Multispectral and hyperspectral compressor CCSDS-123 model with Intel® CoFluent™ Studio

Ana Gómez, Sebastián López, Roberto Sarmiento, Lucana Santos
Institute for Applied Microelectronics (IUMA)
Division of Integrated Systems Design (DSI)
University of Las Palmas de Gran Canaria (ULPGC), Spain

***Abstract*—This paper presents an approach to the modelling of the current standard algorithm for on-board hyperspectral and multispectral data compression (CCSDS-123) using the Intel® CoFluent™ Studio. The obtained results are useful to identify avoidable data dependencies in the reference software implementation and the impact of some parameters in performance.**

Keywords - Intel® CoFluent™ Studio; CCSDS Standard hyperspectral compression, ESL.

I. INTRODUCTION (HEADING 1)

Remote sensing multispectral and hyperspectral imaging has become increasingly important due to its multiple applications and the introduction of new high-resolution sensors, able to collect a huge amount of data that need to be compressed. Among the many available algorithms for compression, the CCSDS-123 Recommendation for lossless Multispectral and Hyperspectral data compression [1] is currently the reference for the upcoming space missions.

In this work, we present an approach to model the CCSDS-123 algorithm using Intel® CoFluent™ Studio tool, with the aim of providing an initial model, which once modified could provide in the future the right specifications that will make it possible to design an FPGA implementation of the algorithm which achieves the throughput and occupancy desired by a potential user. The Intel® CoFluent™ Studio tool [3] offers a visual model-driven development solution, which enables for performance prediction and design space exploration of electronic systems..

II. OVERVIEW OF THE CCSDS-123 ALGORITHM

The CCSDS-132 describes the compressor as a two-part functional system: prediction and entropy coder. It offers two options for the entropy coding stage: the sample-adaptive entropy coding and the block-adaptive entropy coding, which corresponds to the specifications of the CCSDS-121 standard [2]. A short overview of the algorithm is presented next.

A. Prediction

First a *local sum* $\sigma_{z,y,x}$ of the neighbouring sample values is computed by one of two possible configurations: *column-oriented*, using the neighbor on top of the current sample, or *neighbour-oriented*, using 4 neighbouring samples.

Local sums are used to calculate the *central local differences* values $d_{z,y,x}$ and the *directional local differences* $d^{N}_{z,y,x}$, $d^{W}_{z,y,x}$ and $d^{NW}_{z,y,x}$ according to the user's selections to perform *full* or *reduced* prediction. After obtaining the predicted sample, $\hat{s}_{z,y,x}$, the prediction residuals are calculated and mapped to positive integer values.

B. Coding

The mapped prediction residuals $\delta_{z,y,x}$, are sequentially encoded in the order selected by the user: band-sequential (BSQ) or band-interleaved (BI). This encoding order specifies likewise the order in which the encoded samples are arranged in the compressed file.

Under the sample-adaptive entropy coding approach, each mapped prediction residual is encoded using a Golomb power-of-two variable-length binary codeword, based on adaptive code selection statistics such an *accumulator* and a *counter*.

The block-adaptive entropy coder utilizes the Rice coder defined in the CCSDS-121 standard, where coding is applied to a block of J consecutive pre-processed samples.

III. DESIGN METHODOLOGY

A reference software implementation developed by the European Space Agency (ESA), which is available in [4], was used as basis for this work. The reference code has been split into several block functions, which represent the main stages of the compression, in order to develop the application model. The functions obtained from the CCSDS-123 reference software are located inside each block, which have been modified in order to adapt the C code to the CoFluent ™ Studio environment. The different parts of the designed application model are detailed below:

- **Read Parameters block** is in charge of storing all the encoder input parameters.
- **Predictor block** aims at calculating the image residuals, through three different functions: *readSamples* function stores the image samples; *localDifferences* function computes local differences used during the prediction process; and *weights* function, where the prediction residuals of the samples are calculated.
- **Encoder block** is composed by a header generator function and the entropy coding function, with either the sample-adaptive or the block-adaptive encoding option.

For each block, different components are placed in to achieve the desired behavior. Functions might read or write in shared variables and might activate or receive event signals; these events guarantee the synchronization between different

functions, ensuring that shared variables are only written and read at the appropriate time. Functions are sequentially executed by loops, which depend on a simulation value (parameter *Iterations*). Fig. 1 shows, as example, the *readParameters* and *predictor* blocks implemented in CoFluent™ Studio.

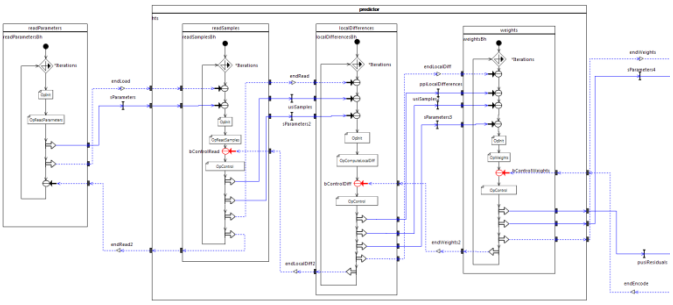

Fig. 1: Application model readParameters and predictor block

Intel® CoFluent™ Studio offers different simulation charts, as the one shown in Fig. 2, where transactions between different functions can be analyzed, through visualization of function's activity, shared variables and events. The application model has been verified through these kind of charts and the resulting compressed images in comparison with the resulting images from the reference software.

Fig. 2: Application model *timeline* simulation chart

IV. RESULTS

Performance in terms of throughput has been analyzed, as shown in Fig. 3, for both encoding options.

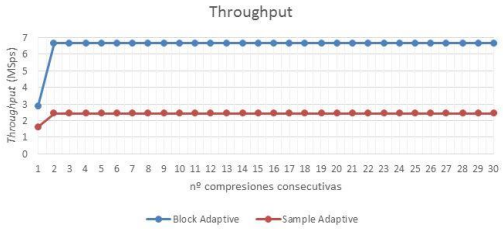

Fig. 3: Throughput of application model

Different simulations have been executed for the application model, including platforms considerations shown in Fig. 4, while varying different application and design parameters.

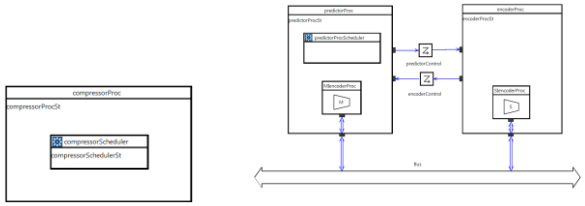

Fig. 4: *simplePlatform* and *combinedSWHWPlatform*

From the results obtained from the simulations, a final model has been developed, where prediction and coding work at sample level, through a residual queue as highlighted in Fig. 5, obtaining a higher throughput as illustrated in Fig. 6.

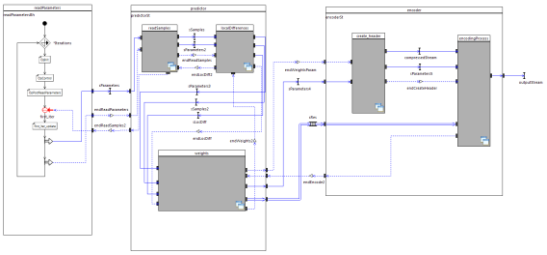

Fig. 5: Improved application model

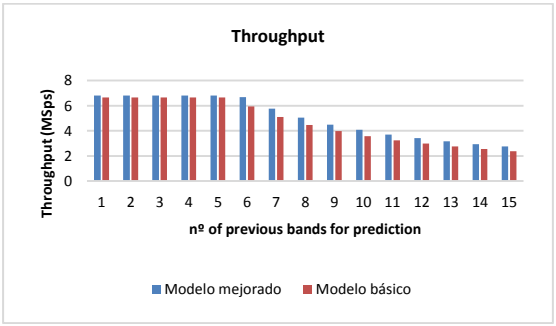

Fig. 6: Throughput comparison between improved and initial application models

V. CONCLUSIONS

A functional model of the CCSDS-123 standard compressor has been implemented with CoFluent™ Studio software tool. This model has been verified through a reference images set tested with the reference software results. The model has been analyzed in order to obtain influence from different parameters. Finally, an improved model has been implemented able to obtain a higher *throughput* than the original.

REFERENCES

[1] Lossless Multispectral & Hyperspectral Image Compression. Recommendation for Space Data System Standards, CCSDS 123.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 2012.

[2] Lossless Data Compression. Recommendation for Space Data System Standards, CCSDS 121.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, May 2012.

[3] IntelCoFluentStudio_v5.2_Reference_Guide, Intel Corporation.

[4] ESA Data Compression Tools, available, 2015: http://www.esa.int/Our_Activities/Space_Engineering_Technology/Onboard_Data_Processing/Data_compression_tool