Successfull Multicore certification with Software-Partitioning

Efficient Implementation for DO-178C, EN 50128, ISO 26262

Sven Nordhoff, SYSGO AG

Abstract

The usage of multi-core processors (MCPs) in modern systems is state-of-the art and will also come to reality in safety-critical domains like railway, automotive and avionic industry in near future. This fact is driven by different aspects. More and more functions traditionally implemented as separated electronic hardware units will be hosted on platforms where different functionality will be combined into one hardware. The industry wants to reach with the future family of high performance platforms larger scale integration on function level. Additionally new safety and security functions, information services and comfort features will increase the demand for more processing performance. Furthermore the industry has to deal with the fact that most of the manufacturers of CPUs will reduce the number of mono-core CPUs because the mass market does not demand this piece of hardware anymore. The possible technology improvements on mono-core CPUs has reached its limit to provide even more processing bandwidth with higher CPU clock frequencies, increase pseudo-parallel processing on instruction level through instruction pipelines and speculative program executions and by increasing the cache size and number of cache levels. Therefore the chip industry has switched to a multi-core design to accelerate the performance of processors. The question is: "Can a multi-core based platform reach the same level of determinism as a single core platform and can this be demonstrated?"

This paper addresses the current state on multi-core certification in the industry mainly based on experiences from the railway and avionic industry. We will address the certification aspects of multi-core based platforms with the focus on today's technologies and processes related to the new requirements of avionic certification authorities for multi-core processors. The paper provides an overview of certification concerns on multi-core processors and possible answers which can be given by a multi-core supporting hypervisor operating system.

Introduction to MCP-Hypervisor

The concept of a real-time hypervisor OS is well introduced and we do not want to repeat this again [7]. An MCP-hypervisor (multicore-hypervisor) uses all concepts of a traditional hypervisor which is mainly the spatial and temporal separation of software running on top of a hypervisor. The question will be if a traditional hypervisor can cope with the MCP specific challenges adequately or do we have to handle additional aspects in regards to MCP.

The following section gives a short overview about hypervisor attributes which are immanent for a MCP-hypervisor:

Real-Time: The MCP OS shall be a hard real time operating system to ensure that timing-critical application can be executed and a worst-case timing measurement scenario is possible and will be supported by the MCP OS e.g. by providing WCET analysis and measurement results to help the WCET analysis of the application developers.

Temporal Segregation: Within a hypervisor OS the concept of temporal segregation is adopted by using a time partitions concept to provide time fences to allow terminating the execution of an application if it over-runs its allocated execution time or deadline and guarantee that no critical application can be starved of processor time. For applications running on an MCP, the concept of time partitioning can be used to enable a limitation to execute application on different cores on the same time if an exclusive access to shared resources is needed to ensure e.g. adequate timing behavior and WCET measurement/analysis entry points.

Spatial Separation: The concept of spatial separation is adopted by using resource partitions. Any critical application is able to be executed in its own process with its own virtual memory space supported by hardware memory protection (MMU). Furthermore partitioning of I/O resources is needed to ensure correct separation to handle access to shared resources. This can be supported by protection mean like IOMMU/PAMU. In an MCP environment, applications can be executes on different cores in parallel having access to shared resources. An MCP concept will add the number of shared resources by using shared caches and busses to enable communication between the cores (e.g. interconnect). An MCP hypervisor shall enable concepts to deal with this adequately. A static configuration can limit the usage of cores for a dedicated resource partitions therefore conflict of executing different applications on different core can be managed.

Safe & Secure: The MCP OS shall be safe and secure meaning that the OS can be used for safety and security critical systems in different industry domains that such kind of software shall be certifiable against industry standards for safety related systems e.g. DO-178C [1], EN50128, ISO26262 but also against security-related standards (e.g. Common Criteria evaluation criteria, Airbus SAR, DO-356). The MCP specific certification requirements are currently under discussion in the different industry domains. This paper will address the MCP specific additional requirements addressed currently by EASA/FAA.

Mixed Criticalities: The MCP-Hypervisor shall support mixed criticalities, meaning that applications with different safety and security levels shall run on the same hardware, protected from each other by means of software partitioning. Here resource partitioning and time partitioning are the main concept to ensure correct separation of software functionality on one hardware. The MCP aspects need to be addressed especially if applications with the same criticality level will be executed on different cores on the same time "fighting" for the shared resources.

Support of Guest OS: Due to the need to run large scale operating systems like Linux or to run application APIs like POSIX or ARINC 653 (APEX) it will be needed to enable guest OS on top of a MCP-Hypervisor. Therefore a hypervisor shall enable

multiple personalities (OS environments, APIs or run-time environments). An MCP-hypervisor OS shall cope with objectives of different personalities dealing with multi-core. The ARINC 653 standard has introduced the usage of multi-core in the last revision. Furthermore standard operating systems like Linux deals with MCP since a long time. If such an OS are running on top of a hypervisor OS all needed functionality shall be supported.

Portability: A commercial MCP OS shall be highly portable to supports all important CPU Architectures like x86, PowerPC, ARM, MIPS and Sparc. Furthermore a concept needs to be enabled to implement an abstraction layer to separate the CPU architecture specific details from the OS core components.

Multi-Core Certification Concerns

Using multi-core architecture in industries for safety critical application raises concerns by the certification authorities in the following main areas:

Design Assurance: Leak of processor design documents may lead to undetected interference channels meaning that the documentation which is available for inspection do not have such kind of maturity to enable the user of such kind of hardware to answer to all certification objectives. E.g. the compliance to avionic electronic hardware assurance guidance DO-254 [2] and related CAST and CRI paper cannot be answered adequately. However this topic is similar to the discussion on microcontrollers started with the development of the Airbus A380 which lead to a more specific guidance in this field [3].

In-Service-History: The best certification story for electronic hardware without design assurance evidences in place is to collect adequate In-Service-History. Unfortunately the available multi-core based designs are quite new on the market and adequate. E.g. In-Service-History in avionics applications is hard to achieve due to the limited operation in this field. Consequently the approach used in the microcontroller domain to collect sufficient In-Service-History with the help of non-avionic industry fields including commercial non-safety related service history can be served as a proof [3].

Hardware Interference Channels: Cores within a chip interacts with each other and resources of the chip will be shared (e.g. cache). This interaction and/or sharing leads to interferences (e.g. timing) which need to be analyzed and possible mitigation / verification activities to cope with this interference need to be defined.

Measurement of worst case timing (WCET): Due to the sharing of resources in MCP and the usage of complex infrastructure (like busses) the analysis and / or measurement of the worst case timing is hard to achieve. Traditionally the concepts to provide adequate WCET analysis is usually not easy for traditional mono-core systems but will be more complicated in multi-core systems. A hypervisor OS strategy to support WCET analysis and measurement will be one part of the whole picture to cope with this certification-critical objective.

MCP Interference Channels

This paper is not intended to discuss the amount of interference channels related to MCP in detail. For a detailed research see [5], a paper provided to the 29th Digital Avionics Systems Conference 2010. The following section is based on this paper.

A generic MCP architecture is built like this:

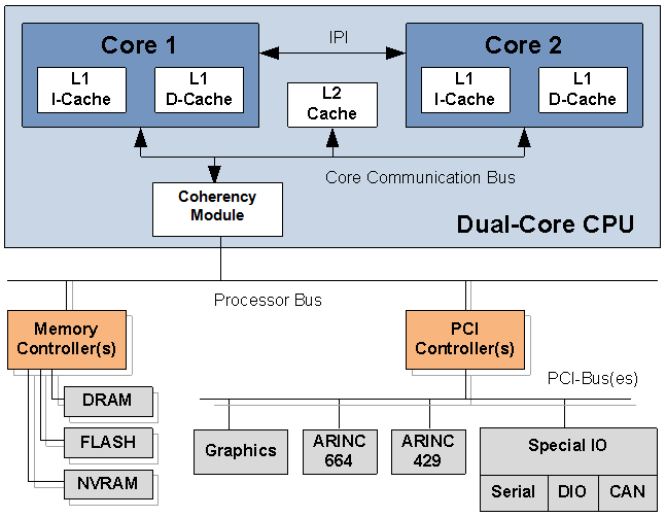


Figure 1. Typical MCP architecture

Typically the following MCP related components produce interference channels:

- Shared caches:
  - one L1 cache per core
  - one L2 cache shared between cores (e.g. Intel, MPC 8572D)
  - L3 cache typically shared
- Cache coherency protocol located in the coherency module
  - Complexity grows with number of cores
  - Configurable on some CPUs through core cross bar
  - Global / Local cache flush and invalidate
- Shared buses (Core Connection, Processor, Memory, PCI)
- Shared Interrupts
- Shared devices (Memory, Timer, I/O)

Cache Sharing: The L1 cache is typically divided into a data and an instruction cache while all other levels store data and instructions. Most multi-core processors have a dedicated L1 data and instruction cache per core while the architecture of the L2 and L3 cache varies with the CPU family.

Shared caches are an essential cause of interference in a multi-core processor. A comparison of read and write throughput between an Intel Pentium Dual Core E5300 and an AMD Athlon II x2 processor shows the impact of a shared L2 cache. Figure 2 shows the results for the two CPUs. If the data set is small enough to fit in the L1 cache (private for each core) the Intel and AMD processor show no loss of performance if the second core becomes active. If the data set is smaller than the L2 cache accessible by the cores and the L2 cache is not shared (AMD processor), the second core again causes no performance degradation. If the data set is smaller than the L2 cache visible to a core and the L2 cache is shared (Intel processor), the worst case performance loss through the second core depends on the data set size and is between 30% and 95% for write operations and 19% and 92% for read accesses. The largest impact (92%) is observed if the data set has exactly the size of the L2 cache because with one core the data still completely fit into the L2 cache while with 2 cores all data need to be fetched from memory which means that we are comparing the performance of the L2 cache with the performance of the memory bus.

If the data set is significantly larger than the L2 cache, the worst case performance losses caused by the second core is ~50% for read and write operations.

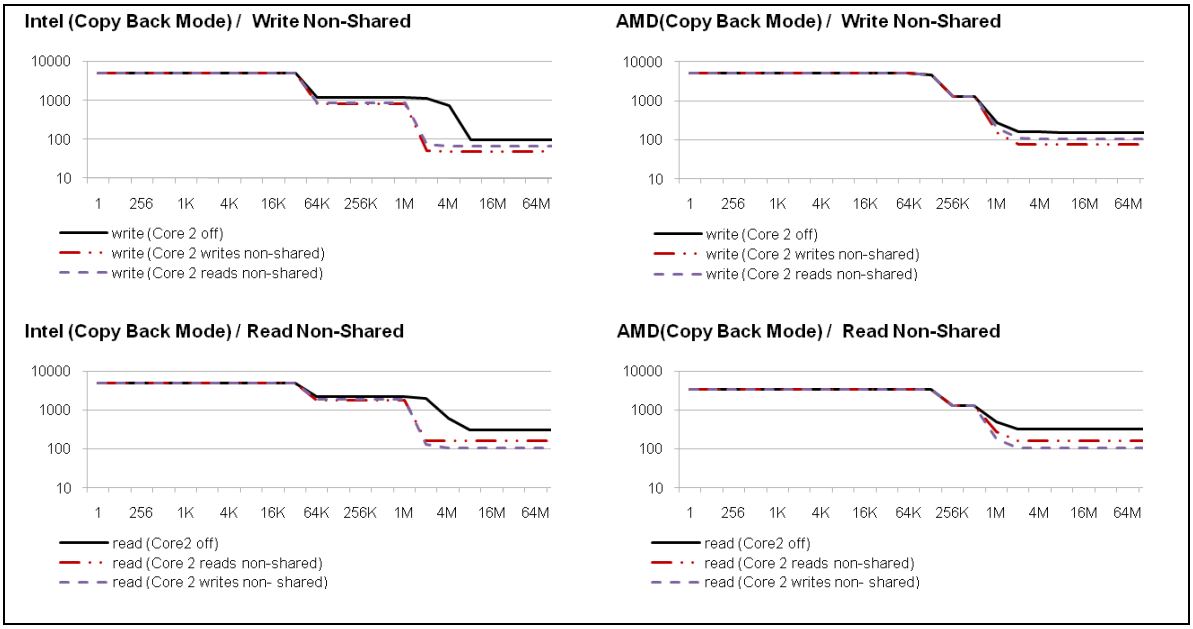


Figure 2. Read and write throughput on separated data sets (Mbyte)

Cache Coherency: Another important aspect related to the use of caches is the consistency of local caches connected to a shared resource. Cache coherency is of particular importance in multi-core systems with unified memory. Cache coherency means: "*If one of the local caches of core C_A contains a reference to a physical resource P_x and the cached value is more recent than the value stored in P_x, any read access from any core (including core C_A) must provide the value cached by core C_A.*"

Coherency between caches is maintained by means of the cache coherency protocol. A detailed description of the different cache coherency protocols can be found in [6].
If both cores are only reading, the second core causes almost no performance impact over the entire range of data sets in case of the Intel processor while on the AMD processor the performance drops down to 50% if the data set is larger than the L2 cache. If both cores are only writing, the Intel processor suffers much less from the concurrent read than the AMD processor but the dependency on the data set size is similar (see Figure 3).

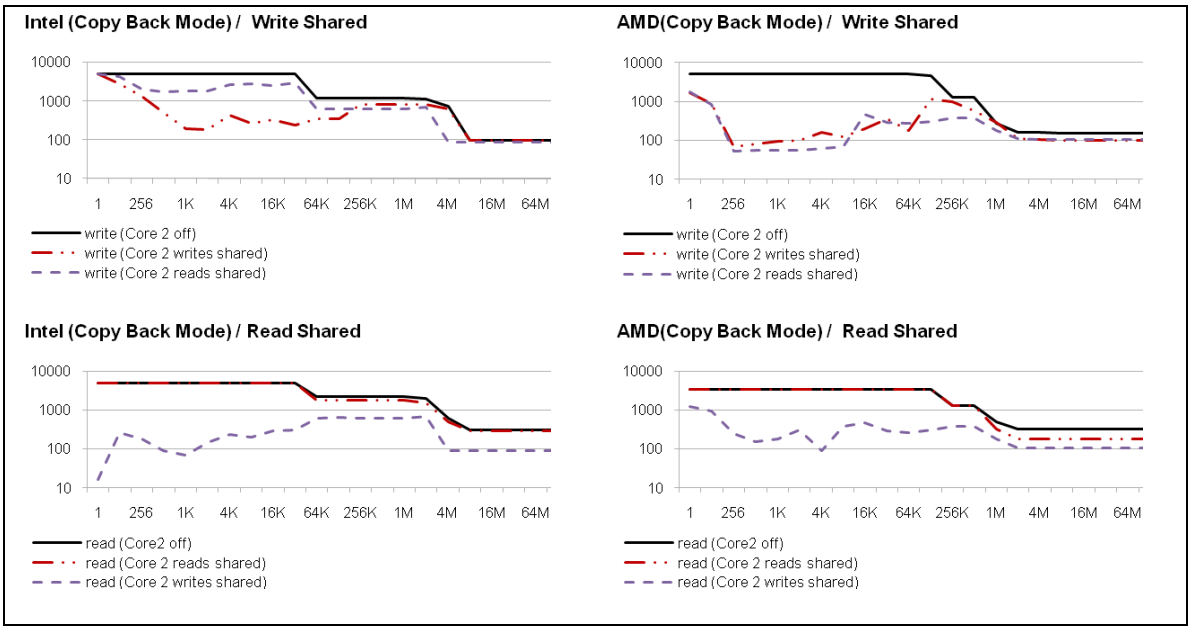


Figure 3. Read and write throughput on shared data sets (Mbyte)

If one core is reading while the other core is writing the same data set, Intel and AMD processors behave completely different. Figure 4 shows the relative throughput for both processors compared to the throughput if only one core is active. On the Intel processor, the writing CPU suffers much less from a concurrent read than on the AMD processor, however the maximum performance loss is also 90% for the case that the data set size is 4 Mbyte. On the AMD processor the performance loss is 99% on small data sets and it moves towards 50% for large data sets. If we compare the read performance loss we see that on the Intel processor, that the reader is almost completely blocked on very small data sets. This effect does not appear on the AMD processor. If the data sets get larger, Intel and AMD behave similar. For midsize sets the loss is still around 90% on both processors.

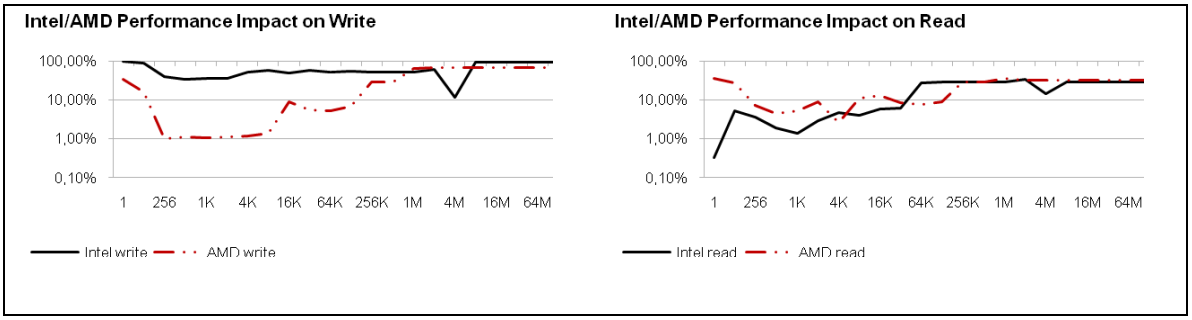


Figure 4. Relative performance loss on concurrent read and write operations

Data Buses: The results of the performance measurements (Figure 2 and 3) show that the bandwidth of the memory bus is shared between the cores. If the cores operate on a data set which is so large that the caches have no effect, the performance drops down to 50% if both cores are active. The same effect has been measured on the PCI bus. While a cache hit rate of 0% may be very unlikely when accessing memory, it is the normal case on the PCI bus since PCI devices are typically accessed with caches disabled.

Shared I/O Devices: The reduction of performance caused by concurrent access to a shared I/O device mainly depends on the bus which connects the device to the processor (e.g. the PCI bus) and on the device itself. A device which can only handle one request at a time may block a second request for hundreds of microseconds.

Multi-core Processors certification guidance

The following chapter will summarize the avionic authorities MCP concerns based on EASA/FAA CAST 32 paper (rev0) which was published in 05/2014 [4]. This CAST 32 paper can be used as a good introduction to the avionic authorities MCP concerns.
The newest EASA/FAA certification position is based on the EASA MCP CRI which is currently under development by EASA/FAA. The industry was invited to participate and discuss the content of this paper in 2016. The good news is that this paper does not limit the usage of only two active cores anymore and IMA systems are considered yet. The bad news is that this paper has no public availability. Hopefully the contents will be shared to the public using a new revision of the CAST 32 or a new and public EASA MCP guidance paper.

The following summary gives a short overview about the current status of the CAST 32:
The MCP papers are valid for avionic systems on DAL level A, B and C. As mentioned before there is a smaller subset of objectives for DAL C applications. If only one core is active, only two objectives need to be considered how to ensure that the deactivated cores are not able to be activated. Depending on the specific project, some of the objectives do not to be addressed. For example, if a system on DAL C does not require robust partitioning some of the objectives dealing with this can be neglected. A summary table in the MCP CRI shows which of the objectives apply for which development assurance level (A, B, C). Some configurations are precluded from the CAST 32 and the MCP CRI considerations. This includes:

- the usage of Hyperthreading CPUs,
- the usage of two identical cores run in lock step
- processors linked by conventional data buses, and not by shared memory, shared cache, a coherency fabric / module / interconnect

The CAST-32 [4] - Topics are the following:

- **Configuration Settings**: Configuration of required, unused and dynamic features needs to be analyzed, determined and documented (see also DO-254 [2])

- **Processor Errata:** Process needs to be in place to assess MCP Errata sheets regularly
  (same approach like COTS/microcontroller certification)

- **SW Hypervisors and MCP HW Hypervisor Features** needs to be identified in plans (PSAC, PHAC) and development of such operating system shall be compliant with DO-178B/C objectives [1].

- **MCP Interference Channels:** Identification of interference channels and verification means of mitigation

- **Shared memory and caches (between processing cores):** Description of the shared resource approach in SW plans (PSAC)**.** Identification of any problem of deterministic software execution caused by the MCP approach. Analysis and tests shall be developed to determine worse case effects of shared memory and caches.

- **Planning and Verification of Resource Usage:** The used approach shall be documented in the SW plans (PSAC)**.** Allocation of resource and interconnect usage and adequate management and measurement of used capacities. Verification of resource and interconnect demands to ensure capacity limit compliance.

- **Software Planning and Development Processes**: Identify the MCP software architecture (in SW plans like PSAC) and describe the development and verification planned to demonstrate deterministic execution (which is a typical DO-178B/C objective [1]).

- **Software Verification:** The execution of verification activities shall be on the target MCP environment. Developed software need to comply with DO-178B/C [1]**.** Data & control coupling between all software components hosted via shared memory need to be verified.

- **Discovery of Additional Features or Problems:** Any other problem of MCP not described in CAST 32 need to be considered

- **Error Detection and Handling and Safety Nets:** Errors and failures (of the MCP) need to be addressed by Safety Net approach on system level.

Hypervisor / Segregation Kernel solutions to support Multi-Core certification

This chapter gives a short overview how to use hypervisor techniques to ensure proper handling of MCP issues as discussed in the previous chapters. Consequently the methodology of a separation kernel will be used.
Interference between software components running concurrently on different cores mainly depend on the software architecture and the way the software utilizes the cores. Different concepts of using multi-core processors are possible on operating level:

Asymmetric Multiprocessing (AMP): The AMP approach utilizes a multi-core processor platform very much like a multi-processor system. Each core runs its own single-core system software layer. The cores are loosely coupled through distinct communication channels which may be based on inter processor interrupts, specific shared memory regions or other external devices. The major advantages of the AMP approach are:

- The system software layer does not need to be multi-core aware which simplifies the design.
- There is no implicit coupling through shared data and critical sections inside the system software layer.
- Each core may run a system software layer which is optimized for the task to be performed. An example for this is a dual-core platform where one core is responsible for I/O processing and data concentration while the other core runs an ARINC 653 compliant OS which hosts the applications.

The disadvantages of the AMP approach are:

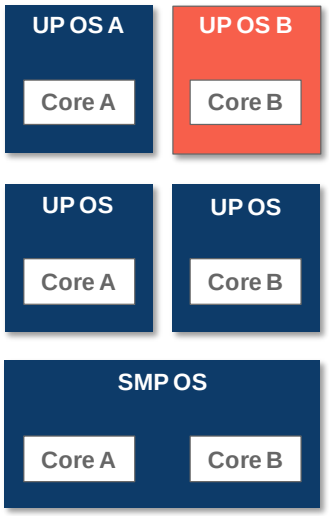
- All system software instances must be certified to the highest level applicable for the platform since they have full access to privileged processor registers and instructions.
- Partitioning of platform resources is more complicated, especially if different system software layers are used. This limits the use of the APM concept to CPUs with a small number of cores.
- Synchronization between applications running on different cores is more complex.
- The AMP approach does not support parallel execution on application level.

Interference on an AMP platform is mainly caused by shared caches, memory and I/O buses and concurrent access to shared devices. Interference on the memory bus is hard to avoid while access to I/O devices may be limited to one core. Coherency problems are limited to distinct communication buffers and interference caused by the system software is limited to shared device handles.

Symmetric Multiprocessing (SMP): The SMP approach uses one system software instance to control all cores and platform resources. The OS typically provides inter and intra partition communication services which transparently manage cross core communication. Device drivers are responsible to manage concurrent access to platform resources.

The advantages of the SMP approach are:

- There is only one system software instance responsible for the partitioning concept which limits the certification effort to one software package.
- There is only one platform configuration layer required.
- The SMP system software can completely isolate execution of critical tasks, e.g. by temporary disabling concurrent execution of non-trusted partitions.
- SMP provides much more flexibility and allows a better load balancing than an AMP configuration.
- Parallel execution on application level can be supported where interference between cores is of no concern, e.g. for non-safety related partitions.



The main disadvantages of the SMP approach are:

- The system software layer is more complex since it needs to protect its internal data from concurrent access without significant impact on parallel service requests.
- The internal data need to be arranged very carefully in order to avoid false sharing effects.
- Due to the shared system software layer, an implicit coupling of unrelated execution threads cannot be completely avoided.

Compared to an AMP configuration the SMP approach adds an important source of potential interference which is the shared system software layer. A careful design however can limit the impact by the implementation of fine-grain critical sections. The internal data of the system software layer must be carefully arranged to avoid unintended coupling due to false sharing.

The following example shall show the usage of adequate resource partitioning and time partitioning concepts to cope with multi-core issues in real-time safety-critical applications.

In future safety-critical platforms, besides the critical applications, an increasing number of applications with high performance requirements but lower criticality will be needed. These applications may also not necessarily be based on avionic APIs like ARINC 653 but they may require run-time environments like POSIX, Java or even Linux. They also may require multi-processing at application level. Due to the potential interference between applications it seems not to be feasible to run a safety critical application concurrently with a non-trusted application. The operating system must support exclusive access to the platform for the most critical applications. Intra partition multi-processing for safety critical applications also seems to be questionable because the worst case execution time analysis may become impossible.

When no critical application is running, the platform may be shared between partitions or all cores may be made available to the most demanding application.

The assumed platform is based on a quad-core CPU. The major time frame is divided into three timing partition windows. One critical single core application shall have exclusive access to one of the cores and have exclusive access to the entire platform during its time window. One performance-demanding partition shall have exclusive access to the remaining three cores during its time window. One time slot is shared between two resource partitions providing two cores for one partition and another core for the other partition.

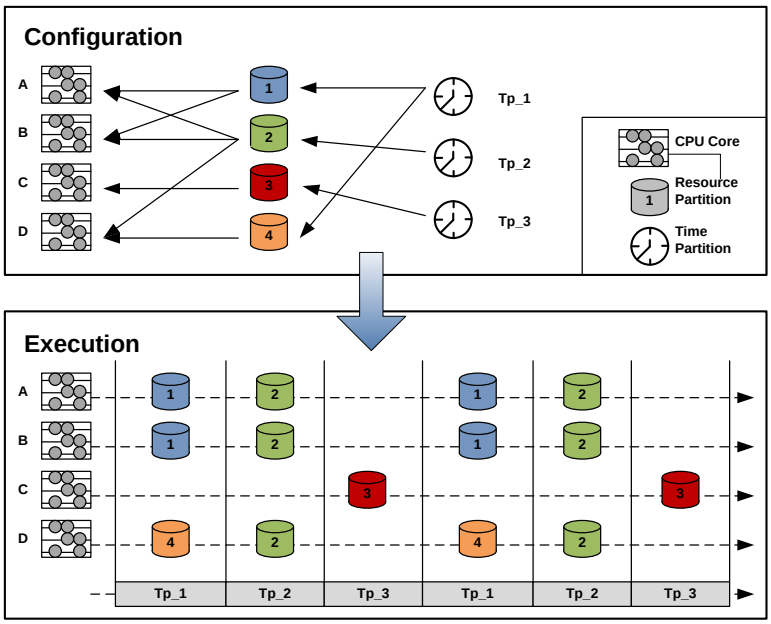


Figure 7. Example of possible MCP configuration

The selected configuration focus on a maximum level of isolation for the safety-critical real-time application accepting a significant waste of CPU time. Partition 3 is the only partition executing on core 'C' and during the time slice of time partition 3 there is no other partition execution. This eliminates any interference on hardware and software level. The level of determinism in this configuration is even better than on a traditional

single core platform since the critical application does not share the core with other partitions which also keeps the state of the private caches unchanged. This is of course a quite expensive configuration since 5 of 12 time windows in a Major Time Frame are unused.

Nevertheless the setting of caches and TLBs need to be considered. Therefore a MCP-hypervisor OS shall provide means to invalidate instruction caches and TBLs and to flush the data cache between time partition switches. This ensures that caches and TLBs are in a defined state when a partition starts execution. The cache / TLB flush and invalidate operation takes place during the time partition switch, so it will steal the CPU cycles from the partition to be activated. A possible approach is to define a small time partition window which is allocated to an unused time partition ID and to insert this before the time critical application shall be executed. This eliminates the jitter of the time critical application. This is shown in the figure below.

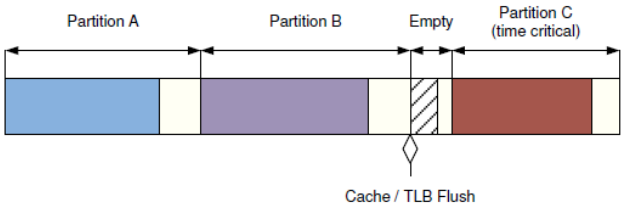


Figure 8. Cache / TLB considerations during partition switch

The platform specific worst case execution time analysis must provide the value for the worst case jitter to be considered on integration level.

MCP Compliance – Additional Analysis Documentation

Typically a Hypervisor OS shall be developed to be compliant to DO-178B/C [1]. Therefore a huge amount of development life cycle data has to be generated. Additionally analysis documentation needs to be established to help the OS integrator to be complaint to certification objectives:

- Analysis to justify that the partition concepts have been adequately specified and implemented
- Analysis of correct stack handling and definition of limits of stack usage for the OS integrator
- Analysis of WCET behavior and definition of WCET limits (e.g. jitter definition) for the OS integrator

Documentation	Means of Compliance
PikeOS Partitioning Analysis (X86)	SW Partitioning
PikeOS Stack Analysis (X86)	Stack Usage
PikeOS WCET Timing Analysis	WCET Analysis
PikeOS Safety Manual	How to use OS in a „safe“ & „secure“ way
Safety Bulletin / Errata (for cert projects)	Management of OPRs
PikeOS MultiCore Analysis (for a given CPU architecture)	Statement of EASA/FAA MCP CRI Compliance

Figure 8. Need of MCP analysis activities

It will be a good decision for MCP Hypervisor OS vendors to provide additionally an MCP analysis document to answer to the relevant objectives of the EASA/FAA CAST32 and MCP CRI. Of course this analysis will not answer the questions and problems related to the internal implementation of the MCP but will help to justify the usage of MCP with the help of a real-time hypervisor.

Current and future work

SYSGO AG provides PikeOS in version 3.4 already as certified for multi-core projects against EN 50128 SIL 4 (Railway). This approach is based on a dual-core approach and follows the MCP strategies introduced in the previous section. SYSGO AG and Thales are currently working on the preparation of next generation of PikeOS to be certifiable for DO-178C [1] SW level C, ISO26262 ASIL A/B, EN50128 SIL 1/2 multi-core projects. The next step will be a PikeOS version to be certifiable for DO-178C SW level A, ISO26262 ASIL C/D and EN50128 SIL3/4 multi-core projects.

SYSGO Engagement in multi-core Research:

Due to the fact that the certification of MCP in the avionic industry is still in the research stage the participation to research projects is important. The SYSGO AG is involved in the following research projects to push forward the MCP certifiability.

ARAMiS stands for Automotive, Railway and Avionics multi-core Systems. ARAMiS is a three-year research project that has started on December 1, 2011. It has received funding from the German Federal Ministry of Education and Research.

EMC² – ‘Embedded multi-core systems for Mixed Criticality applications in dynamic and changeable real-time environments’ is an ARTEMIS Joint Undertaking project in the Innovation Pilot Program ‘Computing platforms for embedded systems’ (AIPP5).

Ashely: Extension of DME Concepts and solutions. Multi-Domain, secured Data Distribution services to streamline aircraft data distribution.

PROXIMA pursues the development of probabilistically time analysable (PTA) techniques and tools for multi-core/many-core platforms. The project will selectively introduce randomization in the timing behaviour of certain hardware and software resources as a way to facilitate the use probabilities to predict the overall timing behaviour of the software and its likelihood of timing failure.

MCFA: SYSGO AG is supporting the working group multi-core For Avionics (MCFA) to support the EASA/FAA MCP related rulemaking process.

Summary

The use of multi-core CPUs is necessary for future safety-critical systems to deal with the increasing performance requirements. A major concern of the authorities is the lack of determinism introduced by the increased complexity of MCP CPUs adding components to systems with major impact on functionality and timing. Multi-core platforms introduce additional hardware and software interference channels between software executing concurrently on different cores. Therefore the need for additional assurance activities is raised by the authorities. As shown in this paper the MCP usage domain need to be described adequately and the major interference channels shall be identified and in best case eliminated by a smart system software design. An adequate process to select the processor needs to be established to take the effects of shared components between cores into account. The platform design must handle any usage of shared interrupt, I/O buses and I/O devices. If a real-time MCP compliant hypervisor will be used the concept of spatial and temporal partitioning can help to cope with MCP related issues. Nevertheless the safest way to ensure certification concerns in regards to MCP is the complete deactivation of all other cores if a real-time safety-critical application is executed on one core having a dedicated small amount of time exclusive access to all shared resources. This scenario is adequate for some special configurations but limits the usage of a MCP. If parallel execution of real-time safety critical software application is needed the usage of a MCP is recommended that can cope or eliminates the worst case timing and determinism problems as identified in this paper. Additionally the adequate and deterministic handling of shared caches and the usage of MCP internal busses (e.g. interconnect) shall be addressed accordingly by the MCP chip vendors.

References:

[1] RTCA, DO-178C, December 13, 2011, Software Considerations in Airborne Systems and Equipment Certification
[2] RTCA, DO-254, April 19, 2000, Design Assurance Guidance For Airborne Electronic Hardware
[3] EASA CEH, August 11, 2011, Development Assurance of Airborne Electronic Hardware, EASA CM – SWCEH – 001 Issue: 01
[4] CAST-32, May 2014, Position Paper CAST-32 on Multi-core Processors, rev0
[5] Rudolf Fuchsen, October 3-7, 2010, How to address certification for multi-core based IMA platforms: Current status and potential solutions, 29th Digital Avionics Systems Conference
[6] Udo Steinberg, 29.04.2010, Parallel Architectures - Memory Consistency & Cache Coherency, Technische Universität Dresden, Department of Computer Science-Institute of Systems Architecture, Operating Systems Group, http://os.inf.tu-dresden.de/Studium/DOS/SS2010/02-Coherency.pdf
[7] Robert Kaiser, Combining Partitioning and Virtualization for Safety-Critical Systems, SYSGO Whitepaper

Author

Sven Nordhoff is responsible as *"Director Certification"* for certification of all safety-critical products at SYSGO AG.
His work includes:

- Management and coordination of projects/products, which need to be certified against DO-178B/C, EN 50128, IEC 61508, ISO 26262.
- Preparation and execution of audits.
- Communication with customers and certification authorities / notified bodies.
- Participation to SYSGO internal process improvements of SW development, verification and validations processes.

The main activities and responsibilities are:

- Member of international working group EUROCAE/RTCA WG71/SC205 to establish DO-178C.
- 12 years experience in qualification and approval of several avionic projects (e.g. Airbus, NATO, Boeing, Eurocopter) from DAL Level A-D
- Tremendous experiences in the qualification and approval of projects in the medical, industry, rail and automotive domains based on IEC 62304, IEC 61508, EN50128 and ISO26262.
- Participation to process improvement programs based on CMMI / ISO15504/SPICE
- Principal ISO-15504/SPICE Assessor, Coach for different seminars for process improvement and quality assurance & management. Coach for DO-178B/C, DO-278 and DO-254 seminars.

Contact

Internet: www.sysgo.com
Email: info@sysgo.com