RECOMP Project

Multicore In Real-Time Systems – Temporal Isolation Challenges Due To Shared Resources

Ondřej Kotaba (Honeywell)
Jan Nowotsch, **Michael Paulitsch**, Dietmar Geiger (Airbus Group)
Stefan M. Petters (CISTER)
Henrik Theiling (SYSGO)

The Project RECOMP

Reduced Certification Costs Using Trusted Multi-core Platforms

- ARTEMIS Project with 42 partners from 8 European countries
- Period of performance: 1.4.2009 to 31.3.2013

- **Overview:** The increasing demand for processing power poses new challenges on the design of modern embedded systems. The adoption of multi-core processors seems to be a promising approach to tackle these challenges and to achieve further performance improvements combined with a reduction of energy consumption. Multi-core architectures are well known from the domain of desktop computing. However using these architectures in safety-critical applications such as aerospace, automotive, health or industrial automation leads to additional requirements, because such systems have to be certified according to domain specific safety-standards. Corresponding certification processes have been developed and established for the area of single-cores only, without taking the issue of multi-cores into account until now. An optimal solution for the field of multi-cores would be given by a modular certification process. Modularization enables saving of costs and time due to the reuse of previously certified components.

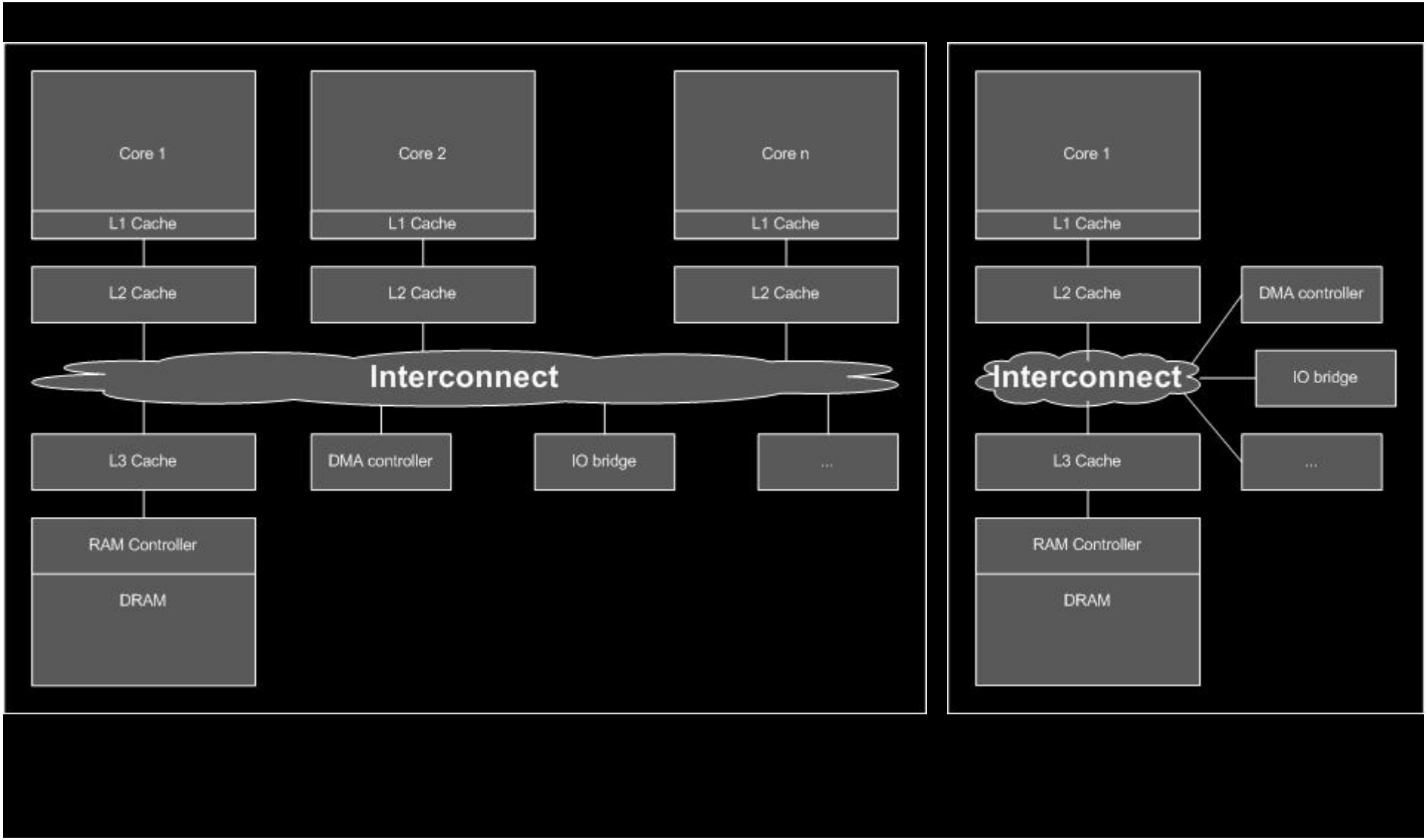
Introduction to issues

… or what this is about?

Assumptions taken

- Primarily Aerospace applications, but applicable in other fields as well
- Mixed-criticality applications
- Hard realtime required
- Temporal determinism required
  - Per application, not per function or instruction

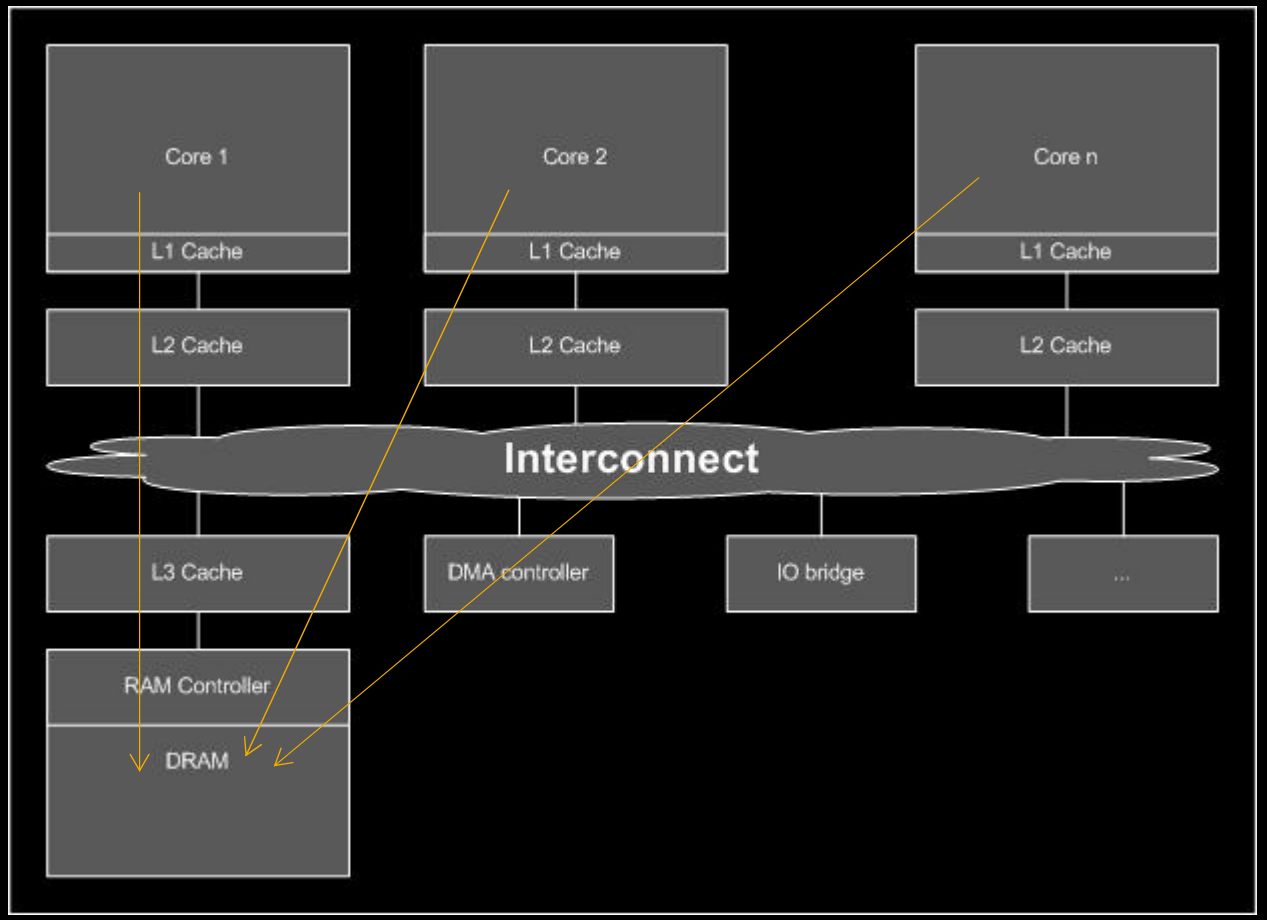
Multicore vs. Singlecore



Realtime in singlecore

- Scheduling the CPU time
  - Cooperative, Event driven, Time sharing , …
- Other chip resources almost dedicated for the CPU … or
  - Serve within a certain deadline (ongoing job)
  - Can be disabled (e.g. DMA)
  - Impact is analyzable (e.g. HW interrupt request)
- … in other words, scheduling of other resources is tied to CPU time cheduling

Shared resources problem



State-of-the-art

- Special hardware available but expensive
  - Focus on General Purpose Processors
- GPPs are optimized to achieve low Average Case Execution Time
- Many research activities, each focusing on a specific problem (or subset of problems)
- Most of the problems solved **when isolated**

Analyses performed

- Looked at what resources are shared by cores
- Looked at what problems that causes
  - Specific mechanisms
- Looked at solutions available
  - Various layers: HW, Hypervisor, OS, Application, development and verification process

Challenges per resource

… or how specific shared resources affect the time determinism

Shared resource	Mechanism
System bus	Contention by multiple cores Contention by other device IO, DMA, etc. Contention by coherency mechanism traffic
Bridges	Contention by other connected busses
Memory bus and controller	Concurrent access
Memory (DRAM)	Interleaved access by multiple cores causes address set up delay Delay by memory refresh
Shared cache	Cache line eviction Contention due to concurrent access Coherency: Read delayed due to invalidated entry Coherency: Delay due to contention by coherency mechanism read requested by lower level cache Coherency: Contention by coherency mechanism on this level
Local cache	Coherency: Read delayed due to invalidated entry Coherency: Contention by coherency mechanism read
TLBs	Coherency overhead
Addressable devices	Overhead of locking mechanism accessing the memory I/O Device state altered by other thread/application Interrupt routing overhead Contention on the addressable device - e.g. DMA, Interrupt controller, etc. Synchronous access of other bus by the addressable device (e.g. DMA)
Pipeline stages	Contention by parallel hyperthreads
Logical units	Contention by parallel applications
	Other platform-specific effects, e.g. BIOS Handlers, Automated task migration, Cache stashing, etc.

O. Kotaba, J. Nowotsch, M. Paulitsch, S. Petters, H. Theiling. Multicore In Real-Time Systems - Temporal Isolation Challenges Due To Shared Resources - WICERT2013 (DATE)

System bus

- Problems with
  - Contention
    - By multiple cores
    - By other device - IO, DMA, etc.
    - By coherency mechanism traffic
  - Typically no sufficient information to fully analyze behavior of system bus

RAM, RAM bus and controller

- Problems with
  - Contention
    - By concurrent access
  - Interleaved access causes line set-up delay
  - Typically no sufficient information to analyze behavior of RAM controller

Shared Cache

- Problems with
  - Cache line eviction
  - Contention
    - By concurrent access
  - Coherency
    - Read delayed due to invalidated entry
    - Delay due to contention by coherency mechanism read requested by lower level cache
    - Contention by coherency mechanism on this level

Local Cache

- Problems with
  - Coherency
    - Read delayed due to invalidated entry
    - Contention by coherency mechanism read

Logical units, pipeline stages

- Problems with
  - Contention
    - of instruction pipeline stages by parallel hyper-threads
    - of shared logical units (some processors)

Addressable devices

- Problems with
  - Overhead of locking mechanism
    - Accessing the RAM or any other device
    - Delay of ensuring coherency of locking – if possible at all
  - I/O Device state altered by other thread/application
  - Interrupt routing overhead
  - Contention on the addressable device - e.g. DMA, Interrupt controller, etc.
  - Synchronous access of other bus by the addressable device (e.g. DMA) (already mentioned)

and other effects

- Additional problems are very platform specific
  - Non-determinism introduced by cache stashing
  - Thermal-related capabilities
    - Automatic frequency adjustments
    - Automatic task migration to even out the load
  - BIOS functions
    - Handlers and microcode
    - Emulation

Conclusions

… or what can we do then?

What is unresolved without HW

- Contention on System bus / Network-on-chip
  - HW Solutions exist
  - Detailed documentation needed
- Memory-related effects
  - HW Solutions exist
- Memory controller
  - Detailed documentation needed

High-performance applications

- Typical COTS General Purpose Processors are very complex and depth of published documentation may be insufficient
- Focus should be on **software mechanisms** for arbitration of access to shared resources Can be achieved by
  - Hypervisor
  - OS (scheduler)
- No mature solution exists

(Relatively)

Low-performance applications

- Modern architecture of multicore COTS Microcontrollers provides appropriate mechanisms to solve the issues
  - At least for singular of multiple cores
- In the near future, we expect further expansion of the market
- Multicore microcontroller can even today provide a feasible hard-real-time solution.

Niche applications

- Special requirements and custom designs
  - Defense and space, research, etc.
- Thinking "manycore" makes system more deterministic
  - If the architecture is scalable to high number of cores, sharing effects must be bounded or controllable
  - Not necessarily having many cores

Recommendations

- Research is needed in software arbitration of access to shared resources
- Small HW architecture modifications would bring significant benefits – or a plea to chip designers
- Use dataflow or other easily transformable algorithm design paradigm

Some applied things ...

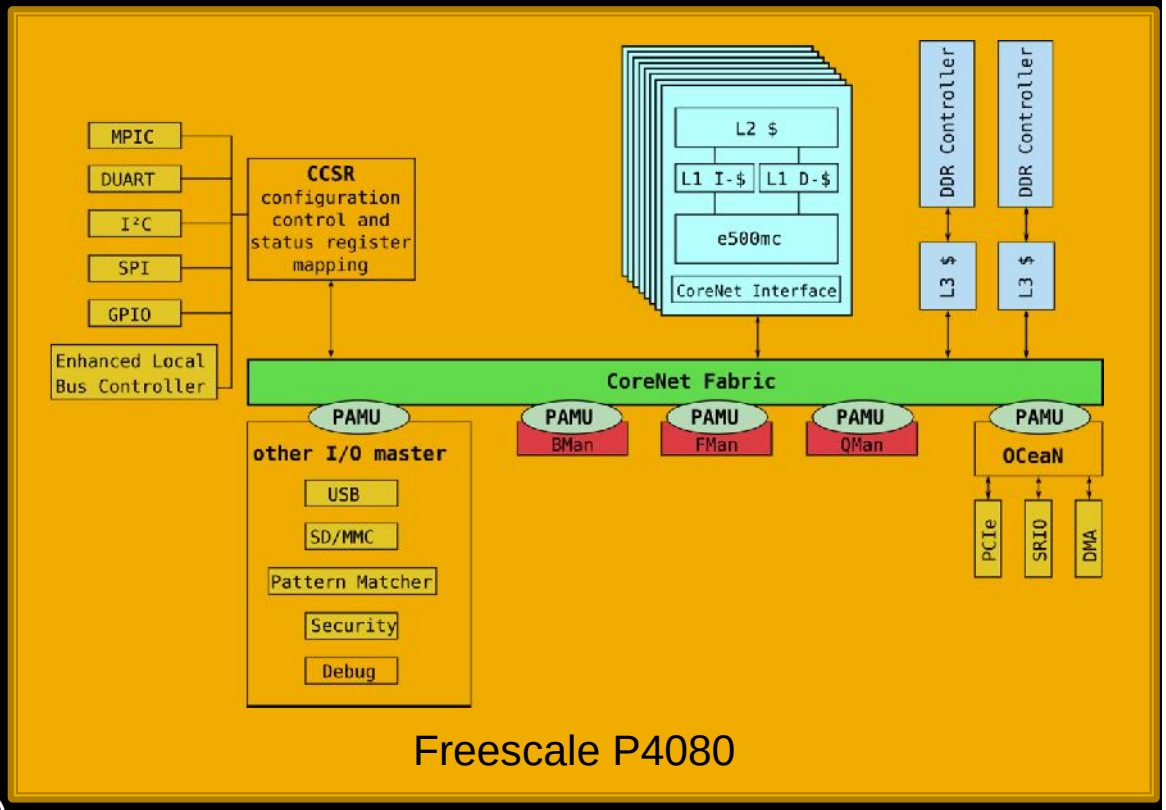
Assessment of Multi-Core Worst-Case Execution Behavior

Motivation:

- Integration leads to common use of shared resources. Partitioning impact needs to be evaluated for safety-critical applications, such as IMA

Goal:

- Analysis of partitioning features of modern multicore computer in context of use in IMA
- Impact of integration on worst-case timing (WCET) of application

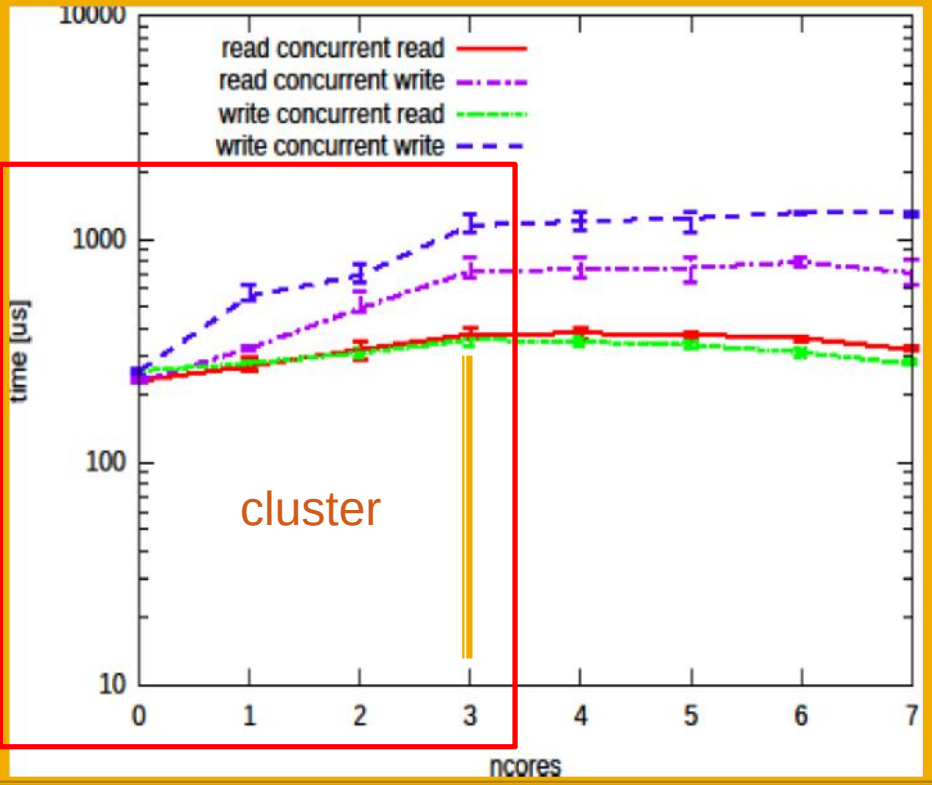


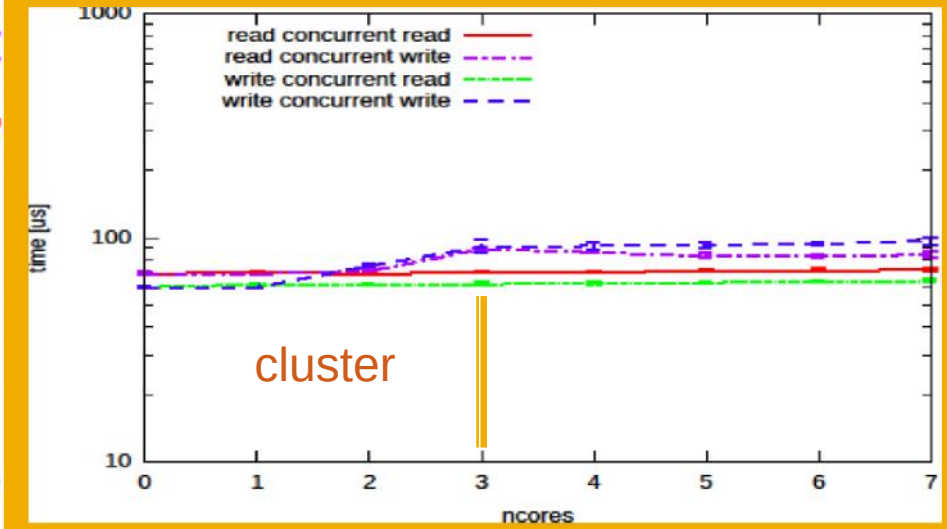
Freescale P4080

J. Nowotsch, M. Paulitsch. Leveraging Multi-Core Computing Architectures in Avionics. EDCC2012

Assessment of Multi-Core WCET Results: Access DDR vs. SRAM on Freescale P4080

- Concurrency setup; Parameter: 4kB regions, 64B gap, 1 to 8 cores, coh. req. flag off
- Key take away: **worst case access time increases over-proportionally with more cores**





Increasing number of cores active

Increasing number of cores active

SRAM - Results: no influence of certain accesses

..and that's all

We hope you liked it.

Ondrej Kotaba	ondrej.kotaba@honeywell.com
Jan Nowotsch	jan.nowotsch@eads.net
Michael Paulitsch	michael.paulitsch@eads.net
Dietmar Geiger	dietmar.geiger@cassidian.com
Stefan M. Petters	smp@isep.ipp.pt
Henrik Theiling	hth@sysgo.com