S4Pro

Smart and Scalable Satellite High-Speed Processing chain

Next Generation Computing & Software Development for Space

Presenter: J. Naghmouchi (iTUBS)

Contributors: Bacem Ferjaoui, Wadid Foudhaily, Lei Jia, Harald Michalik (iTUBS),
Ole Bischoff, Daniel Smith(DSI), Rolf Scheiber (DLR), Hamid Amiri (ENIT), Riccardo Freddi (OHB-I)

This document has been produced under Grant Agreement 822014.

Development Objectives

Application Objectives

- Improved SAR data filtering decimation and compression
- Optimized data transmission through on-board data elaboration
- Near real time data processing
- Improved pattern recognition
- Multi architecture design

Demands on Algorithms

- Processing Efficiency
- High Accuracy
- Compression of Data during elaboration
- High Degree of Flexibility

- Massive Data Elaboration at Low Power
- Leverage Miniaturized HW Modules based on COTS
- High Degree of Modularity and Portability
- Run on Real-Time OS

Demands on SW

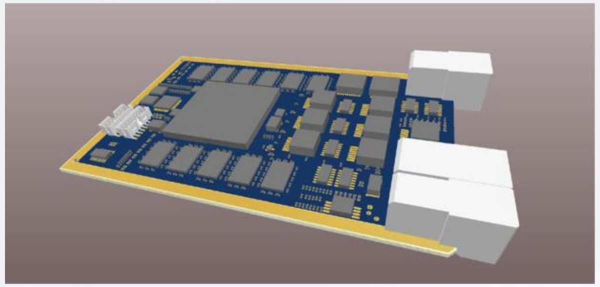
Implemented Features

- ✓ Modified Real-Time Kernel for Linux
- ✓ Self-Optimizing SW Libraries
- ✓ Portability Layer for SW Libraries
- ✓ Algorithm and Software Co-Design

SUMMARY OF RESULTS

Target Payload Processing Unit

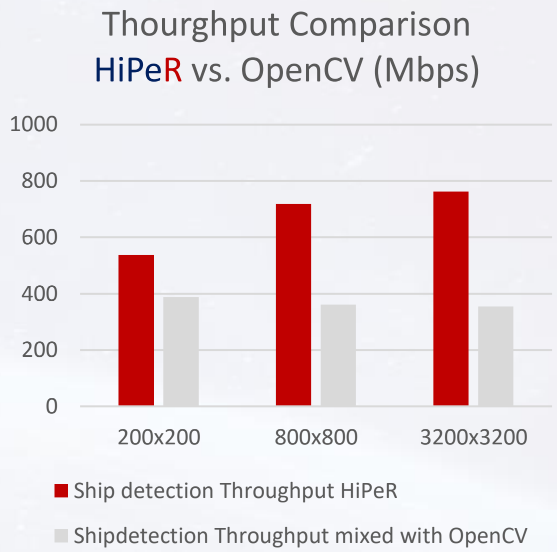
- Zynq ultra scale ZCU102
- ARM cortex A53
- Frequency 1200Mhz
- PCI Express x4
- Peta-Linux (Modified Kernel)



Optical Ship Detection Performance

- Ref throughput: 86 Mbit/s
- Achieved throughput:

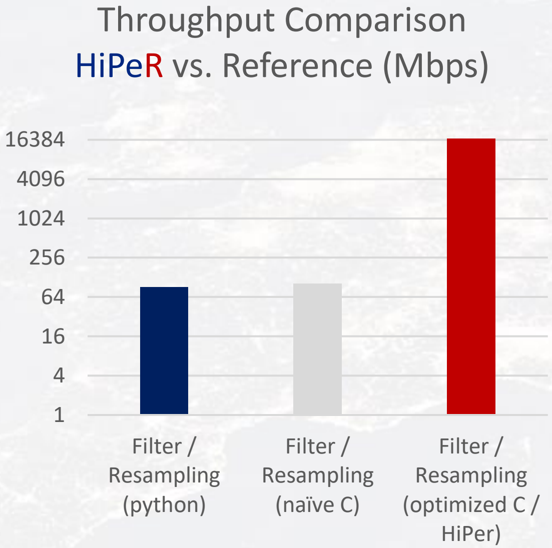
  762 Mbit/s
- Accuracy : 99.98 %
- Meets Requirements of future Ship detection applications for OHB nanosat platforms! (600Mbit/s)



Staggered SAR performance

- Ref Throughput: 0.097 Gbit/s
- Achieved throughput:

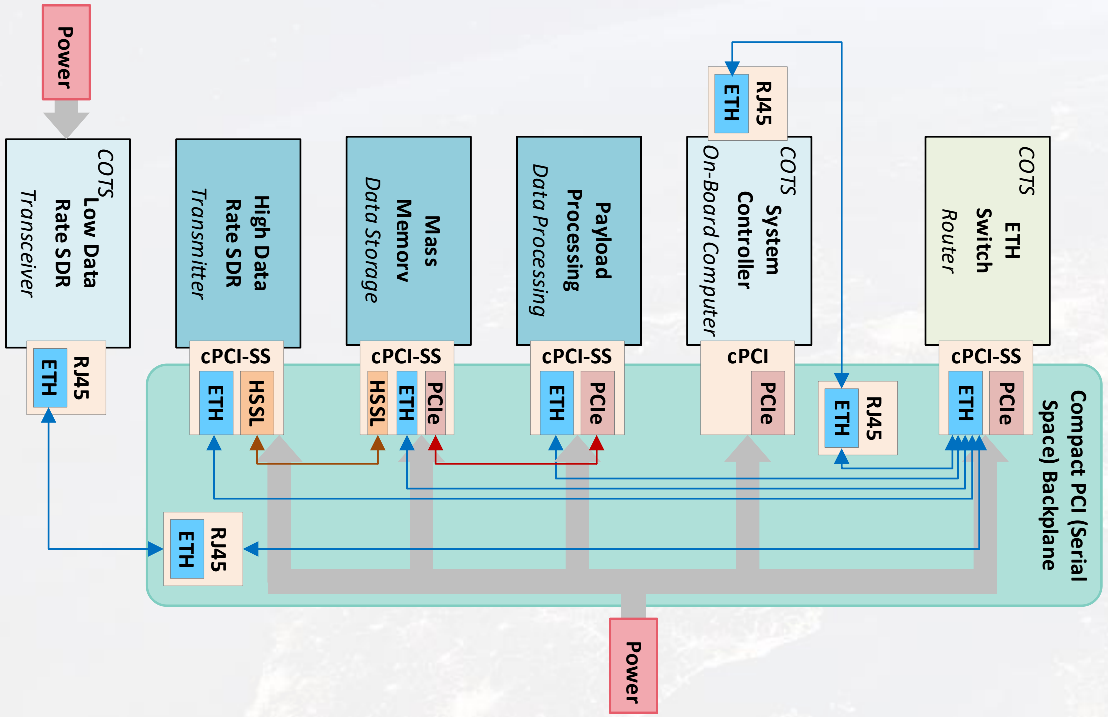
  16.8 Gbit/s
- Accuracy : 99.91 %
- Meets requirements similar to ROSE-L missions and beyond



HW Architecture overview

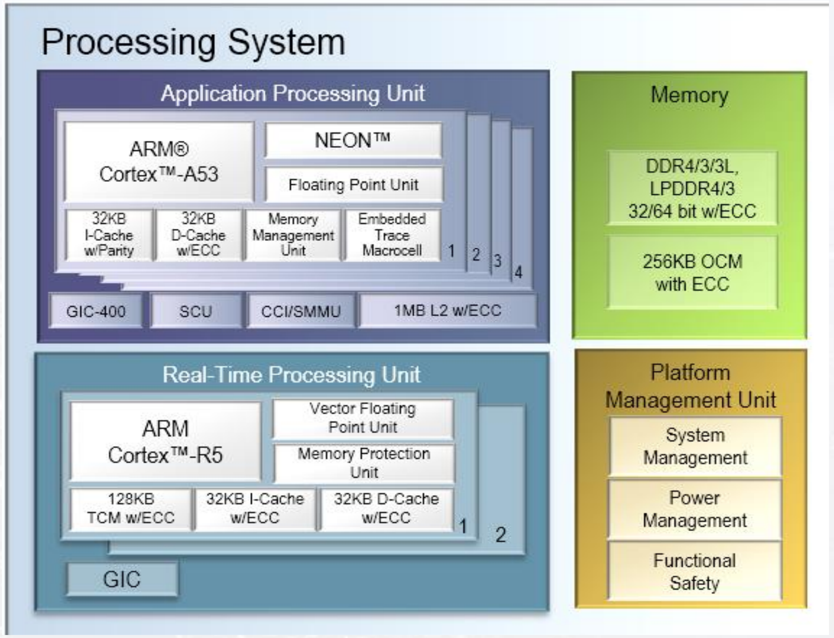
System Architecture

- In S4Pro an on-board computer system is developed that includes the Ultrascale+ as a payload processing unit due to its computational performance, interfacing capabilities (PCIe Gen2/HSSL, DDR3/4, ethernet etc.), and programmable logic.
- For Mission critical tasks such as command and control, space qualified HW is being employed in the project, though.



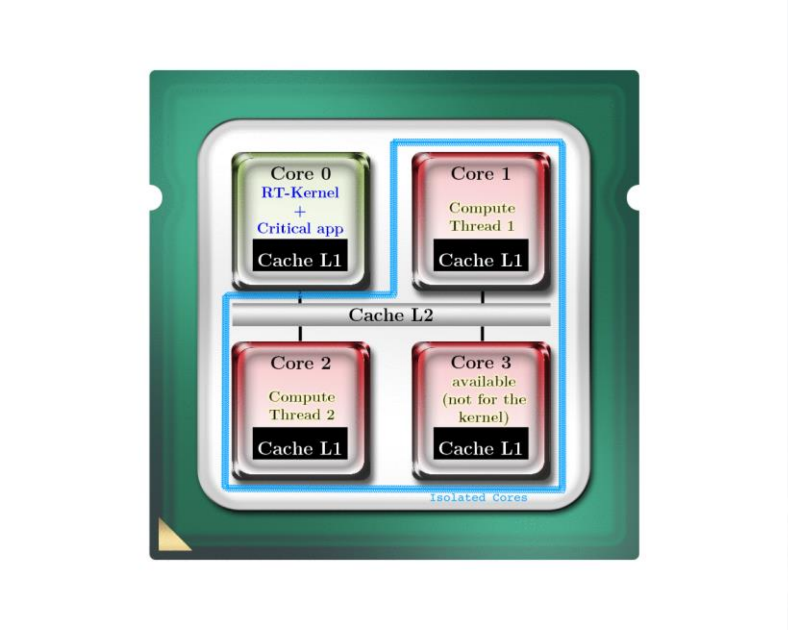
PPU Architecture

- Driving is the Xilinx Zynq Ultrascale+ as representative HW for New Space Technology.
- Although the Processor is not (yet) space qualified, it has been under investigation for use in space.

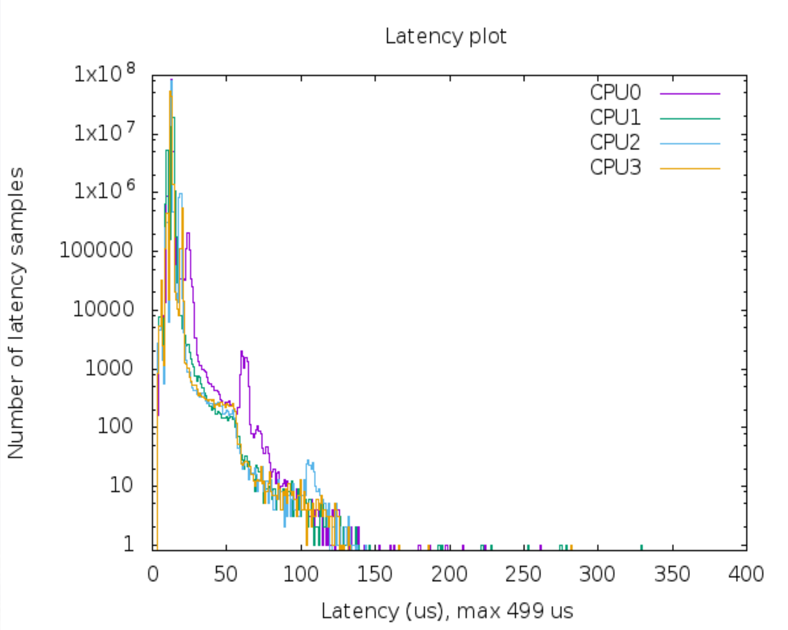


RT OS Overview

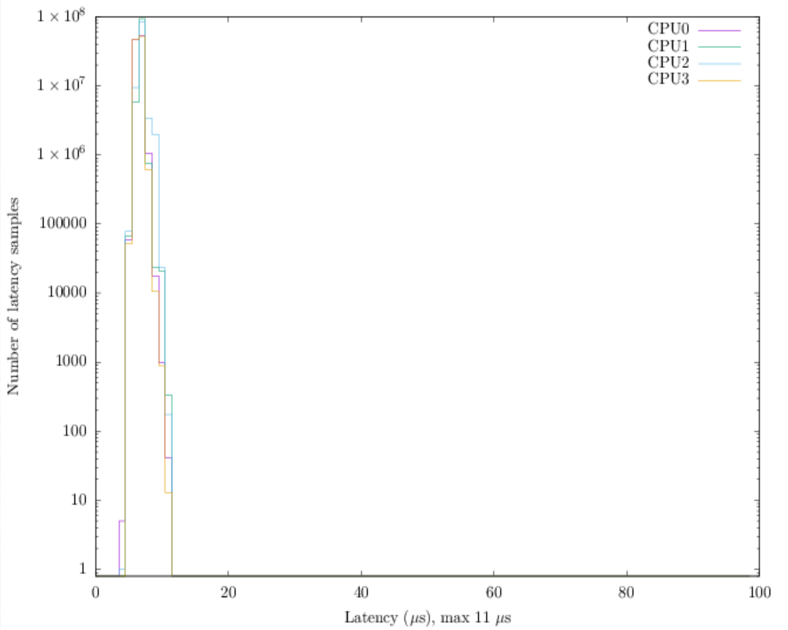
Architecture



Default behavior



Beyond RT behavior



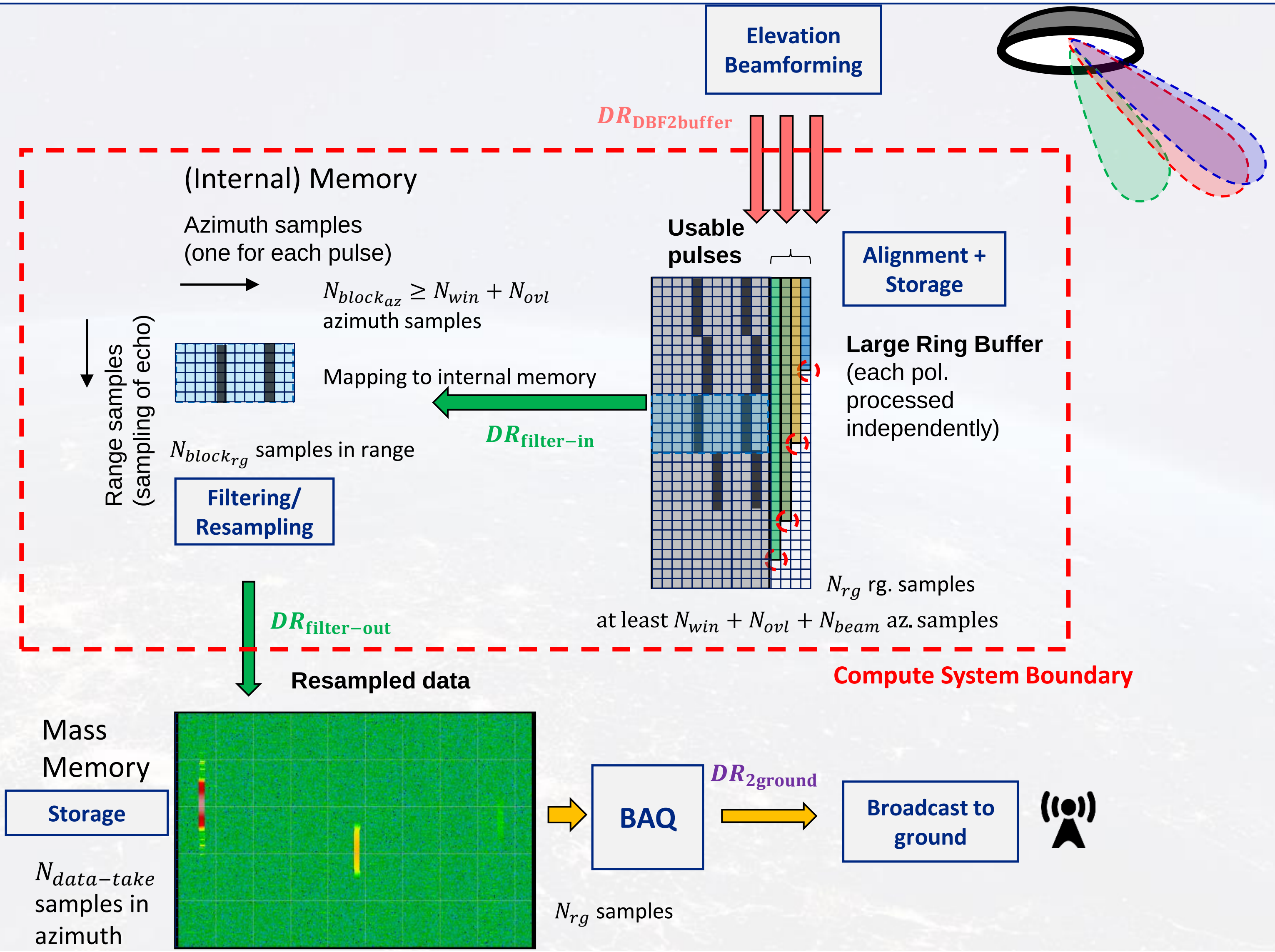
Changes

- RT Kernel (Full preemption)
- (SOFT) Overclocking
- TLB tuning, kernel tweaking
- Isolated cores used solely for our application.

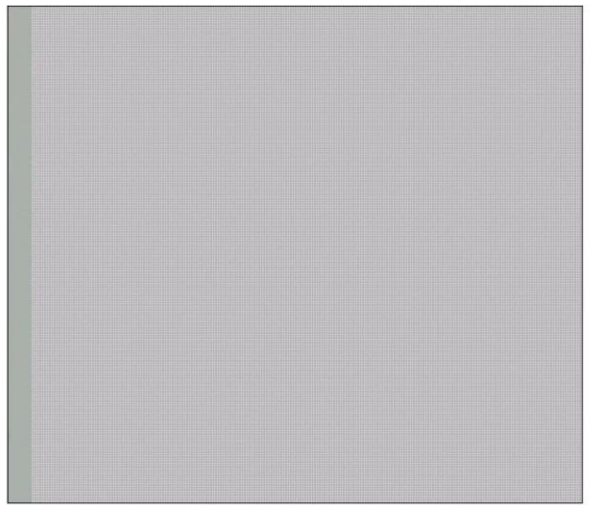
SAR ALGORITHMS OVERVIEW

The main issues:

- Due to the temporal order of the incomming data stream, the spatial representation of the data is not optimal for stream processing
- The data size is relatively large
- The incomming data stream reaches a very high throughput
- Data Access is based on pre-generated tables
- Transpose is NO option!



SAR Processing and Mapping overview



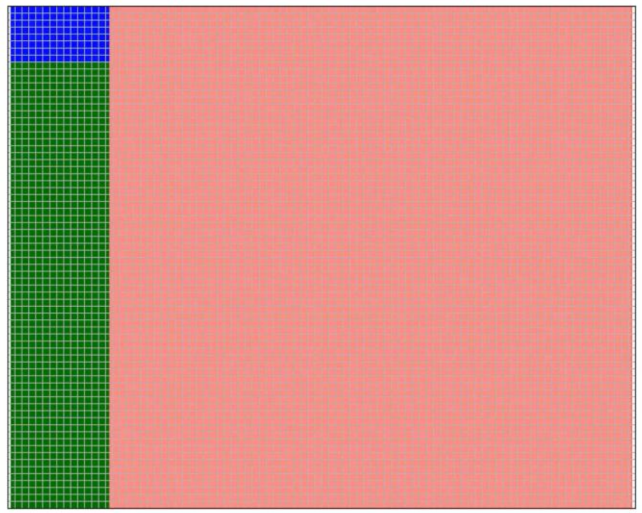
This is how a filter can run through the data, if it were nicely mapped and aligned.

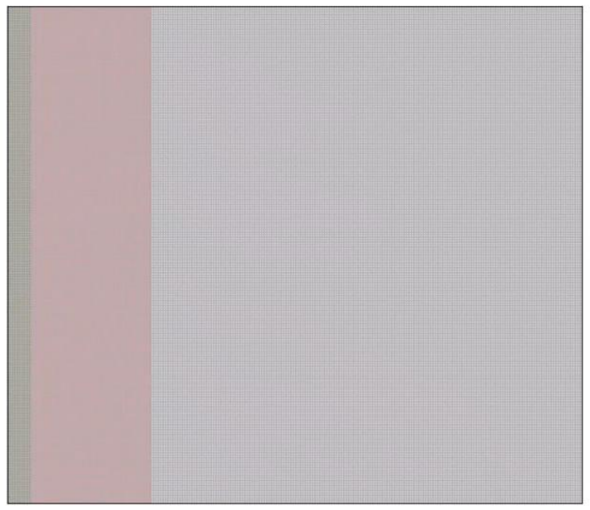
First of all, we need to assume that the data is already mapped in memory.

Remapping of data streams comes at high cost.

So let's make it fit!

This still works for an ***output oriented strategy****.*



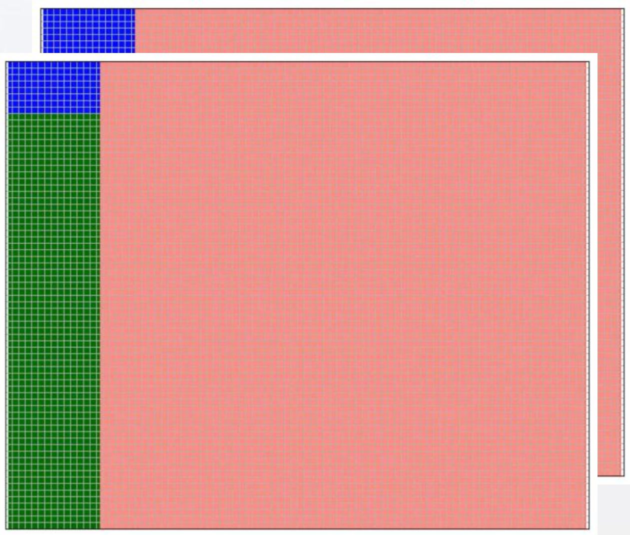


This is how a filter can run through the data (even for multiple PRI cycles), if it were nicely mapped and aligned.

Secondly, processing the data streams inline will cause intermediate results to be evicted from local memories (D$).

Inline Processing will be slow, due to zero reuse.

Also works for multiple threads!



Performance Comparison – Staggered SAR

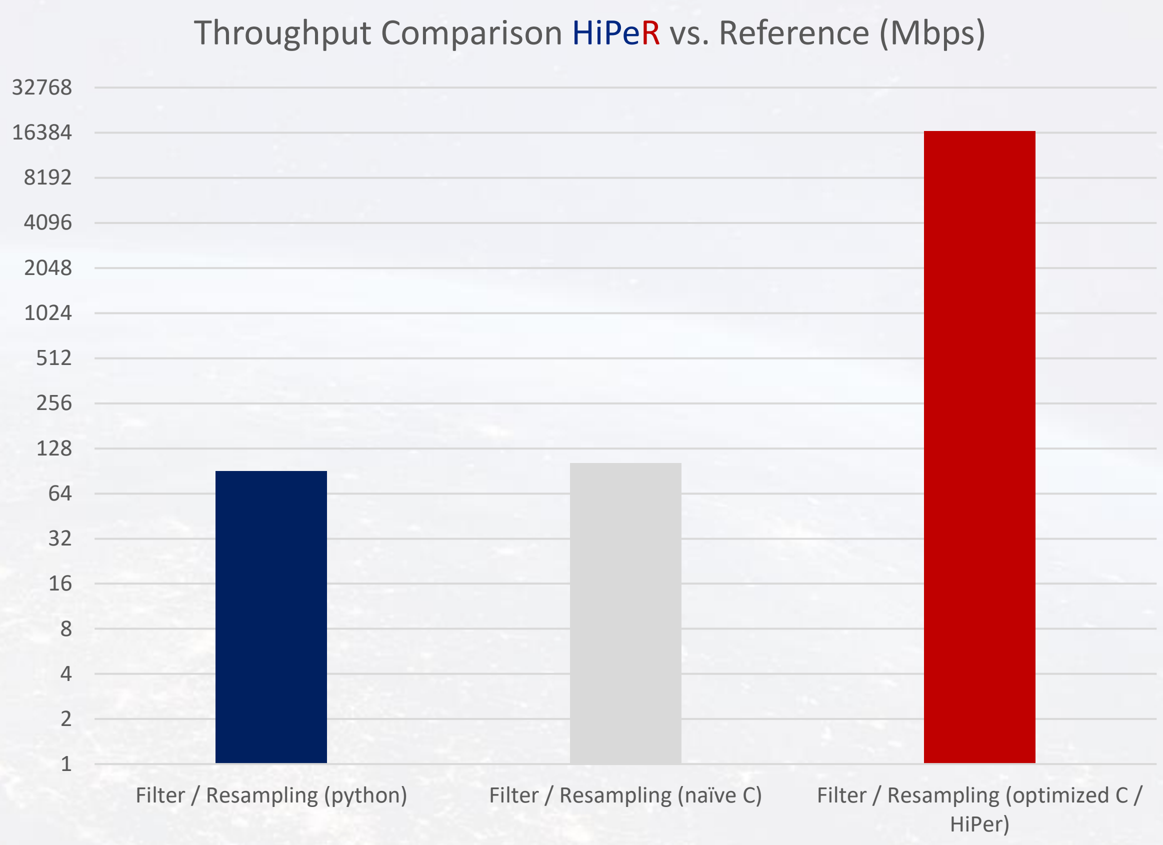
Since no performance results have been published on this application before, we show the improvement throughout our own development:

Reference Throughput: 0.097 Gbit/s (Python)
First Implementation: 0.102 Gbit/s (Naïve C)

<u>Achieved throughput: 16.8 Gbit/s</u> (HiPeR)

-> **Acceleration by** a factor of **164.7x** over a naïve implementation

This meets requirements similar to ROSE-L missions and beyond!

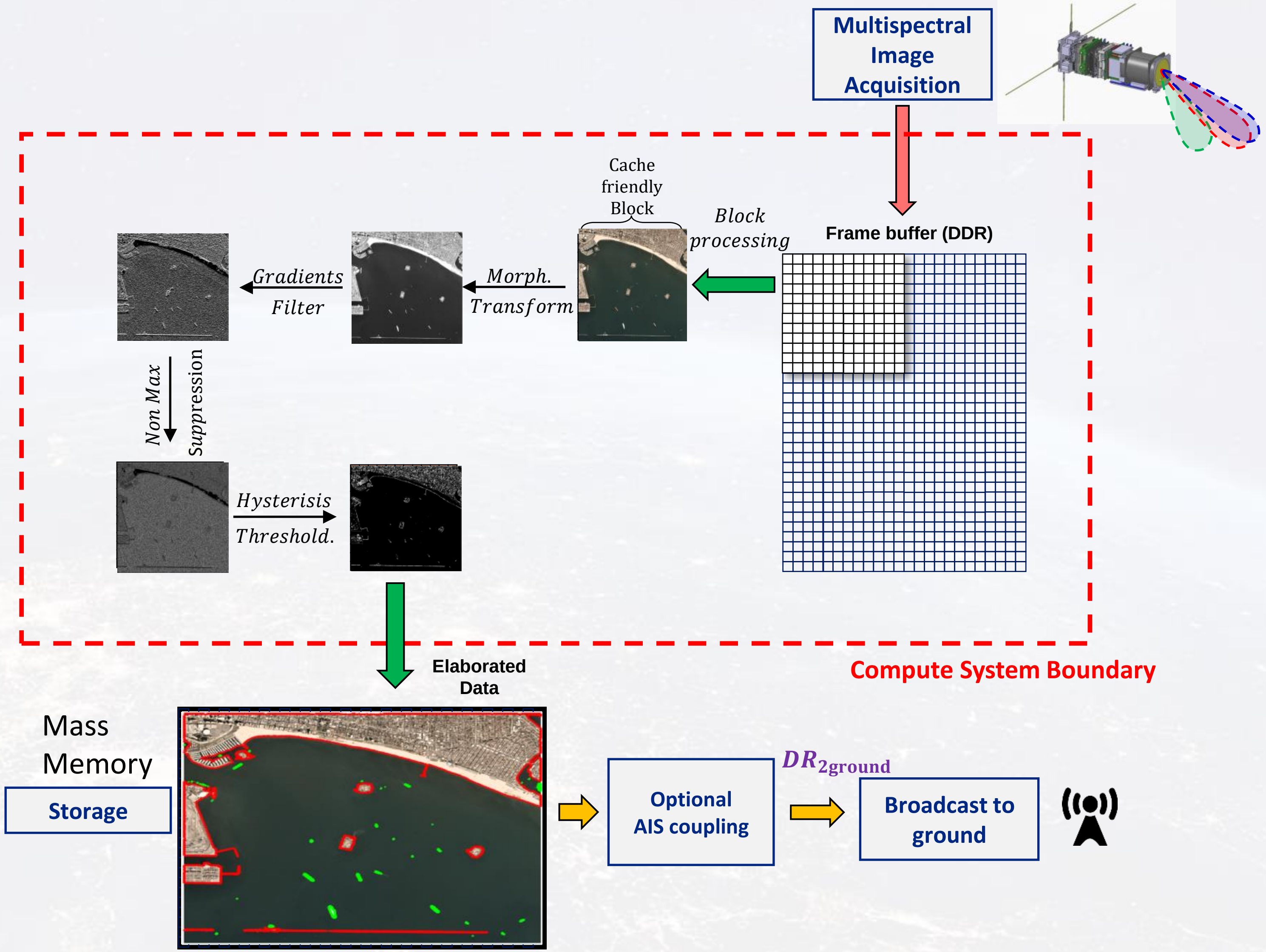


OPTICAL SHIP DETECTION ALGORITHM OVERVIEW

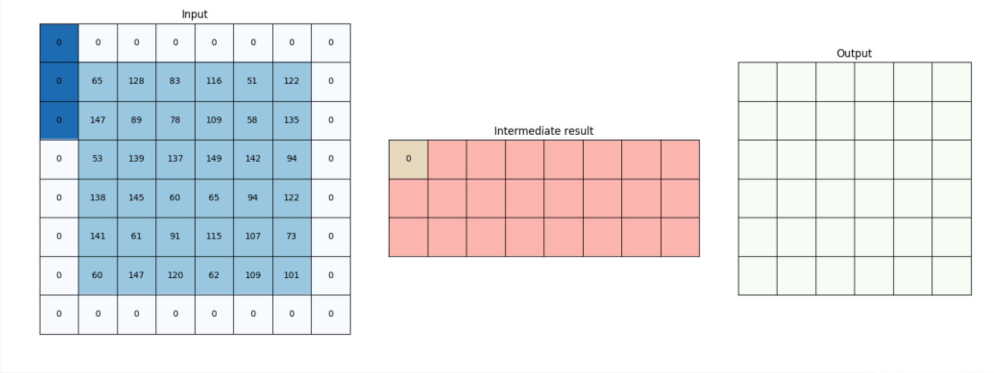
S4Pro has identified edge-detection as the most performance-critical onboard processing task for ship detection.

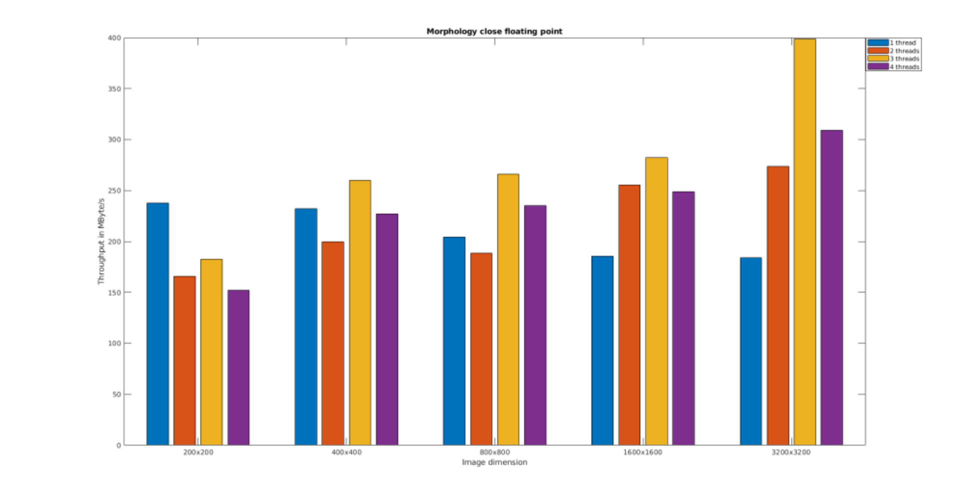
Therefore, benchmarking activities have focused on the acceleration of corresponding software routines.

Benchmarking activities were carried out in parallel to the algorithm design. In order to correct the path.



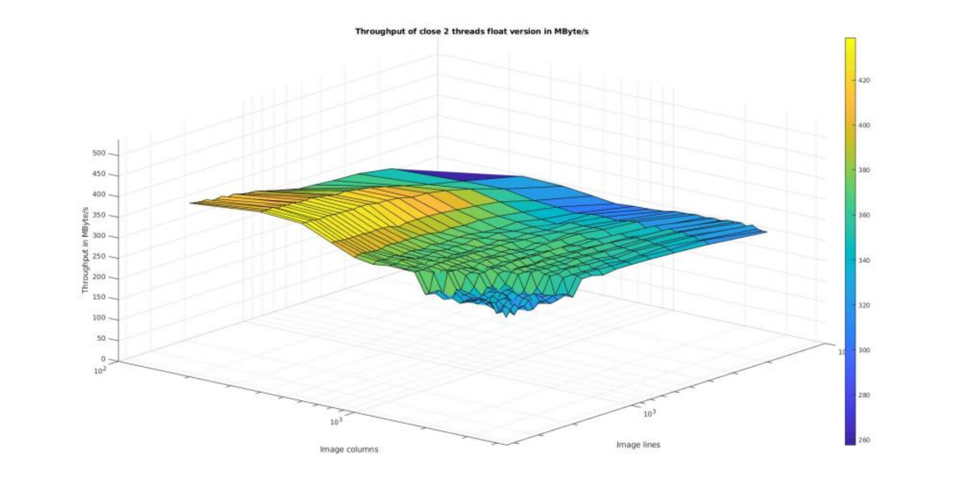
Optical Image Processing and Mapping overview



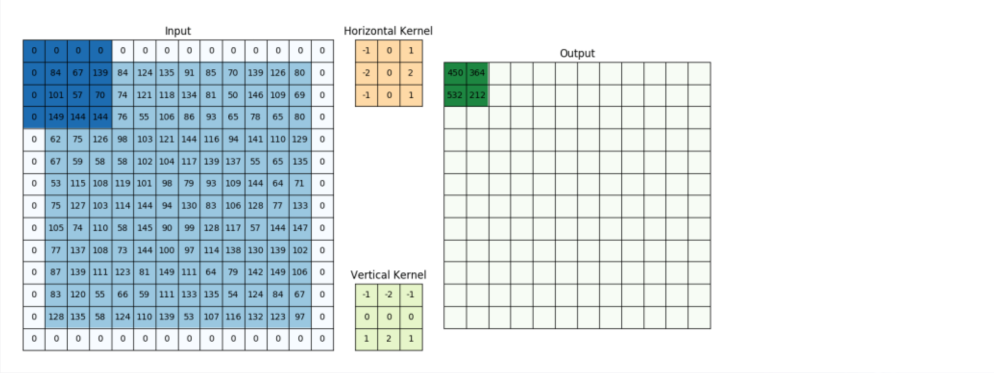


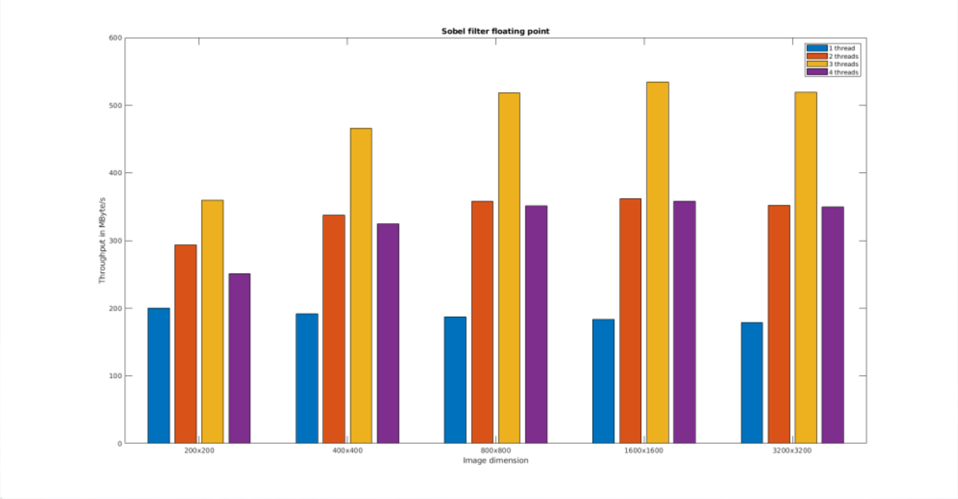
Morphological Closing is useful for filling small gaps and sharpening edges of an image, while preserving shape and size of the objects in the image.

- It consists of a dilation operation followed by erosion.
- The algorithm is memory bound.
- SIMD instructions have been used to vectorize the code.
- Multithreading improved the throughput.

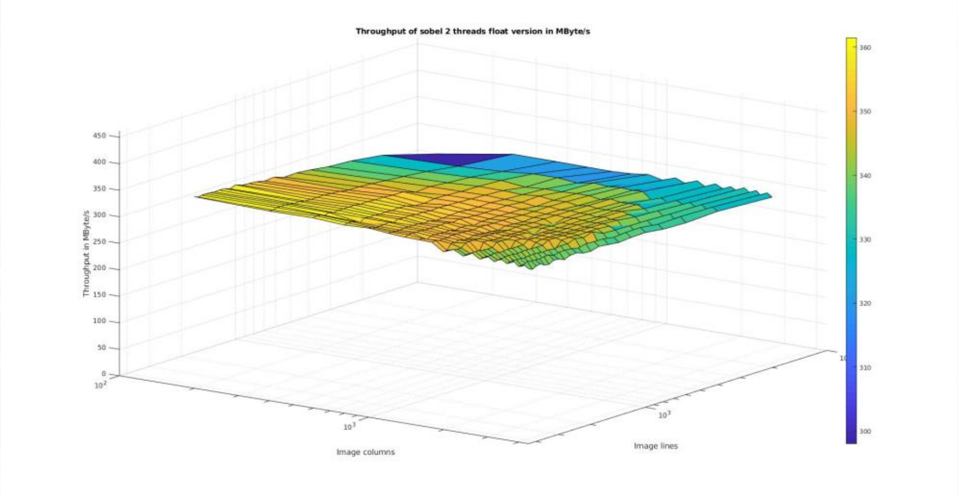


Optical Image Processing and Mapping overview



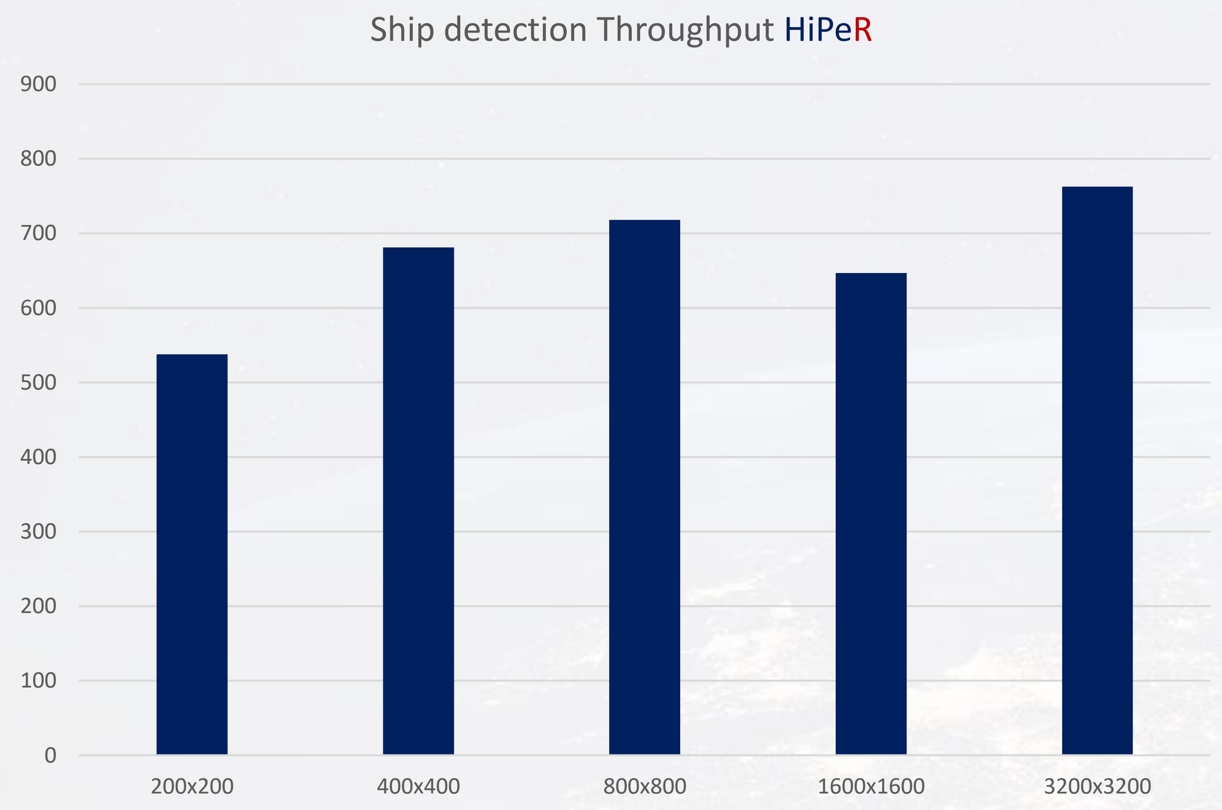


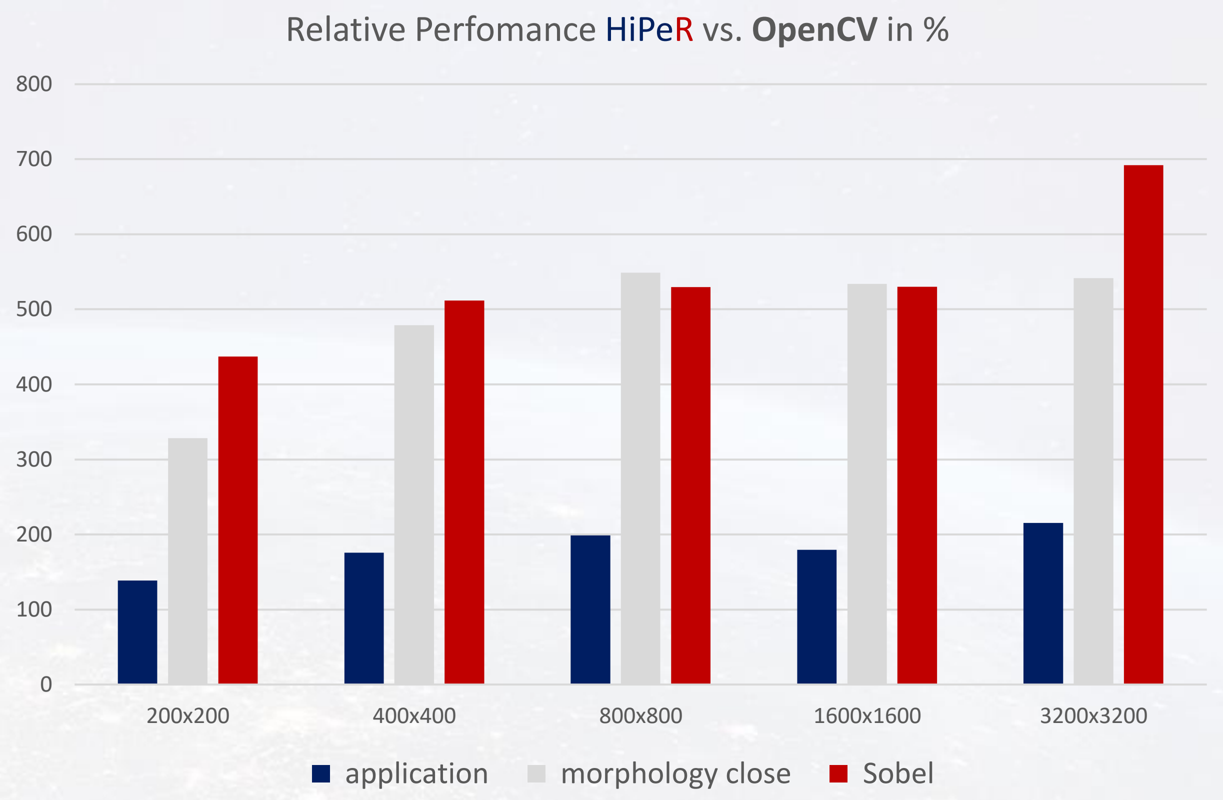
The **Sobel Filter** consists of two 3x3 convolution kernels. Then gradient magnitude and phase are calculated.



- The algorithm is compute-bound
- Data access is improved by using loop tiling
- SIMD and multithreading were used as well

PERFORMANCE COMPARISON - SHIP DECTECTION





Achieved throughput: up to 762 Mbit/s (depending on window size)

Achieved acceleration compared to known solutions: up to 691%

Note: Routines not included in the OpenCV Distribution were replaced by fully optimized HiPeR routines, in order to make a fair comparison for the overall application performance.

Outlook and Future Work

- Implementation and Qualification of pattern recognition SW in commercial nanosats

- Further developments of Staggered SAR applications

- Development of the Next Gen PPU (initial benchmarks have been started)

- Porting the SW to other available processing platforms

- Enlarging the HiPeR library

THANK YOU

ANY QUESTIONS?

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822014.
This text reflects only the author's views and the Commission is not liable for any use that may be made of the information contained therein.