FAPEC in an FPGA: a simple low-power solution for data compression in space

Alberto G. Villafranca*[a,b,c], Shan Mignot[d], Jordi Portell[a,e], & Enrique García-Berro[a,b]

[a]Institut d'Estudis Espacials de Catalunya,
c/ Gran Capità 2–4, 08034 Barcelona, Spain
[b]Departament de Física Aplicada, Universitat Politècnica de Catalunya,
c/ Esteve Terrades 5, 08860 Castelldefels, Spain
[c]PCOT, Institut Cartogràfic de Catalunya,
Parc de Montjuïc, 08038, Barcelona, Spain
[d]Observatoire de Paris,
5 Place Jules de Janssen, 92190, Meudon, France
[e]Departament d'Astronomia i Meteorologia, Universitat de Barcelona,
c/ Martí i Franquès s/n, 08028 Barcelona, Spain

ABSTRACT

Future space missions are based on a new generation of instruments. These missions find a serious constraint in the telemetry system, which cannot download to ground the large volume of data generated. Hence, data compression algorithms are often mandatory in space, despite the modest processing power usually available on-board. We present here a compact solution implemented in hardware for such missions. FAPEC is a lossless compressor which typically can outperform the CCSDS 121.0 recommendation on realistic data sets. With efficiencies higher than 90% of the Shannon limit in most cases – even in presence of noise or outliers – FAPEC has been successfully validated in its software version as a robust low-complexity alternative to the recommendation. This work describes the FAPEC implementation on an FPGA, targeting the space-qualified Actel RTAX family. We prove that FAPEC is hardware-friendly and that it does not require external memory. We also assess the correct operation of the prototype for an initial throughput of 32 Mbits/s with very low power consumption (about 20 mW). Finally, we discuss further potential applications of FAPEC, and we set the basis for the improvements that will boost FAPEC performance beyond the 100 Mbit/s level.

Keywords: Lossless compression, Adaptive compression, PEC, FAPEC, FPGA, CCSDS 121.0

1. INTRODUCTION

Space missions cover a large range of strategies and instrumentation approaches. In the case of scientific missions, it is desirable to retrieve the maximum amount of data from a given experiment and, thus, the instruments are designed accordingly. The evolution of instrumentation technology is constantly pushing the telemetry resources of space missions to the limit. Consequently, despite having faster data links, telemetry bandwidth continues to be a limiting factor for most missions. One of the options to mitigate this effect is data compression.

The space environment presents a set of characteristic restrictions which are rare in other cases. Due to the nature of the communications channel [1], the block size of the compressor is limited to a length of at most a few kilobytes. In this way the loss of information in case a transmission error occurs is minimized. Unfortunately, adaptive compressors usually require a large amount of data to perform optimally. Furthermore, if data from several instruments are time multiplexed, non-uniformities will most probably arise in the data stream which will decrease the compression ratio. Finally, the processing power is extremely limited, and low-complexity algorithms are thus desirable. Therefore, standard compression solutions applied in most on ground systems are not feasible in space.

* agonzalez@ieec.cat

The Consultative Committee for Space Data Systems (CCSDS) issued a proposal – CCSDS 121.0 – aimed to generic lossless data compression [2]. Since its release in 1993 this solution has been widely adopted. It has been employed in several missions during the last years [3], using both software and hardware implementations [4]. The success of this proposal lies on its simplicity, using the Rice coder as the basis for the compression strategy [5]. Basically, the compression follows a two-stage strategy. Firstly, there is a pre-processing stage which modifies the statistics of the input data through a reversible process. The goal of this stage is simply to remove data redundancy in order to improve the performance of the second stage. Subsequently, the coding stage applies to data blocks of either 8 or 16 samples a coding method which consists on Rice codes in most cases – which belong to the family of variable length codes (VLC) [6]. Rice codes can be easily implemented in both software and hardware owing to their simplicity, as they are calculated with a few binary operations.

Despite the satisfactory results of this strategy, the CCSDS 121.0 Lossless Data Compression recommendation is not exempt of problems either. The main issue arises at the coding stage, because the Rice coder does not behave optimally when compressing data with statistics deviating from geometric-like distribution profiles [7]. Despite being a typical profile for instrumental data, there are many cases where this distribution does not fully apply. Examples include outliers in the measurements, a pre-processing stage which does not perform as expected (not predicting the samples adequately), or simply data following different statistics – such as Gaussian or Gamma distributions. In particular, although the small block size of the recommendation provides a good adaptation level, the efficiency significantly decreases with increasing levels of noise or outliers in the data. This makes the CCSDS 121.0 solution significantly under-optimal for some missions, especially considering that space-based measurements are often contaminated due to prompt particle events (PPEs). These PPEs can account for up to 10% of the total cases in certain cases [8].

Within the frame of a compression study for the Gaia mission [9] a new coding method was developed to overcome these problems. The result was a VLC called PEC (Prediction Error Coder), a new entropy coding algorithm specially devised for space [7]. Specifically, PEC is capable of dealing with prediction errors contaminated with outliers, presenting a reduced affectation in the compression ratio. Compared to Rice, such outliers lead to a very small increase in the size of the output code. Therefore, PEC improves the compression performance of the Rice coder while still presenting a low computational cost. Both PEC and Rice must be calibrated to the expected statistics of the data, i. e., they are static coders. Nonetheless, as Rice is wrapped within the CCSDS 121.0 to become adaptive, an adaptive version of PEC – FAPEC or Fully Adaptive PEC – was also developed [7]. The adaptive stage automatically selects the most suitable calibration, thus adapting to changes in the data distribution and becoming a completely autonomous coder. The tests performed on the FAPEC coder showed that it is a very simple and robust alternative to the CCSDS recommendation for noisy environments. On the other hand, a software implementation is not always the best solution in space, since dedicated hardware is more efficient in terms of power consumption and usually provides a higher throughput. Thus, in some cases a hardware implementation is preferred over the more flexible software solution. Consequently, FAPEC also needed a hardware implementation in order to offer a complete solution. The aim of this paper is, precisely, to fill this gap. The paper is structured as follows. In §2 we briefly introduce the operation of FAPEC and the changes implemented in its hardware adaptation. Section 3 presents the selected test platform and in §4 we describe the structure of the FPGA implementation. In §5 we explain the validation procedure, while in §6 the results obtained with the prototype are presented. Forthcoming work is analyzed in §7 and, finally, in §8 we summarize our results and elaborate our conclusions.

2. THE ALGORITHM

2.1 Description of PEC

PEC combines three coding strategies into one single VLC family and uses 4 different coding segments for each strategy [7]. The best strategy is chosen depending on the data statistics. Each segment is associated to a range of values which is defined by the segment size. In this way, the maximum length of a PEC code can be maintained within a reasonable size – not much larger than the original symbol size (unlike the Rice codes), while the first segments still allow a good compression ratio for small values. An automated software calibration process calculates the optimal coding table, which consists of 4 parameters defining the strategy chosen as well as the size of each of the 4 segments. Although a training dataset is required to calibrate PEC, its robustness provides a high resilience when confronted to changing data statistics or not well-fitted datasets. Thus, PEC is less sensitive to variations of the data statistics than the Rice coder. In a way, PEC can be considered a semi-adaptive entropy coder thanks to its multi-segment strategy.

Consequently, if the statistics of the data to be compressed are relatively constant, an algorithm like PEC can be enough as a coding stage. Otherwise, an adaptive algorithm is highly recommended.

2.2 Description of FAPEC

FAPEC essentially implements an adaptive stage on the top of the PEC coder. It automatically selects the coding table for each data block in a highly optimized manner [7]. This selection is performed through the accumulation of probabilities in a logarithmic-like histogram. FAPEC presents decreasing sensitivity for increasing values. After finishing with an input block, the histogram is analyzed and the coding table which better suits the probability distribution is selected. Finally, the PEC coder is applied and the resulting codes are output. Obviously, the calibration procedure performed by FAPEC is much quicker than the exhaustive calibration software developed for PEC, although it provides a nearly optimal configuration. Another important feature of FAPEC is that it performs optimally for block sizes within the 100-1000 range, while CCSDS 121.0 works with blocks of 8 or 16 samples. This already sets an advantage in terms of the necessary coding overhead to allow the decompression.

Its simplicity and robustness seems to position FAPEC as an interesting alternative to the current standard for universal lossless data compression for space. Nonetheless, as previously stated, the feasibility of a hardware implementation also needs to be assessed to offer a complete solution. A hardware implementation of FAPEC is not straightforward from its software counterpart neither from its initial algorithmic definition due to the peculiarities of FPGAs and ASICs. In fact, several features of the original algorithm need to be modified to achieve an optimal hardware implementation. For example, no floating-point operations must be used. Also, the logarithmic-like histogram used by FAPEC to reduce its complexity must be modified to allow an easier (binary-like) rule of construction and analysis. After implementing these changes it is mandatory to evaluate the correct operation of FAPEC again. Section 4 shows the results obtained by this modified version of FAPEC.

3. SELECTION OF THE TEST PLATFORM

Before proceeding with the hardware implementation of a compression algorithm such as FAPEC we must define a reference case considering both budget and time constraints. This will allow to estimate the hardware specifications and, thus, to select the target technology. In our case, we must note that FAPEC has been developed from concepts initially proposed for the Gaia mission [9]. The Gaia CCDs use 16-bit A/D converters with a high conversion rate. We have established as an initial goal the compression of a raw CCD output stream of Gaia, which operates at 2 Msample/s (i. e., 32 Mbit/s). This processing speed fixes an initial sensible requirement for the implementation. Subsequently, further modifications can be studied to obtain a higher throughput.

3.1 Requirements

The initial input interface must read 16-bit words at 2 MHz. Nevertheless, the output interface has been defined as a serial port able to operate up to 40 Mbps owing to the intrinsic variability of the output data rate. In fact, the minimum clock frequency allowed when working at a full duty cycle is 37 MHz. This value is deduced from the following statement: the minimum compression ratio which can be obtained with FAPEC (worst-case scenario) is 0.89. In this case, there is no compression but expansion instead. When applying this compression ratio to the input data rate – 32 Mbps – we obtain the 37 MHz. On the other hand, a parallel output interface could also be used, allowing lower clock frequencies, but also presenting a higher complexity.

Once the interfaces are fixed, the requirements of FAPEC must be analyzed to determine a suitable target technology. The drivers are, on one hand, development effort and budget restrictions. On the other hand, we find the operating frequency and the internal memory required. The prototype has been designed to minimize the memory requirements. Usually, FPGAs feature small amounts of internal memory. This memory offer interesting advantages, such as a higher integration level and a faster operation. Furthermore, potential problems may arise from the use of external memories [10]. Therefore, to keep the memory as low as possible only two data blocks will be present on the system at a given time: one being buffered and the other one being compressed. Besides, the block size has been reduced to 255 samples, thus trading-off the memory and the adaptive requirements. Regarding the histogram, we have opted for a new binary-friendly version. Only 37 bins are required to store a complete histogram — and each histogram bin only requires a depth of 8 bits. Summarizing, the use of 2 histograms plus 2 sample blocks implies only 8784 bits. Finally, there is another small memory used as a look-up table (LUT) which maps each histogram bin to its maximum associated input

value. This LUT is very small, as it only uses 37·16 = 592 extra bits. Hence, the overall memory required by FAPEC is less than 10 Kbits.

3.2 Selected platform

A test platform has been devised balancing the prototyping needs with the representativeness of a realistic model. The ASIC alternative was discarded from the very beginning, because it presents a longer developing time and much higher costs. Moreover, ASICs also reduce the flexibility of designs, which is incompatible with the prototyping task. Therefore, FPGA naturally arises as the best option: reprogramming is usually allowed and it is a low-cost alternative. A well-known target for space applications is the radiation-hardened ACTEL RTAX series [11], which feature an anti-fuse technology. For our prototyping, flash-based FPGA from the same manufacturer has been chosen. The main reason is that RTAX are not reprogrammable. Flash-based FPGAs provide re-programmability and portability of synthesis, thus reducing costs and assuring a high degree of representativeness and compatibility with the final platform. An ACTEL PROASIC3L development kit was finally chosen for the sake of simplicity. It includes an M1A3P1000L FPGA offering 1 Mgate and a 48 MHz clock [12]. Additionally, the board offers 1 MByte of SRAM and 16 MByte of flash memory. Both memories have been used to simplify the interface communication. Since the data will be read from and written to these memories in the tests, the I/O throughput will be guaranteed.

The compression chain has been split into three stages. Firstly, an input file is loaded to the flash memory from the computer, bridging a UART through the FPGA (write mode). Secondly, the FPGA reads the input values at the specified rate – 2 MHz – from the flash memory and stores the output values in the SRAM (operation mode). Finally, there is a retrieval of the SRAM data from the computer, again through the UART (read mode). The read/write modes are managed from the PC through Python scripts. Python language is very convenient for prototyping as it is intuitive and, hence, quick to develop and debug. These scripts interact with the serial ports allowing the communication with the UART. Obviously, additional resources must be implemented in the FPGA (not only the FAPEC compressor) so that it can interact with the external memories. Specifically, an IP core module simulates an AMBA–AHBL bus to interface with the memories, which simplifies the development of an interface manager. This architecture, illustrated in Figure 1, allows a flexible testing of FAPEC.

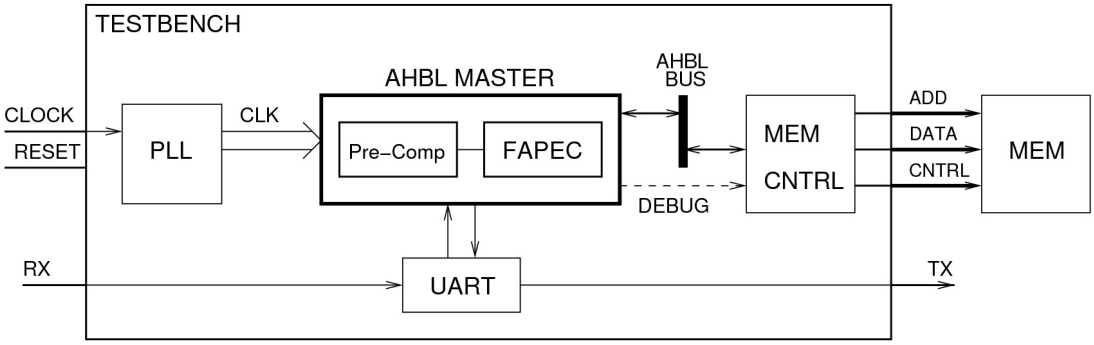

Figure 1. Schematic view of the FAPEC test bench.

4. ALGORITHM IMPLEMENTATION

The FAPEC compressor has been fully implemented in VHDL. Neither IP cores nor non-standard functions have been used in order to guarantee the simplest porting process of the prototype to the RTAX – or devices from other companies such as Xilinx or Atmel. In Fig. 2 the structure of the algorithm and of its implementation is outlined. A modular approach has been adopted to simplify the validation of the prototype. The modules have been validated incrementally, that is, validating each new module using also its predecessors.

The first stage is the *pre-compressor*, simply consisting of a delta predictor in this case for the sake of simplicity. It subtracts the previous value to the current one. The FAPEC block size (255 samples) is also applied to the pre-compression routine. In this way each packet can be independently recovered, as no memory is kept from one packet to another. The *histogram accumulator* identifies the pre-compressed samples and increments the corresponding bin value

in the *histogram memory*. Because of timing constraints, two different streams alternatively process the incoming values to identify their corresponding histogram bin. Data samples are simultaneously stored in the *block memory*, where they wait to be compressed. After having processed the 255 samples of a data block, the *histogram parser* activates. It parses the histogram while accumulating the occurrences stored in the bins and determining the ceilings – maximum values – for each of the 4 segments of PEC. Additionally, this module also selects the PEC variant and the size of the first segment. Until here, the different modules have built and analyzed the histogram. Once completed, the next stage is the *table constructor*, which calculates the size of the remaining PEC segments, i. e., the coding table, from the ceilings provided by the *histogram parser*. The small bin memory previously mentioned – operating as a look-up table (LUT) – is required to map the histogram bins to equivalent pre-compressed values. We remind that FAPEC uses a logarithmic-like histogram, mapping the 2^{16} possible values (16-bit) to only 37 bins. All the previous modules constitute the adaptive stage of PEC, namely, the FAPEC algorithm in itself. The last module of Figure 2 is the *PEC coder*. It receives the segment sizes and ranges from the *table constructor* for each block and uses them to code the stored samples into the *block memory*. Finally, the coding table is transmitted as a packet header, followed by the coded samples.

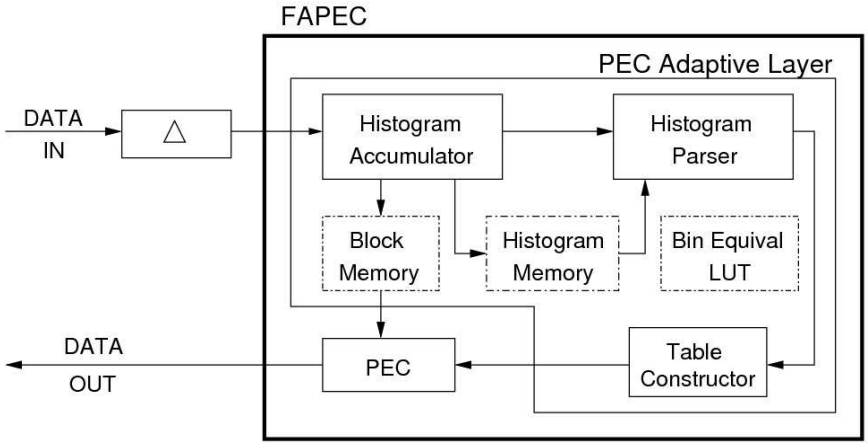

Figure 2. Block diagram of the VHDL design of FAPEC.

5. VALIDATION

As indicated before, FAPEC has suffered small changes to simplify the hardware implementation. Consequently, a new validation study is mandatory, being the simplest way to repeat the software tests run with synthetic simulations and with real data [7]. The goal is to assess that the hardware implementation of FAPEC still outperforms the CCSDS 121.0 recommendation, even in the presence of moderate levels of outliers.

5.1 Synthetic simulations

The synthetic test data are generated using a Monte Carlo method to produce geometric or Gaussian distributions, which usually arise after a good pre-processing stage [13]. The outliers are simulated as additive noise uniformly distributed over the entire dynamic range. The results are shown in Figure 3 and they are close to those obtained with the original software version [7]. Essentially, the modified FAPEC algorithm presents larger oscillations in the redundancy curve, opposed to a more uniform behavior of the software version. This phenomenon basically owes to the simplification in the rules that determine the coding table. In terms of redundancy, the worst-case of these oscillations imply an increase of 0.3 bits/symbol — it depends on the Shannon limit and hence on the data entropy. Nevertheless, note that FAPEC keeps offering redundancy values lower than 10% in most scenarios. Hence, it guarantees that the hardware implementation of FAPEC will offer excellent results even when dealing with noisy or unexpected data statistics.

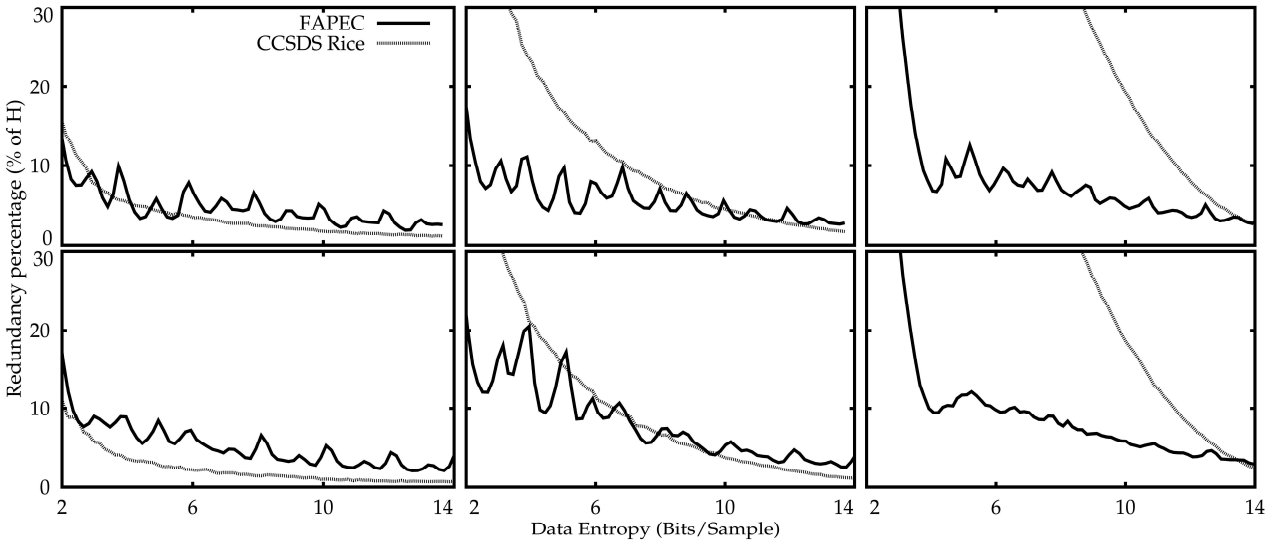

Figure 3. Coding redundancy (as compared to the Shannon limit) of CCSDS 121.0 and modified FAPEC. From left to right: 0.1%, 1% and 10% of uniform noise added to the simulated pre-compressed values. Top: Discrete Gaussian distribution; bottom: Geometric distribution.

5.2 Realistic simulations

In addition to the synthetic tests, the implementation of FAPEC has also been applied to highly realistic simulated data of the Gaia mission instruments [14]. More specifically, we have included data from the Sky Mapper (SM), the Astrometic Field (AF), the Blue and Red Photometers (BP and RP) and the Radial Velocity Spectrometer (RVS). The SM generates small images of 20×3 or 40×6 pixels, covering 4.7×2.1 arcsec^2 around the image of a star. The AF typically captures one-dimensional profiles of the star image with 6×1 samples (0.3×2.1 arcsec^2) except for the brightest stars, where it generates tiny images covering 18×12 pixels at most (1×2.1 arcsec^2). BP and RP instruments obtain dispersed images covering the bluer and redder portions of the stellar spectrum respectively. They contain 60×12 pixels at most (3.5×2.1 arcsec^2). Finally, the RVS analyzes the stellar spectrum around 850 nm, projecting it in up to 1260×10 pixels (74.3×1.8 arcsec^2). The results are summarized in Table 1. FAPEC offers the largest improvement with respect to CCSDS 121.0 for the SM and RVS data. The reason is that the SM is specially affected by cosmic rays and solar protons (that is, outliers in the data). On the other hand, the RVS contains large amounts of readout noise. RP data show some improvement as well, while for the AF and the BP the results are similar in both compressors. It is noteworthy that during all these realistic tests the efficiency of FAPEC has always been above the 89%. Furthermore, it is also worth noting that the largest improvements obtained with FAPEC correspond to the files where CCSDS 121.0 offers lower efficiencies. In other words, Table 1 demonstrates the nearly constant efficiency of FAPEC under diverse scenarios and the high sensitivity of CCSDS 121.0 to noise and outliers.

6. HARDWARE PERFORMANCE

The power consumption in a hardware device is generally dominated by the dynamic consumption in the current technology of integrated circuits. Hence, the slower the operation, the better the power figures that can be achieved. This is one of the main reasons of the success of the pipeline design in many applications. On the other hand, pipelines may complicate the logical design. We have tried to balance these two aspects in the prototype. We have also kept it as simple as possible, as the current design is just a feasibility study. In this way, small modifications and tuning of the design can be implemented easily. This leaves an open door for future improvements to reduce the system power consumption. In our case, the central modules — *histogram parser* and *table constructor* — operate with a slow clock (2 MHz) owing to pipeline structures, while the *histogram constructor* and the *PEC coder* modules need a 40 MHz clock. As previously mentioned, the FAPEC compressor has been implemented in a PROASIC3L development board, with an M1A3P1000L FPGA. The estimated power consumption for the design operating in the FPGA (alone) is just ~20 mW.

Table 1. Relative redundancy of the CCSDS 121.0 and FAPEC algorithms (lower is better). The third column shows the relative improvement that FAPEC achieves compared to the CCSDS 121.0. The last column is the Shannon Limit. GP stands for Galactic Plane data (with higher densities of stars), while NO_GP refers to measurements outside the Galactic plane.

	CCSDS	*FAPEC*	*FAPEC/CCSDS Improvement*	*Shannon Lim.*
SM_{GP}	16%	8%	**9%**	2.37
AF_{GP}	8%	8%	**0%**	1.66
BP_{GP}	12%	11%	**1%**	2.75
RP_{GP}	10%	6%	**4%**	2.61
RVS_{GP}	20%	11%	**10%**	2.10
SM_{No_GP}	15%	8%	**7%**	2.54
AF_{No_GP}	7%	8%	**-1%**	1.69
BP_{No_GP}	7%	6%	**1%**	3.61
RP_{No_GP}	7%	4%	**3%**	3.65
RVS_{No_GP}	8%	5%	**3%**	2.51

Working at 2 Msample/s, the goal of 32 Mbit/s is achieved. When compared to the software implementation, although its analysis is not as accurate as in the hardware case, a first estimation yields a required processing power of ~200 MIPS to execute a highly optimized software version of FAPEC with the same 32 Mbit/s throughput. Regarding the FPGA cell usage, the algorithm shows very modest requirements. In Table 2 we summarize the approximate FPGA usage of each module. Theoretically, it could be possible to implement up to 7 FAPEC cores in parallel inside this modest prototyping FPGA, thus leading to a maximum theoretical throughput of 224 Mbit/s.

Table 2: Approximate usage of the PROASIC3L M1A3P1000L for FAPEC, detailed for the different modules.

	Cell Usage
Pre-Compressor	~ 1%
Histogram Accumulator	~ 3%
Histogram Parser	~ 2%
Table Constructor	~ 3%
PEC	~ 4%
Total	**~ 13%**
FPGA Internal Memory	9 %

The latency associated to the input samples is variable as it depends on two parameters. Firstly, there is a fixed value associated to the statistical analysis of the data block. Secondly, a variable value depending on the processing time must be added. This processing time is different for each sample for two reasons. On one hand, the sample position inside the given data block must be considered. Compression is a serial process and, therefore, the initial samples present a higher latency than the final ones because the compression time of a block is, by definition, shorter than its intrinsic reception time (127.5 µs). If longer times were allowed, infinite delays would be theoretically possible. Consequently, the latter samples are kept buffered less time than the initial ones. On the other hand, the specific compression ratio also affects the latency. Higher compression ratios imply shorter processing times – less bits to output – thus also reducing the latency. Table 3 shows representative latency values obtained during the tests, whereas Figure 4 displays the associated delays as a chronogram.

Finally, regarding the RTAX target, and considering the information provided by the manufacturer [11], with their low-end product (RTAX250S) it would be possible to comfortably implement 2 FAPEC cores (64 Mbit/s operating in parallel). The high-end RTAX4000S should theoretically allow more than 30 cores, hence leading to more than 1 Gbit/s of throughput. Due to the underlying technology of RTAX, different from ProASIC (anti-fuse versus Flash), it is only possible to roughly estimate its power consumption. Both technologies share the benefits of low start-up and static power

consumption. In addition, their dynamic consumption is similar as well. Therefore, the estimated consumption of FAPEC inside an RTAX FPGA should be close to the 20 mW of the PROASIC3L prototype developed.

Table 3. Typical latency times, observed for the case of a compression ratio of 2.24.

Input block period	127.5 *μs*
Module	***Latency (μs)***
Pre-Compressor	0.09
Histogram Accumulator	1.05
Histogram Parser	14.33
Table Constructor	83.90
PEC	1.07 – 56.87
Block position	***Accum. Latency (μs)***
1st sample	227.10
Last sample	155.40

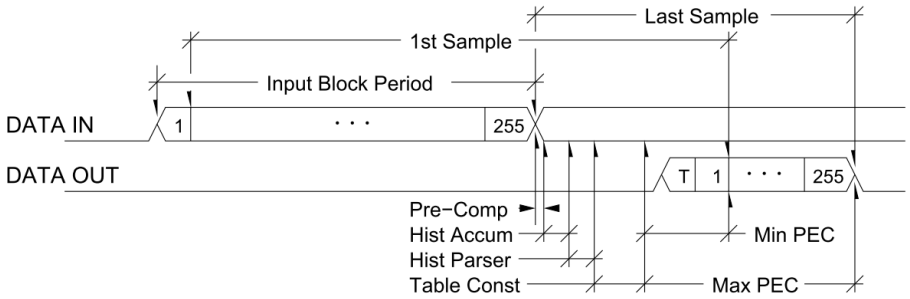

Figure 4. Chronogram of the FAPEC delays. On the top section we show the delays associated to the samples themselves. The bottom section shows the delays related to the different FAPEC modules.

7. FURTHER WORK

An initial version of the prototype was running the central modules at 24 MHz while the remaining ones worked at 48 MHz. As shown here, we have succeeded in reducing the fast clock to 40 MHz and the slow one to 2 MHz. Regarding the slow clock no further improvement is foreseen. However, the 40 MHz one may still be reduced as the coding block could operate at a minimum rate of 37 MHz. Currently, this 40 MHz boundary is set by the pipelines in charge of allocating the pre-processed samples in the histogram bins. It is possible to introduce an additional analysis pipeline to mitigate this constraint. This solution will be implemented and tested in the next version. Nevertheless, the 37 MHz limitation due to the PEC coder will still remain, and a major change on the architecture should be implemented to overcome this limitation.

A higher throughput will require higher clock frequencies, which is not advisable in some cases. A potential alternative would be the integration of a parallel output. Because of the intrinsic variable length of the compressed values, the easiest solution envisaged is the integration of a buffer memory. Although FAPEC is a serial process by construction, its output could be stored in a buffer. Once an entire block had been compressed, it could be output through a parallel interface. In this case, the minimum clock frequency would be set by the PEC coder although the addition of several of these blocks operating in parallel would reduce this constraint. However, further complexity would be introduced on the system. Thus, in the end the solution is a trade-off between clock frequency and complexity. Once an acceptable compromise is found, we expect to implement another version of the prototype offering a throughput higher than 100 Mbit/s, that is, more than 6 Msamples/s, while keeping the power consumption below 100 mW.

8. SUMMARY AND CONCLUSIONS

An FPGA implementation of the FAPEC lossless data compressor has been presented. As in the software case, this data compressor yields a performance similar to that of the CCSDS 121.0 recommendation on well-behaved data distributions, but it has proved to offer significant improvements for data contaminated with outliers or noise. We have assessed the feasibility of the hardware version. Therefore, we consider that FAPEC is definitely an excellent alternative for space missions, especially considering that many instruments are affected by radiation and prompt particle events.

Embedded (hardware) implementations are becoming increasingly necessary in space. We have developed a prototype to demonstrate that a hardware implementation of FAPEC is not only feasible, but also fast and efficient. The benchmarked implementation on the PROASIC3L FPGA successfully validated its operation at 32 Mbit/s (2 Msample/s) with a relatively simple design. In addition, the possibility of several compression cores operating in parallel could easily take advantage of the low device usage to multiply the throughput of the system beyond 100 Mbit/s. Future modifications may allow for even higher data rates. Thus, the FAPEC compressor seems to be a realistic alternative to be implemented in space missions that require high-performance and resilient lossless data compression systems, either in software or in hardware, while keeping low levels of complexity. Future developments will provide a space-qualified implementation in the RTAX FPGA family.

ACKNOWLEDGEMENTS

This work was partially supported by the MICINN-FEDER grant AYA2008-14648-C02-01, by MEC grants PNE2006-13855-C02-01 and TME2008-01214, and by the AGAUR.

REFERENCES

[1] J. Portell, E. García-Berro, X. Luri, & A. G. Villafranca, “Tailored data compression using stream partitioning and prediction: application to Gaia,” *Experimental Astronomy*, vol. 21, no. 3, 125-149 (2006).

[2] CCSDS, “Lossless data compression, blue book,” Tech. Rep., 121.0–B–1, (1997).

[3] P.–S. Yeh, “Implementation of CCSDS lossless data compression for space and data archive applications,” Tech. Rep., (2002).

[4] R. Vitulli, “PRDC: an ASIC device for lossless data compression implementing the Rice algorithm”, *IEEE International*, (2004).

[5] R. F. Rice, “Some practical universal lossless coding techniques”, Tech. Rep., (1979).

[6] K. Sayood, *Lossless Compression Handbook,* Academic Press, (2003).

[7] J. Portell, A. G. Villafranca, and E. García-Berro, “Quick outlier-resilient entropy coder for space missions”, Journal of Applied Remote Sensing 4 (2010).

[8] M.A. Nieto–Santiesteban et al., “Data Compression for NGST”, *Astronomical Data Analysis Software and Systems VIII*, ASP Conf. Series, Vol. 172, (1999).

[9] M. A. C. Perryman, K. S. de Boer, G. Gilmore, E. Hoeg, M. G. Lattanzi, L. Lindegren, X. Luri, F. Mignard, O. Pace, & P. T. Zeeuw, “Gaia: Composition, formation and evolution of the Galaxy”, *Astronomy & Astrophysics* 369, 339-363 (2001).

[10] S. Mignot, *Towards a demonstrator for autonomous object detection on board Gaia*, PhD thesis, Observatoire de Paris (2008). URL: http://tel.archives-ouvertes.fr/tel-00340279/

[11] ACTEL RTAX FPGA family: http://www.actel.com/products/milaero/rtaxs/

[12] ACTEL M1A3P1000L FPGA: http://www.actel.com/products/pa3l/default.aspx

[13] P.–S. Yeh et al., “On the optimality of Code Options for a Universal Noiseless Coder”, Tech. Rep., JPL Publication 91-2 (1991).

[14] C. Babusiaux, “The Gaia Instrument and Basic Image Simulator”, in *The Three-Dimensional Universe with Gaia*, ESA SP-576, 125-149 (2005).