Multicores in Space

J.L. TERRAILLON
Stockholm
14/06/2012

Content

- Why multi-core exists?
- Do we need multi-core in space?
- How to use multicore?
- What products do we have in space?
- How do they perform in real-time…?
- Can we be Hard Real Time ever?…
- Conclusion

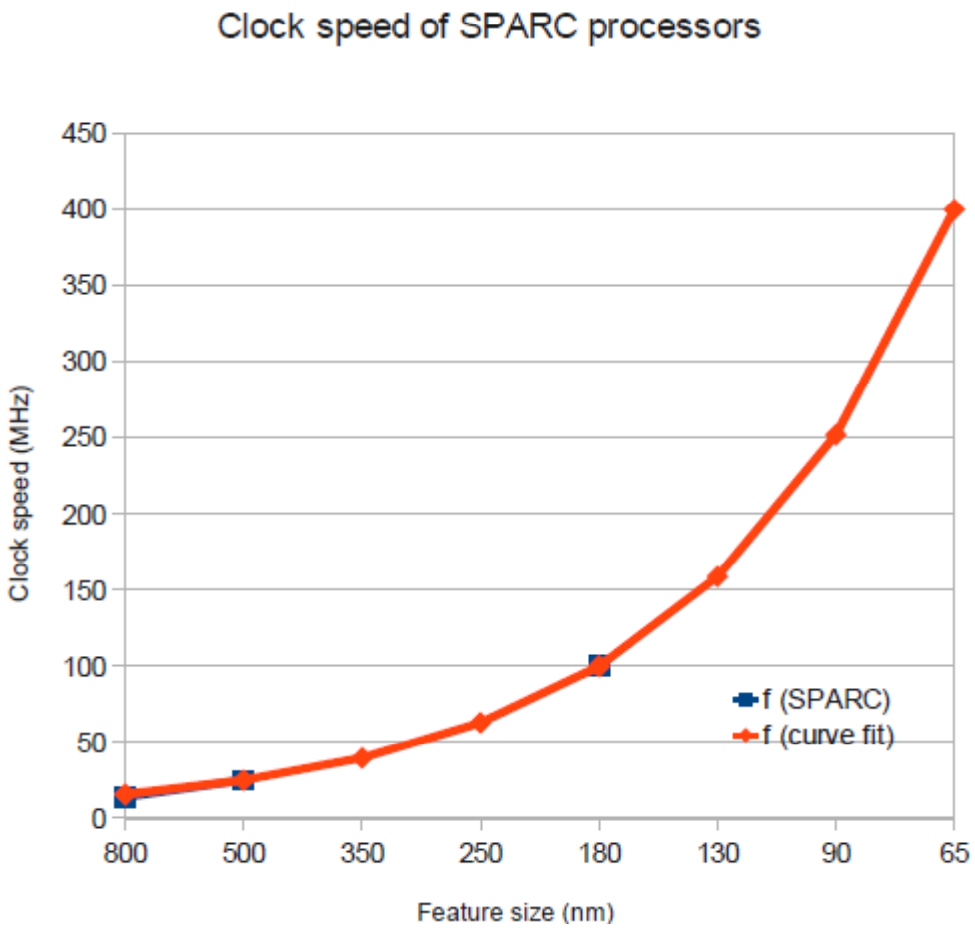
WHY MULTICORES EXIST?

Clock frequencies of SPARC processors for Space

Clock Speed of SPARC Architectures*

mid 90's	end 90's	mid 00's
ERC32 3-Chip Set	ERC32 Single Chip	LEON2-FT AT697
ATMEL (TEMIC)	ATMEL (TEMIC)	ATMEL
0.8 mm	0.5 mm	0.18 mm
SPARC V7	SPARC V7	SPARC V8
10 MIPS 14 MHz	20 MIPS 25 MHz	85 MIPS 100 MHz

Extrapolating to 90 / 65 nm we obtain 250 / 400 MHz



Multi-Core: beyond the Power wall

Increase frequency ➔ Power Wall

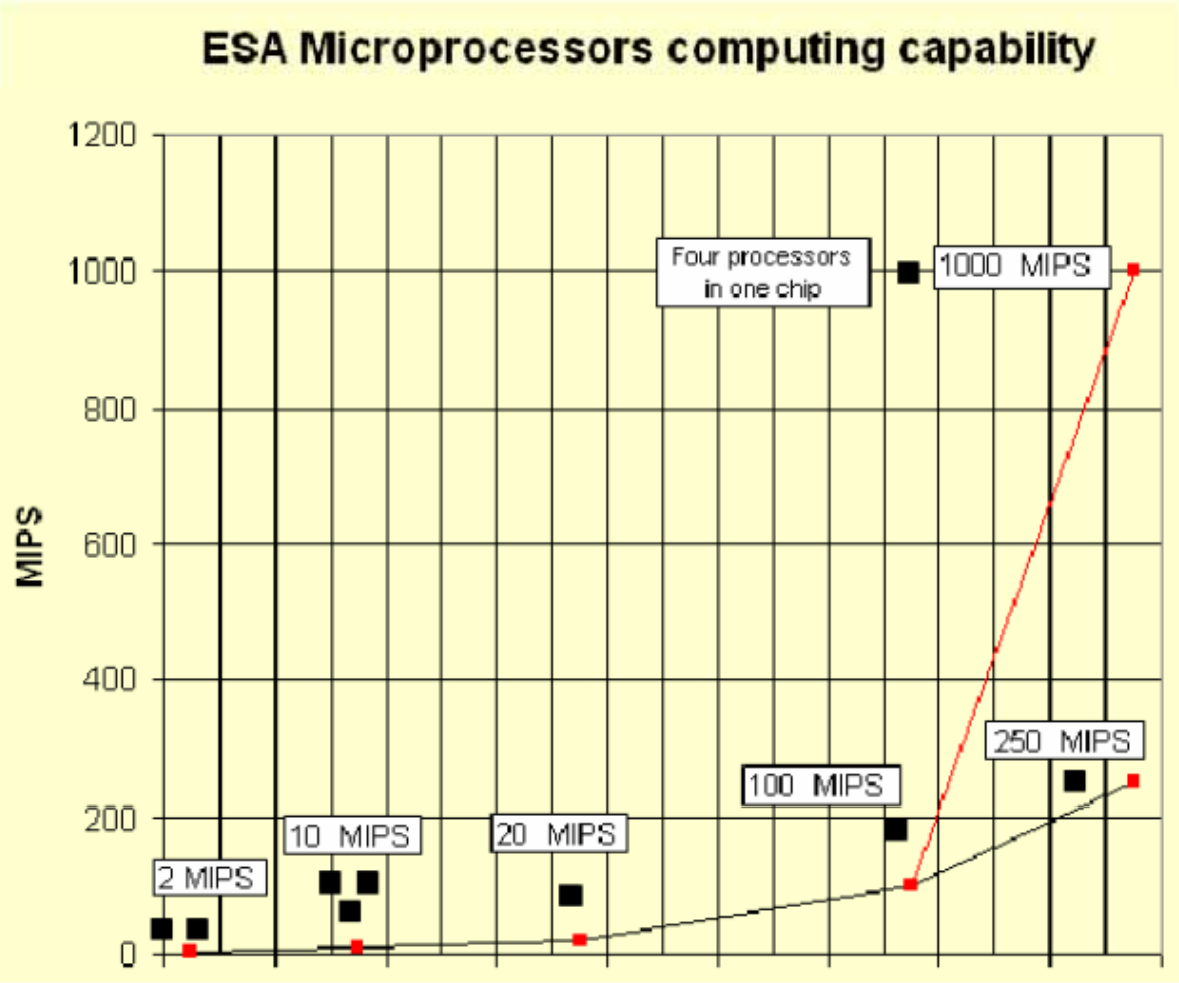
- Intel Tejas Samples Dissipate 150W of Heat at 2.80GHz. In 2004 Planned to run 7GHz. Abandonned…

Spacecraft has limited power (mono-core < 1W)

Technology & architecture limitation (on 65nm) to 400MHz/250MIPS

➔

Multi-Core



Experience from commercial world

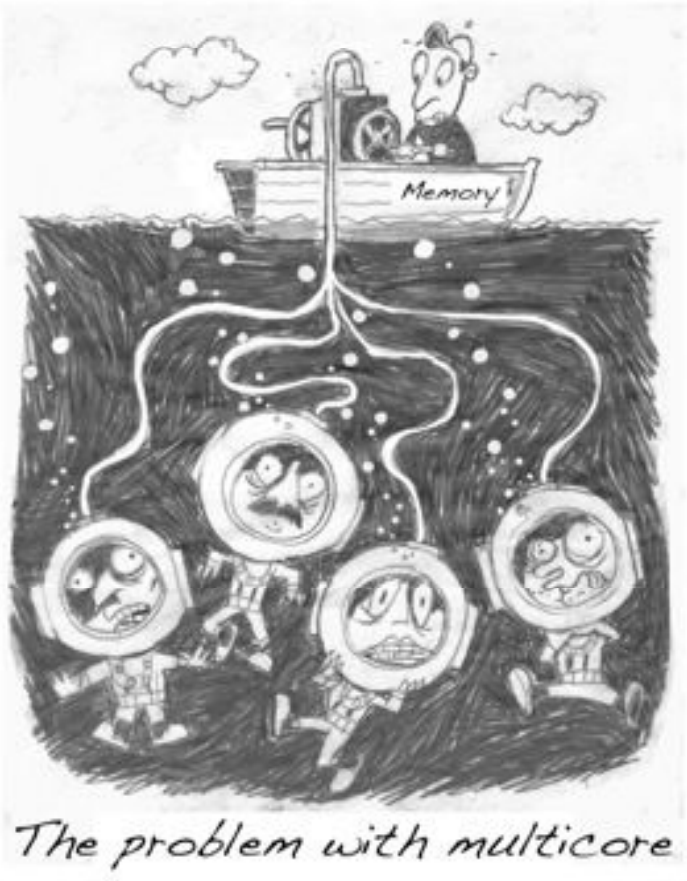
Craig Barrett, Intel's CEO at the time, used a Q&A session at the 2005 Intel Developer Forum to explain the shift:

*Question: "**How should a consumer relate to the importance of dual core?**"*

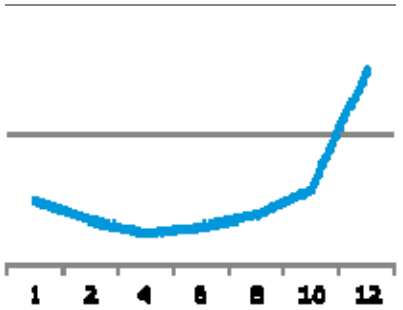
*Answer: Fair question. I would only tell the average consumer... it's the way that the industry is going to continue to follow Moore's Law going forward - to increase the processing power in an exponential fashion over time... Dual core is really important because that's how it's happening. Multicore is tomorrow... Those are the **magic** ingredients that the average consumer will never see, but they will experience [them] through the performance of the machine.*

➔ We (Space) deal with **(critical) (hard) real-time embedded** systems

The Memory wall



- Each core has a fast (expensive, small) local memory
- But all cores need to access to the main memory through a bus which becomes a bottleneck



(Illustration by Doug Davis, courtesy of Venray Technologies.)

Memory bus bandwidth

There are three ways to increase memory bus bandwidth:

- Increase memory transfer speed ➔ power increase
- Increase memory transfer size ➔ too many pins, power increase
- Move data closer to CPUs, increase cache size ➔ expensive.
- Move CPU closer to data, use (D)RAM techno to make a CPU ➔ need to simplify CPU, we don't have the right technology for space.

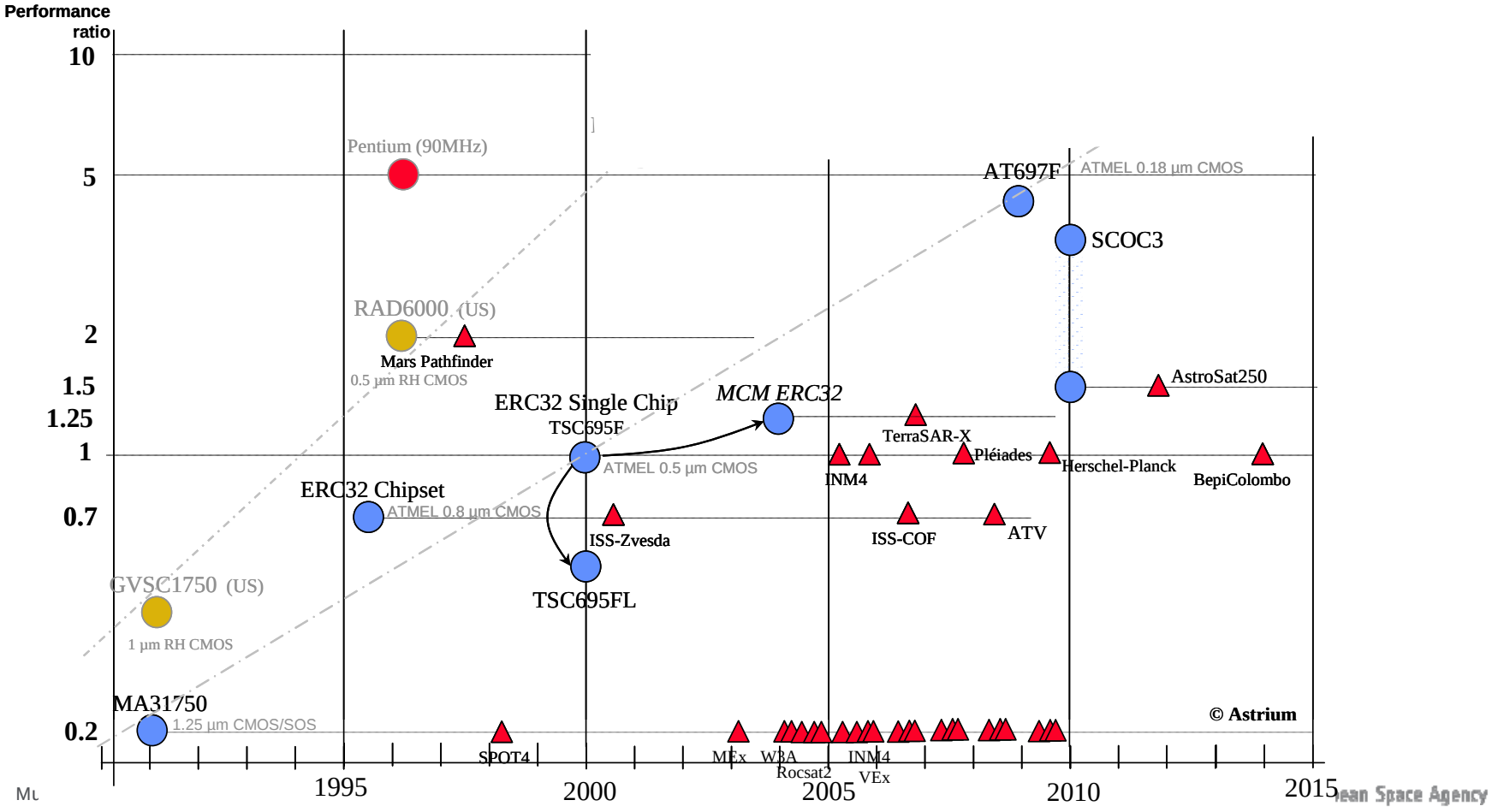
Cache hierarchy

- Write through : use the bus
- Write back : cache coherency, snooping

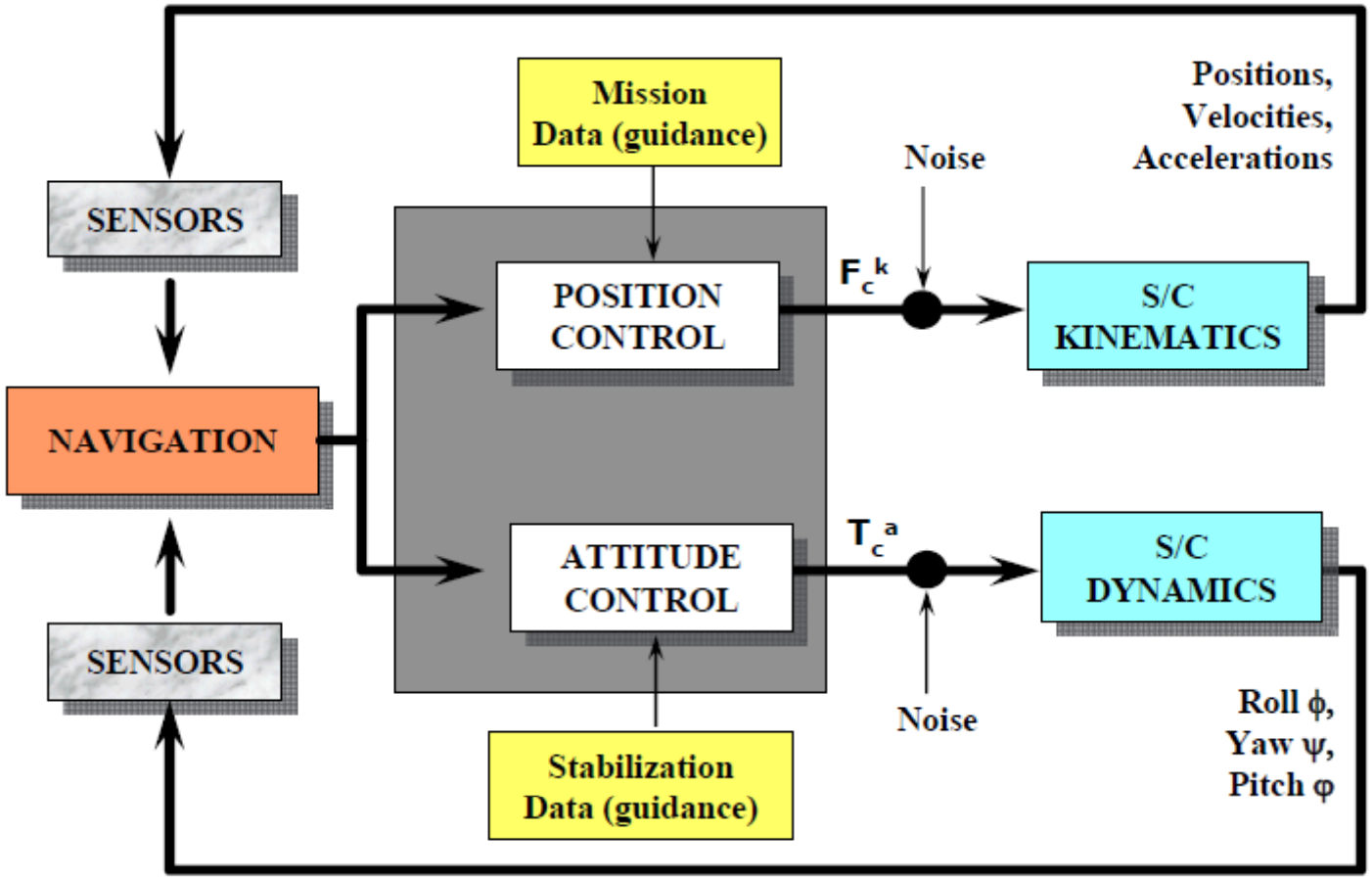
➔ Multi-cores do not look software-friendly…

DO WE NEED MULTI-CORE IN SPACE?

Processor landscape
(courtesy Astrium)



Guidance, Navigation, and Control subsystem



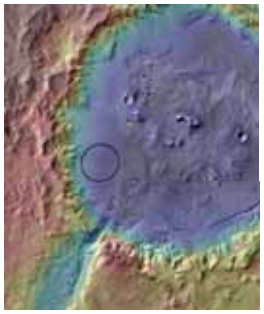
Can advanced GNC be “fit” into current computer architectures?

GUIDANCE:

- Real-time re-targeting function (trying to change the trajectory based on an acquired image)
- On-board trajectory optimization function (creation of a new trajectory after a failure or mission or vehicle reconfiguration)

1 Hz / 8 Gb RAM / 1 process / 100 MIPs

10 Hz / 1 Gb RAM / 1 process / 120 MIPs



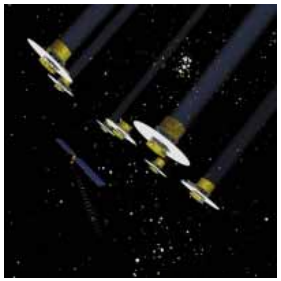
NAVIGATION:

- Image processing (point extraction and point tracking (200 points, 20 Hz, fast image processing))
- Data fusion
- State vector filtering

20 Hz / 8 Gb RAM / 1 process / 200 MIPs

10 Hz / 8 Gb RAM / 1 process / 200 MIPs

1 Hz / 1 Gb RAM / 1 process / 10 MIPs



CONTROL:

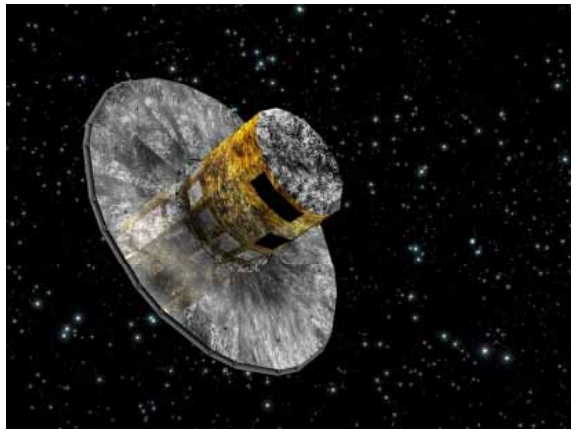
- Model-based predictive (formation flying) (prediction of the plant state in the near future)
- H-infinity (rendezvous) (robust control when many perturbations are affecting the vehicle under control)
- Dynamic inversion (entry)

10 Hz / 1 Gb RAM / 1 process / 50 MIPs

1 Hz / 3 Gb RAM / 1 process / 50 MIPs

5 Hz / 1 Gb RAM / 1 process / 50 MIPs

High performance processing



Payloads require more and more processing capabilities

- 25 years of **Spot** family: 25 m down to 1.5 m resolution, higher acquisition rate, 1 to 30 MIPS
- **Gaia** is an ambitious mission to chart a three-dimensional map of our Galaxy, the Milky Way, in the process revealing the composition, formation and evolution of the Galaxy.
  - CCD all around the spacecraft watching stars
- 3 years between Gaia and **MTG**: payload processing need x 1.3
  - Leon3 max 140 Dhrystone MIPS.
  - PowerPC used in Gaia is 1400DMips (use 1000)
  - PowerPC envisaged in MTG 1800 Dmips
  - But PowerPC consumes too much and is not rad hard so it is triplicated

Scientific payloads

- **Euclid**, a mission devised to provide insight into the nature of dark energy and dark matter by accurate measurement of the accelerated expansion of the Universe
  - CCD, photospectrometer, impossible to do on ground (L2, 4h/day download link)

- Science algorithms could be done with DSP…
- … but we don’t have

Autonomy

Autonomy

- Short contact periods with the spacecraft
- Little number of ground stations
- Ground control of several spacecraft per centre
- Limited downlink with high data rate: need for compression & filtering

Autonomy levels

- 1-none
- 2-time tagged sequence and on-board scheduler
- 3-on board procedure triggered by event and on-board interpreter
- 4-execution of **goal oriented operations on-board**: planner, scheduler, on-board inference engine, model based autonomy, model based FDIR on board

Autonomy, on-board FDIR

ARPHA = on-board diagnosis, prognosis and recovery

- model based FDIR
- prognosis on-board with two purposes: detecting the future belief state, and evaluating the future effects of recovery policies in order to select the most suitable to deal with anomalies or failures.
- ARPHA Diagnosis time depends on model complexity,
- ARPHA Prognosis time depends on n_prog and model complexity,
- ARPHA Recovery time depends on time steps of recovery and model complexity),
- ARPHA deadline depends on the system under control.
- With model of case study: 8.68 sec (diagnosis) + 43.77 sec (prognosis) + 175.37sec (recovery) = 3 minute and 47 sec
- Trade-off between model accuracy and execution time
- Need faster execution in real project, Leon3 is at the limit

➔ co-processor or multicore.

Integrated Modular Avionics

Spare electronics, group software on the same computer:

- Several payload control software (e.g. LVCUGEN Cnes)
- Platform + Payload control
- Star Tracker software on platform (star tracker head alone)
- IMU, GPS and GNC software on a single computer in rockets

- Power/mass/volume reduction

But:

- increase of complexity of the central computer SW (e.g.: TSP mechanisms are required).
- increase of data flow in/out the central computer.
- The IMA/TSP implies an incremental integration approach, which in turn assumes to master the system architecture early in the life cycle, which is a risk.
- Necessity of tools to manage the increased complexity of the system (e.g. schedulability analysis).

So, multicore or not multicore?

- COTS multi-core processors are designed for high average performance, not for timing predictability
- They bring in additional handicaps for hard real-time tasks
  - Bus is shared among the cores, bus conflicts possible
  - Shared secondary cache
  - Access to memory must be arbitrated
  - I/O and interrupt handling

➔ multi-core is a necessary evil… [ERTSS2012]

➔ we still have a bit of time to prepare ourselves

HOW TO USE MULTICORE?

Programming models survey

Autosar

- Tasks grouped by application, application allocated to cores, local scheduler to each core.
- Message queues

Cell Superscalar (Barcelona Supercomputing Centre)

- Central processor plus 8 specialized processors working on vectors
- Specific compiler aware of the architecture (annotations),
- Memory transfer are also programmed

MARS (Multicore Application Runtime System)

- Assume a master processor and several slaves
- The master load the (sub)program and data to each slave and recover the results ➔ heavy workload of the master
- Applies to GPU architectures
- Slaves may also pick up their program/data themselves

Programming models survey

STEricsson Nomadik

- Component based approach
- Components are deployed on either master ARM or slave DSP
- The software architect can bind any DSP component to his application
- Synchronisation ARM/DSP with HW semaphore

Various development environments

- TI SYS/BIOS real-time kernel
- MCAPI API based on network nodes
- OpenMP API, where code is tagged before threads must be formed
- Go[ogle] concurrent language including parallelized *Goroutines*

Lessons learned from non space

- Dedicated programming model for embedded
- Component model using standard API

Current options for space

Software choice

- Bare metal without multithreading: not appropriate to the space platform software nature
- Asymmetric multi-processing: one OS per core, appropriate if the communication between cores is limited
- Symmetric multi threading: one OS distributes the tasks to the core depending on current workload and pre-defined policy/strategy,
→ but predicting real-time behaviour seems impossible (WCET with caches, correlation of tasks sharing data between cores)

Therefore

- Platform software (Hard Real Time): Asymmetric with hypervisor
- Payload software (performance): Symmetric or bare metal

An interesting coincidence…

Integrated Modular Avionics, Time and Space and Secured Partitioning…

Use cases for Partitioning

- Spin in from aeronautical IMA
- Multiple levels of criticality in the same software: to save electronics, several software in the same computer (e.g. platform/payload, multi-payload, GPS/inertial, etc.)
- Decouple life cycle of software applications, rationalise integration, optimize (re)validation
- Multiple levels of security in the same software

Cores are natural partitions

- One core, one partition
- Several partitions in a core
- A partition on several cores

TSSP and Multi-Core: Strategies and Benefits

Partitioning similar in principle to a distributed system

- With additional design constraints

Partitioning/multi-core is a logical and powerful combination

- Designing for partitioning is a good way to move to a multi-core system

Partitions could be assigned to cores statically or dynamically (AMP vs SMP)

- Static assignment preferred
- Permits load balancing
- Partition-core assignment could be changed without a full re-validation

Dependability

- If one core fails, there are others to recover.
- However, they need to be available (spares), and hardware generally fails per chip, not per transistor…

Multi-Core Hardware Issues with TSSP

BUT issues for safety and mainly for secure (dual-use) software

- Real-time uncertainty of multicore (through bus contention) affects the "time partitioning"
- Sharing of the bus affects "space partitioning"
    - Note: on a mono-core, interrupts and DMA are not permitted in a secure system…
- Space partitioning of DMA access can be re-established through IOMMU
- Time partitioning can be re-established through bus arbitration by use of bus controller and interrupt controller.
- Cache must be flushed after partition switch and hypervisor call
- Security holes between supervisor and user mode in cache data. Need hypervisor mode with more restrictions?
- Permissions to be added to cache access and interrupt handler.

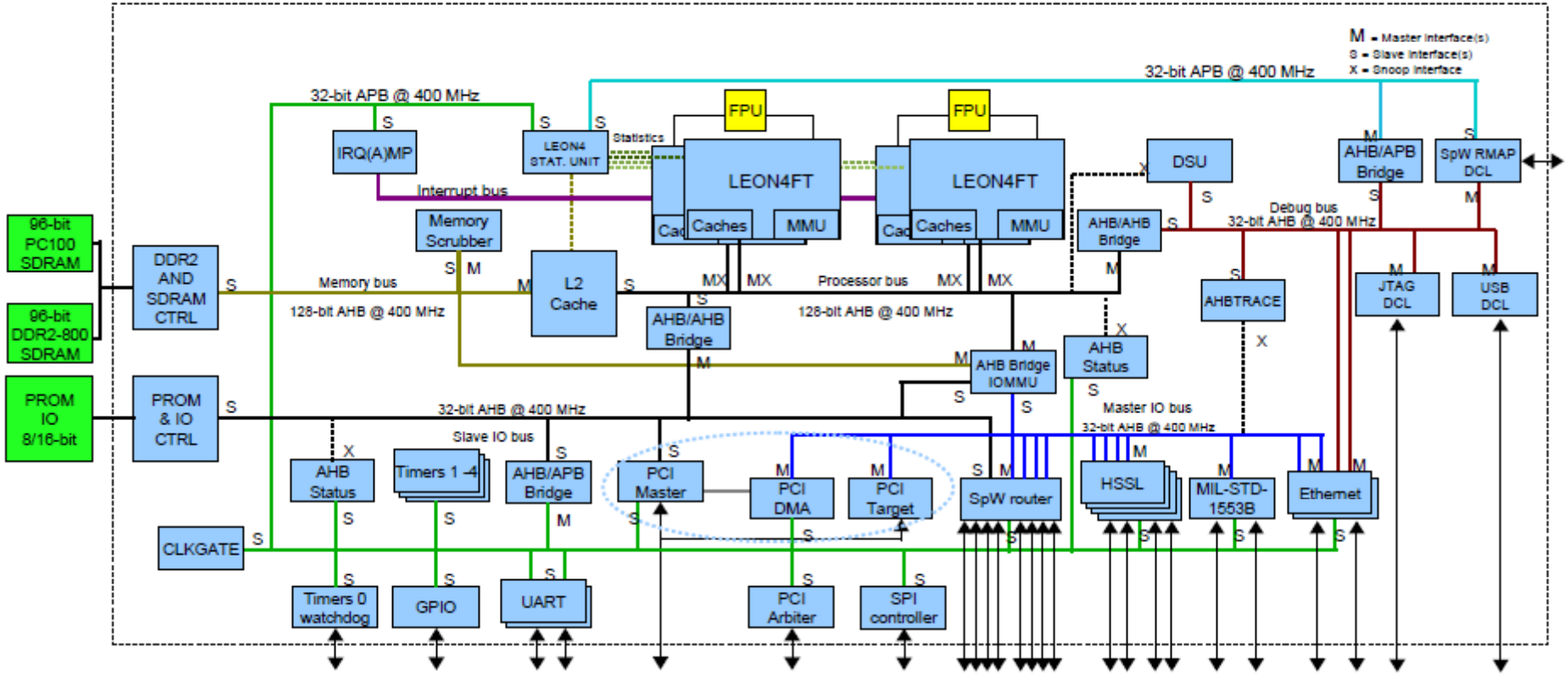
WHAT PRODUCTS DO WE HAVE IN SPACE?

The NGMP data sheet

Need: No ITAR – Low power – Low cost ➔ European Sparc based

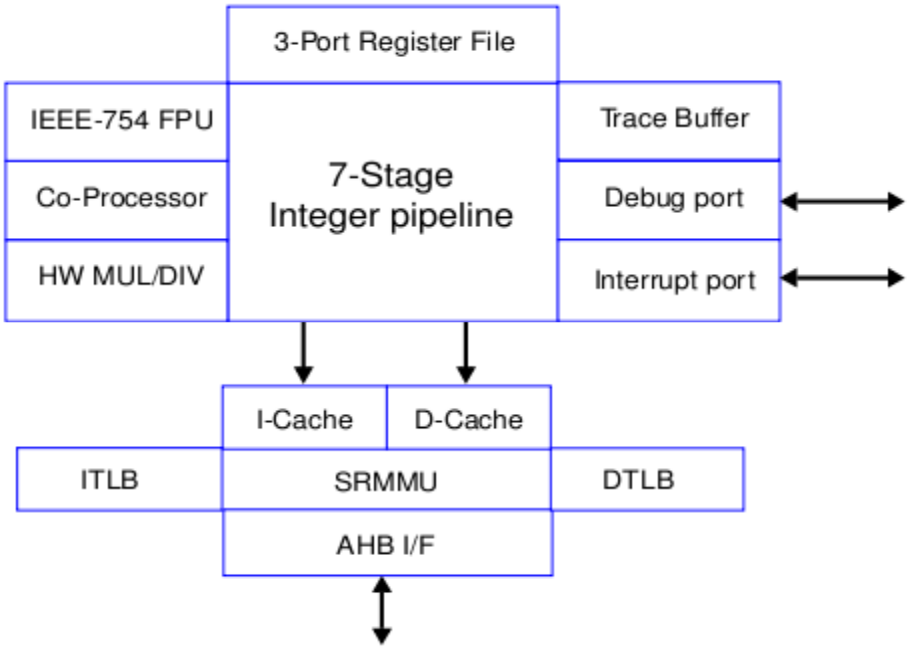
- Quad-core LEON4FT with two shared GRFPU floating point units
- 128-bit L1 caches connected to 128-bit AHB bus
- 256+ KiB L2 cache, 256-bit cache line, 4-way LRU
- 64-bit DDR2-800/SDR-PC100 SDRAM memory interface
- 32 MiB on-chip DRAM (if feasible, not applicable for FP)
- 8-port SpaceWire router with four internal AMBA ports
- 32-bit, 66 MHz PCI interface
- 2x 10/100/1000 Mbit Ethernet
- 4x High-Speed Serial Links (if available on target tech, not applicable for FP)
- Debug links: Ethernet, JTAG, USB, dedicated SpW RMAP target
- 16xGPIO, SPI master/slave, MIL-STD-1553B, 2xUART
- Target frequency: 400 MHz
- Maximum power consumption: 6W. Idle power 100 mW.

NGMP diagram



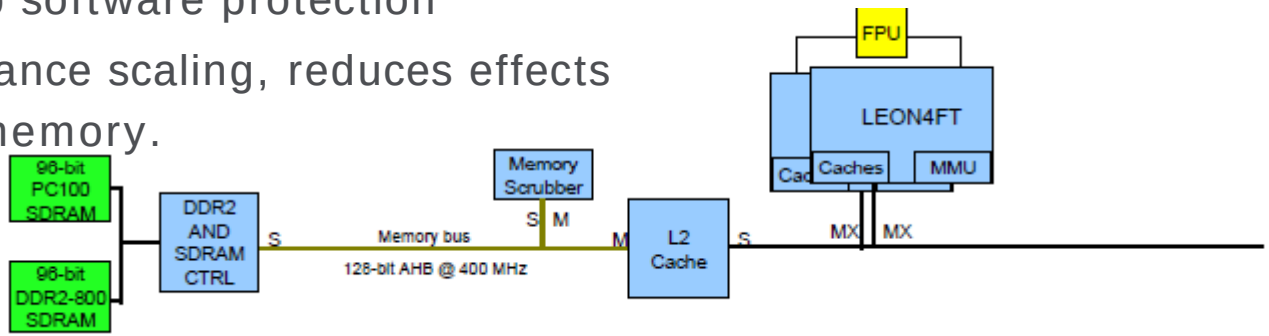
Architecture - LEON4FT

- IEEE-1754 SPARC V8 compliant 32-bit processor
- 7-stage pipeline, multi-processor support
- Separate multi-set L1 caches with LRU/LRR/RND, 4-bit parity
- 64-bit single-clock load/store operation
- 64-bit register file with BCH
- 128-bit AHB bus interface
- Write combining in store buffer
- Branch prediction
- CAS support
- Performance counters
- On-chip debug support unit with trace buffer
- 1.7 DMIPS/MHz, 0.6 Wheatstone MFLOPS/MHz
- Estimated 0.35 SPECINT/MHz, 0.25 SPECFP/MHz
- 2.1 CoreMark/MHz (comparable to ARM11)



Architecture - Level-2 Cache

- 256 KiB baseline, 4-way, 256-bit internal cache line
- Replacement policy configurable between LRU, Pseudo-Random, master based.
- BCH ECC and internal scrubber
- Copy-back and write-through operation
- 0-waitstate pipelined write, 5-waitstates read hit (FT enabled)
- Support for locking one more more ways
- Support for separating cache so a processor cannot replace lines allocated by another processor
- Fence registers for backup software protection
- Essential for SMP performance scaling, reduces effects of slow, or high latency, memory.



Software elements

Gaisler software products

- Operating systems will be ported on NGMP (RTEMS 4.8 and 4.10, WindRiver VxWorks 6.7 with SMP support, eCos 2.0, Linux 2.6)
- The GNU C/C++ toolchain will be used (Versions 4.1.2 and 4.4.2 have been successfully tested, OpenMP requires GCC 4.2 onwards and a pthreads implementation
- MKPROM2 with support for booting SMP and AMP
- NGMP simulator based on GRSIM (accuracy goal is above 90% over an extended simulation period)
- NGMP will be fully supported by the GRMON debug monitor

PikeOS (SysGo) (Leon 3 monocore, some multi-core, but not on NGMP)

ESA investigations

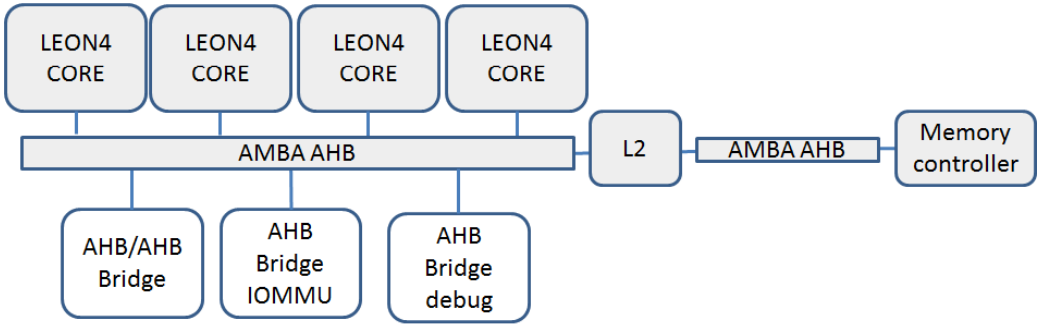
- Hypervisor Xtratum for AMP (with Astrium)
- RTEMS 4.11 for SMP on dual core (Cnes as well)
- Development environment for multi-core (ITT TRP to be publish soon)
- Schedulability analysis for multi-core (intended TRP 2013)

HOW DO THEY PERFORM IN REAL-TIME...?

Barcelona Super Computing Center Benchmarks

Inter-task interference analysis

- Four of the main sources of inter-task interferences: the AMBA AHB processor bus, the shared L2 cache, the shared memory controller and the write-through policy of the L1 data cache.
- Benchmarks have exercised the impact of various source of inter-task interferences (using Linux and RTEMS operating systems)



Benchmarks results

- When several bus-hungry tasks try to access at the same time the AMBA AHB processor bus, they may suffer an overhead of up to **1.75**x in Linux and **1.95** in RTEMS.
- The combined effect of the interaction in the memory controller and the AMBA AHB processor bus introduces a variation of **2.8**x for Linux and **3.4**x for RTEMS.
- The combined effect of the three resources, i.e. AMBA AHB processor bus, L2 cache and memory controller makes the execution time increase up to **4x** for Linux and more than **9x** for RTEMS.
- Finally, and more surprisingly, applications with high number of stores may suffer slowdowns higher than **19x**. This is due to the fact that the first level data cache is write-through, so every single store has to go to L2, suffering significant slowdowns.

- On the one hand, this provides a way to provide safe WCET bounds to the execution times of applications in the NGMP. On the other hand, this determines how applications have to be scheduled to reduce inter-task interferences effect on WCET bounds.

L1 is write-through

- For NGMP time composability is affected by:
  - the percentage of store instructions and
  - if the application has few number of stores, whether it fits in the 1st level data cache.
- L1 cache write-through nature affects significantly the execution time of application with high percentage of stores (up to a 20x slowdown). This because store operations always access a shared resource, the processors main bus.
- At hardware level, a write-back policy for the L1 data cache may be considered as it will significantly reduce the overhead on applications execution time due to inter-task interferences.

- However, write-back for L1 means EDAC instead of parity-bit checking and reload from L2.

EUCLID: Computational needs LEON2 results

- Need: real time processing of 2048*2048 pixels (16 bit per pixel) after 560s of exposure time
- baseline : 30 groups of 8 samples and 5 discarded
- Software has between 15s to 32s (time between groups) to process each frames (2048*2048 pixels)

- Operations can be done in parallel for each outputs, odd and even pixels
- For each frame (1024*1024), tprocess= 9.8s, (LEON2 80MHz) so **40s for 2048*2048 pixels!**
- Improvement can be done (only integer is on going) but multi-core architecture is beneficial

NGMP preliminary results

- Execution time speed up on NGMP (4 cores) = **1.8**

- But:
  - on simulator, not on real hardware (representativeness)
  - with less observables than real hardware
- If not due to simulator, potential explanation : the way L1 cache is used
  - the algorithm loop and write in L1. As L1 is write-through, each loop use the AMBA bus to write in memory ➔ contention
- MMU not disabled (slow down cache miss)

- ➔ careful programing style when in need of performance

- ➔ better use a DSP? None yet for Space

CAN WE BE HRT EVER?...

MERASA

- Multi-Core Execution of Hard Real-Time Application Supporting Analysability
- EC FP7 2.1M€ 2007-2010
- MultiCore hardware designed for Hard Real Time
- WCET analysis tools for multi-core (OTAWA (Toulouse) & RapiTime)
- Software cofing rules & adapted OS: Isolation of HRT threads, time-bounded thread synchronisation

RAPITA

IRIT

BSC Barcelona Supercomputing Center

Universität Augsburg

Honeywell

Core:

- Two-way in-order superscalar, 4-way simultaneous multithreaded (SMT) core
- Internal **hard real-time scheduler** performs SMT execution
- All pipelines are designed **non-blocking**
- Long running instructions implemented by **preemptible microcode sequences**
- Hard real-time tasks uses instruction and data **scratchpads** local to core (to limit interferences)

Multi-Core:

- Core-local scratchpads combined with Dynamically Partitioned Cache decrease number of accesses to shared memory
- Round-robin/time triggered (TTA) access to bus and memory bounds access times
- Hard real-time capable DDR2 SDRAM controller
- Support for thread synchronisation implemented in SDRAM controller to decrease WCET overestimation

ParMERASA

technische universität dortmund

DENSO Europe

- Multi-Core Execution of parallelised Hard Real-Time Application Supporting Analysability EC FP7 3.3M€ 2011-2014
- Techniques to parallelise applications and verify HRT, 16 to 64 cores

- ➔ low expectations to have a specific HRT multicore for space
  space does not drive the market
  too expensive

SO WHAT?

Let's face reality

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Dipartimento di Matematica Pura ed Applicata

INRIA

RAPITA SYSTEMS LTD

AIRBUS

PROARTIS
Probabilistically Analyzable Real-Time Systems

Probabilistically Analysable Real-Time Systems

- FP7 2010-2013
- Estimation of WCET needs accurate hardware knowledge:

➔ less and less possible

- Uncertainty in WCET relates to history of previous software execution e.g. cache miss:

➔ less and less analysable

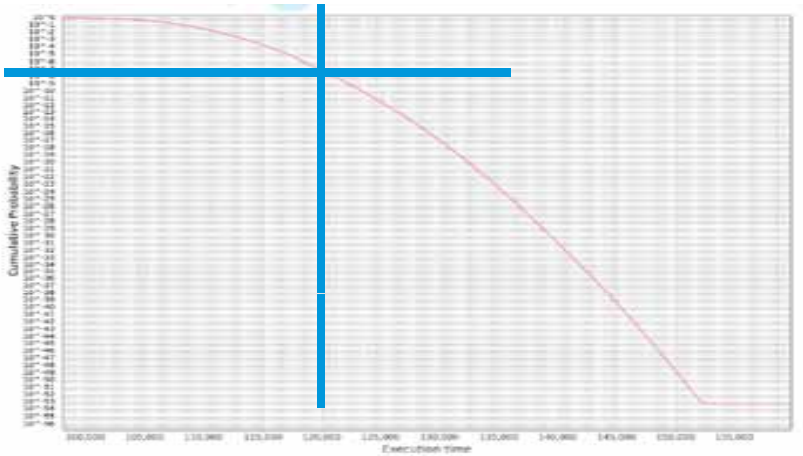
- Time composability does not work, too many dependences.
- ➔ pessimistic WCET, not used
- Instead, average execution time with a "margin"…

...and think out of the box

Novelty: remove dependencies on history and hardware knowledge

- Be fully stochastic: random hw execution and probabilistic timing analysis
- Therefore, software has a probability to miss a deadline
- Need random hardware: (clever) random cache placement and replacement
- Average execution time is worst (10+%)

10-7
20%



CONCLUSION

Conclusion

- Multi-cores seem to be a necessary evil
- Space domain will have to use them but has still time
- For some tasks, DSP may be used with heterogeneous architecture and a software component model, but we don’t have yet a DSP…
- Multi-cores will probably be used in combination with TS_P with hypervisor for platform, and maybe in SMP for payloads.
- But probably not for Security (dual-use)…

Conclusion

- We need work on hardware understanding, parallel programming models, software products…
  - Qualified Hypervisor
  - Hypervisor with dynamic scheduling policy for partitions, interrupt driven
  - Tools to analyse and design the integrated system, especially for what concerns time partitioning
  - Qualified SMP RTOS;
  - Adequate design, debug and analysis tools to manage SMP SW systems in real time systems
  - Language extension(s) to ease concurrent programming and/or Ada 2012

- We have to prepare our software Product Assurance colleagues to think in terms of stochastic software errors…

Credits

CREDITS

- ESA colleagues' work (André Pouponnot, Augustin Fernandez-Leon, Roland Weigand, Luca Fossati, Giorgio Magistrati, Guillermo Ortega, Marco Zulianello, Andreas Jung, etc)
- Astrium study "System Impact of Distributed Multicore Systems" (Contract ESTEC 4200023100)
- BSC study "MultiCore OS benchmarks" (Contract ESTEC 4000102623)
- MERASA, parMERASA, ProARTIS