EUROPEAN WORKSHOP ON ON-BOARD DATA PROCESING (OBDP2021), 14-17 JUNE 2021

EVALUATION OF NEW GENERATION RAD-HARD MANY-CORE ARCHITECTURE FOR SATELLITE PAYLOAD APPLICATIONS

Patricia López Cueva[(1)], Remi Barrère[(2)], Kevin Eyssartier[(3)], Mickaël Bruno[(4)], Clement Coggiola[(5)]

[(1)]*Thales Alenia Space, France, Email:patricia.lopezcueva@thalesaleniaspace.com*
[(2)]*Thales Research and Technology, France, Email: remi.barrere@thalesgroup.com*
[(3)]*Thales Research and Technology, France, Email: kevin.eyssartier@thalesgroup.com*
[(4)]*CNES – French Space Agency, France, Email: mickael.bruno@cnes.fr*
[(5)]*CNES – French Space Agency, France, Email: clement.coggiola@cnes.fr*

ABSTRACT

Computing power requirements of space applications, in particular for payload functions, are constantly rising with the objective of achieving higher spatial, spectral, temporal and radiometric resolution. RC64 is a many-DSP rad-hard processor that offers processing performance close to FPGAs or specialised ASIC but with the flexibility of software programming and low power consumption.
This paper presents the results of a CNES study aimed at evaluating RC64 component for satellite payload applications by using as an example an implementation of the CCSDS 123.0-B-2 "Low-Complexity Lossless and Near-Lossless Multispectral and Hyperspectral Image Compression" standard [4].

1. INTRODUCTION

Space applications, more specifically payload functions, are requiring increasing amount of computing power, the aim being to increase the number of processing operations that can be carried out on board to achieve higher spatial, spectral, temporal and radiometric resolution. Moreover, next generation projects are seeking a breakthrough by embedding innovative processing designs to gain efficiency, flexibility and autonomy, putting more emphasis on the necessity to have powerful processing components, and being usable for deep space missions.

Indeed, COTS components that provide the necessary performance while being tolerant to radiation are today available for missions with low to medium requirements on radiation tolerance, such as *Kalray's MPPA®* Coolidge™ or Xilinx ACAP Versal. However, these components are not suitable for deep space missions. On the other hand, very few radiation-hardened components exist today that offer the needed performance. The RC64 is a good example of this kind of components.

The RC64 component from Ramon.Space is a many-core radiation-hardened processor, built around DSPs [1], and designed for spatial applications requiring high computing power and very high speed processing [2][3]. This component has a massively parallel architecture that offers processing performances close to FPGAs or specialised ASICs, but also provides the flexibility of software programming and low power consumption. However, its architecture requires rethinking the algorithms in order to make the most of the RC64 capabilities.

Last few years, CNES has been working on the evaluation of this component for different space applications. As part of this effort, a new study was launched to evaluate the suitability of using the RC64 component with high complex algorithms.

For this study, the selected algorithm is the CCSDS 123.0-B-2 standard, which introduces, among other new features, *near-lossless* compression. However, the difficulty is that this standard has been designed in a sequential way, which means that the execution model has to be changed to parallel: this is a required step before any attempt at efficient implementation on a many-core.

2. RC64 ARCHITECTURE AND PROGRAMMING MODEL

RC64 is composed of 64 DSP CEVA-X1643 cores [1] that share a 4MB memory as shown in Figure 1. Each core has also a private 8KB DTCM memory, as well as 8KB instruction and data caches.

Its programming model is task oriented with the particularity of depending on a hardware scheduler for the execution of the defined tasks. In order to program the RC64, the developer defines a series of small sequential tasks and a dependency graph that describes the dependencies between the tasks. This approach seeks to provide a simpler and more effective way of programming a massively parallel architecture such as the RC64.

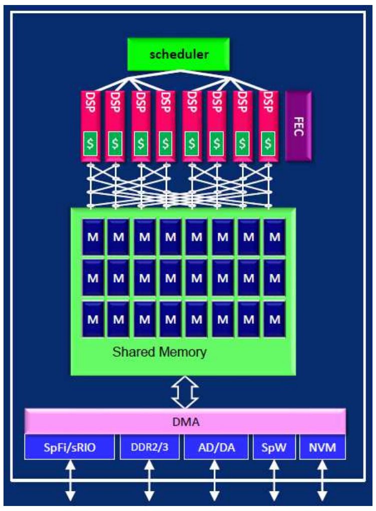

Figure 1 : RC64 Architecture

Concerning inputs/outputs, RC64 offers SpaceWire (SpW) and SpaceFibre (SpFi) interfaces to connect the RC64 with other systems. Moreover, it contains a DDR memory necessary to store large data buffers due to the limited amount of shared memory. All accesses to external interfaces and DDR are managed by DMA requests to transfer data to/from shared memory, as internal processors have no direct access to DDR.

3. CCSDS 123.0-B-2 ARCHITECTURE

The CCSDS 123.0-B-2 low-complexity lossless and near-lossless multispectral and hyperspectral image compression is a data compressor for spatial imagers.

The compressor is composed of two layers: predictor and encoder (see Figure 2). The predictor estimates the value of current image sample based on the values of nearby samples in a small three-dimensional neighbourhood, and then sends to the entropy coder the quantized residual difference between the estimated value and the current sample value projected on a positive interval. The encoder then encodes the mapped quantizer indices $\delta_z(t)$ in order to limit the size of the compressed image.

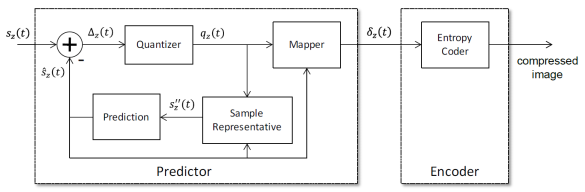

Figure 2 : Compressor Architecture

The input of the compressor is a three-dimensional image composed of an array of integer values. This array is composed of $s_{z,y,x}$ samples organised in N_x columns, N_y lines and N_z bands. The compressed output is an encoded bitstream from which the input can be reconstructed, exactly or approximately. This optional loss of information takes place in the quantizer.

4. CCSDS 123.0-B-2 PARALLELISATION STUDY

The prediction calculation of $\hat{s}_z(t)$ is performed using previously computed nearby sample representatives in the current band and the *P* preceding spectral bands. The user-specified parameter *P* is an integer in the range [0; 15], which determines the number of bands used for prediction. This parameter is important for memory and computation usage on the RC64, as it requires to store previously computed *s”(t)* of the *P* previous bands and the bands to be computed in order.

The algorithm can be configured in four different types of neighbourhood: neighbour-oriented mode uses sample representative values from the lateral columns whereas the column-oriented mode is only using the current column, wide mode uses sample representative values from the current line whereas the narrow mode only uses values from previous line. This spatial dependency requires that the lines in the same band shall be computed in order.

In this study, we propose three parallelisation methods, which could be applicable for any parallel architecture: sub-image splitting, sequential parallelisation and interleaved parallelisation. The sub-image splitting consists in splitting the input in multiple samples which can be compressed in parallel, which eliminates inter sample dependencies. The sequential parallelisation splits the compute load into bands; each thread can process a band if the current line has been computed in the *P* previous bands. The interleaved parallelisation splits the compute load into lines and each thread advances through bands. These last two parallelisation modes require synchronisation points at the end of each line.

5. BENCHMARK RESULTS

To better anticipate our implementation choices, the RC64 is first benchmarked, the aim being to determine the best practices and make the best of this platform. The DDR bandwidth, computation speed and hardware scheduler are evaluated to this end.

Table 1: DDR performance in MB/s

Buffer size in bytes		***4k***		***64k***		***128k***		***256k***	
	CDA[1]	3G	12G	3G	12G	3G	12G	3G	12G
Core frequency	**Rx/Tx**								
40Mhz	Tx	24	-	210	-	280	-	338	-
	Rx	25	-	210	-	280	-	336	-
137Mhz	Tx	72	89	298	714	333	955	353	1159
	Rx	72	84	297	717	332	957	353	1151

DDR performance is measured by transferring buffers of different sizes in different configurations. The measured time includes all overheads, from the DMA call to the callback entry point (activated when the DMA transfer finishes). The results, presented in Table 1 show a much better performance when using large buffers instead of small buffers, in the order of fifteen time faster, which indicates that it is more efficient to use large buffers in the implementation when possible.

Computation benchmark compares convolution time between 16bits, 32bits and 64bits inputs. DSP optimisation through intrinsics is also measured. On average, 32bits and 64bits computations are 2.3 and 9.3 times slower respectively than 16bits computations, and the use of intrinsics –limited to 16bits - gives a speedup of 2.7.

The efficiency of the hardware scheduler is measured considering software events, regular task terminations and duplicable task terminations. The software event is an API call that might trigger the activation of a task and it takes on average 227 cycles between the API call and the activation of the task. Regular task termination is the scheduling event triggered by the return of a task and it takes on average 189 cycles to be performed. The duplicable task termination is the scheduling event triggered when all the duplicated tasks return, and it takes on average 462 cycles to synchronise the cores and to activate the next task. These low values confirm that the parallelisation is possible despite its many required synchronisation barriers.

6. CCSDS 123.0-B-2 IMPLEMENTATION ON RC64

In order to minimise data dependencies and so to allow a better efficiency, the selected algorithm is implemented with the following limitations: only wide sum mode, only sample adaptive coder, only sequential order mode.

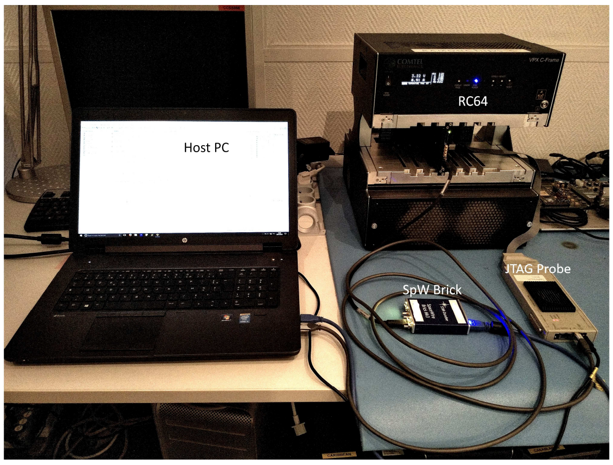

Figure 3 : Test Setup

Figure 3 shows the test set-up where the RC64 card is installed on a VPX rack and accessed via JTAG from the Host PC. Moreover, the Host PC is connected to the card via SpW to be able to have console output on the Host PC and exchange data files with RC64.

Figure 4 shows the activity diagram of the algorithm once ported to RC64 platform. All files are transmitted between the host PC and the RC64 DDR via SpW. Then, only data corresponding to bands that can be processed is loaded from DDR to shared memory via DMA. The prediction and encoding of samples of the loaded bands is then parallelised. Once a band has been treated, it can be stored in DDR memory and this write operation is done in parallel to sample treatment of other bands as an optimisation.

The encoded bands are not written in memory in a sequential way and it is impossible to predict the size of an encoded band in advance. Therefore, all bands are written in DDR memory leaving enough space so that no data of other bands is overwritten.

Once all bands are processed, the unnecessary padding introduced during the parallel loop is then removed in order to obtain the final compressed image file.

[1] DMA Arbitration Configuration

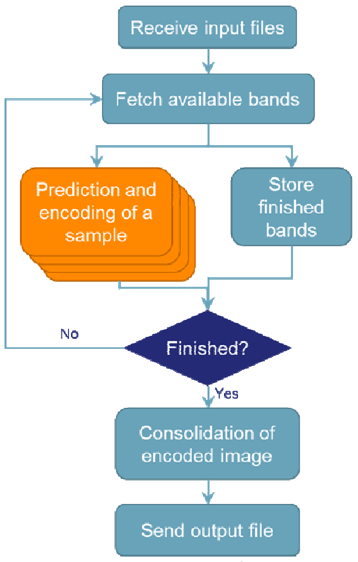

Figure 4 : Activity Diagram of CCSDS 123.0-B-2 algorithm

6.1. Possible Optimisations

During the parallelisation study and the implementation on target, several optimisations have been identified that could improve the performance of the current implementation. The identified optimisations have not all been implemented yet due to lack of time and budget on the current study.

Input double buffering
The goal of double buffering is to hide memory transfer time behind computation time. This is achieved by loading the next required inputs during the current inputs computations. This could bring an important gain, especially when the load image step takes a substantial proportion of the time, as is the case with AIRS image when using 32 or 63 threads due to the low number of lines on each band. On the other hand, this technique raises memory consumption by doubling input buffers so a good balance needs to be found.

Modification of data width
Predictor internal registers have a resolution of R in the range $\max\{32,D+\Omega+2\} \leq R \leq 64$ with D the resolution of the input and Ω the resolution of the weight. This ensures that no overflow occurs. The current implementation uses the maximum range of R=64 in order to stay compatible with all configurations. As the benchmark in Section 5 shows that the resolution has a significant impact on the computation performances, lowering precision from 64 bits to 16 bits could drastically reduce predictor latency, to the detriment of the numerical stability. This use of a 16-bit precision would also allow using intrinsic functions, which would offer a much higher performance, up to double, on some parts of the algorithm.

The drawback is that the algorithm precision needs to be lowered in order to avoid overflows, which introduces a quality impact on the image after decompression.

R also drives the precision of sample representatives and can help saving internal memory used by the predictor.

Partial line for each task
Currently, each duplicable task instance processes the data of a full line, which limits the possible input size as well as the number of cores that can be run in parallel. This is due to the limited size of the shared memory, which is the main drawback of RC64 architecture for this algorithm.

By splitting data lines into multiple blocks, the global memory usage can be lowered while still meeting the data dependency requirements. Due to the fact that the algorithm is parallelised in spectral bands, images with low number of spectral bands could benefit of a higher parallelism with this optimisation.

A disadvantage of this method is that it creates a dependency between blocks that does not exist when using full lines. It can be solved either by keeping this data in memory, which generates more data dependency and complexity, or by using the reduced prediction mode and the column-oriented local sums, which would need to be implemented.

Another drawback is the usage of smaller buffers for DMA transfers to and from DDR. As shown in Section 5, the buffer size has a significant effect on the bandwidth of the DDR: transferring small data set is inefficient.

7. PERFORMANCE RESULTS

The characteristics of the AIRS sample are N_x=90, N_y=135 and N_z=1501: the important number of bands allows the full usage of the RC64 cores. Encoding is parallelised onto 63 threads to free the last core for memory transfers, synchronisation issues can be observed otherwise. Different values of the parameter P, which represents the number of spectral bands used for the prediction, are used: a higher P implies a higher computation intensity, a higher memory footprint during the encoding step and a higher compression rate. The AIRS sample has a size of 34MB of 16bits sequential inputs. The compressed bitstream sizes are 14MB, 9.4MB, 9.1MB, 9.0MB and 9.0MB for P at 0, 1, 4, 8 and 15 respectively.

Using higher number of threads results in a drastic improvement, mainly observed for the prediction and encoding step. This speedup is proportional to the number of threads: this confirms that the overheads in the prediction/encoding steps are not significant, and also that threads do not interfere with each other. The parameter P has no impact on the speedup achieved when using higher number of threads.

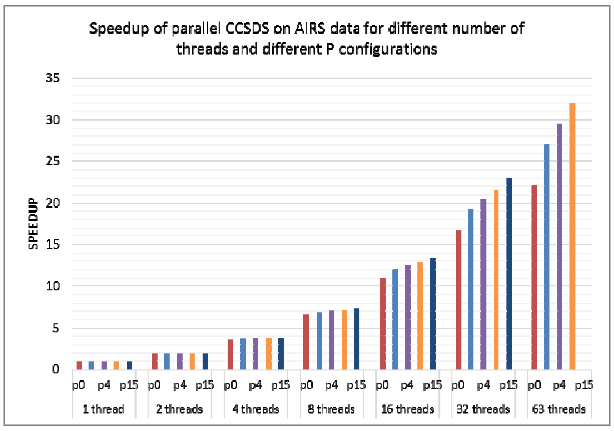

Figure 5 : Speedup Results on AIRS samples

Bandwidth measured during the loading image step is between 3.2 and 4MB/s, much less than the best case achieved in the benchmark in section 5**,** but load image and consolidation stages latency becomes significant only when using 63 threads. Thus, future performance optimisations should address these two steps.

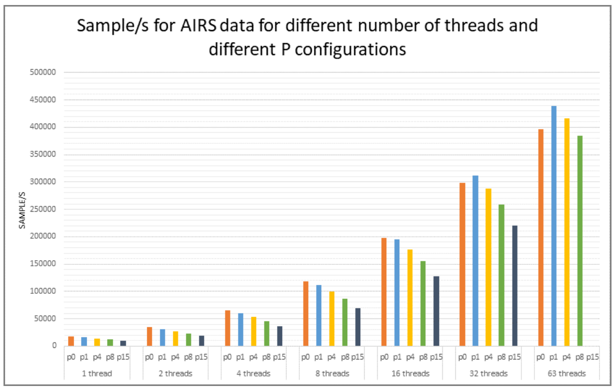

Figure 6 : Samples/s results on AIRS samples

Pleiades sample is of dimensions N_x=296, N_y=2448 and N_z=4. Pleiades sample is only tested with P=0 because of memory consumption: indeed, the N_x*N_y samples needed for processing dependencies cannot fit in internal shared memory for bigger values of P. In addition, it is not possible to use more than four threads with the implemented parallelisation, which means that 93% of the compute power of the RC64 is out of reach because of the low number of bands N_z.

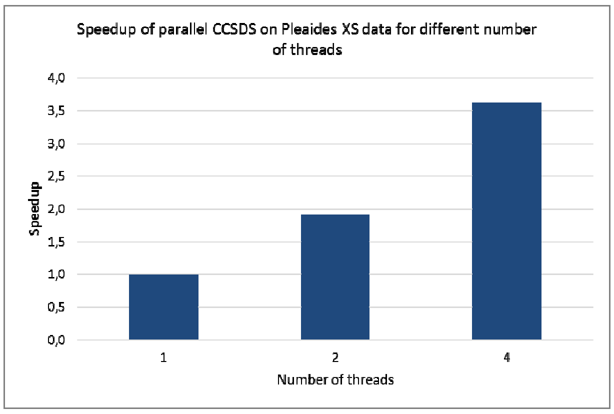

Figure 7 : Performance Results on Pleiades sample

As already noticed with the AIRS sample, a linear speedup can be observed when there are no interferences between threads. But the four threads parallelisation cannot achieve the same level of performance as with the AIRS sample: prediction and encoding step can still bring significant gains on its own. The interleaved parallelisation, proposed during the parallelisation study, would allow a higher number of threads in this case and should lead to a better performance.

8. EVALUATION RESULTS

It is important to note that the implementation of the CCSDS 123.0-B-2 standard on the RC64 platform was a porting of an existing algorithm. This algorithm was initially designed in a sequential way, with the main objective of being fully compliant of the standard, i.e. supporting all modes and data sizes. As already presented, this approach is not the most effective way to achieve good performances on a manycore such as RC64.

Another point concerns the programming model of the RC64, which is different from a typical threading model: tasks are supposed to be much smaller than typical threads, and scheduling and synchronisation between tasks does not use the same mechanisms. This new programming model is very interesting and is intended to facilitate the user experience, but a learning curve phase has to be considered when introducing this platform on a system. As with any major change, using this new programming model requires to redesign an existing algorithm before porting it to the RC64.

On the other hand, the performance results show the high potential of the RC64 for on-board data processing algorithms. As an example, the speed-up and performance obtained on the AIRS sample, with a maximum speed-up of 58.35 and throughput of 0.45 Msamples/s, are significant, taking into account that there is still room for further optimisations.

Nevertheless, we identified a few improvements, although not mandatory, that could make the usage of the RC64 platform more convenient to use.

First, a more detailed documentation explaining how to program efficiently the RC64 would be of great value: if the documentation provides main guidelines, some “tips and tricks” could be very helpful.

Another important aspect to consider is the tracing system: currently only software execution traces are supported by the platform, which in some cases can be intrusive. Hardware data and instruction traces would allow a finer analysis and then a better understanding of the application execution on the platform. Another point related to the preceding one, access to the DDR via JTAG probe would be of great help when debugging applications. Otherwise, the user has to print the data while reading or writing it into DDR via DMA transfers and he never has a complete view of the memory content at a given moment.

Last point, direct DMA transfers between I/Os (SpW, SpFi) and DDR would allow to save shared memory for the applications, as no deallocation of memory is supported.

9. CONCLUSIONS

Regardless of the possible improvements mentioned in the previous section, it is clear the interest of using the RC64 platform for on-board data processing.

The performance that it offers is substantial and at a level that is unseen anywhere until now in the area of radiation-hardened platforms. It certainly opens the door to increased on-board data processing capabilities of deep space missions, with the consequent increase on flexibility on the type and amount of applications that could be embedded.

Even if this study has been focused on data compression, other types of applications could benefit from this high performance platform, such as image processing or even artificial intelligence.

Regarding the CCSDS 123.0-B-2 standard, the parallelisation of the algorithm has shown that the reference implementation is too generic to be efficiently implemented on-board while keeping good performances on all possible modes.

For its implementation on a mission, a study of the dimensions of the images to be processed, the needed compression rate, and the resolution needs to be carried out in order to be able to specialise the implementation for the given mission.

REFERENCESS

[1] Gellis, H et al., “CEVA DSP processors for MacSpace”, MacSpace Symposium and Summer Seminar, 2016

[2] Ginosar R. et al., "RC64: High performance rad-hard manycore", *2016 IEEE Aerospace Conference*, 2016

[3] Ginosar R et al., “RC64 Architecture”, MacSpace Symposium and Summer Seminar, 2016

[4] Low-Complexity Lossless and Near-Lossless Multispectral & Hyperspectral Data Compression, Recommended Standards. CCSDS-123.0-B-2. Blue Book, 2019