Optimized multicore implementation and benchmark for on-board image compression algorithm - on dedicated space HW

Authors: Marius Olaru
Alexandru Bulumac
Valentin Picos

ENEA

esa
European Space Agency

AIRBUS
DEFENCE & SPACE

Outline

- Context
- Purpose
- Hardware environment
- Software environment
- Algorithm's parallelization
  - Aim and objectives
  - Algorithm overview
  - Proposed solution
    - Image decomposition
    - The master/worker pattern
    - Prediction stage
    - Encoding stage
    - Parallel packing of code-words
- Implementation results

ENEA

esa
European Space Agency

AIRBUS
DEFENCE & SPACE

Context

- ENEA is the prime contractor and Airbus is the sub-contractor.
- ESA intends to introduce multicore processors in future space missions.
- The processing power on satellites is limited and multicore processors offer a good balance between performance and power consumption.
- The necessity for performing hyperspectral image compression on multicore systems running on-board satellites has increased as hyperspectral sensors generate a very large amount of data.
- The storage space and the bandwidth used for transmitting images is a scarce resource on satellites.

ENEA

esa
European Space Agency

AIRBUS
DEFENCE & SPACE

Purpose

- To parallelize the lossless compression algorithm for hyperspectral and multispectral images described in the CCSDS 123.0-B-1 standard.
- To optimize the software implementation of the parallelized algorithm so that it can run effectively on multi-core processor systems.
- To verify the developed code on three hardware platforms and benchmark its performance.

Project's team: 4 members

Project duration: 25 months

ENEA

esa
European Space Agency

AIRBUS
DEFENCE & SPACE

Hardware environment

- Regular x86 PC with 4 cores running Linux.
- GR-CPCI-LEON4-N2X board (Quad-Core 32-bit LEON4 SMP processor) running RTEMS RTOS with SMP support.
- NXP (Freescale) QorIQ P4080 board running Linux.

ENEA

esa
European Space Agency

AIRBUS
DEFENCE & SPACE

Software environment

- Linux for cross-platform application development
- RTEMS RTOS (Real-Time Executive for Multiprocessor Systems)
- MTAPI (Multicore Task Management API) library for parallel programming
- Transport layers for Ethernet (TCP)
- C Programming Language for the parallel program and Python for the test framework

ENEA

esa
European Space Agency

AIRBUS
DEFENCE & SPACE

Algorithm's parallelization - aim and objectives

- To evaluate different parallel programming models and select a parallelization scheme;
- To investigate the types of parallelism (e.g. task, data, and pipeline parallelism) that exist in the algorithm and how the interactions between cores (contention) can be minimized;
- To investigate what processing efficiencies can be gained from distributing and balancing the processing workload across the available cores;
- To exploit the identified parallelism in the sequential algorithm and provide a parallel implementation that maximizes throughput and reduces the execution time of the implementation;
- Performance evaluation and analysis.

Algorithm overview (1)

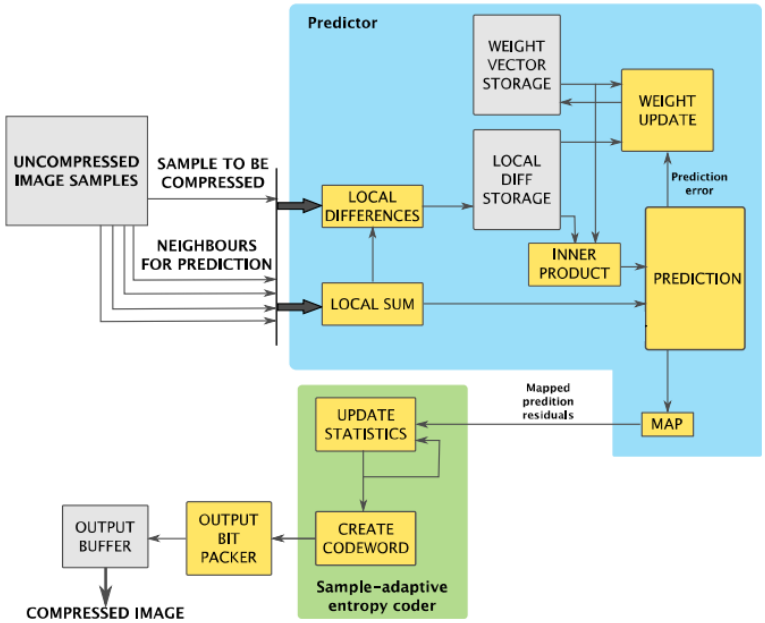

Figure 1 – Simplified schematic of the compression algorithm with Sample-Adaptive Entropy Coding (CCSDS 120.2-G-1 standard)

Algorithm overview (2)

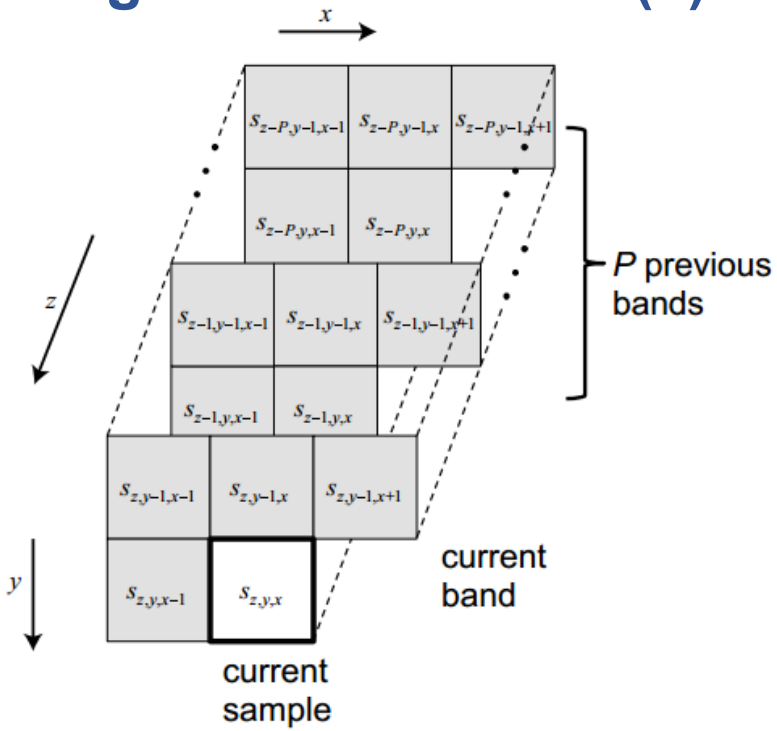

Figure 2 - Current sample and the nearby samples used for calculating the local mean value, the local differences vector, and the prediction error

ENEA

esa
European Space Agency

AIRBUS
DEFENCE & SPACE

Image decomposition (1)

- The strategy employed for decomposing the problem space into sub-spaces is based on geometric decomposition.
- The image is divided along the y and z axes in blocks of samples from entire x axis.
- The input image is decomposed into independent blocks which are reformatted to BSQ format in parallel.
- Neighbouring blocks along the y axis must be processed in FIFO order in order to solve the data dependencies that exist between these blocks.

Image decomposition (2)

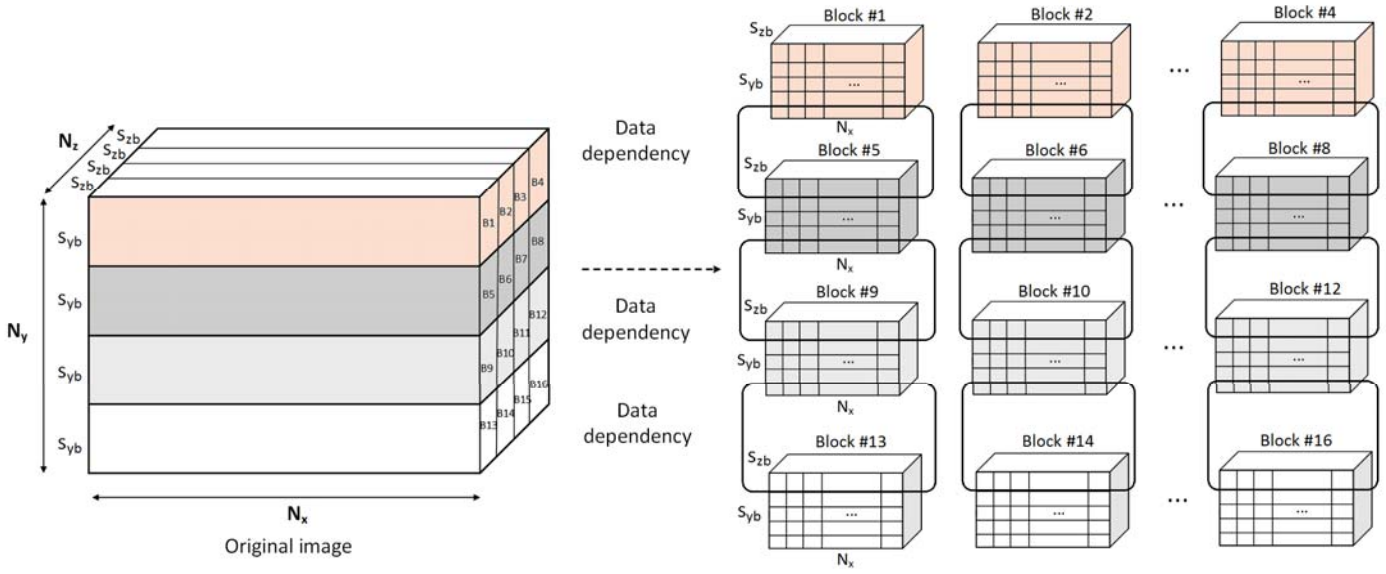

Figure 3 - Original image is split into 16 blocks of Nx·syb·szb samples (e.g. Nz = 4·szb and Ny = 4·syb)

Master/Worker pattern

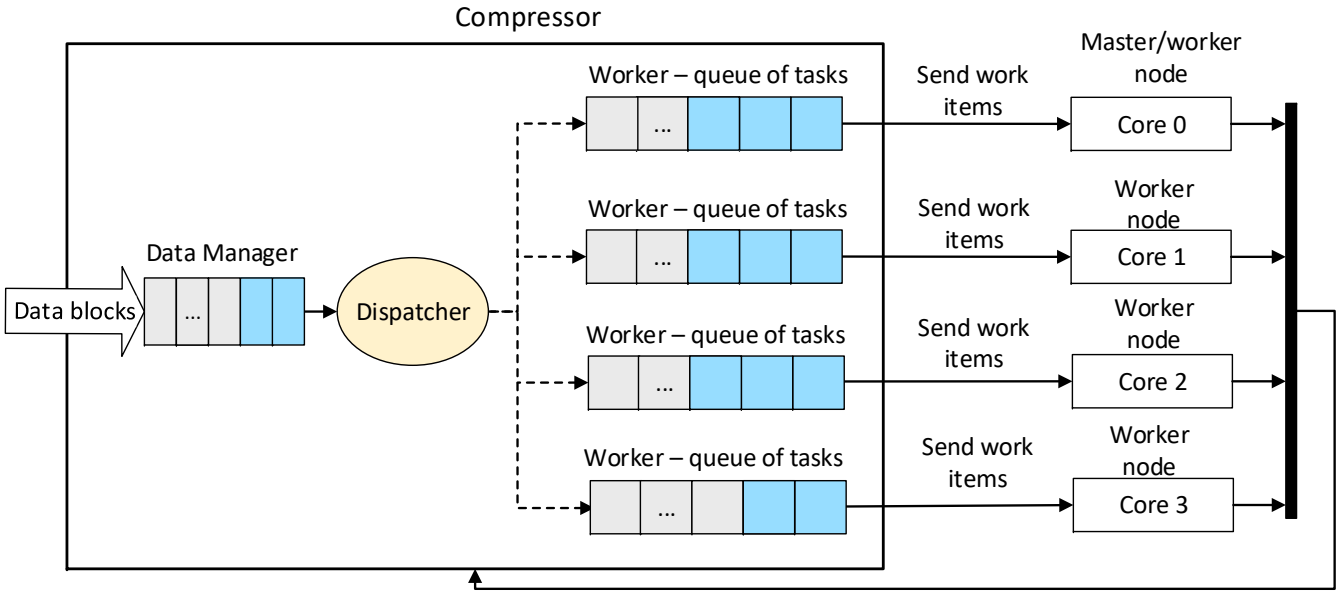

Figure 4 - Master-worker architecture for distributing the computational effort of the compression algorithm

Prediction stage

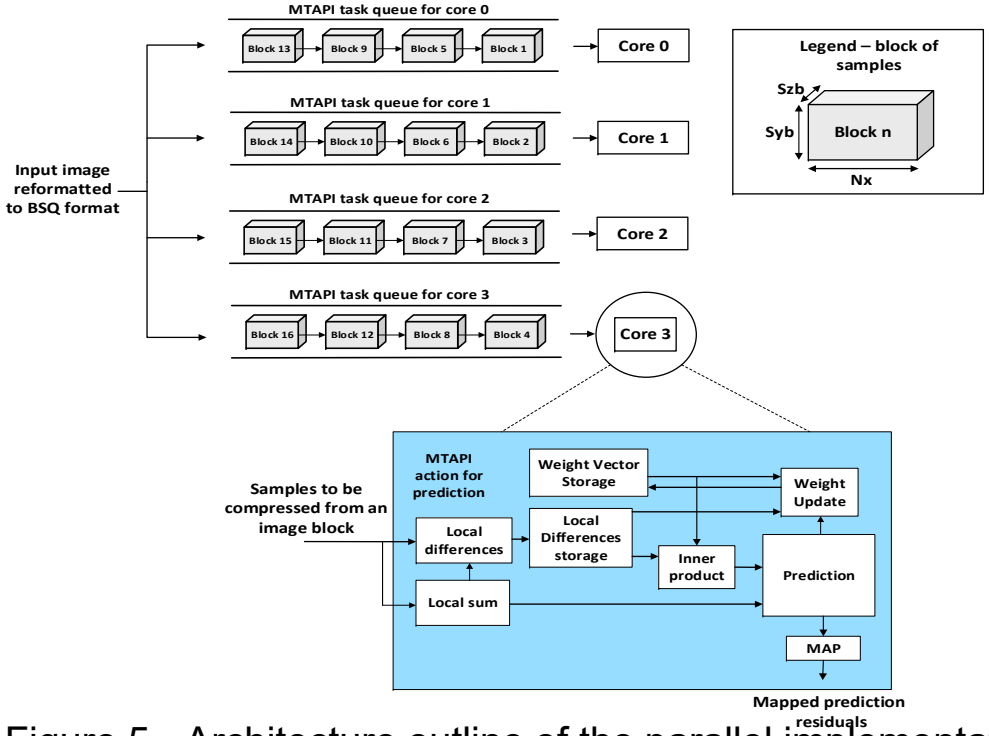

Figure 5 - Architecture outline of the parallel implementation of the prediction stage of the compression algorithm (e.g. for four processor cores)

Encoding stage – SA, BSQ

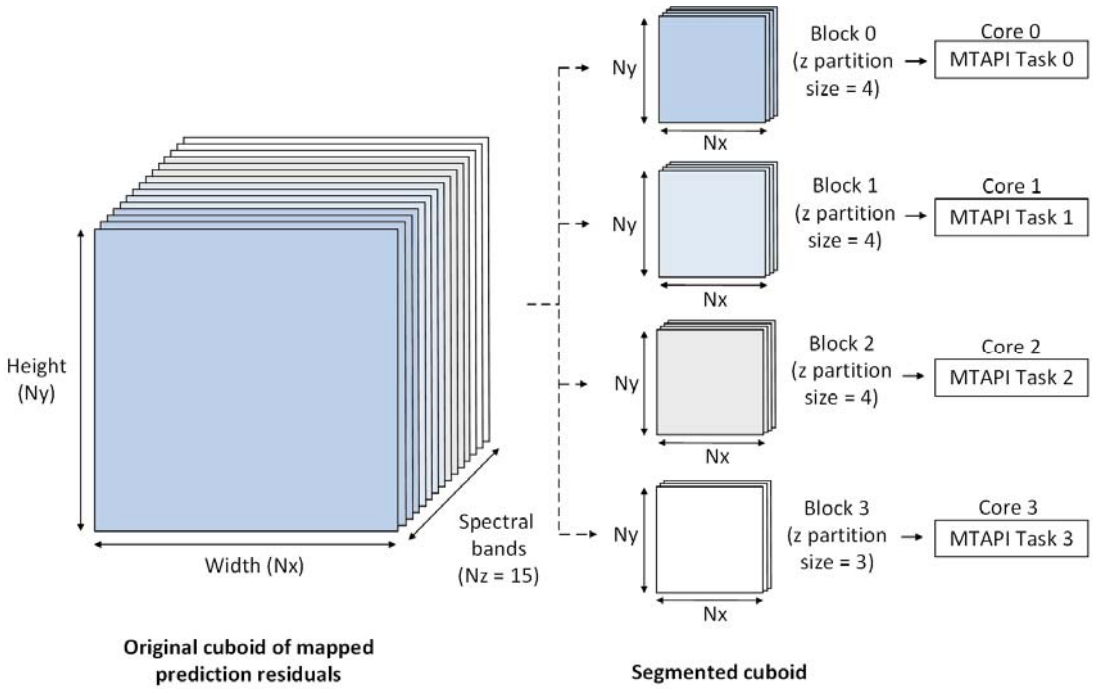

Figure 6 - Data partitioning required to encode the mapped prediction residuals in parallel (Sample adaptive, BSQ order, Number of cores = 4)

Encoding stage – SA, BI

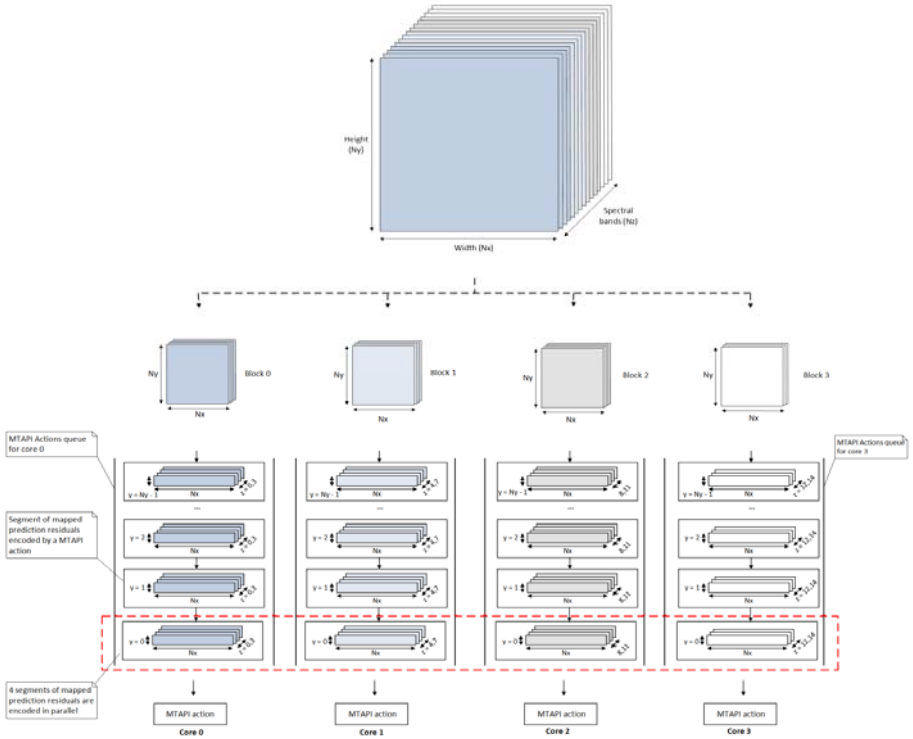

Figure 7 - Data partitioning required to encode the mapped prediction residuals in parallel (for the case when 4 processor cores are used)

Encoding stage – BA

• The parallelization scheme used for encoding the mapped prediction residuals using the block-adaptive encoding method has similarities to the one described for the sample adaptive, BSQ case.

• The most significant difference is the way in which the size of each partition is calculated. For the block-adaptive encoding method, the size of each partition is calculated based on the number of reference blocks that exist in an image, while for the sample-adaptive method the size of each partition was calculated based on the number of spectral bands, N_z.

• For instance, if the number of reference blocks is 15 and there are 4 processor cores, there will be four partitions as follows:

[0, 3] - 4 groups of blocks (r) are processed by core 0
[4, 7] - 4 groups of blocks (r) are processed by core 1
[8, 11] - 4 groups of blocks (r) are processed by core 2
[12, 14] - the last 3 groups of blocks are processed by core 3

Parallel packing of code-words

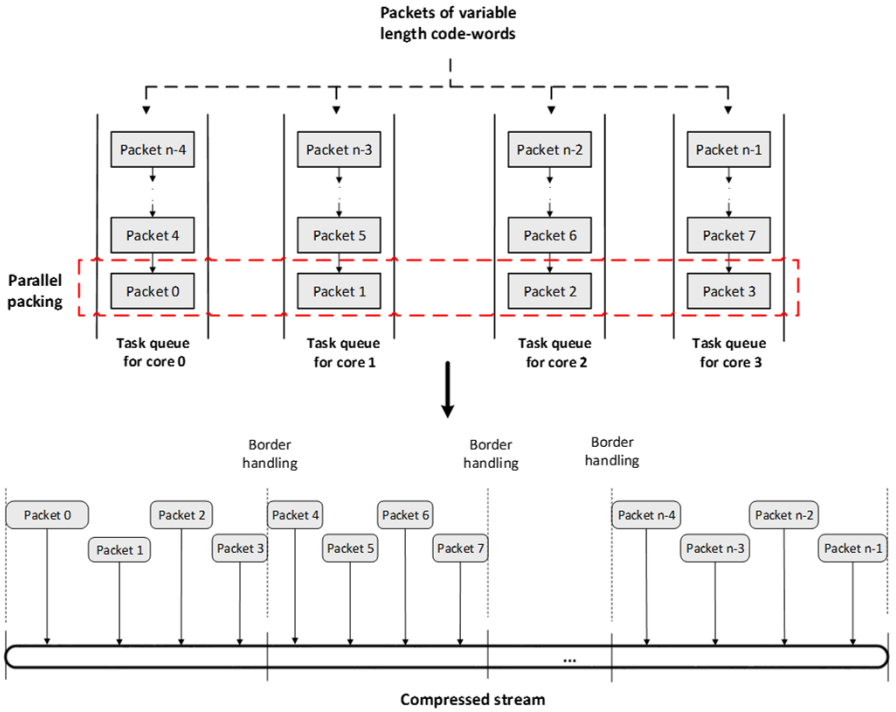

Figure 8 – Packing of variable length code-words on a quad-core processor

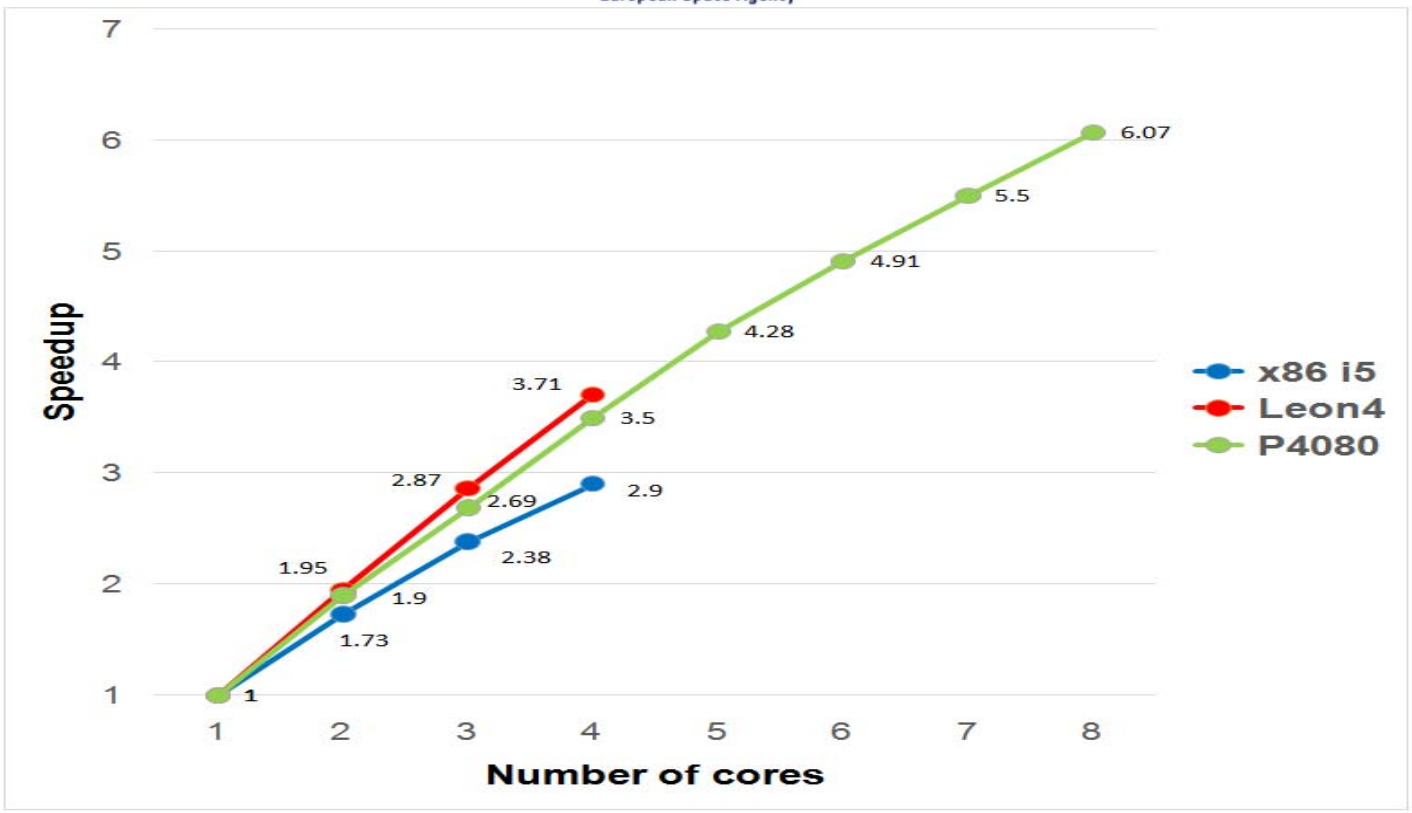

Figure 9 - Overall performance improvement vs. number of cores used

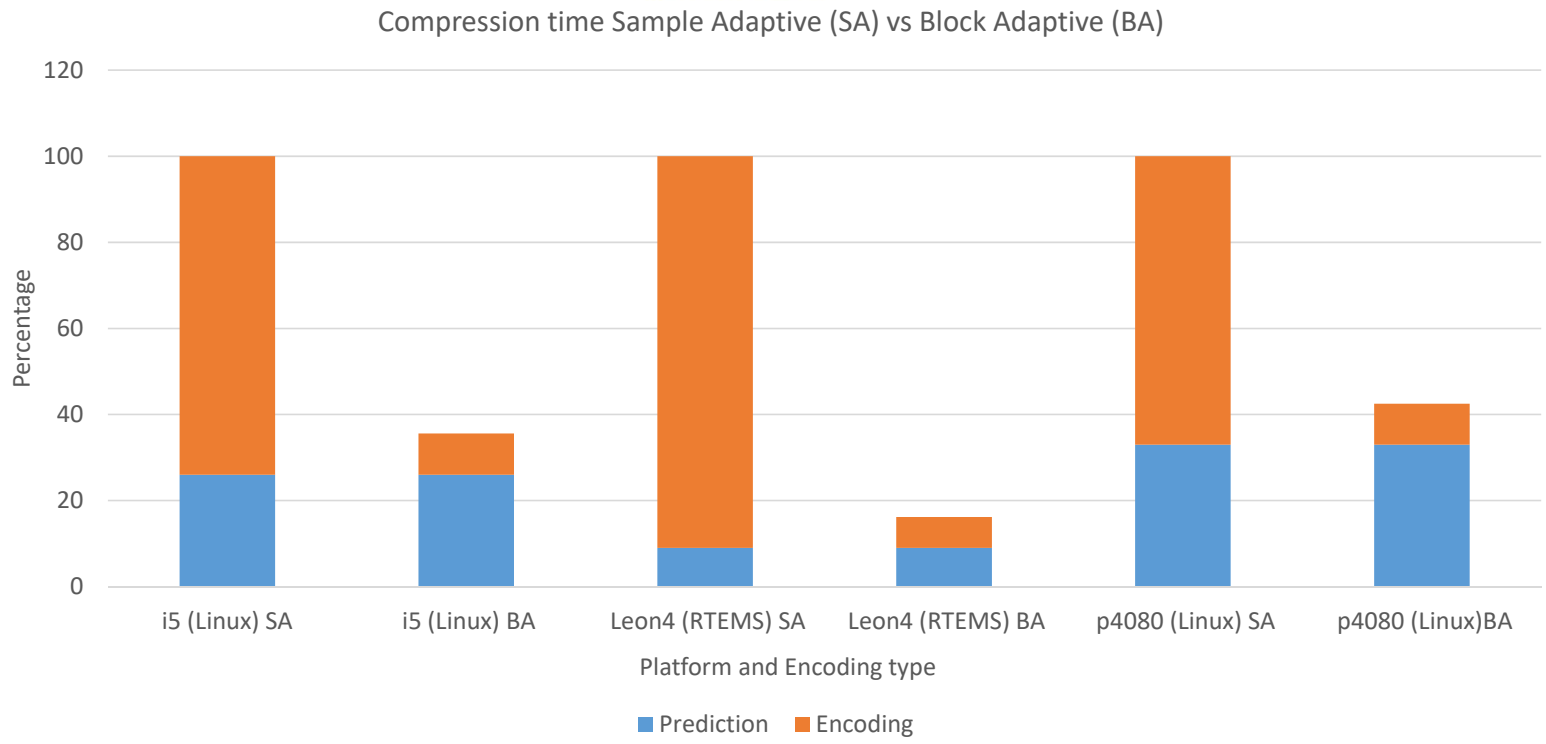

Figure 10 - Time used for prediction vs. encoding

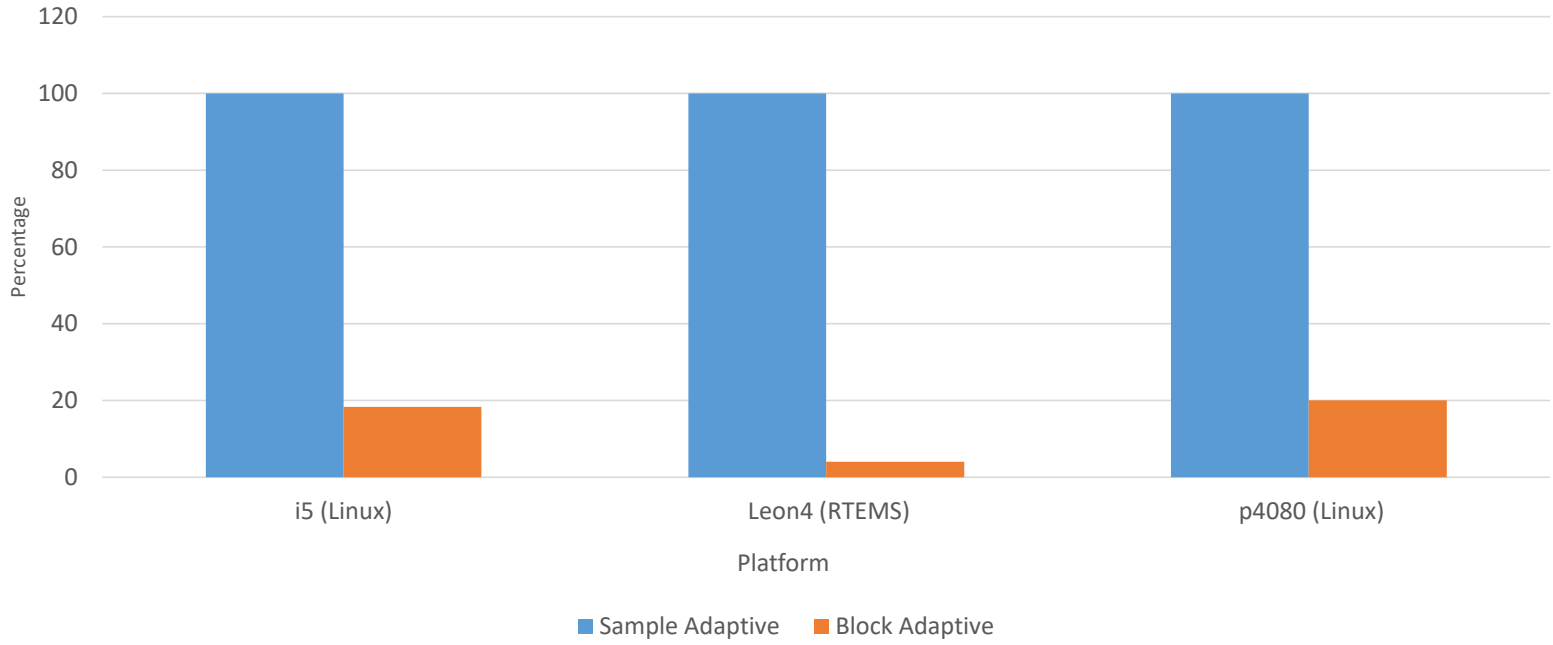

Figure 11 – Encoding time depending on encoding type

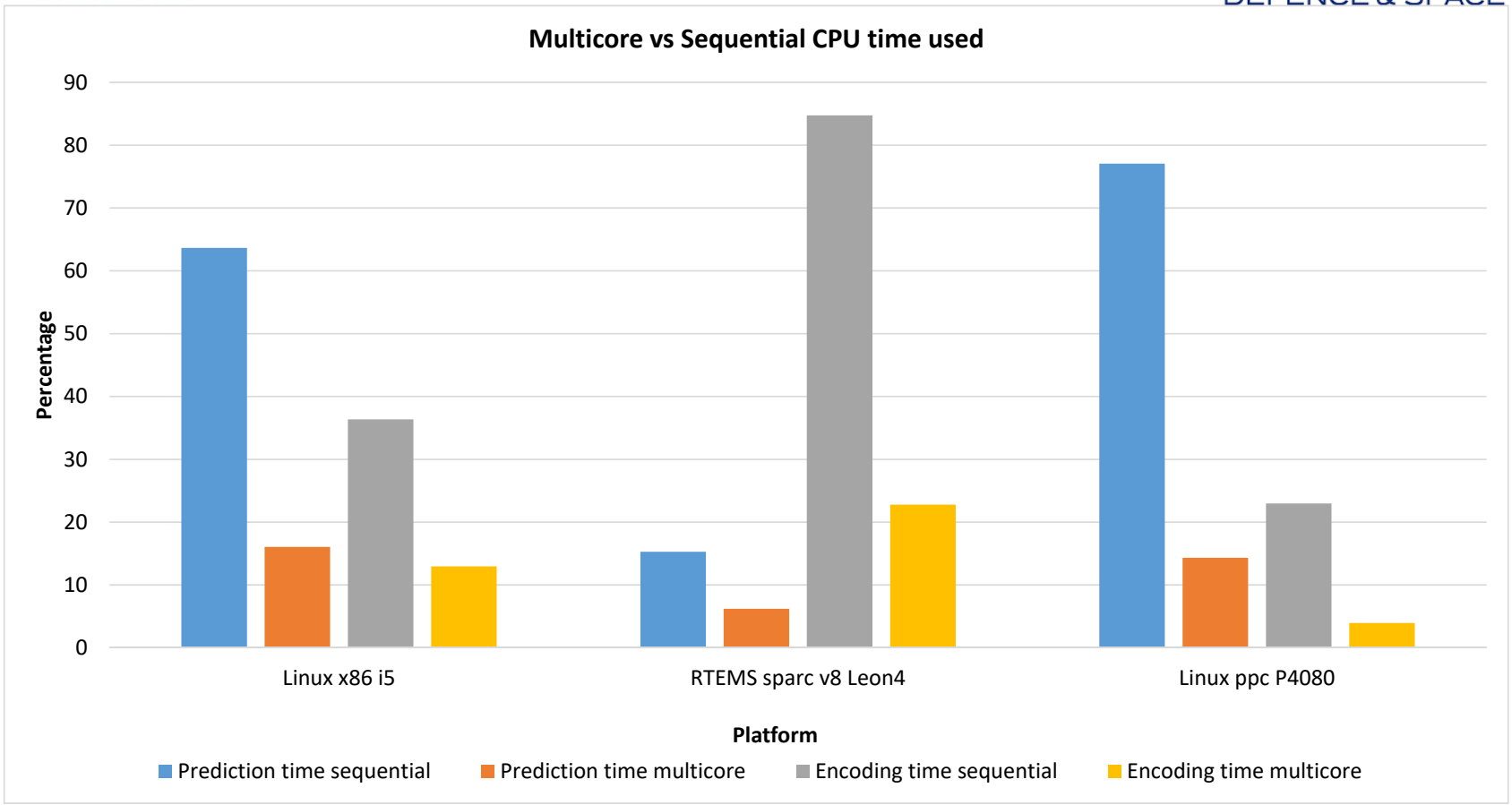

Figure 12 – Multicore vs Sequential CPU time used.

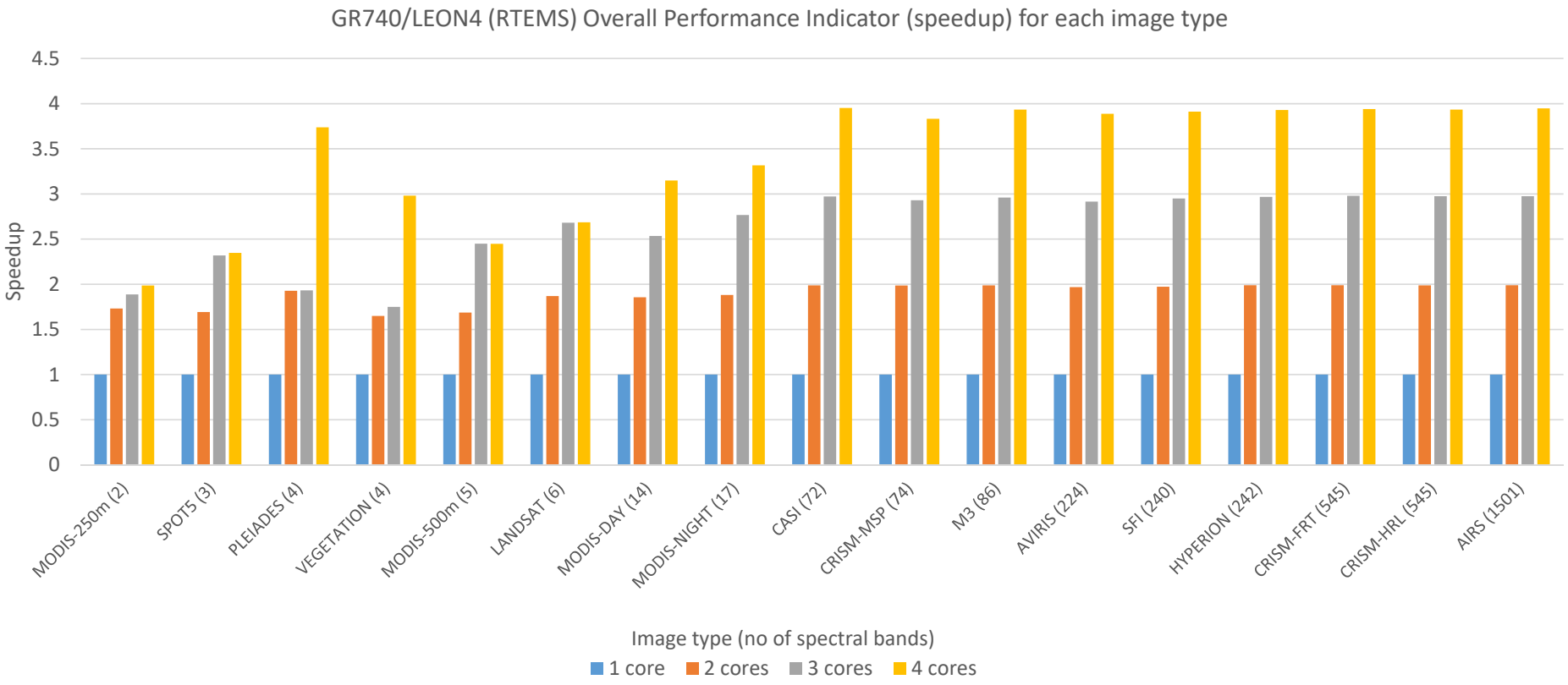

Figure 13 – Performance indicator on RTEMS Leon 4 for each image type

ENEA

esa
European Space Agency

AIRBUS
DEFENCE & SPACE