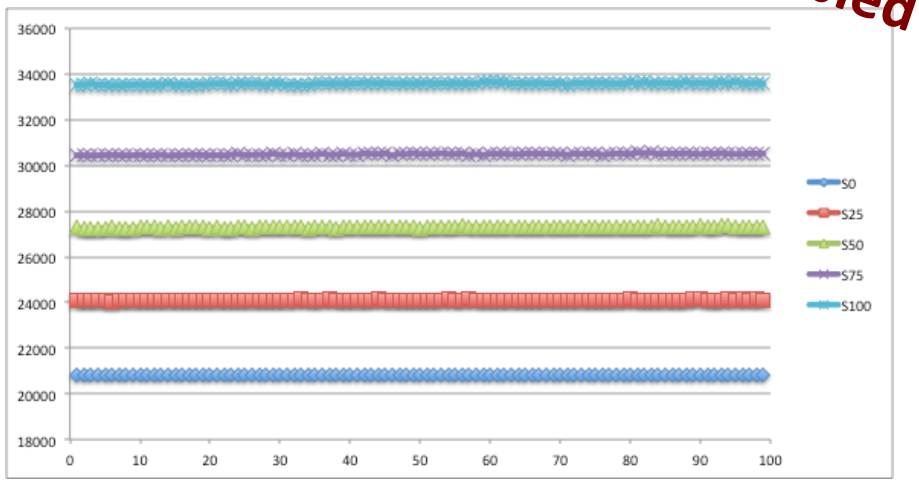
  - **The interference can be modeled**
    - **Evaluating the Inteference**
    - **Limiting it by construction of the scheduling plan**




| | S25 | S50 | S75 | S100 |
|---|---|---|---|---|
| Interference | 15,74% | 31,05% | 46,39% | 61,22% |

**LEON3 @ 50MHz**

**Cache disabled**



# Outline

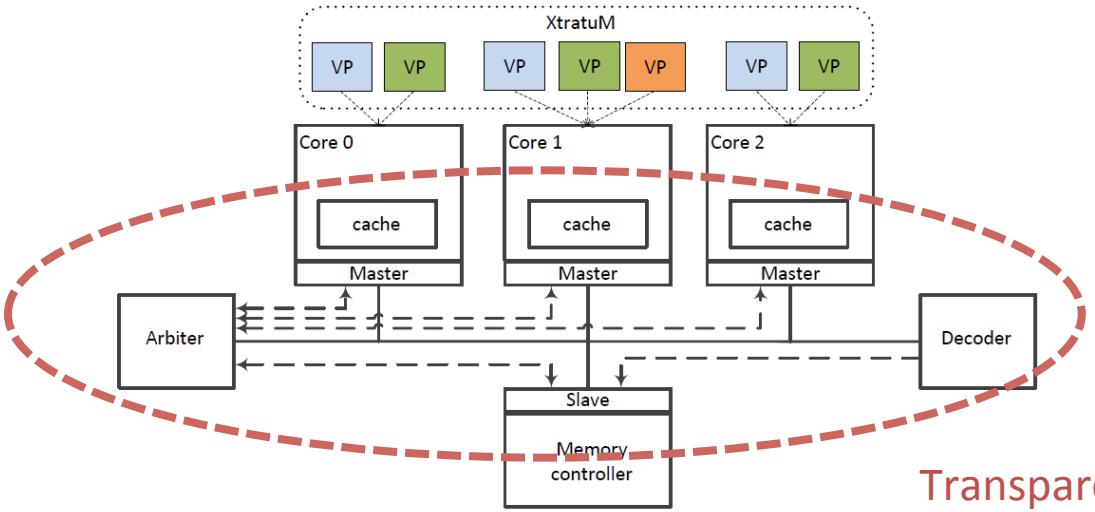
- Introduction
- Hypervisor
- **Memory arbitration based on TDMA**
- Evaluation
- Conclusion

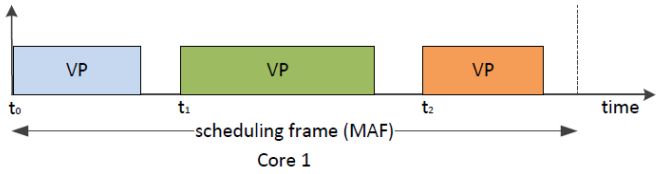
# Hardware

- LEON3 is a synthesizable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture.
- The model is highly configurable, and particularly suitable for system-on-a-chip (SOC) designs.
- Source code is available allowing free and unlimited use for research and education.

# Temporal Interference

- Hardware solution:
  - AMBA Bus policy can be RR or Priority base
  - Perform the hardware modifications to include TDMA
  - No impact in the software architecture (transparent)



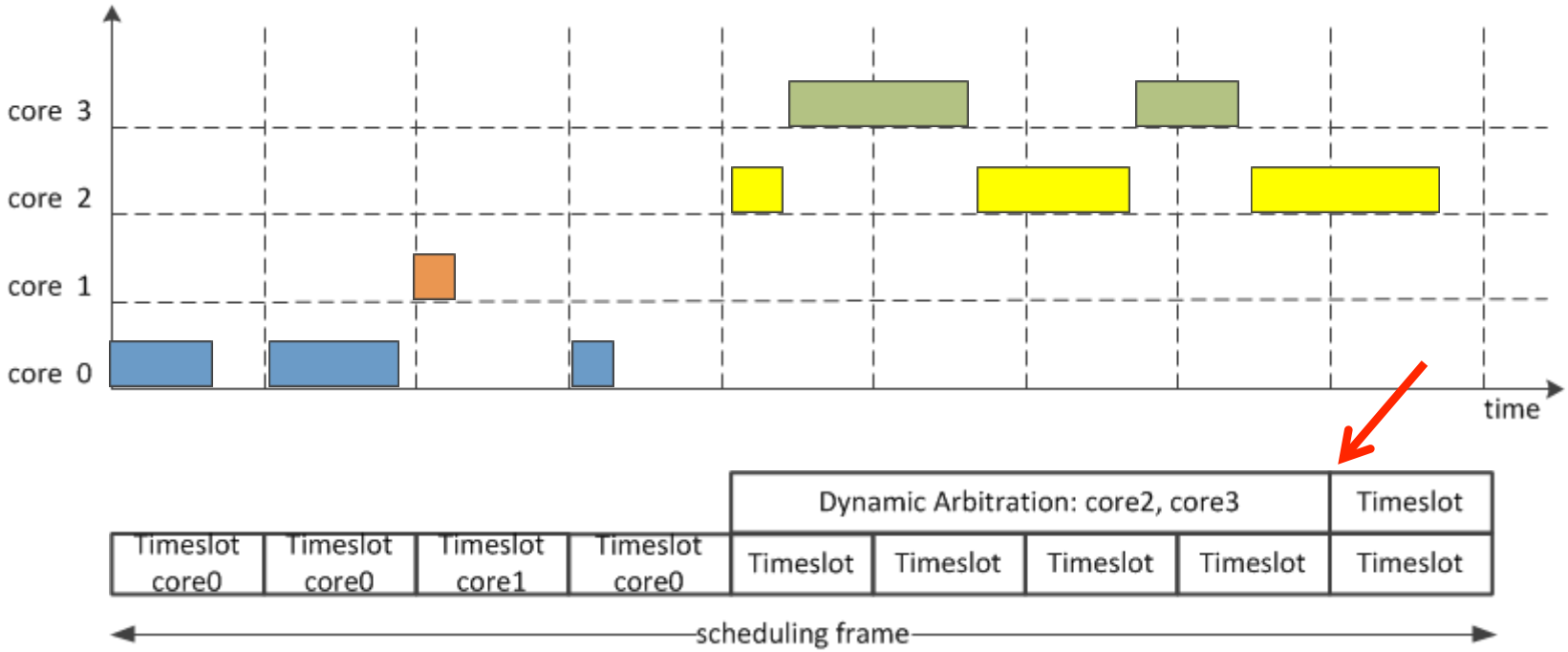

Transparent to the hypervisor

# Requirements in Bus-Based Communication

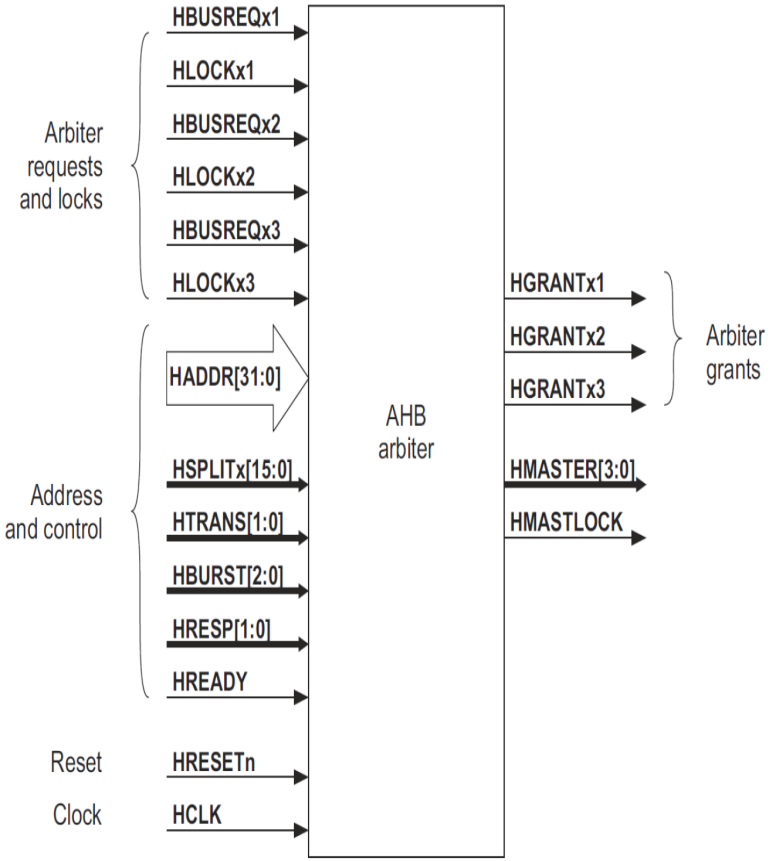
- On-chip communication must satisfy response time requirement for critical VP:
  - Guarantee max-latency and min-bandwidth to ensure that performance constraints are satisfied
  - Explicitly define communication transactions in temporal and spatial domain

- Arbitration:
  - Non-preemptive;
  - TDMA for critical and RR for non-critical partitions;

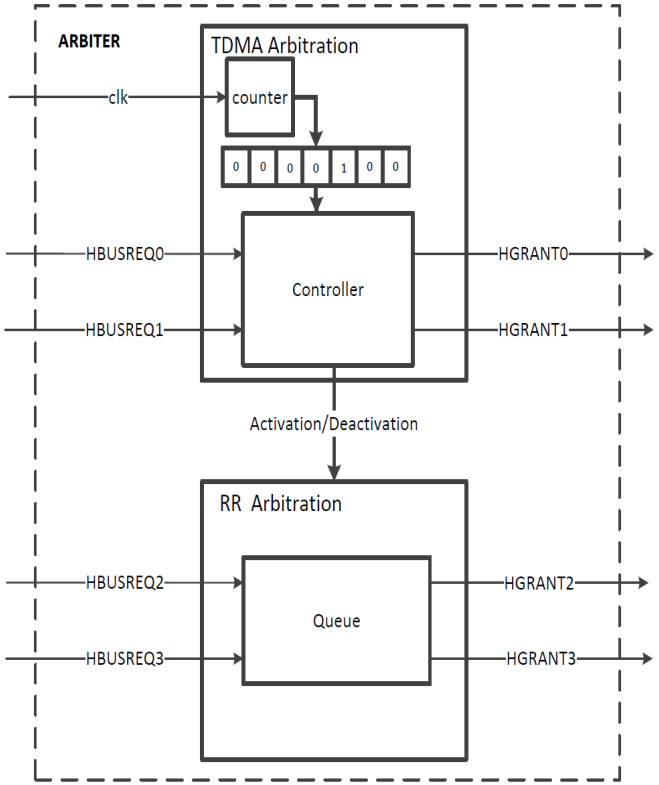
# Dual-layer arbitration policy



# Arbiter architecture




AMBA arbiter*



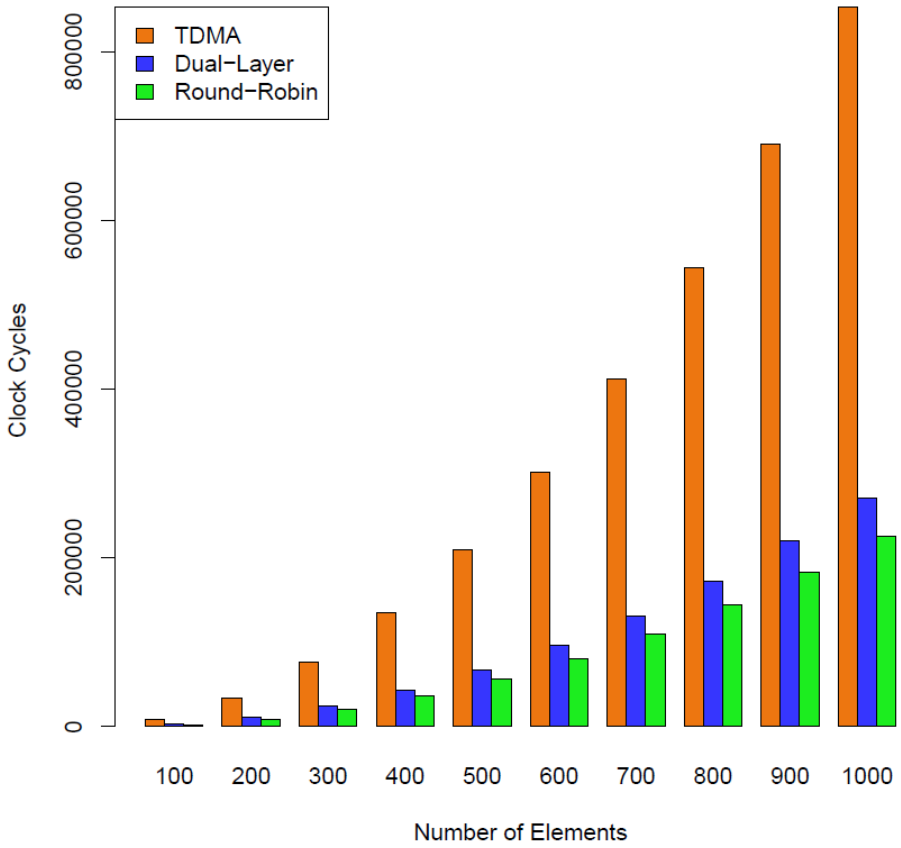

Dual-layer arbiter

# Outline

- Introduction
- Hypervisor
- Memory arbitration based on TDMA
- **Evaluation**
- Conclusions

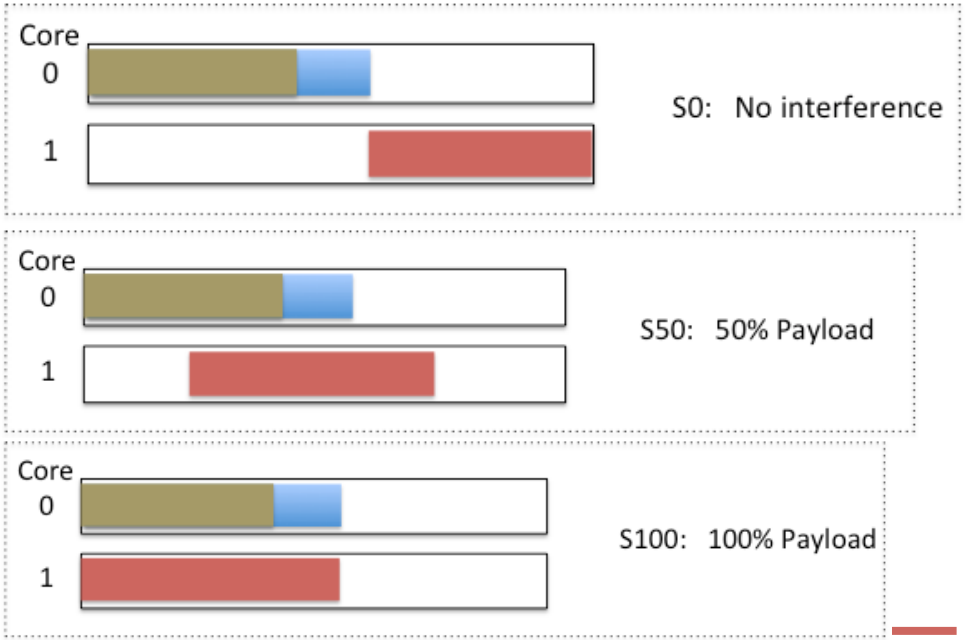
# Evaluation: cost

- Bubble sort algorithm:
  - Vector with size 100-1000 elements;
- TDMA, RR, DL policies
- Performances for noncritical tasks:
  - 3.1 faster than TDMA
  - 1.2 slower than RR



# Evaluation: Interference

- Memory interference
  - Goal: Measure the temporal interference of the cores
  - Scenarios:
    - A defined payload executed in 2 cores (Cache disabled)
    - Several scenarios with difference overlaps
      - S0, S25, S50, S75, S100



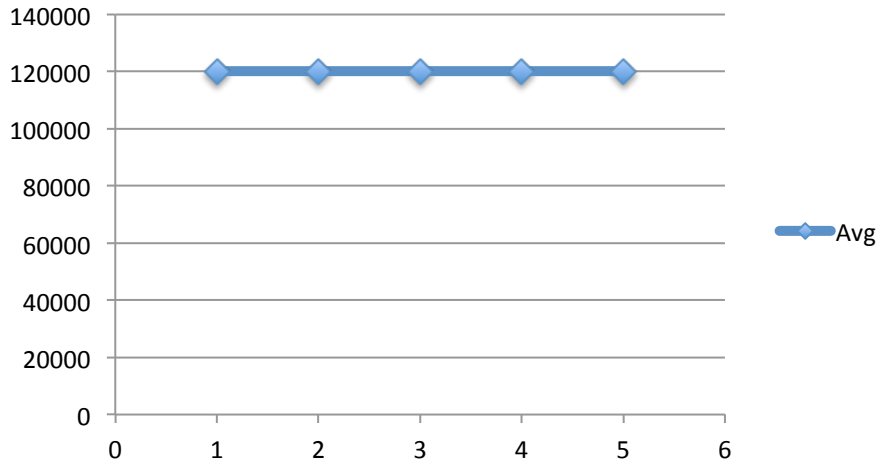
# Evaluation: Interference

- Memory interference tests: Results

| | S0 | S25 | S50 | S75 | S100 |
|---|---|---|---|---|---|
| **Avg** | **120104** | **120104** | **120104** | **120104** | **120104** |
| **Max** | **120111** | **120109** | **120111** | **120111** | **120109** |
| **Min** | **120099** | **120099** | **120099** | **120099** | **120099** |
| **Stdev** | 2,441 | 2,447 | 2,483 | 2,506 | 2,251 |
| **Interference** | | **0,00%** | **0,00%** | **0,00%** | **0,00%** |

Time to complete the payload in usecs
Time granularity 1 usec

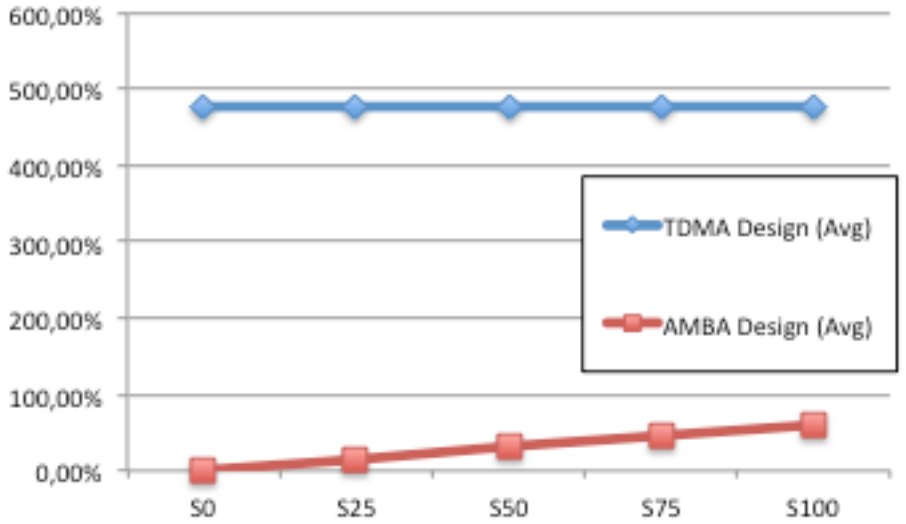
| | S25 | S50 | S75 | S100 |
|---|---|---|---|---|
| Interference | 15,74% | 31,05% | 46,39% | 61,22% |



# Evaluation: Execution Costs

- Execution costs are 5 – 7 times higher than the AMBA bus policies (LEON3 @ 50MHz and SDRAM)

| | Avg Time (µsecs) |
|---|---|
| TDMA bus arbitration | 120099 |
| RR bus arbitration | 20689 |
| Increment cost factor | 5.80 |




- Services of the hypervisor are strongly increased
- PCS is 1 ms which limits the usability for some applications

| | TDMA | AMBA | ARM Cortex A9 @100MHz | x86 @1.6GHz |
|---|---|---|---|---|
| PCS | 1 ms | 224 usec | 8usec | 2usec |
| Int Latency | 322 usec | 34 usec | 1usec | 1usec |

# Conclusions

- Temporal isolation can be modeled or provided by the hardware.
- Tradeoff functionality/cost
- Platform that provides full temporal and spatial isolation between partitions
- Certification can be done for each Partition in isolation
- TDMA for critical partitions and RR for non-critical ones
- Performance improvement for non-critical applications

# Thank you