Final Presentation - Development Environment for Future Leon Multi-core

Cobham Gaisler, Sweden
OAR Corporation, USA
Airbus Defence & Space SAS, France

ESA Technical Officer: Marco Zulianello
Presenters: Daniel Hellström (Cobham Gaisler), Fabrice Cros (Airbus D&S)

Introduction

- Activity Overview and approach
- RTEMS SMP improvements
- Parallelisation Library - MTAPI
- Demonstrator - GAIA VPU SW
- Conclusion

Overview

Activity background

- Previous work/studies like SIDMS identifies issues adopting multi-core in flight SW. We need:
  - better multi-core OS support for flight software
  - better multi-core analysis tools
  - libraries to help exploiting multi-core hardware
- Space qualified dual-core GR712RC already present and ESA drives GR740 quad-core LEON4 for 2015
- Translates into study missions:
  - RTEMS SMP OS – RTEMS is a commonly used OS in flight SW
  - Existing GR712RC and LEON4-N2X (NGMP prototype)
  - Parallelisation library/technique
  - Demonstrate technology with existing flight SW

Overview

ESA funded activity – 18 months

ESA Activity:

“Development Environment for Future Leon Multi-core”

Milstone	Date
KO	Aug 2013
PDR	Dec 2013
CDR	Sept 2014
FR	March 2015

Overview

Work separation

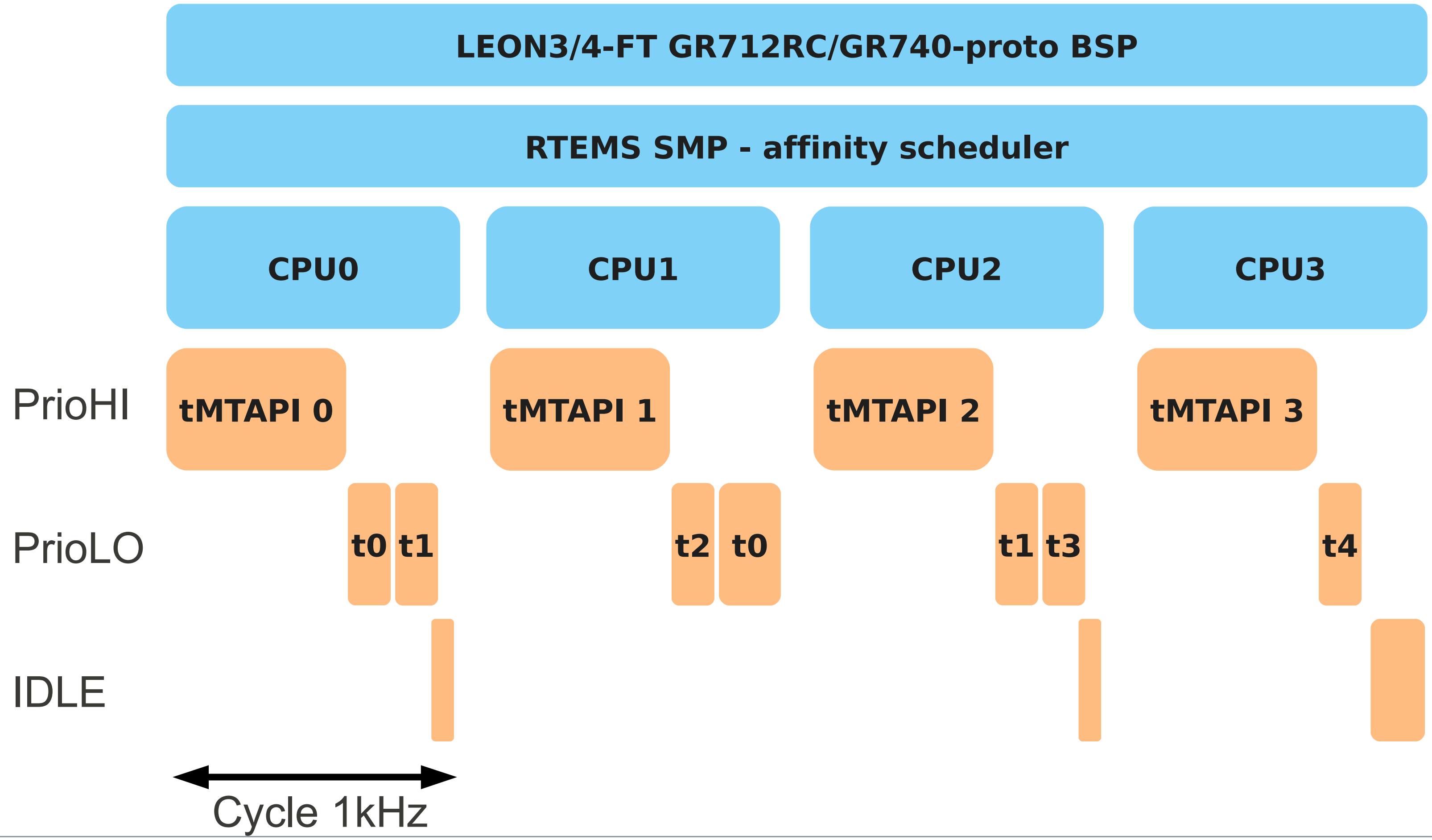
Development work split up in three major tasks:

	OAR Corporation	Cobham Gaisler	Airbus Defence and Space
TASK1	RTEMS SMP Development	RTEMS SMP LEON/SPARC Development	
TASK2		MP Library Development	MP Library Analysis
TASK3			Demonstrator

Overview

Proposed approach - Parallelisation technique (1)

Work split up in RTEMS Tasks:



Overview

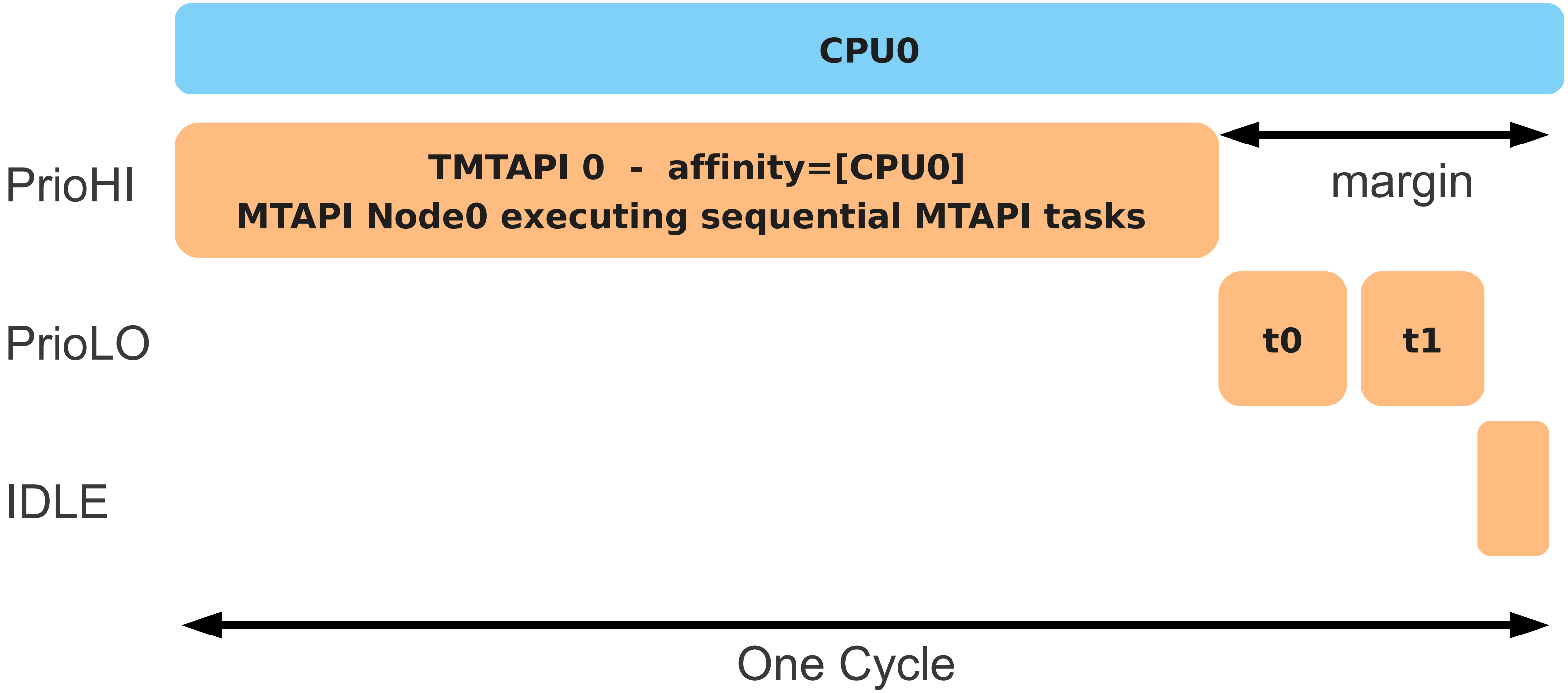
Proposed approach - Parallelisation technique (2)

Time split in cycles executing every 1kHz.

MTAPI node executing parallelised Demonstrator

Scheduler CPU Affinity bounds MTAPI Node[N] to CPU[N]

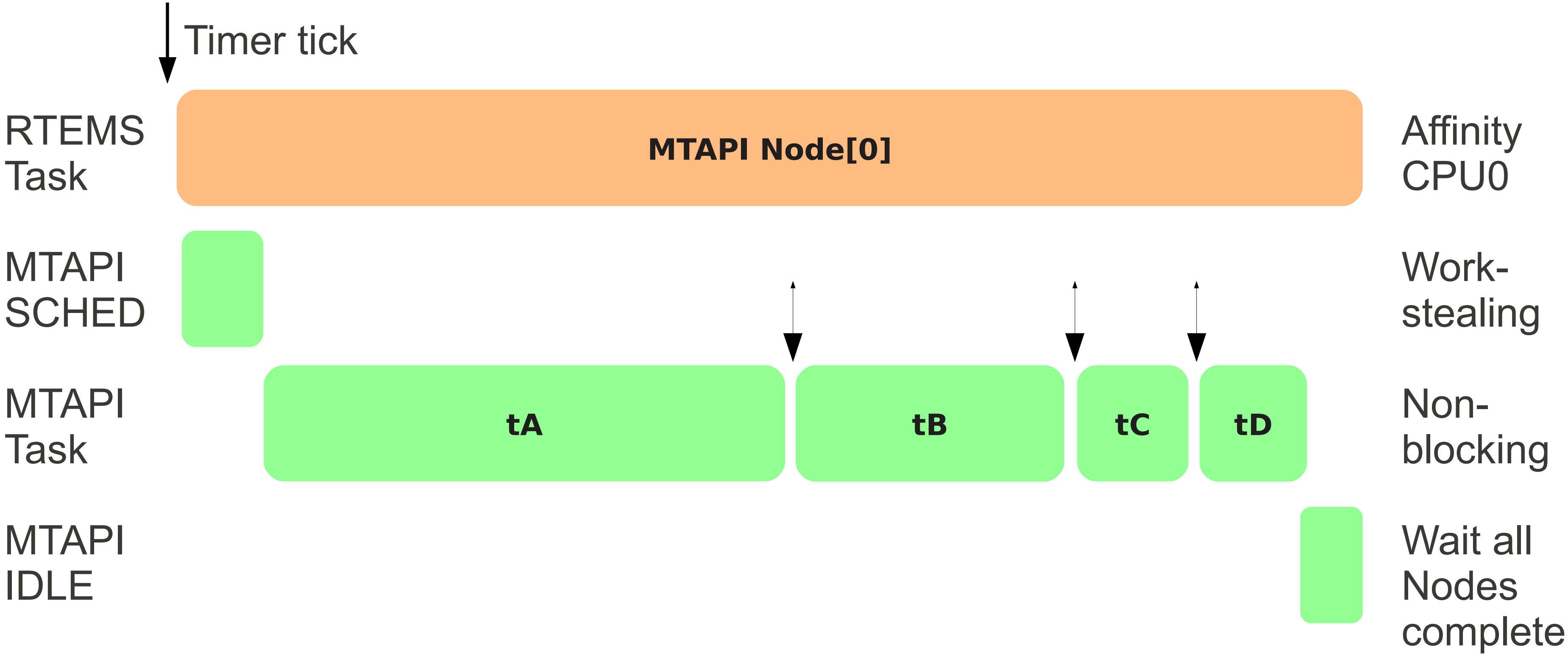
Avoid Context switches. Isolation from other cores



Overview

Proposed approach - Parallelisation technique (3)

- MTAPI node 0 runs scheduler which controls start & end of cycle, spawns work every @ 1KHz
- MTAPI task a non-blocking piece of code
- Work stealing protected by atomic locks



TASK1 - RTEMS SMP

Primary goals

- Early analysis: Test, Analyze, Port, Fix existing SMP
- Processor Set API
- Extend SMP Scheduler with Affinity
  - Pluggable Scheduler
  - New Affinity User APIs (RTEMS, POSIX)
  - Scheduler itself
- Scheduler on/off-line mode
- Atomic Layer
- Trace Library - RTEMS Capture Engine
- BSP - GR712RC and LEON4-N2X (GR740 prototype)
- Build warnings
- Test & document & submit to RTEMS Community

TASK1 - RTEMS SMP

Early analysis results

- I-Cache inconsistency
- Interrupt support broken in BSP
- BSP/RTEMS start-up/shut-down code and sequence issues
- Scheduler off-line/on-line mode not very useful on LEON, due to idle-mode impl. and added Complexity
- Interrupt CPU Affinity needed
- RTEMS Scheduler Simulator analysed
- Atomic layer had evolved since proposal
- GCC-4.8.3/master required for C11 atomics
- Existing test checks fails undetected
- SpaceBel & EB consortium coordination

==> BSP & OS fixes, now able to boot using old toolchain

TASK1 - RTEMS SMP

Primary goals – updated

+ Cache manager

– Scheduler off-line/on-line mode not very useful on LEON, due to idle-mode impl. and added Complexity

+ Static Interrupt CPU Affinity needed

+ New toolchain based on GCC-4.9.x supporting C11 and atomic instructions, require testing & review.

– Replaced atomic layer support work planned.

+ RTEMS Scheduler Simulator used for testing

TASK1 - RTEMS SMP

Processor Set API

- Defined and implemented new Processor Set API
- Operations, for example:
  - Bit-mask – AND(set0, set1)
  - Counting - Count(set0)
  - Boolean tests - is CPU[n] set, is empty, etc.
- Integrated into newlib and RTEMS

TASK1 - RTEMS SMP

SMP scheduler affinity support (1)

- Define/develop new task affinity User APIs

  POSIX Pthread:

```
int pthread_getaffinity_np(const pthread_t id, size_t cpusetsize, cpu_set_t *cpuset)
int pthread_setaffinity_np(pthread_t id, size_t cpusetsize, const cpu_set_t *cpuset)
int pthread_attr_getaffinity_np(const pthread_attr_t *attr, size_t cpusetsize, cpu_set_t *cpuset)
int pthread_attr_setaffinity_np(pthread_attr_t *attr, size_t cpusetsize, const cpu_set_t *cpuset)
```

  RTEMS Classic:

```
rtems_status_code rtems_task_get_affinity(rtems_id id, size_t cpusetsize, cpu_set_t *cpuset);
rtems_status_code rtems_task_set_affinity(rtems_id id, size_t cpusetsize, cpu_set_t *cpuset);
```

- Pluggable Scheduler API
- SMP Deterministic Priority Scheduler extended
- Implementation constraints:
  - Support affinity without require it in other schedulers
  - Affinity possible in other schedulers too
  - Using the same API

TASK1 - RTEMS SMP

SMP scheduler affinity support (2)

- Testing performed using
  - RTEMS Scheduler Simulator – Algorithm testing
    - Fix and extend sched-sim with respect to multi-core
    - Main test development performed here
    - Affinity algorithm tested using standard use cases and corner cases described by “test scenarios”
  - RTEMS test-suite extended
    - User API testing (affinity storage, error paths etc)
    - Simple tests
- MTAPI system level test
- Demonstrator

TASK1 - RTEMS SMP

LEON3/4 BSP

- UART driver
- Interrupt controller
  - Macros for CPU operations
  - Extended (16..31) Interrupt support
- System Clock driver fixes
- Start-up, shut-down, error handling, etc
- GR712RC errata: sleep-mode
- Static Interrupt CPU affinity support
  - BSP weak table to define CPU-to-IRQ map, defaults all to CPU0.
  - RTEMS Interrupt handling layer is unaware

TASK1 - RTEMS SMP

Affinity - results

- CPU Affinity use cases:
    - Load balancing
    - Performance - increase cache hit rate
    - Reduce application locking
    - Reuse single-core code and drivers by setting task to execute on same CPU as Interrupt is handled
    - Used by parallelization libraries
    - ...
- Demonstrated NGMP GBit Ethernet driver reuse. Low-priority background NFS traffic in SMP configuration.

TASK1 - RTEMS SMP

Capture engine

- Tracing kernel/application SMP support
- One Capture Engine instance per core
- Improve SMP locking, avoid spin-locks
- Remove dynamic mem allocation in trace path
- Variable records, a new storage class
- Wrapper responsible to call Capture Eng.
- Function tracing, not modifying compiled code
  - GNU LD -wrap
  - Future: auto-generate wrapper from DWARF information.
  - Tools for generating wrappers in *rtl-host.git*
- Reading out trace buffer
- Tests

TASK1 - RTEMS SMP

Capture engine wrapper example

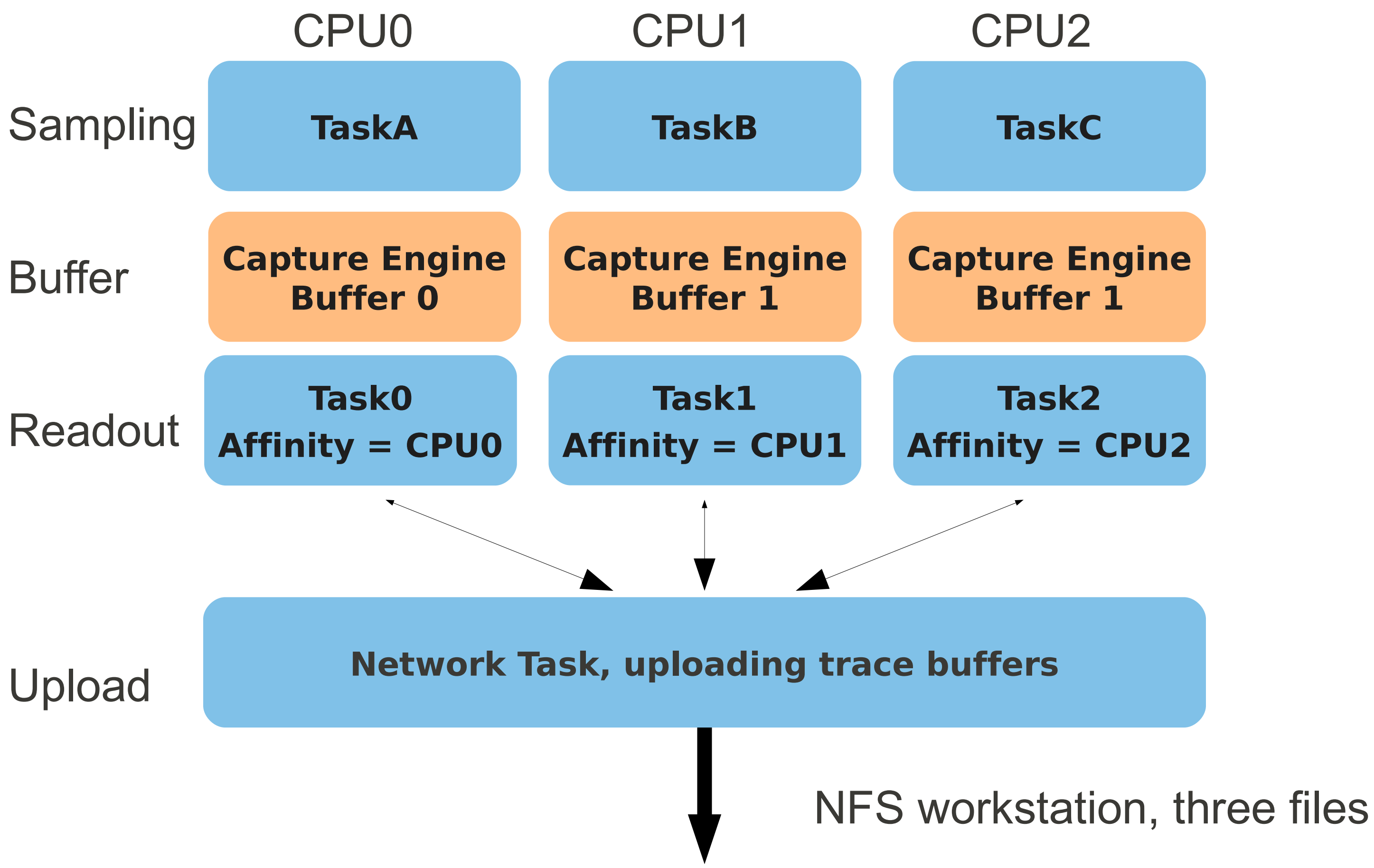
- LD –wrap=func
- No changes to compiled code, link-time only

```
func(argA, argB)
{
       SAMPLE ENTRY → Call capture engine (FUNC_ID, ENTRY, argA, argB)
       tmp = __wrap_func(argA, argB)
       SAMPLE RETURN → Call capture engine (FUNC_ID, RETURN, tmp)
       return tmp
}
```

TASK1 - RTEMS SMP

Capture engine buffer readout

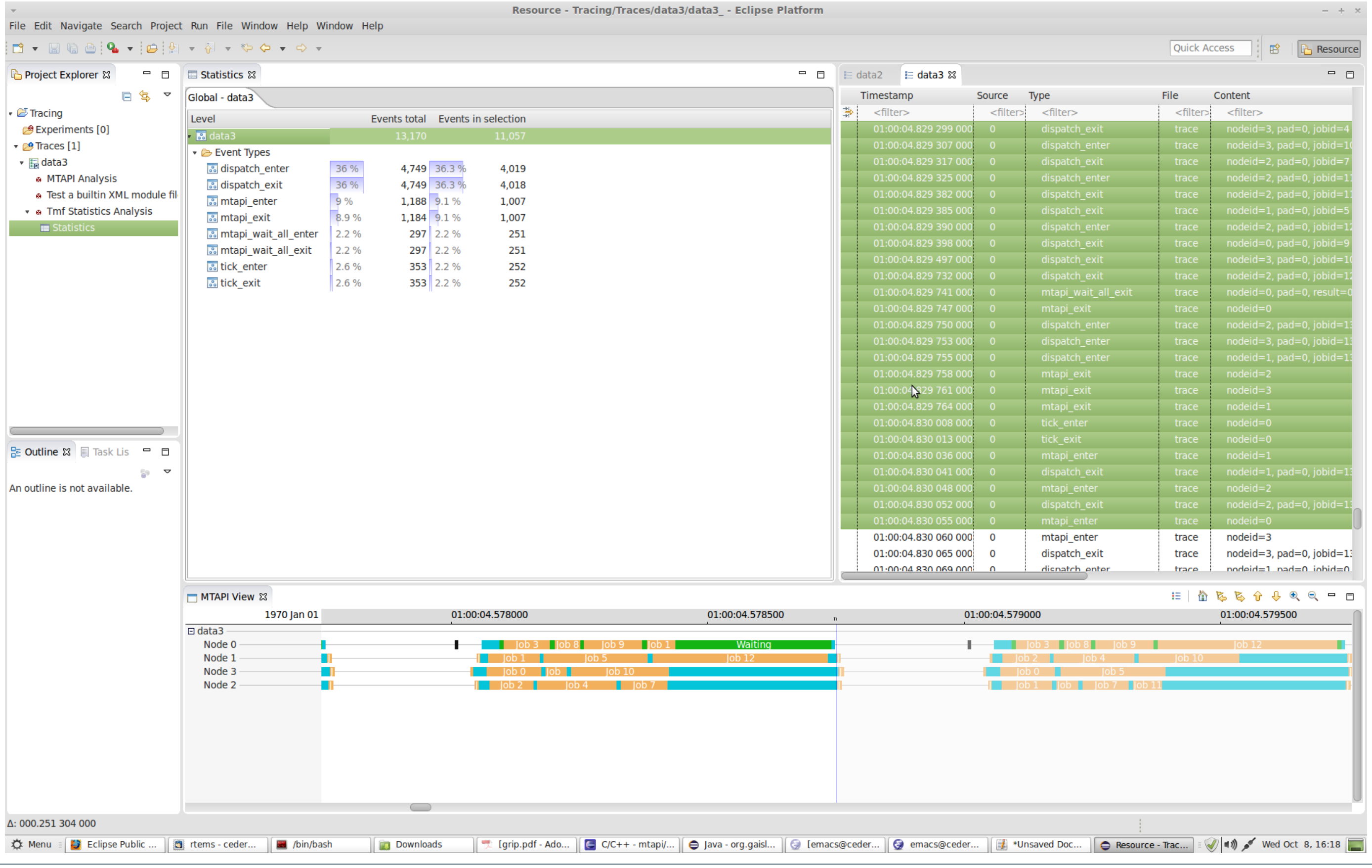
- Set up to avoid locking recording task/CPU
- Readout task always on same CPU as the capture buffer instance



TASK1 - RTEMS SMP

Capture engine visualisation example

- Proof of concept – capture engine on SMP
- Common Trace Format (CTF)
- Babeltrace or Eclipse LTTng



TASK1 - RTEMS SMP

Toolchain based on GCC-4.9

- GCC-4.9, Binutils 2.23, Newlib 2.10, GDB 7.7.1
- Code review of atomic support in GCC-4.9 LEON backend. RTEMS spin-locks use this too. Major problems found and fixed/reported:
  - GCC LEON3 write-buffer
  - RTEMS spin-lock implementation
- Automated Toolchain testing using GRMON2 Tcl:
  - GCC test-suite
  - RTEMS test-suite
  - Various Linux test-suites
- LEON Compare-And-Swap (CAS) Atomics support required by C11 (mcpu=leon3 enables it)
- muser-mode selects CAS privileged/user
- Added new GCC target mcpu=leon3v7
- DWARF-4 issues

TASK1 - RTEMS SMP

Environment - GRMON2

- GDB 7.7.1 support
- SMP model threading support
- RTEMS 4.11 and RTEMS 4.11 SMP threading support
- Exit code handling improved for automated RTEMS/GCC test-suites script set up
- Added bare metal threads (-bmthreads) support

TASK2 – MP Library

Overview – main objectives

- Survey to summarise existing solutions, analysis, trade-off, selection of parallelisation technique/library to assist Demonstrator parallelisation.
- Implement and/or port selected technique/library
- Implement a test-suite

TASK2 – MP Library

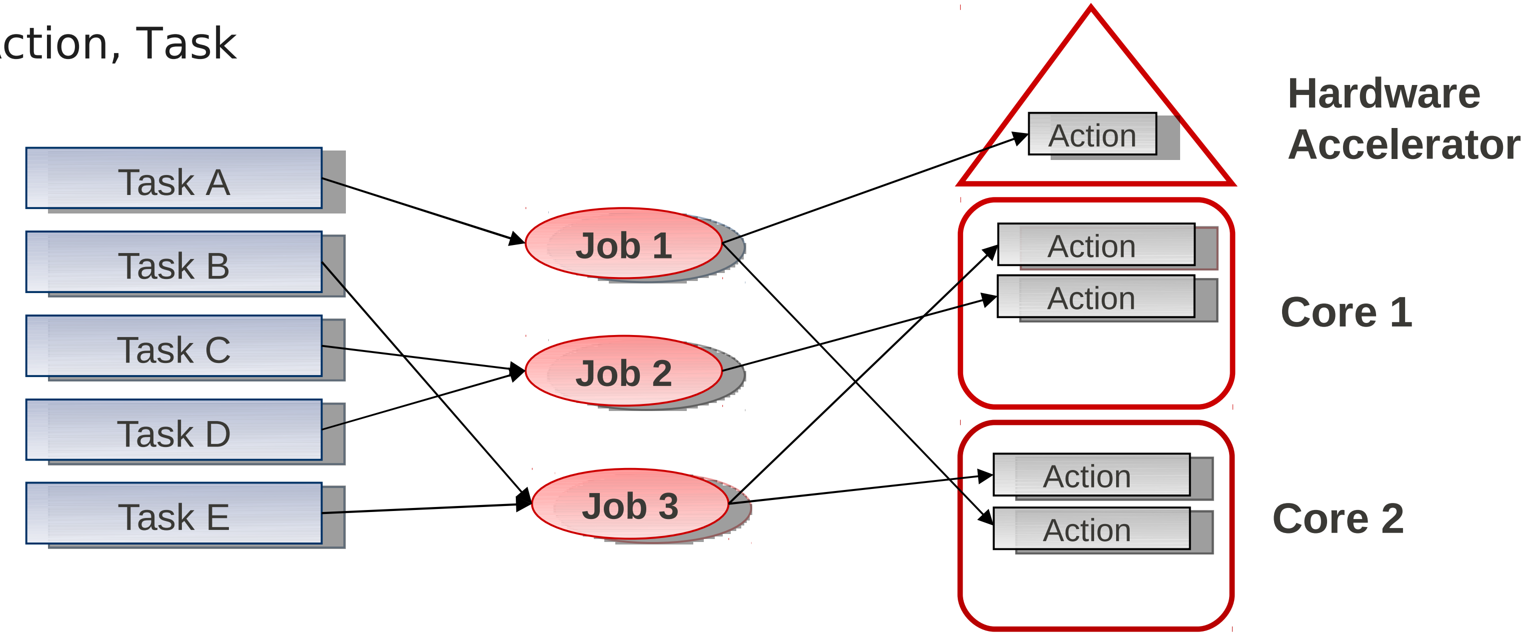
Survey of existing parallelisation techniques

- Focus on payload applications and multi-core issues previously identified
- Comparision criterias used to score:
  - the overall parallelisation scheme
  - task synchronization method(s)
  - task dispatching techniques supported
  - support for hardware accelerators and multi-core systems
  - compiler dependability and modifications required
  - execution determinism
  - software qualification
  - adaptation complexity
- MP parallellization libraries analysed:
  - OpenMP
  - Intel Cilk plus
  - Multicore Association (MCA) APIs
  - POSIX Threads

TASK2 – MP Library

MTAPI

- MTAPI was selected – got overall good score
- Defined by Multicore Association (MCA). The specification: http://www.multicore-association.org/workgroup/mtapi.php
- Task Management API for embedded systems
- Concepts:
  - Execution sequence control, groups, waiting
  - Domain, node
  - Job, Action, Task



TASK2 – MP Library

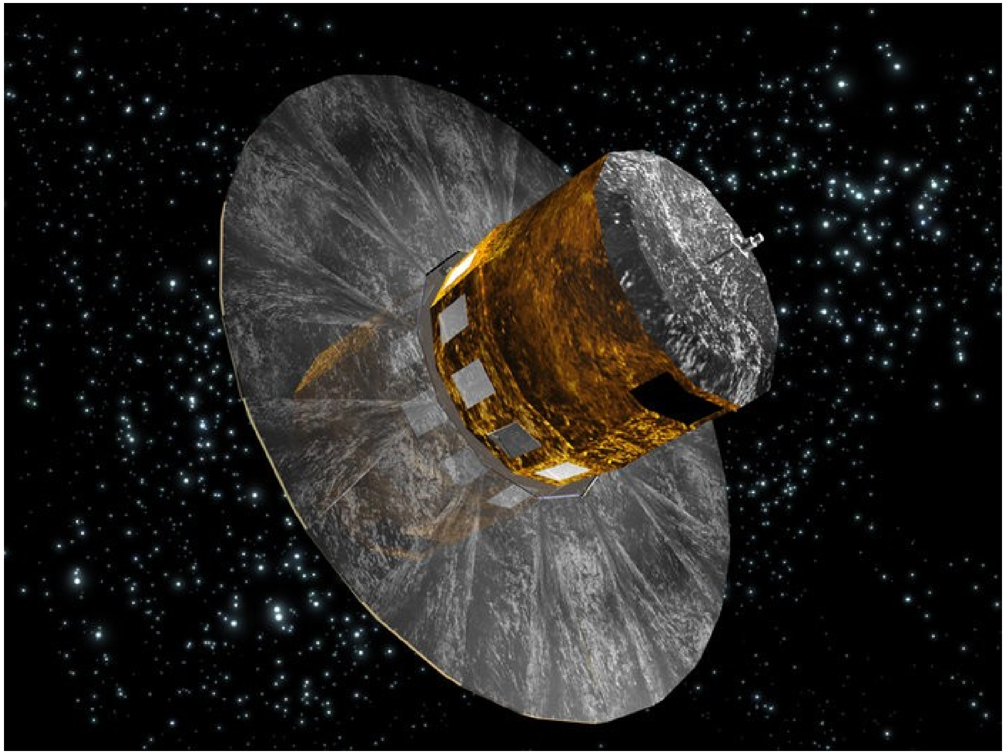
MTAPI implementation

- MTAPI for LEON implementation from scratch
- Concepts:
  - Domain = Multi-core Processor
  - Node = RTEMS task with CPU affinity
- Implemented in C
- No dynamic allocation, small footprint, lock-free design
- Communication between nodes: shared memory and atomic operation protection
- Test-suite with 92.5% code coverage
- ~300K start & joins per second on a 150MHz LEON4-N2X

TASK3 – Demonstrator

GAIA Video Processing Unit (VPU) application software from Airbus Defence & Space:

- Single-core PPC (Maxwell SCS750 - ITAR)
- VxWorks OS, ~32K lines of code



Conclusions

- RTEMS SMP and GR712RC/LEON4-N2X BSPs functional, new fundamental SMP features present and tested.
- Basic development environment support in place:
  - GCC-4.9.x toolchain, GDB, GRMON2. RTEMS SMP trace tool core available. Future improvements.
- All RTEMS SMP, Newlib, GCC code developed within project has been submitted upstreams to respective project. Improved LEON support mainline RTEMS and community interest for LEON & SMP.
- MTAPI helps in adopting multi-core. Plan to be publicly available in 2015. MTAPI reference:
  - http://www.multicore-association.org/workgroup/mtapi.php
- The demonstrator shows that a complex space payload application can operate in RTEMS SMP LEON multi-core environment and parallelisation simplified using MTAPI.
- Results shows significant performance gain and power efficiency going to multi-core LEON.
- Project successfully completed within time frame

Conclusions

Thank you for listening!

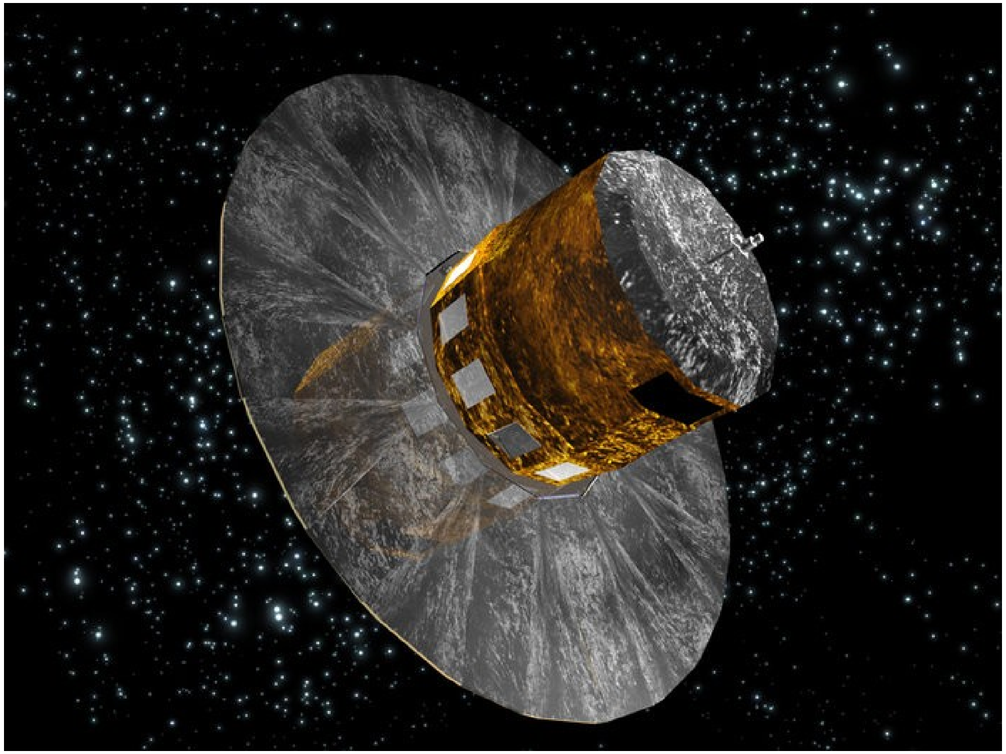
Conclusions

Demostrator slides not part of FP

TASK3 – Demonstrator

Overview

- GAIA Video Processing Unit (VPU) application software from Airbus Defence & Space:
  - Single-core PPC (Maxwell SCS750 - ITAR)
  - VxWorks OS
  - ~32K lines of code



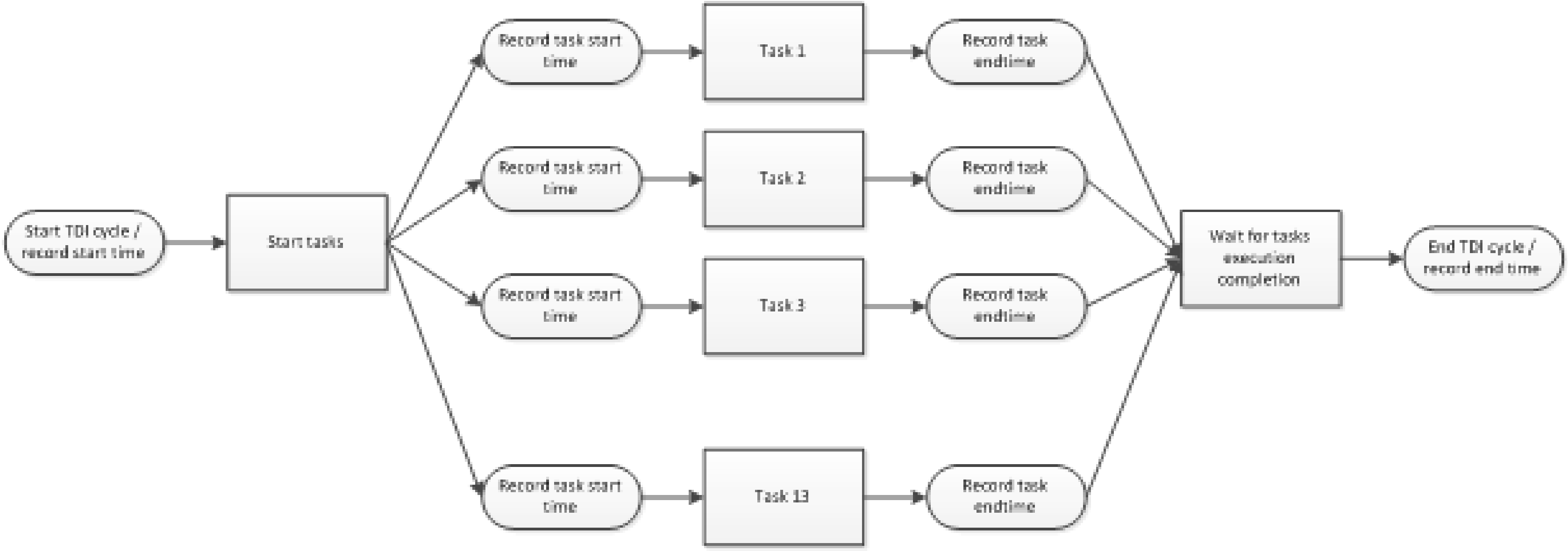
TASK3 – Demonstrator

Work performed

- Ported to SPARC/LEON single and multi-core
- Ported to RTEMS-4.10.2 and RTEMS-4.11 SMP
- Parallellised using MTAPI
- Performance was studied, SW restructured to improved task scheduling, compared against reference and different configurations
- Correctness verified against reference verification environment

TASK3 – Demonstrator

Performance Measurement

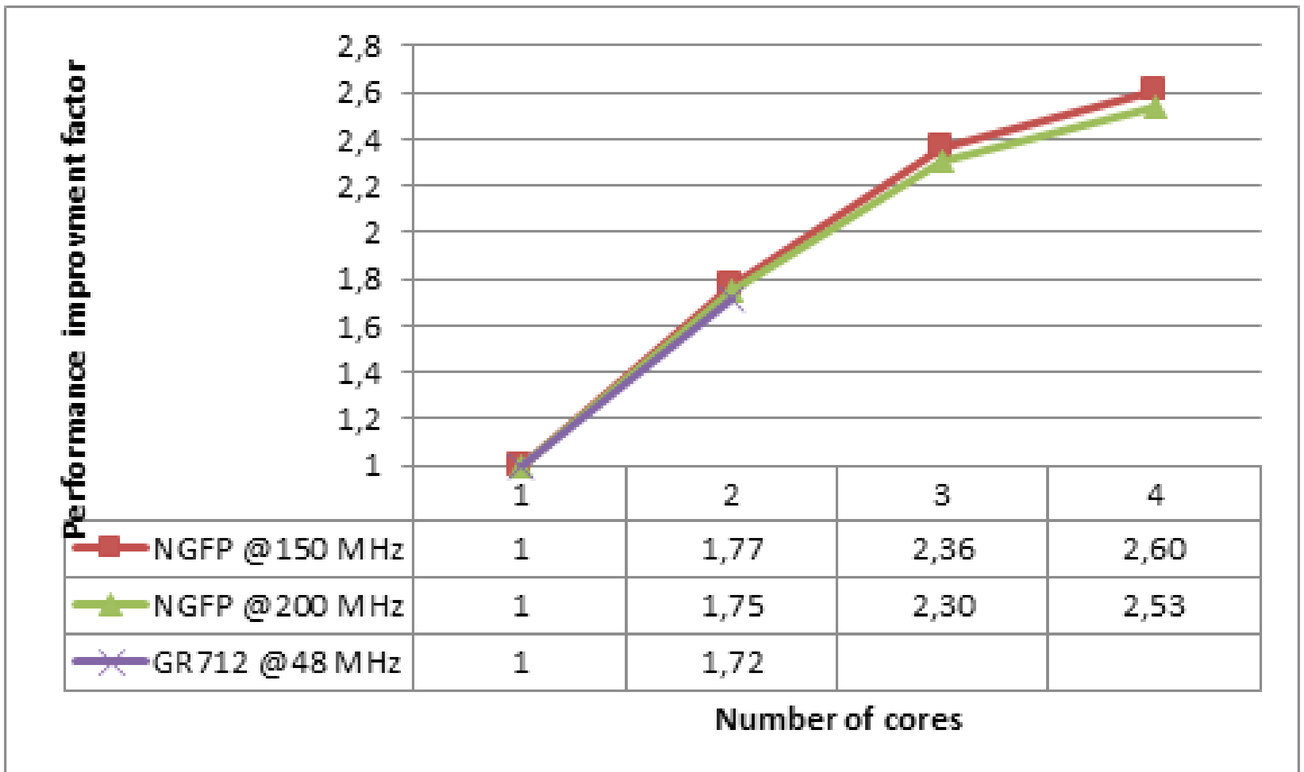


- CCD array picture input at 1KHz. VP algorithm executes 16 tasks @ 1KHz
- Measure execution time of one cycle
- By measuring start/end MTAPI task execution a Figure Of Merit (FOM) was established.
- FOM tells us how many tasks are active at a time, thus a measure on how well the application has been parallellized

TASK3 – Demonstrator

Performance Measurement

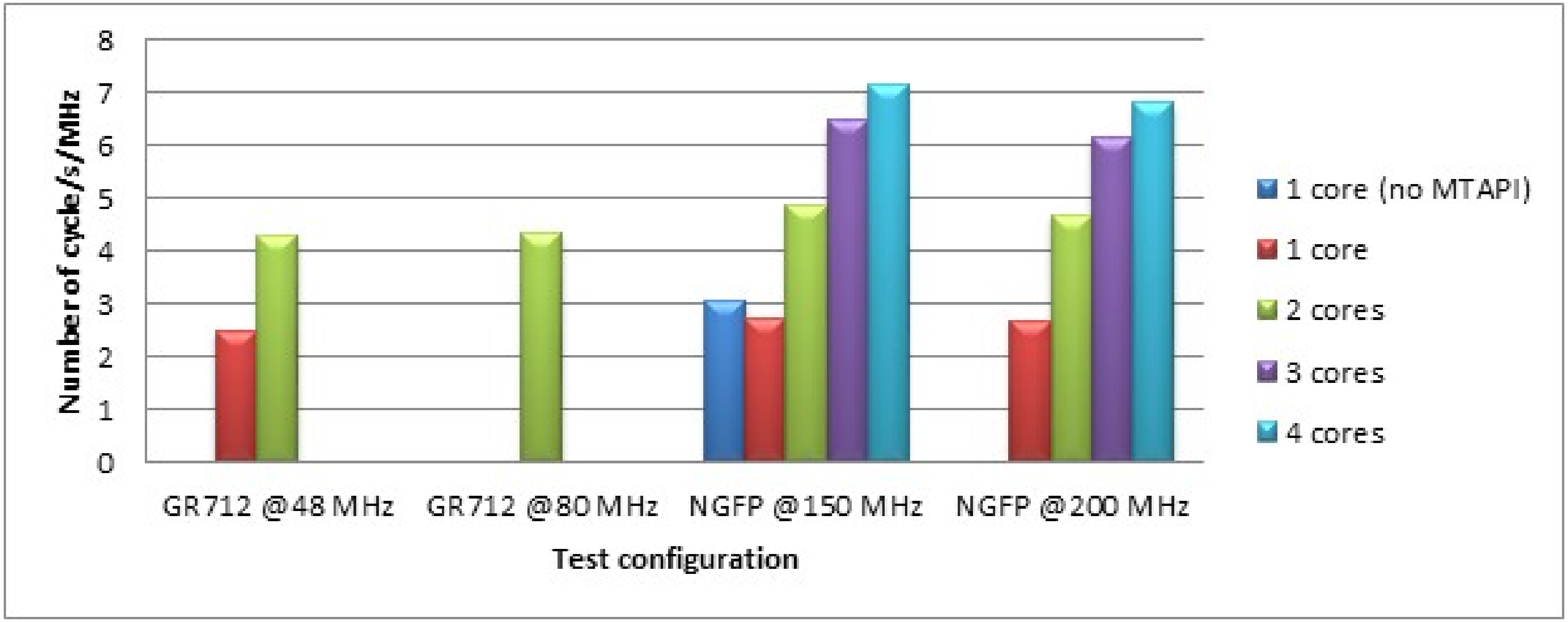
- Scaled well up to three cores (2.42x), minor benefit using four (2.60x). However FOM and overhead tells us it is not a HW issue, CPUs are idle at times.
- L2 Cache hit rate ~96.5% (from L4STAT)



TASK2 – MP Library

MTAPI implementation

- RTEMS SMP+MTAPI overhead measured to about 12.5%
- GR712RC perf scaled well since memory frequency scaled with system frequency (to be expected). NGMP 150MHz best per MHz.
- LEON4-N2X Power efficiency estimated 1/3 of reference



Conclusions

- RTEMS SMP is available in Git master and as a preview at

  http://www.gaisler.com/anonftp/rcc/smp/

- Environment has the standard development support required
- MTAPI helps in adopting multi-core. Publicly available in 2015. MTAPI reference:

  http://www.multicore-association.org/workgroup/mtapi.php

- The demonstrator shows that a complex space payload application can operate in RTEMS SMP LEON multi-core environment.
- Results shows significant performance gain and power efficiency going to multi-core LEON.

Conclusions

Thanks for listening!