High Performance Parallel Payload Processing for Space (HP4S)

OBDP 2021

DEFENCE AND SPACE

June 2021

AIRBUS

Agenda

I. Project Overview

II. Preparatory Phase Outcomes

III. Implementation Phase Outcomes

IV. Evaluation Phase Outcomes

V. Conclusion & Future Work

PROJECT OVERVIEW

High Performance Payload Processing Requirements

❑ **High demand for flexibility in High Performance Payload Processing**

- ASIC < FPGA < Processors (General Purpose Processors)
- Many future processing devices are multi or many cores

⇒ **parallel processing on-board on multi/many cores** for image, new sensors, radar, telecom, robotic, Vision Based Navigation use cases

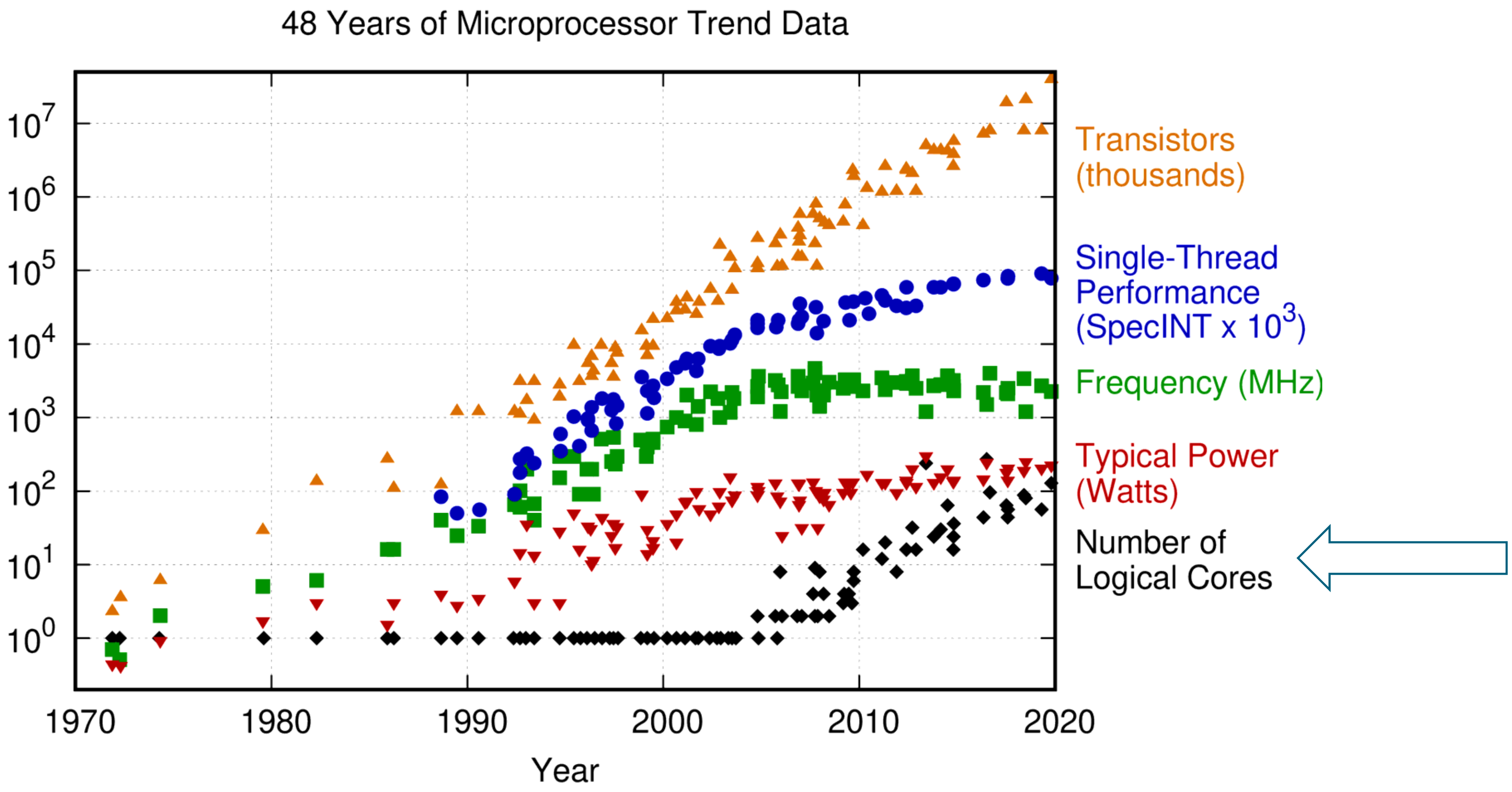
❑ **Airbus DS is working on future high performance space computer (multi-core), both radiation hard and radiation tolerant (New Space approach)**

- Software parallelization needs ***methodologic*** and ***tooling*** support for more efficiency and this can get some momentum from the innovation and academic works in this area

❑ **Need for the parallelization to be rather independent w.r.t. the specific characteristic of the target device**

- **Portability**: The application may be developed without a deep knowledge of the platform. It is the runtime which is adapting the parallelization to the target device.
- Can be used both on Rad-Hard and Rad-Soft (COTS) devices

General Purpose Processors Trend



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp

Why OpenMP ?

- **A parallelism-centric programming model**
  - ***Hardware-centric models*** aim to replace the native platform programming with higher-level, user-friendly solutions, e.g. Intel TBB, NVIDIA CUDA.
    - These models focus on performance by tuning an application to match a chosen platform, which makes their use a neither scalable nor portable solution.
  - ***Application-centric models*** deal with the application parallelization from design to implementation, e.g. OpenCL.
    - Although portable, these models may require a full rewriting process to accomplish productivity, and so impacting on programmability.
  - ***Parallelism-centric models*** allow users to express typical parallelism constructs in a simple and effective way, and at various levels of abstraction, e.g. POSIX threads (Pthreads), MPI and OpenMP.
    - This approach allows flexibility and expressiveness, while decoupling design from implementation.

➢ **OpenMP provides a *good compromise between portability, performance* and programmability**

OpenMP Insights

OpenMP Enabling HPC since 1997

- **Mature language** constantly reviewed (last release Nov 2018, v5.0), since 1997 !
  - Defacto industrial standard in HPC shared memory (and heterogeneous) processor architectures
  - Active research community
- Productivity in parallel programming
  - **Performance**
    - Fine-grain data/task-parallelism and an advanced accelerator model for heterogeneous computing
  - **Portability**
    - Supported by many chip vendors (Intel, IBM, ARM, NVIDIA, TI, Gaisler, Kalray)
  - **Programmability**
    - Interoperability with other programming models (e.g., MPI, CUDA)
    - Currently available for C, C++, Fortran and Ada (under evaluation)
    - Allows incremental parallelization (#pragma omp) that can be easily compiled sequentially
- **Increasing interest** on the embedded computing domain

Main Objectives

❑ HP^{4}S study defined two strategic end goals:

- ***G1. Improve overall system performance***.
  Effectively master and exploit the most advanced parallel embedded architectures targeting the space domain.

- ***G2. Improve the parallel programming productivity*.**
  Reduce the development efforts of systems based on parallel architectures, while fulfilling system's functional and non-functional (time predictability) requirements.

HP4S Project organization

The study was structured around four technical steps:

1) A **Preparatory phase** dedicated to the inventory of possible application use cases that would make sense for being evaluated during this study as well as an in-depth analysis of the available many/multi-cores processing targets an associated resources w.r.t. their suitability for the use cases and for supporting an OpenMP framework. This phase results in the selection of
   - 2 representative use cases
   - 2 hardware targets (one rad-hard, one rad-tolerant COTS)

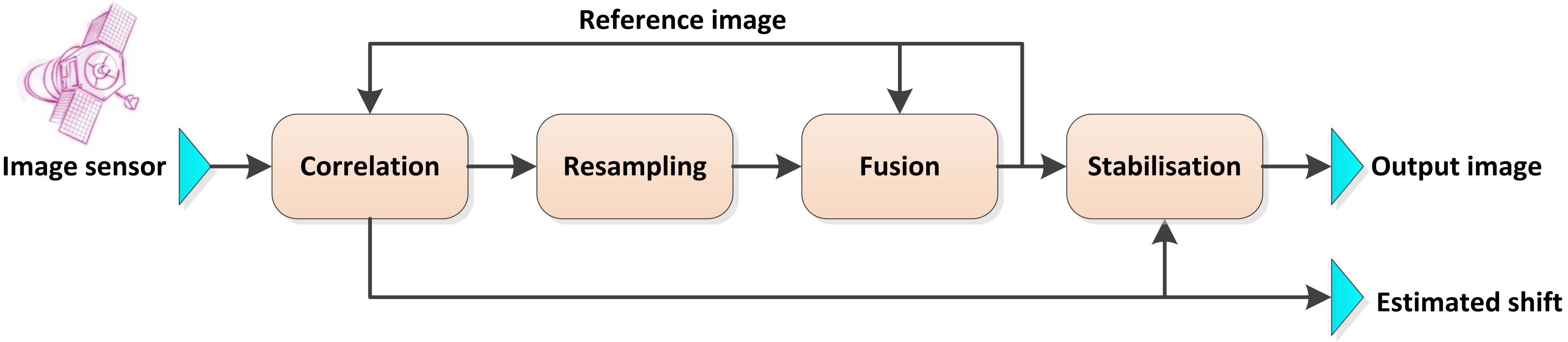
2) A **Development phase** dedicated to the preparation of the OpenMP framework on the selected targets and to the parallelisation work on the two selected use cases.

3) An **Evaluation phase** dedicated to the actual porting of the parallelised use cases and benchmarks on the targets and evaluation according to the criteria's defined during the preparatory phase.

4) A **Synthesis phase** dedicated to the analysis of all collected evaluation results and a synthesis for all the study outcomes as well as recommendations for the way forward with respect to parallel processing and multi/many cores targets.

PREPARATORY PHASE

Software Use Case N°1 - HRGEO

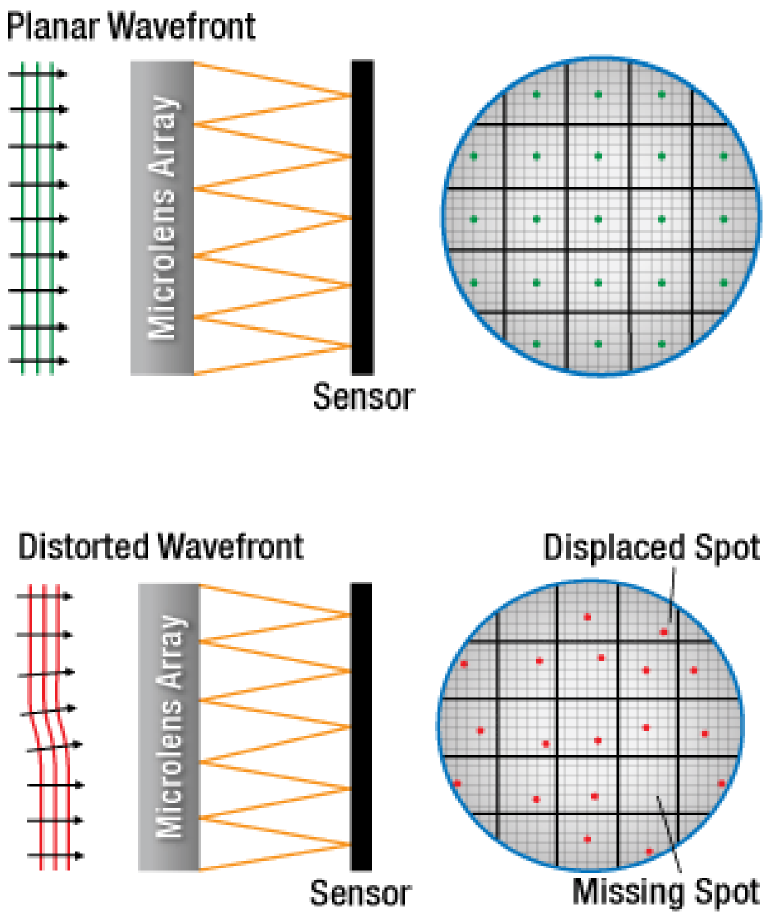


- Goal is to improve the S/N ratio on a static image by merging multiple images from a stream captured by a more or less steady camera (translation / rotation)

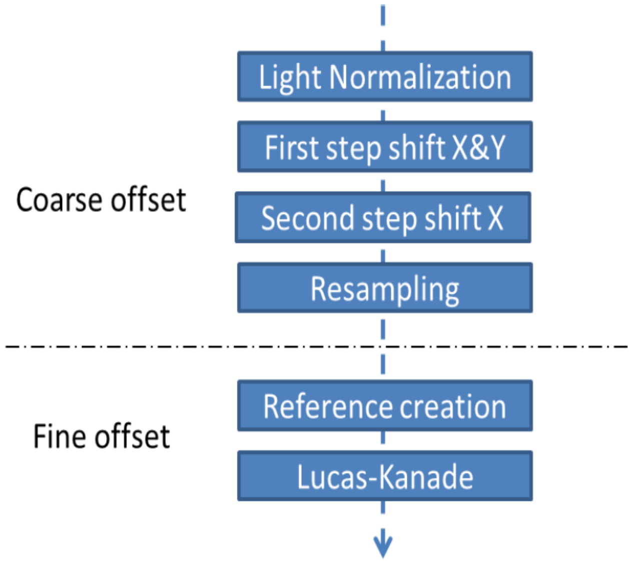
- The algorithm includes image ***registration***, ***resampling*** and ***fusion***

- The algorithm creates and manages a geometric model fitting image displacement, then accumulates successive images into a single reference image after compensating for the displacement with a sub-pixel accuracy

Software Use Case N°2 – Adaptive Mirror



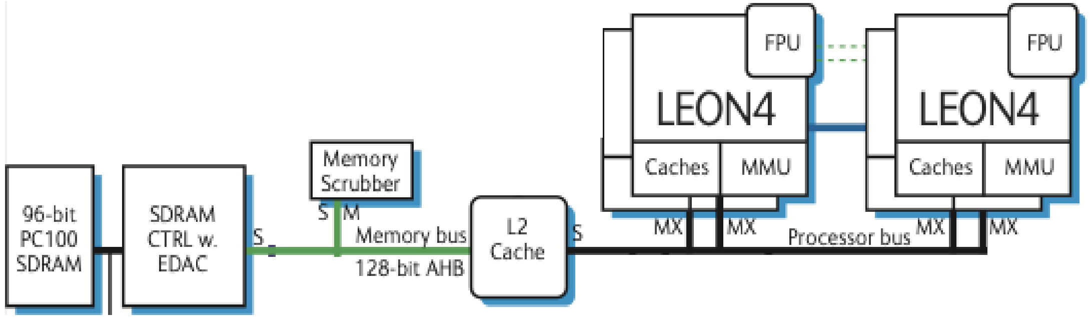
- Adaptive optics allows fixing thermoelastic deformations thanks to a myriad of piezo actuators. The wave front defaults are measured on board by a Shack-Hartmann analyzer with its image sensor
- By comparing the locations of the spots in the measured spot field with those in the reference spot field, the shape of the wavefront can be calculated, and this is the aim of the image processing algorithm of this use case



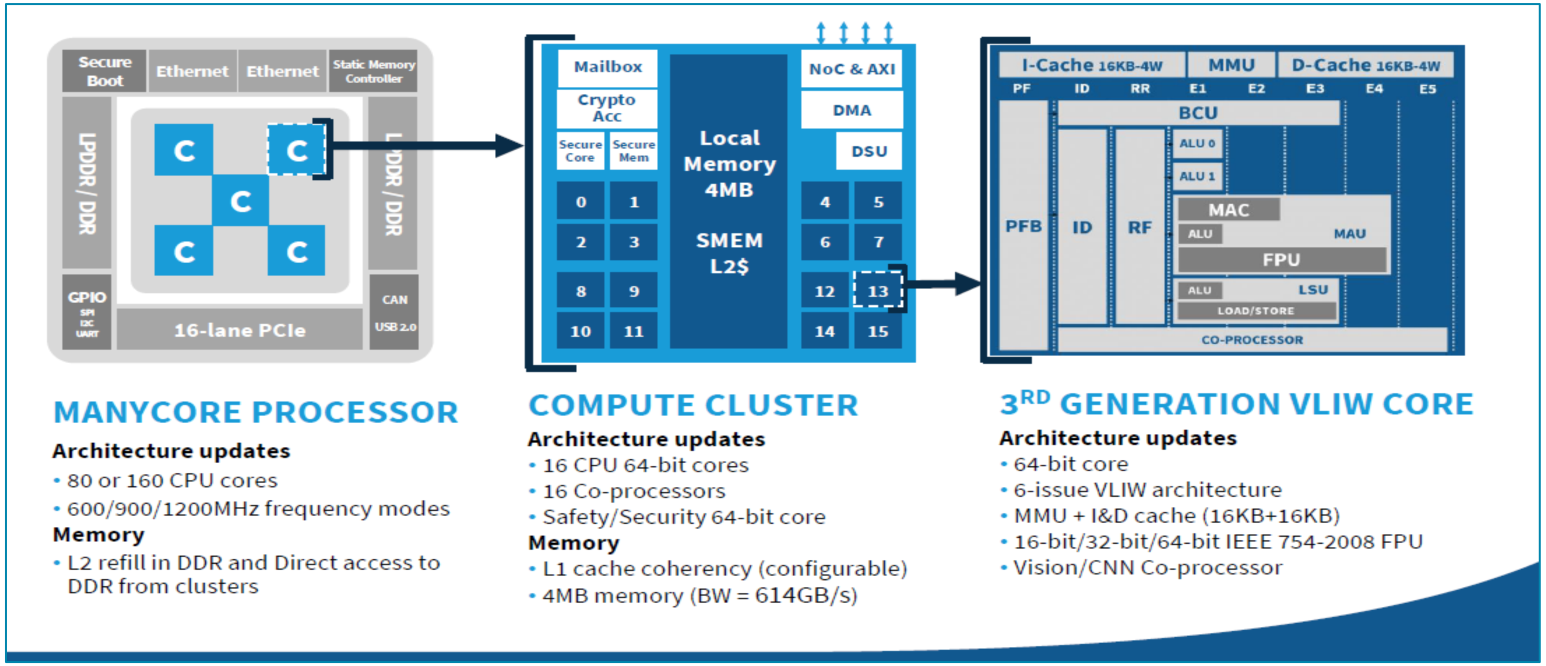
- Selected algorithm handles a 12x12 matrix of lenses for 3 detectors that are independent. Not all lenses must be processed, only the ones with a light intensity above a certain threshold. The light intensity is a constant, thus the number of lenses to be processed can be defined beforehand. 104 lenses have to be processed in our current implementation
- A *Coarse* phase computes a first rough estimate of the shifts and interpolates the image with this shift
- A final *Fine* phase uses the interpolated image to compute a very precise shift.

Hardware Targets

GR740



MPPA Coolidge



MANYCORE PROCESSOR

Architecture updates

- 80 or 160 CPU cores
- 600/900/1200MHz frequency modes

Memory

- L2 refill in DDR and Direct access to DDR from clusters

COMPUTE CLUSTER

Architecture updates

- 16 CPU 64-bit cores
- 16 Co-processors
- Safety/Security 64-bit core

Memory

- L1 cache coherency (configurable)
- 4MB memory (BW = 614GB/s)

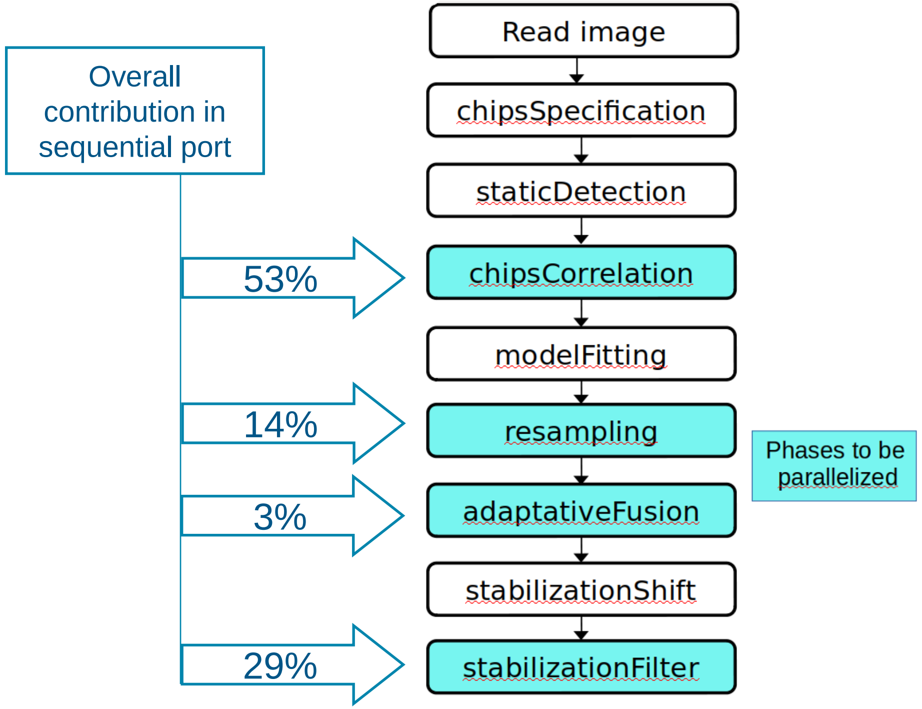
3RD GENERATION VLIW CORE

Architecture updates

- 64-bit core
- 6-issue VLIW architecture
- MMU + I&D cache (16KB+16KB)
- 16-bit/32-bit/64-bit IEEE 754-2008 FPU
- Vision/CNN Co-processor

IMPLEMENTATION PHASE

Use Cases Ported to Open MP
Identifying hotspots



Overall contribution in sequential port

Read image

chipsSpecification

staticDetection

53% → chipsCorrelation

modelFitting

14% → resampling

3% → adaptativeFusion

stabilizationShift

29% → stabilizationFilter

Phases to be parallelized

HRGEO

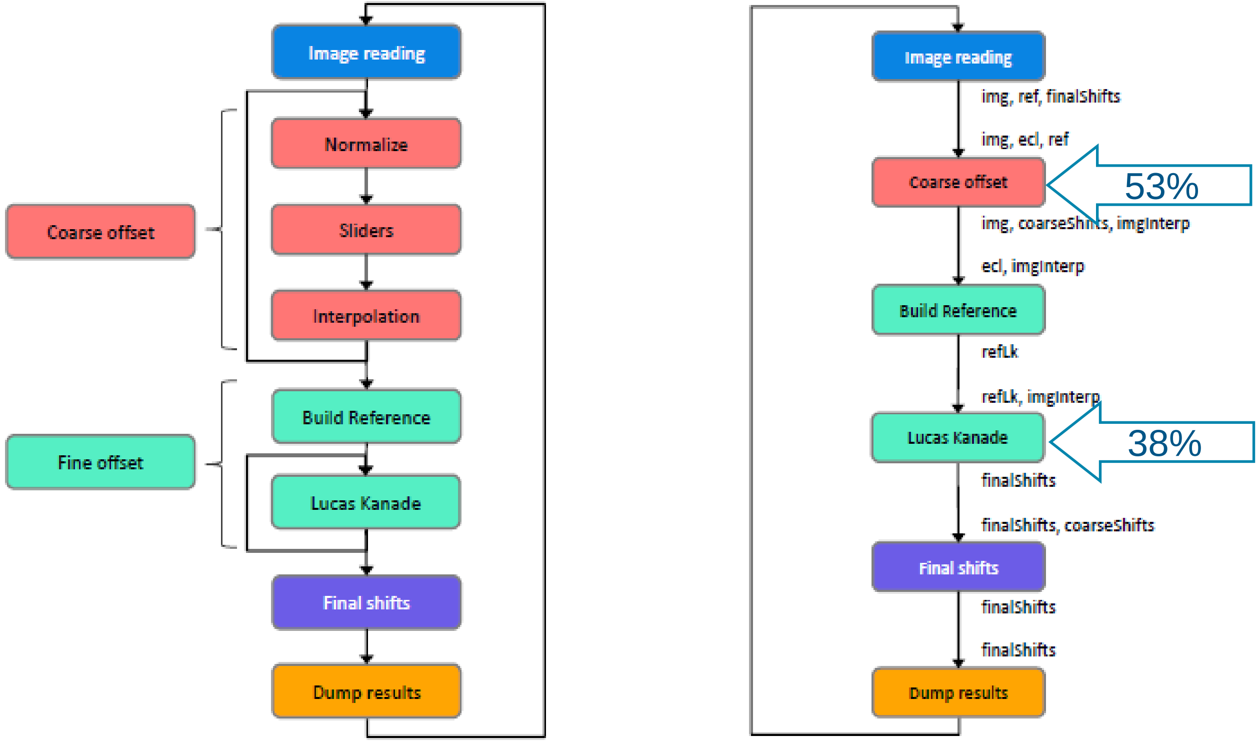


Image reading

Coarse offset: Normalize, Sliders, Interpolation

Fine offset: Build Reference, Lucas Kanade

Final shifts

Dump results

Image reading

img, ref, finalShifts

img, ecl, ref

Coarse offset ← 53%

img, coarseShifts, imgInterp

ecl, imgInterp

Build Reference

refLk

refLk, imgInterp

Lucas Kanade ← 38%

finalShifts

finalShifts, coarseShifts

Final shifts

finalShifts

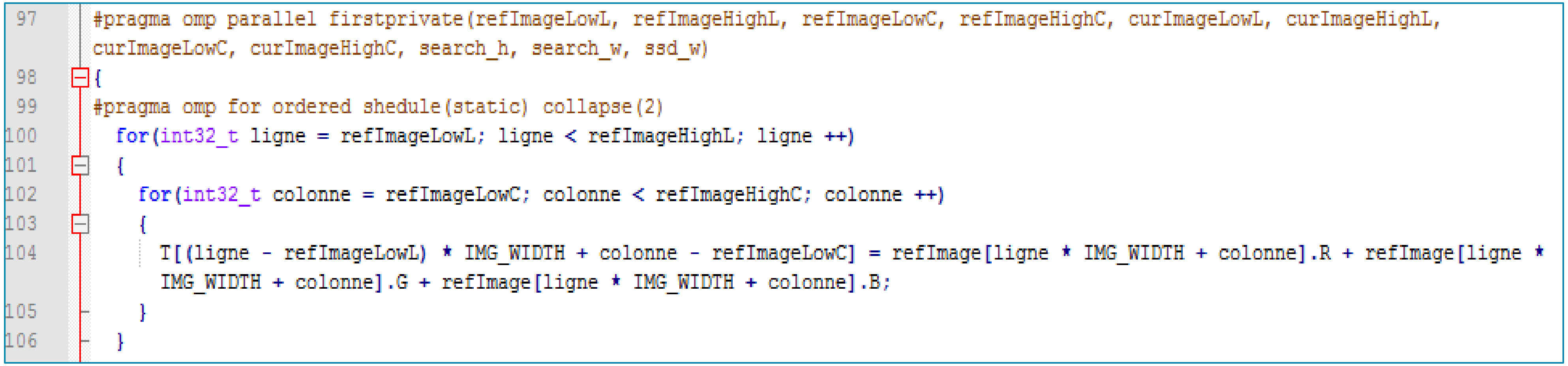
finalShifts

Dump results

MIRROR

Use Cases Ported to Open MP
Choosing an appropriate parallelization strategy for each hotspot

- OMP schedule is set to **`static`** for determinism and predictability across runs
- **`collapse`** clauses are used as fine grain optimization for load balancing
- OMP parallel loops are declared **`ordered`** so as to allow Extrae observability, and force instrumented runtime calls, with no overhead as the parallel sections do not contain any actual **`ordered statements`**
- Functional **legacy** C **code** remained **untouched**, and parallelization of actual algorithmic parts only consisted in **`#pragma omp`** annotations



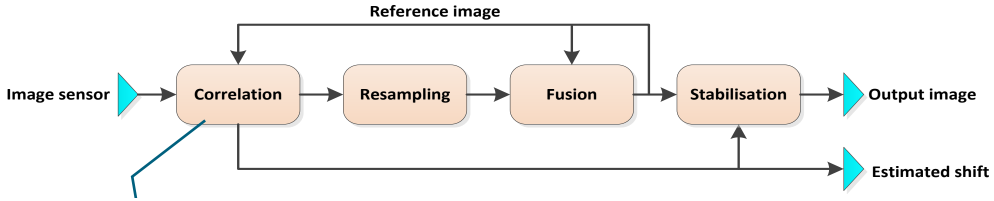
```
#pragma omp parallel firstprivate(refImageLowL, refImageHighL, refImageLowC, refImageHighC, curImageLowL, curImageHighL, curImageLowC, curImageHighC, search_h, search_w, ssd_w)
{
#pragma omp for ordered shedule(static) collapse(2)
  for(int32_t ligne = refImageLowL; ligne < refImageHighL; ligne ++)
  {
    for(int32_t colonne = refImageLowC; colonne < refImageHighC; colonne ++)
    {
      T[(ligne - refImageLowL) * IMG_WIDTH + colonne - refImageLowC] = refImage[ligne * IMG_WIDTH + colonne].R + refImage[ligne * IMG_WIDTH + colonne].G + refImage[ligne * IMG_WIDTH + colonne].B;
    }
  }
```

Extrae[1]: an HPC tracing tool

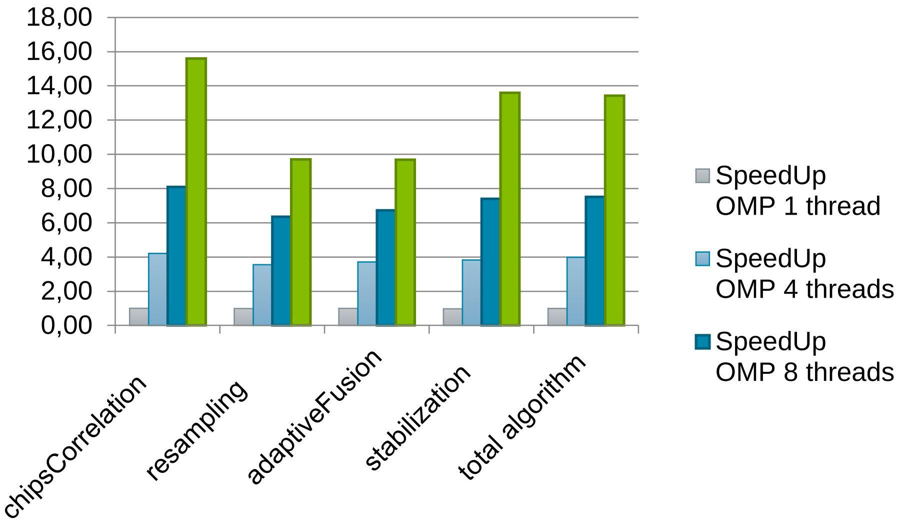
- A dynamic instrumentation package to trace parallel programs
- Capable of automatically capturing the activity of the parallel runtimes
  - No need to access the source code, recompiling, relinking, or having prior knowledge of application internals structure
- Allows reasoning about the execution behaviour of the parallel programing model
  - OpenMP support (other supported programming models are MPI, pthread, etc.)
- **… But not supported in the GR740 and MPPA…**
- **BSC has ported Extrae on these two targets**

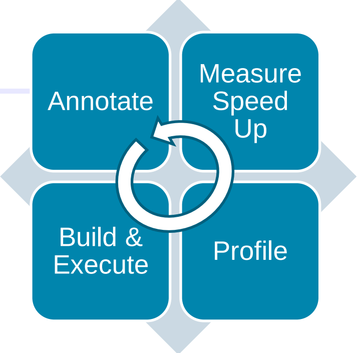
[1] ***Extrae*** means ***Extract*** in spanish

Process Overview



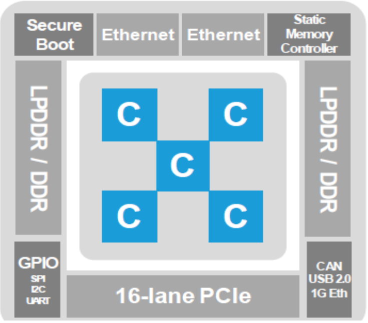
```
#pragma omp for collapse(2) schedule(guided)
  for(int32_t ligne = curImageLowL; ligne < curImageHighL; ligne ++)
  {
    for(int32_t colonne = curImageLowC; colonne < curImageHighC; colonne ++)
    {
    ...
    }
  }
```

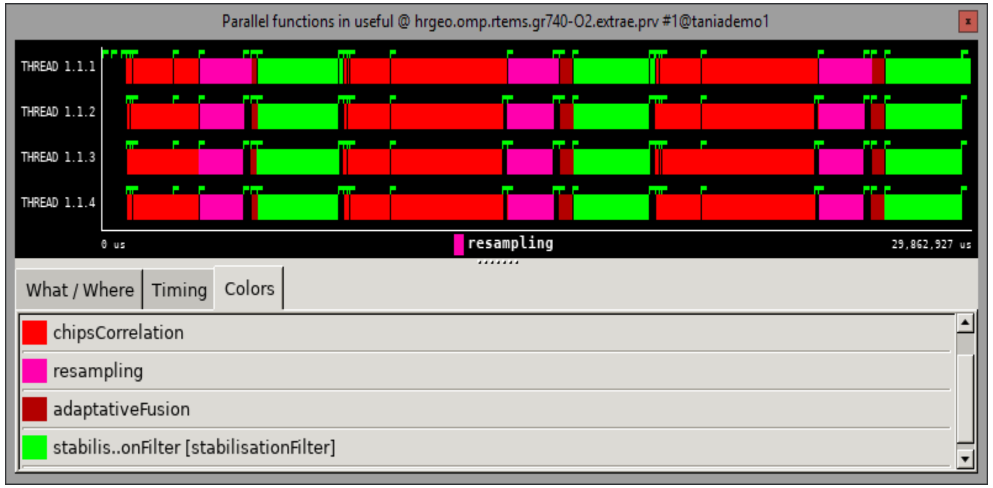




OpenMP
Enabling HPC since 1997

Embedded With RTEMS
www.rtems.org





Parallel functions in useful @ hrgeo.omp.rtems.gr740-O2.extrae.prv #1@taniademo1

THREAD 1.1.1
THREAD 1.1.2
THREAD 1.1.3
THREAD 1.1.4

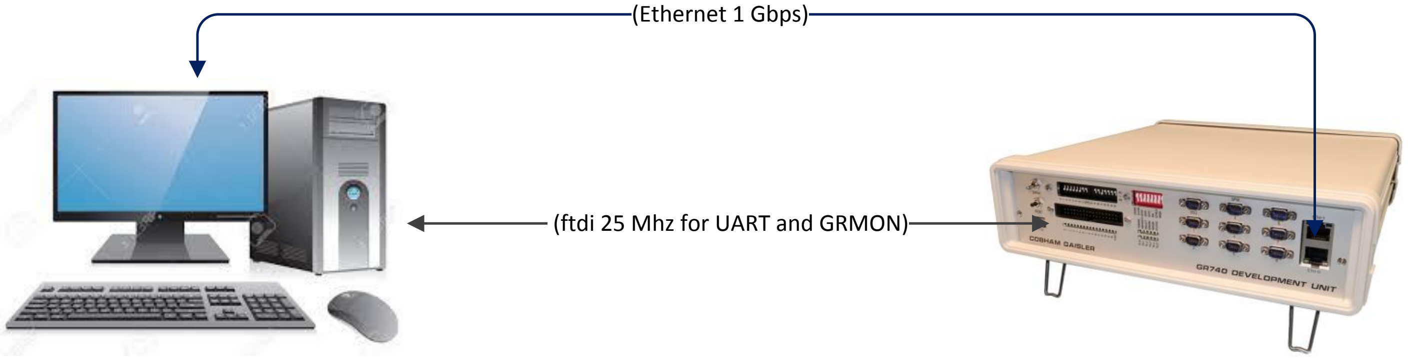
0 us resampling 29,862,927 us

What / Where | Timing | Colors

🟥 chipsCorrelation
🟪 resampling
🟥 adaptativeFusion
🟩 stabilis..onFilter [stabilisationFilter]

EVALUATION PHASE

GR740 Lab Setup

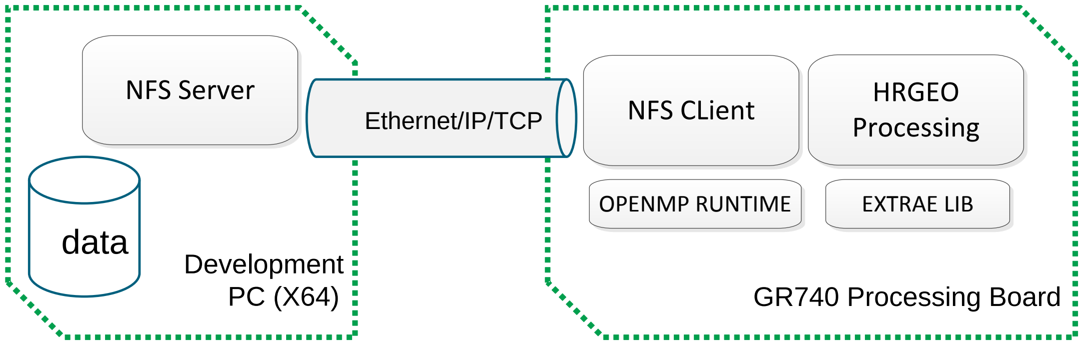


Linux Development Host

GR740

Cobham RCC 1.3-rc6
GRMON for upload/execution/serial
Linux NFS server for I/Os
Paraver traces visualization tool

RTEMS5 SMP OS
OpenMP runtime
Ethernet Driver
NFS Ethernet client for I/Os
Use Case application
Extrae 3.7.1 instrumentation lib for observability



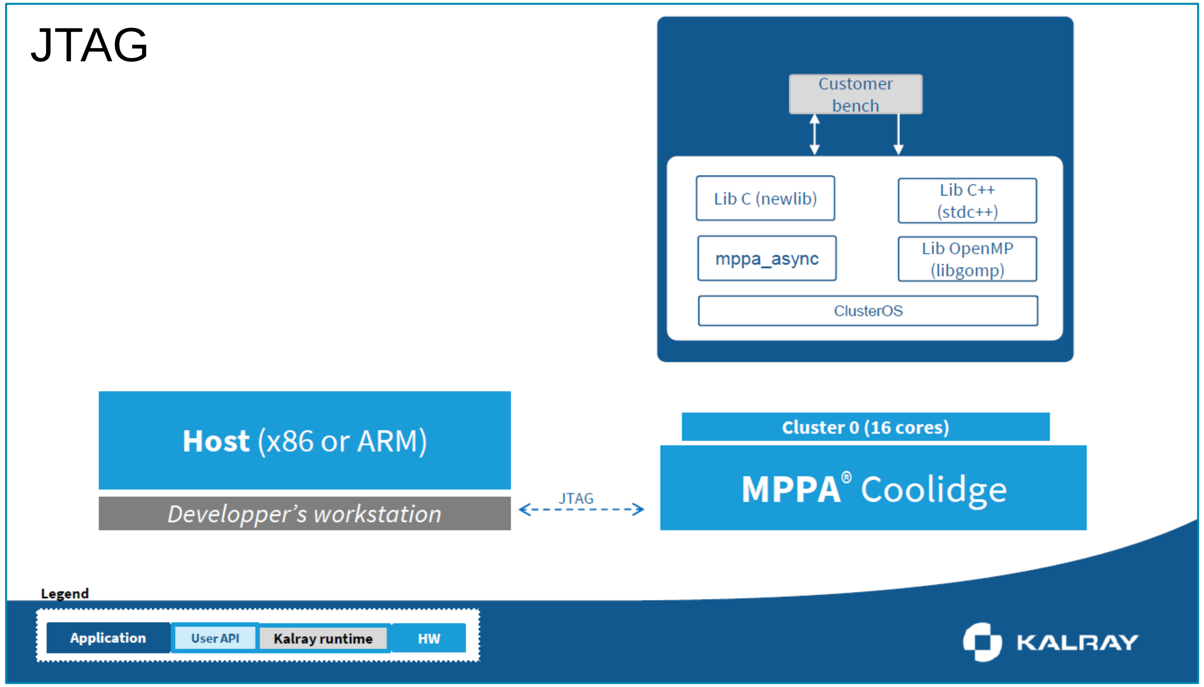
Kalray Lab Setup

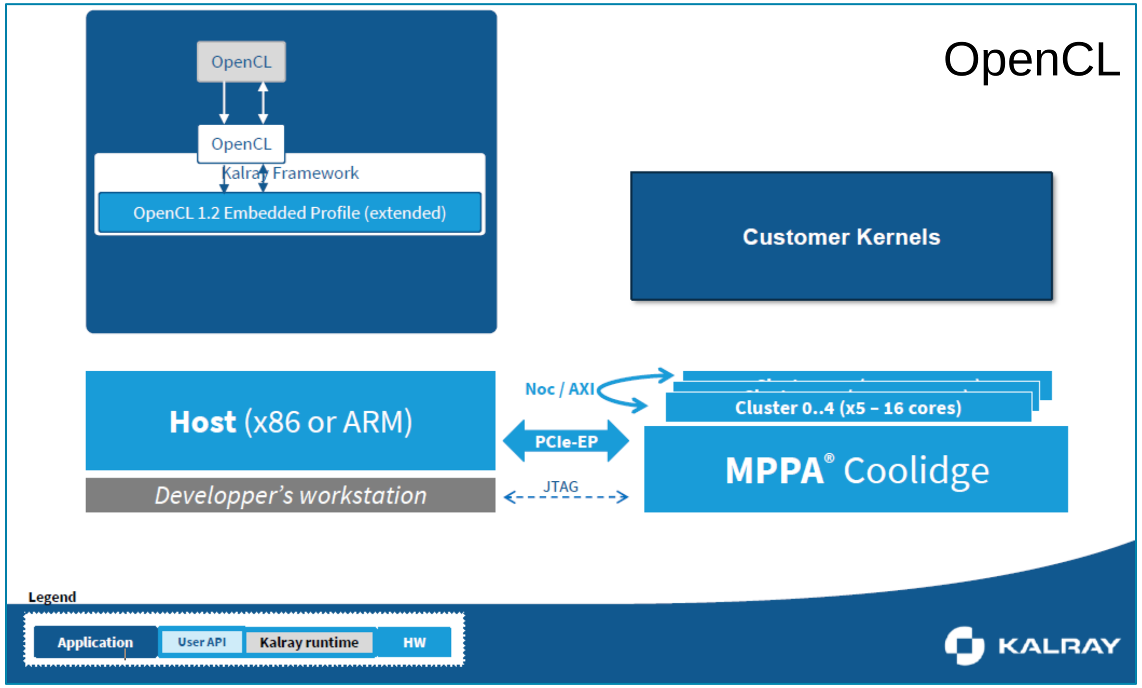
MPPA DEV4 includes all hardware and software to develop your application on the MPPA processor:

- 1x Workstation with
  - Intel Core i5-7500TE x86 Host processor and its motherboard
  - A 500 GB hard drive
  - 8GB DDR4-2133 memory w/ ECC
- 1x Kalray MPPA3-80 "Coolidge" PCIe board (K200)
  - Ready-to-production board
  - PCIe GEN4 16-lane / 2x DDR4-3200 channels w/ ECC
  - Power: up to 40W
  - Full Height Half Length form factor
  - 2x100 Gb Ethernet
- 1x MPPA® trace & debug USB probe

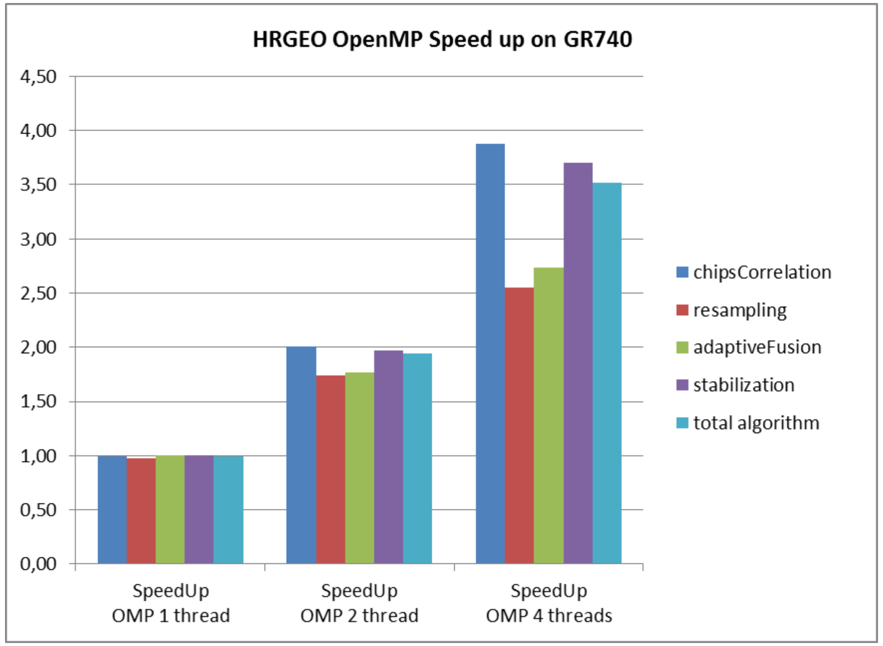
This product is fully integrated and tested before shipment to customer to quickly start evaluations

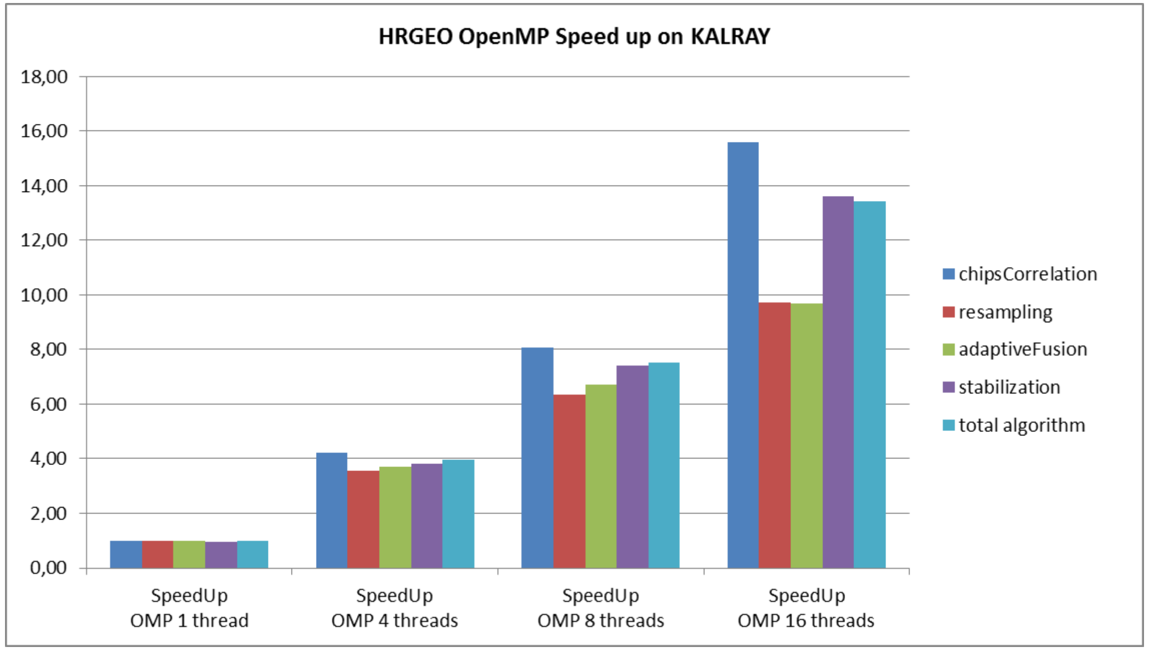
- Ubuntu
- Pre-installed AccessCore™ SDK

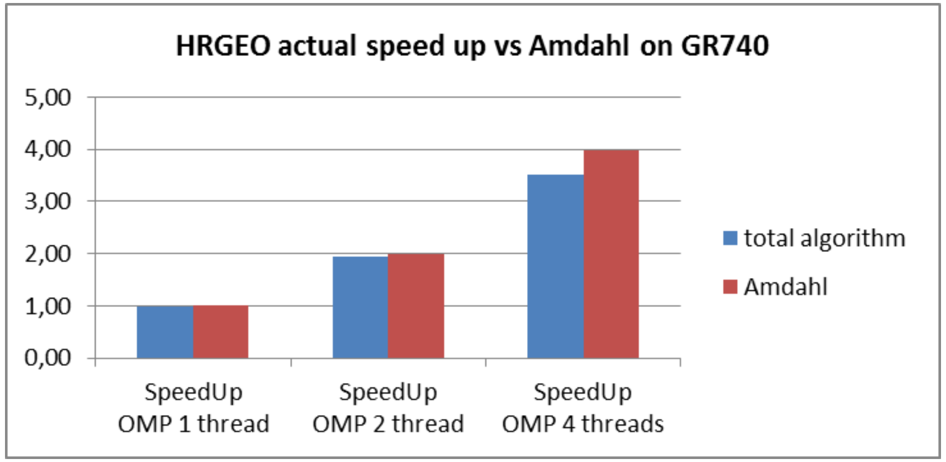


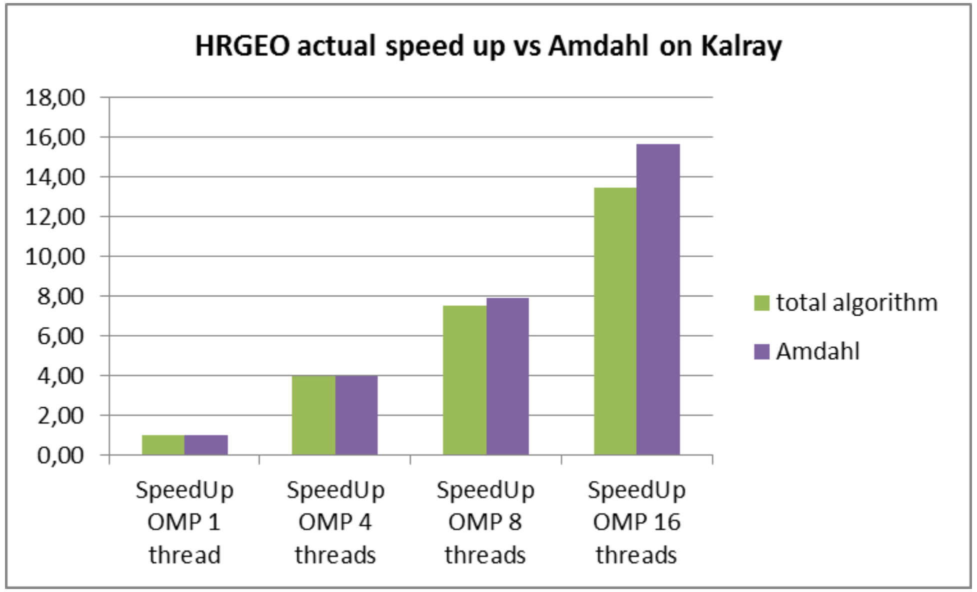


Speed Up HRGEO

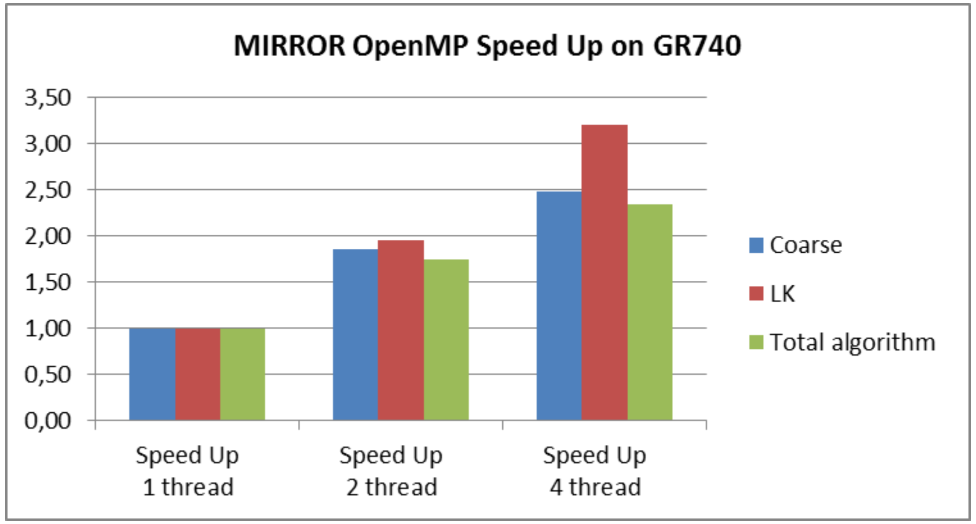


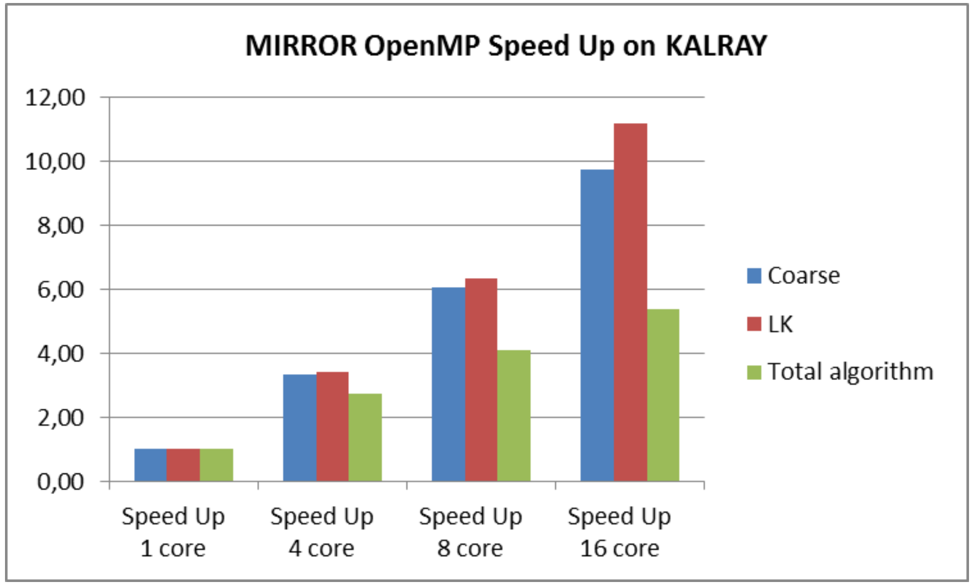


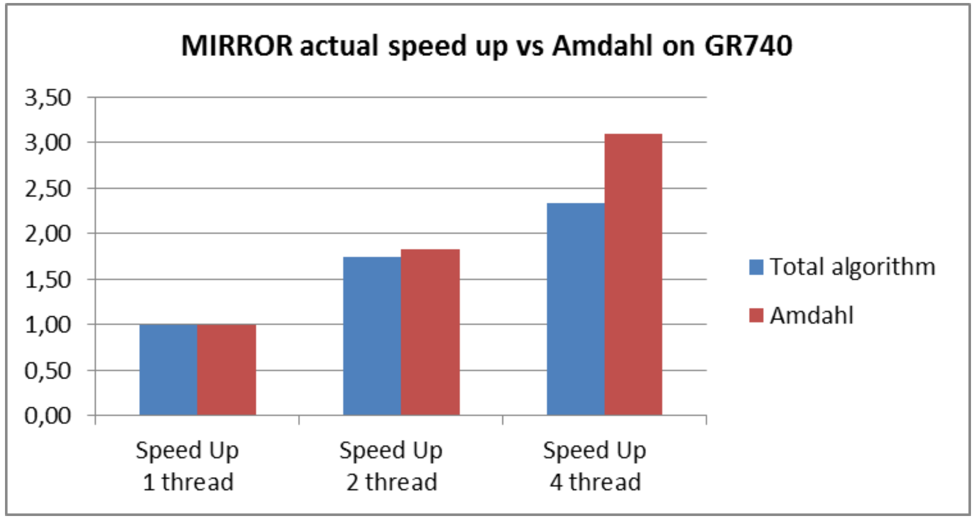


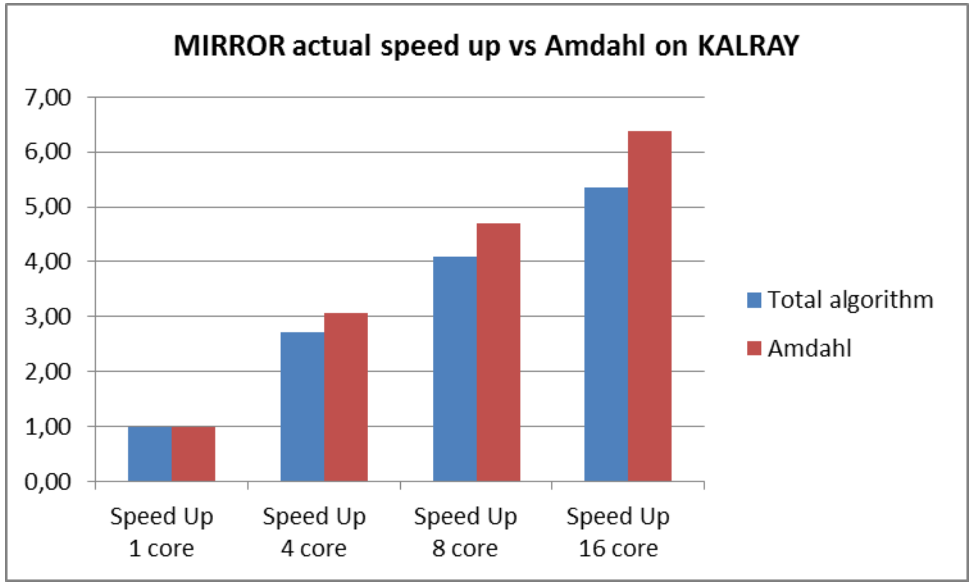


Speed Up MIRROR

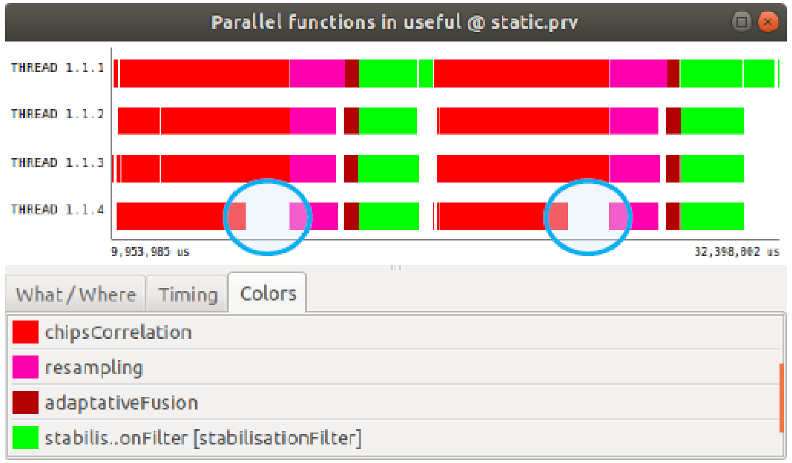

MIRROR OpenMP Speed Up on GR740


MIRROR OpenMP Speed Up on KALRAY

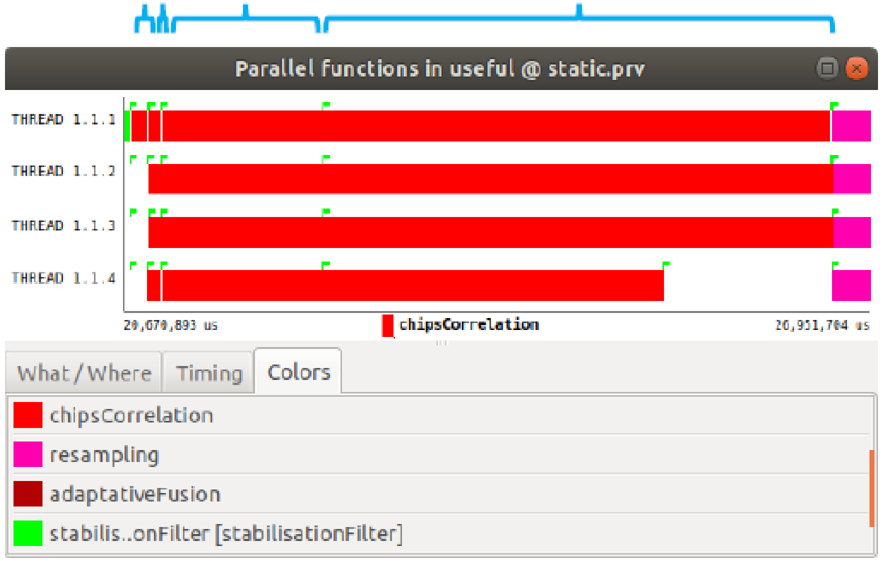

MIRROR actual speed up vs Amdahl on GR740


MIRROR actual speed up vs Amdahl on KALRAY

Extrae benefits : Load balancing

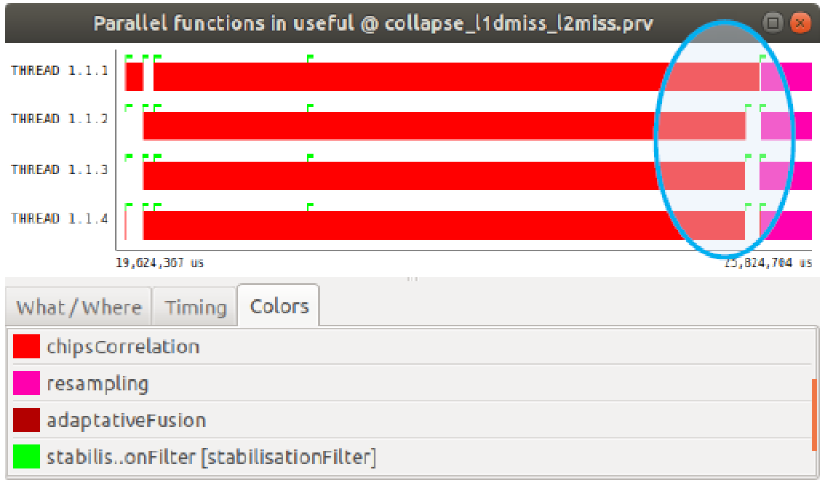


🟥 chipsCorrelation
🟪 resampling
🟥 adaptativeFusion
🟩 stabilis..onFilter [stabilisationFilter]



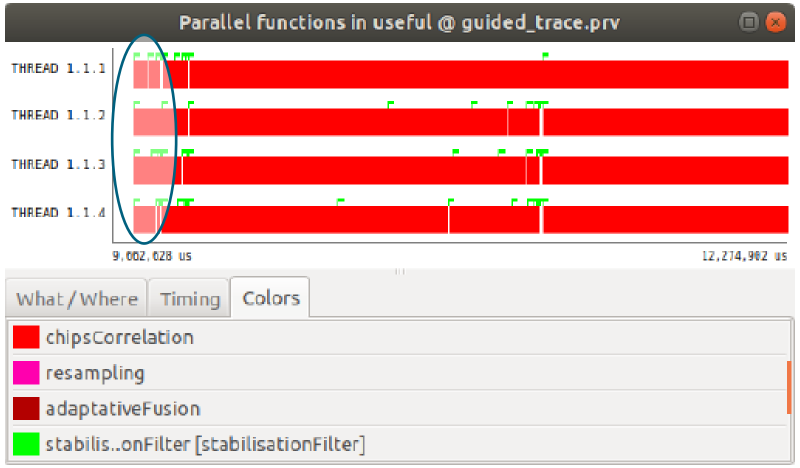
🟥 chipsCorrelation
🟪 resampling
🟥 adaptativeFusion
🟩 stabilis..onFilter [stabilisationFilter]

Collapse(2)



🟥 chipsCorrelation
🟪 resampling
🟥 adaptativeFusion
🟩 stabilis..onFilter [stabilisationFilter]

schedule(guided)



🟥 chipsCorrelation
🟪 resampling
🟥 adaptativeFusion
🟩 stabilis..onFilter [stabilisationFilter]

CONCLUSION & FUTURE WORK

Conclusion

☺ Facilitate the development, timing analysis and execution of parallel real-time space applications using the OpenMP parallel programming model.

☺ Evaluate the interest and porting effort of a list of homogeneous and heterogeneous foreseen COTS and RadHard hardware targets in the space domain with OpenMP programming model and framework.

☺ Demonstrate the portability benefits of the OpenMP parallel programming model.

Future Work

- OpenMP Qualification strategy & development process impacts
- OpenMP Time Predictability and WCET estimation for parallelized software
- OpenMP Offloading features and SIMD support exploration
- Extension of the study to ARM MP SoCs, and heterogeneous targets
- Continue exploring the Kalray MPPA potential as we only scratched the surface during this study

Thank you !

AIRBUS

Adapting Extrae to the GR740

1. Intercepting calls in a static environment
2. POSIX dependence
3. Retrieving function names
4. Trace generation
5. Supporting hardware counters
6. Statically defining the environment

Adapting Extrae to the Kalray MPPA Board

- Intercepting calls in a static environment and POSIX dependence is solved the same way GR740. However sampling is not supported yet.
- Trace generation is only available with the JTAG interface. Important system calls regarding writing files are not available through PCIE.
- Hardware Counters are straightforward, since PAPI is supported. No extra work needed.
- Retrieving function names is not available yet.