OUTLIER-RESILIENT ADAPTIVE ENTROPY CODERS

Jordi Portell[1,2], Alberto G. Villafranca[2,3], Marcial Clotet[3,2], Enrique García-Berro[3,2]

[1] *Dept. d'Astronomia i Meteorologia (ICC-UB), Universitat de Barcelona,*
c/Martí i Franquès 1, 08028 Barcelona, Spain
Email: jportell@am.ub.es

[2] *Institute for Space Studies of Catalonia (IEEC), c/Gran Capità 2-4, 08034 Barcelona, Spain*
Email: agonzalez@ieec.cat

[3] *Dept. de Física Aplicada, Universitat Politècnica de Catalunya, c/ Esteve Terrades 5, 08860 Castelldefels, Spain*
Email: garcia@fa.upc.edu

ABSTRACT

The Rice entropy coder is at the core of the CCSDS recommendation for lossless data compression. The compression efficiency of this coder rapidly decreases in presence of noise and outliers, since it is conceived for noiseless data following geometric distributions. This, in turn, makes the CCSDS recommendation very sensitive to the presence of outliers in the data, leading to an important decrease of the compression ratio for realistic situations. Here we present two adaptive entropy coders which appear to be reliable alternatives to the CCSDS coding stage. First, we study how the substitution of the Rice coder by a sub-exponential one benefits the compression result, keeping the rest of the CCSDS recommendation identical. Next, we analyze the performance of FAPEC, which was specifically conceived for compressing data with large amounts of outliers and noise. We show that both solutions offer high compression ratios under almost any situation and that, in general, the results are substantially better than those obtained when the original CCSDS recommendation is used. This is assessed by testing both compressors with synthetic and real data. The processing requirements are similar for the three solutions. Finally, a hardware prototype of the FAPEC coder on an FPGA board is available, offering a high throughput with a low power consumption and device usage.

Keywords: Lossless data compression, entropy coding, CCSDS, Rice, outlier, resilient, FAPEC, FPGA

I. INTRODUCTION

Data compression systems for satellite payloads use to have tight restrictions on several aspects. First, the data block size should be rather small, in order to avoid losing large amounts of data when transmission errors occur [1,2]. Second, the processing power for software implementations (or electrical power, in case of hardware implementations) is largely limited in space. Thus, the compression algorithm should be as simple and quick as possible. Finally, the required compression ratios are becoming larger as new missions are conceived and launched. When all these restrictions are combined with the need of a lossless operation, the design of such a data compression system becomes a true challenge.

The Consultative Committee for Space Data Systems (CCSDS) proposed a general-purpose compression solution [3,4] based on a two-stage strategy – an otherwise typical approach. Firstly, a simple pre-processing stage changes the statistics of the data by applying a reversible function, often implemented as a data predictor followed by a differentiator. Secondly, a coding stage based on an entropy coder outputs a variable number of bits for each of the symbols calculated by the first stage. While no specific pre-processing method is included in the recommendation, the coding stage is mainly based on the Rice-Golomb codes [5,6], which are simple to calculate and, hence, quick implementations are available – specially in hardware. This CCSDS recommendation (codenamed 121.0) operates with blocks of 8 or 16 samples, determining the best Rice code to be used for them. Summarizing, it is a quick algorithm that yields good compression ratios, and what is most important, it rapidly adapts to the statistical variations of the data to be compressed. Hence, this recommendation is widely used in scientific payloads [7], and it still remains the reference in terms of generic data compression for space missions, owing to its flexibility and speed. Although new techniques improving its performance have appeared, they mostly focus on a specific kind of data, such as image [8] or multi/hyperspectroscopy. However, it is important to realize that these new methods require more computational resources – and what is most important for the purpose of our work, they also require a final stage for entropy coding.

The CCSDS 121.0 recommendation is not exempt of problems. In particular, Rice codes offer excellent compression ratios for data following a geometric probability distribution, but any deviation from this leads to a decrease in the compression efficiency. One solution to this problem is to use a different kind of Golomb or prefix codes with a smoother growth for high values, but within the same framework and adaptive stage as CCSDS 121.0 [9]. Yet another solution is to develop a brand new set of codes, such as the Prediction Error Coder (PEC), introduced in [10,11,12]. Here we describe an adaptive stage for PEC, leading to what we have called the Fully Adaptive PEC (FAPEC) [11], which autonomously determines the nearly-optimal set of codes for a given block of data.

In this paper we first discuss in section II the limitations of the Rice coder and of the CCSDS 121.0 lossless compression standard. Section III presents the concept of outlier-resilient entropy coding, describing the cases of subexponential codes and PEC. Section IV describes the adaptive implementations for both coders, while in section V we show the results obtained on synthetic and real data. Section VI briefly describes a hardware prototype of FAPEC on an FPGA. Finally, in section VII we summarize our major findings and list our conclusions.

II. LIMITATIONS OF RICE AND OF CCSDS 121.0

Rice codes are optimal for data following discrete Laplacian (or two-sided geometric) probability distributions [13], which are expected after the CCSDS 121.0 pre-processing stage [3] – or, in general, after any adequate pre-processing stage. However, this assumes a correct operation of the predictor, which cannot be taken for granted as noisy samples and outliers can modify the expected distribution. This is especially true for the space environment, where prompt particle events (such as cosmic rays or solar protons) will affect the on-board instrumentation. Any deviation from the expected statistic can lead to a significant decrease in the resulting compression ratio.

The Rice-Golomb coder is based on a k parameter that must be chosen very carefully in order to obtain the expected compression ratios for a given set of data. Table 1 hereafter illustrates some Rice codes for small values and low k configurations. The lowest values of k lead to a rapid increase in the length of the output code, although such values are the ones leading to the shortest codes for small values. If we would use Rice codes statically (that is, fixing the k value), an unacceptable risk would appear. That is, it could happen that we expect to code a dataset for which we always expect low values, and thus we select a low k such as 1. With this configuration, receiving a single high value such as 20000 would lead to an output code of about ten thousand bits. The adaptive layer introduced by the CCSDS 121.0 recommendation selects the best configuration for each given data block. It determines the total length of the coded block using $k = 1$ to $k = 13$, and then it selects the value of k leading to the shortest total length. Note that $k = 0$ is not considered, since it coincides with the Fundamental Sequence option already included in CCSDS 121.0. This automatic calibration significantly reduces the effect of outliers present in the data, leading to acceptable ratios even in case of rapidly changing statistics. Nevertheless, this is done at the expense of increasing k when such outliers are found. For example, in a data block where all the values are small (or even zero), a single high value makes CCSDS 121.0 to select a high value of k, thus leading to a small compression ratio. Our goal is to reduce the effect of such outliers even within a data block, making possible to select smaller values of k and, thus, increasing the achievable compression ratios.

III. OUTLIER-RESILIENT ENTROPY CODES

Data compression through entropy coding basically assigns very short codes to the most frequent symbols, while letting less frequent symbols generate longer codes. If a code is not carefully designed, less frequent symbols can lead to prohibitively long codes, affecting the overall compression ratio. Here we define *outlier-resilient entropy coding* as this careful design of an entropy coder in the sense that less frequent symbols lead to relatively short codes, typically of the order of just twice the original symbol size or even less. It is very important to emphasize at this point that we do not refer to error-resiliency or data integrity here. We refer to *outlier-resiliency* as the ability of achieving high compression efficiencies despite of having large amounts of outliers in a data block – that is, despite of compressing data which does not strictly follow an expected probability distribution.

The coders presented here can be used as the coding stage of elaborated data compression systems, such as those used in imaging or hyperspectroscopy. It is also worth mentioning that these coders are especially applicable when high compression ratios are needed but the processing resources available are low, which is the usual case in satellite data compression. Other entropy coders such as arithmetic coding [14] could provide better results, but at the expense of more processing resources. Finally, a lossless operation is assumed throughout the whole chapter, which is the most frequent premise of an entropy coder.

A. Subexponential codes

The main reason for the CCSDS 121.0 performance to drop abruptly when noise or outliers are introduced is that Rice codes are not intended to be used with noisy data. This limitation is due to the fact that the length of Rice codes grows too quickly for large input values, especially when low values are assigned to the k parameter. Subexponential codes are prefix codes used in the Progressive FELICS coder [15, 16]. Similarly to the Golomb codes, the subexponential coder depends on a configuration parameter *k*, with $k \geq 0$. The design of this coder is supposed to provide a smoother growth of the code lengths, as well as a smoother transition from the inherent CCSDS 121.0 strategies (Zero Block, Second Extension or Fundamental Sequence) to the prefix coding strategy. In particular, for small dispersions, moving from these strategies to subexponential coding does not imply a significant increase in the output code length.

Essentially, subexponential codes are a combination of Rice and exponential-Golomb codes. These two coding strategies are used depending on the value being coded and the value of *k*. When $n < 2^{k+1}$, the length of the code increases linearly with *n*, while for $n \geq 2^{k+1}$ the length increases logarithmically. The first linear part resembles a Rice coding strategy and maintains a slow code growth for small values, while the second part resembles the exponential-Golomb code. Table 1 shows some subexponential codes for some values of *n* and *k*. These two strategies combined provide an advantage with respect to both Rice and exponential-Golomb codes. In particular, this strategy allows obtaining similar code lengths to Rice for small input values, and additionally, in the case of outliers or large values, the code length is shorter than that of Rice due to the exponential steps in the second stage. This is also shown in Table 1, which includes the difference in code length (in bits) between Rice and subexponential.

Table 1 – Some Rice and subexponential codes, and bit length differences between them

	$k = 0$			**$k = 1$**		
N	**Rice**	**Subexp**	**Diff.**	**Rice**	**Subexp**	**Diff.**
0	0|	0|	0	0|0	0|0	0
1	10|	10|	0	0|1	0|1	0
2	110|	110|0	+1	10|0	10|0	0
3	1110|	110|1	0	10|1	10|1	0
4	11110|	1110|00	+1	110|0	110|00	+1
5	111110|	1110|01	0	110|1	110|01	+1
6	1111110|	1110|10	-1	1110|0	110|10	0
7	11111110|	1110|11	-2	1110|1	110|11	0
…	…			…		
31	1…(x31)…10|	111110|1111	-22	1…(x15)…10|1	11110|1111	-8

B. Prediction Error Coder (PEC)

Rice codes are adequate for data following a geometric statistical distribution, which often arise after an adequate pre-processing stage [13]. However, any deviation from this statistic can lead to a decrease in the final compression ratio. Despite of the adaptive stage, the CCSDS recommendation also suffers from this limitation. Additionally, and most importantly, even with such adaptive stage we have found important decreases in the compression efficiency when realistic fractions of outliers are found in the data. The improvement based on subexponential codes can mitigate it, but a more flexible and robust solution is preferable.

We have developed the Prediction Error Coder [10,11] to solve this weakness of the Rice-like codes. As its name indicates, PEC is focused on the compression of prediction errors, and hence a pre-processing stage outputting signed values is required – such as a data predictor plus a differentiator. PEC was devised and developed within the frame of the GOCA Technology Research Programme of ESA. Efforts were put on the development of a very fast and robust compression algorithm, and PEC was the outcome – a partially adaptive entropy coder based on a segmentation strategy. PEC is composed of three coding options, namely, Low Entropy (LE), Double-Smoothed (DS) and Large Coding (LC). All of them are segmented variable-length codes (VLC). LC also makes use of unary prefix codes [15], while LE and DS rely on the "minus zero" feature of signed prediction errors – an impossible value that is used as an escape sequence. In Fig. 1 a schematic view of PEC is shown and the coding strategy of each of the options and segments is unveiled. The number of bits per segment (*h*, *i*, *j* and *k*) are independent parameters, which fix the compression ratio achievable for each range of input values. As can be seen, the coding scheme is completely different to that of the Rice coder. The three coding options share the same principles: the dynamic range of the data to be coded is split into four smaller ranges (or segments). The size of each segment is determined by its corresponding coding parameter (*h*, *i*, *j* or *k*), which indicates the number of bits dedicated to code the values of that segment. This set of parameters is called *coding table*. Then, for each value to be coded, the appropriate segment and number of bits to code the value are chosen, following the procedure indicated in Fig. 1.

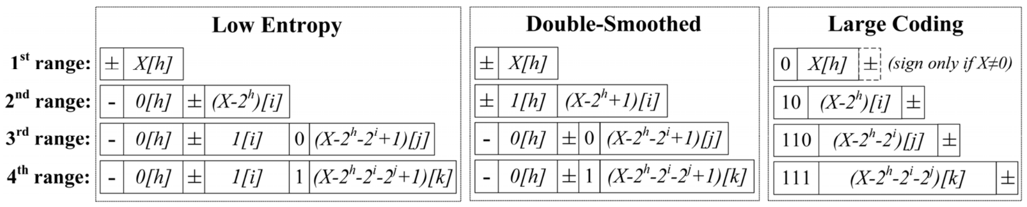

Fig. 1 – Illustration of the three PEC coding strategies, for data with low, medium and high entropy levels.

PEC follows the assumption that most values to be coded are close to zero, although this premise is not really mandatory for a successful operation of PEC. When this is true, the coding parameters must be fixed in a way that the first segments are significantly smaller than the original symbol size, while the last segments are slightly larger. This obviously leads to a compressed output, while the ratio will be determined by the probability density function (PDF) of the data combined with the selected coding table. Additionally, one of the main advantages of PEC is that it is flexible enough to adapt to data distributions with probability peaks far from zero. With an adequate choice of parameters, good compression ratios can still be reached with such distributions.

PEC can be considered a *partially adaptive* algorithm. That is, the adequate segment (and hence the code size) is selected for each one of the values. This is obviously an advantage with respect to the Rice coder, which uses a fixed parameter for all the values – at least within a given coding block, in the case of the CCSDS recommendation. Another advantage with respect to Rice is that, by construction, PEC limits the maximum code length to twice the symbol size in the worst of the cases (depending to the coding table). Nevertheless, despite of these features, it is true that the coding table and coding option of PEC must be adequately chosen in order to get the best compression ratios.

IV. ADAPTIVE IMPLEMENTATIONS

A. Adaptive subexponential coder

Subexponential codes can be directly integrated within the CCSDS 121.0 framework, simply substituting the Rice coder. The rest of the recommendation can be kept unchanged, also including the Prediction Error Mapper (PEM). Only one change is required, namely, the $k = 0$ option must be allowed, in order to lead to a smooth transition between the Fundamental Sequence option and the subexponential coding. On the other hand, the $k = 13$ option is not required anymore (at least for 16-bit samples), so the configuration header output at the beginning of each data block can be of the same size.

B. Fully Adaptive PEC (FAPEC)

An adaptive algorithm for PEC has been designed and implemented. The solution, protected by a patent, has been called *Fully Adaptive Prediction Error Coder* (FAPEC) [11,12]. Similarly to the CCSDS recommendation, where an adaptive stage selects the most appropriate value of k for a given data block, FAPEC adds an adaptive layer to PEC in order to configure its coding table and coding option according to the statistics of each data block. In this way, nearly-optimal compression results can be achieved without the need of any preliminary configuration of PEC, and without requiring any knowledge of the statistics of the data to be compressed. The block length is configurable and not restricted to a power of two, with typical (recommended) values of 250 to 500 samples. One of the main premises in the design of FAPEC was the quickest possible operation, even if at the expense of a slight decrease in the optimality of the PEC configuration – and hence a slight decrease in the compression ratio. The intrinsic robustness of PEC guarantees that such decrease will be negligible.

FAPEC accumulates the values to be coded and, at the same time, a histogram of their moduli is calculated on-the-fly. This is a logarithmic-like histogram, in the sense that higher sample values (which are less frequent) are grouped and mapped to fewer bins, and values close to zero are mapped to independent bins. This reduces the memory required for the histogram and, most important, the time required to analyze it. Once the required amount of values has been loaded, an algorithm analyzes the histogram and determines the best coding option (LE, DS or LC) and coding table. FAPEC contains some parameters which have been set from tests with Laplacian distributions, but they could be fine-tuned to better adapt to other statistical distributions if really needed. This is an interesting feature that the CCSDS recommendation does not have. Finally, once the coding option and the corresponding parameters have been determined, they are output as a small header followed by all the PEC codes for the values of that block. Explicitly indicating the PEC configuration makes possible to change the FAPEC decision algorithms without requiring any modification in the receiver. Fig. 3 illustrates the overall operation of FAPEC.

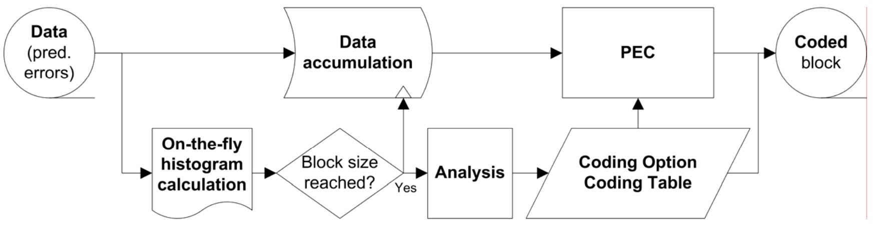

Fig. 3 – Operation of the Fully Adaptive Prediction Error Coder (FAPEC).

V. COMPRESSION PERFORMANCE RESULTS

A. Synthetic data results

We have conducted a campaign of tests compressing some Laplacian distributions using the adaptive entropy coders previously described. We have covered the entire range of dispersions (or entropy levels) typically found in real cases. Figure 4 shows the results, where the abscissa corresponds to the parameter of the statistic, that is, *b* for the case of the Laplacian distribution. Small values of *b* indicate low data dispersion (or, equivalently, low entropy), thus indicating a very good pre-processing stage – or data with implicitly low entropy. Besides the Laplacian distribution, real data is usually contaminated with noise and outliers. Therefore, to obtain meaningful results the coders have been tested under these conditions. Specifically, we have included different flat noise levels, namely 0.1%, 1% and 10% [12]. These levels represent three different scenarios, namely, almost ideal conditions, a realistic scenario, and a fairly extreme situation – a crucial consideration in space applications [17]. The noise introduced in the samples follows a uniform (flat) distribution in the entire data range.

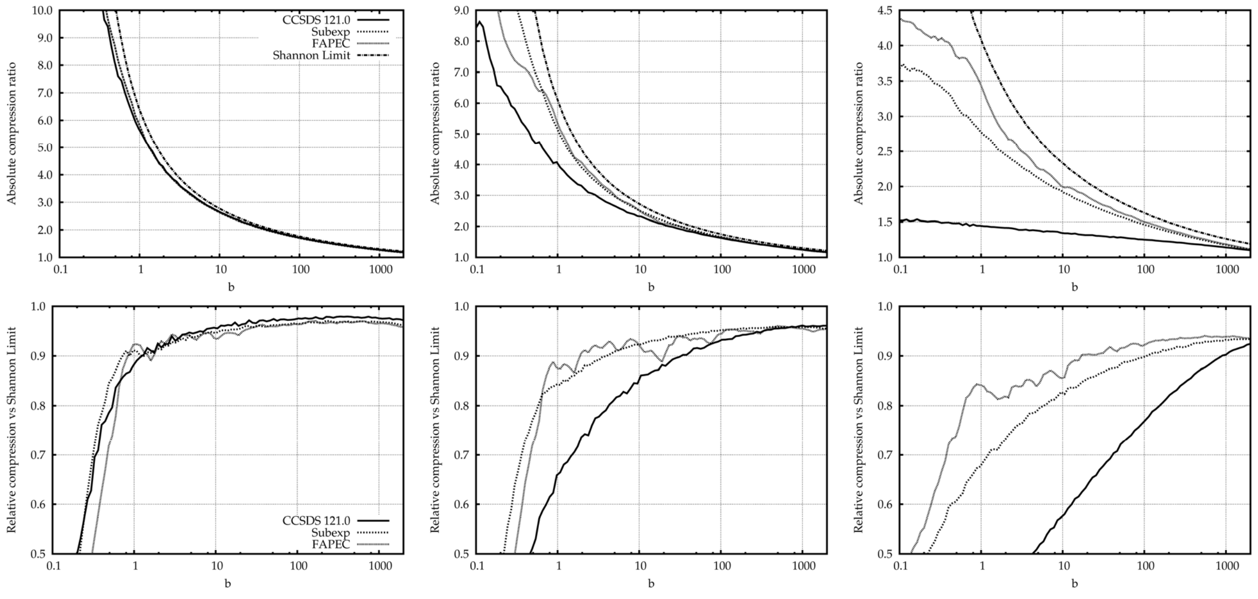

Fig. 4 – Compression ratios (top panels) and coding efficiency (bottom panels) of adaptive subexponential and FAPEC on synthetic data. From left to right, outlier levels of 0.1%, 1% and 10%.

Figure 4 shows the compression performance of our adaptive entropy coders, both in terms of absolute ratios (top panels) and compression efficiency (bottom panels). The first remarkable result that can be seen is that our coders roughly match or even slightly exceed the CCSDS performance when very few outliers are present in the data – that is, for the case in which only 0.1% flat noise is added (left panels). Particularly relevant is that for small values of *b*, which can be rather common in several cases, the subexponential algorithm performs better than the current standard, providing compression ratios which are about 2% larger. In the case of medium to high entropy levels, all the coders perform very similarly. When otherwise realistic noise levels are applied (1%), the new coders keep their compression efficiency mostly unchanged, typically performing better than 85% of the Shannon limit and even reaching 95%. On the other hand, CCSDS 121.0 is strongly affected, offering efficiencies typically below 90% and even dropping below 60%. Finally, in scenarios where the noise or outliers level is rather high (that is, 10%), the current standard is almost unable to compress the data. The current CCSDS standard can just reach compression ratios as low as 1.5 in the best of the cases. That is a compression efficiency well below 50% for medium to low entropies, and typically below 80% even for high entropies. On the other hand, the adaptive subexponential coder obtains compression efficiencies above 50% for almost any case (even for low entropy levels), while the typical result is above 70%. Ratios up to 3.5 can be reached in this way, which is an excellent result considering the large amount of noise in the data. These results are even better with FAPEC, which reveals to be the most outlier-resilient entropy coder. Its efficiency decreases below 60% only for very low entropy levels (where ratios of 4 are reached), while it typically performs better than 80% of the Shannon limit. Thus, when compared to the CCSDS standard, our coders can offer twice of even three times its compression ratio under such extreme conditions. Finally, it is worth mentioning that we have also tested the coders on noisy Gaussian distributions, revealing slightly worse results for CCSDS 121.0 and very similar results for our coders.

B. Results on real data

After evaluating any entropy coder with synthetic data, we should obviously determine how it performs under real conditions in order to assess the initial results. In order to do this, representative space-borne data should be used, such as astronomical imaging, planetary imaging, spectroscopy, etc. We have compiled a *data compression corpus* suitable for testing space-borne entropy coders, compiled from real and simulated datasets, and available upon request to the authors. The testing dataset includes astronomical images [18], realistic simulations of the Gaia instruments [19,20], GPS measurements data [21], photometric and spectroscopic data, and data from instruments of the Lisa PathFinder mission (LPF).

A pre-processing stage was obviously required in order to perform these tests. Very simplistic stages have been used here, typically using just a differentiator – that is, predicting each sample as equal to the previous one. In some cases we have essayed up to third-order filtering (within the prediction loop), thus smoothing the quick variations or noise that the data may have. The pre-processing stage leading to the best ratios has been used for each of the data files, but it is obviously far from the elaborated stages that we can find for imaging, spectroscopy or GPS, just to mention some examples. Thus, the ratios shown here can be seen as the "worst result" that we can get for a given file.

Table 2 – Compression ratios achieved by the coders shown here on real data.

File	CCSDS 121.0	Adaptive Subexp	FAPEC	S_L	W_L *(bits)*	***Best Eff.***
FOCAS img. - ngc0001	1.79	**1.84**	1.83	1.92	16	96%
FOCAS img. - for0001	**3.13**	2.99	**3.13**	3.22	16	97%
FOCAS img. - sgp0002	4.16	4.27	**4.41**	5.44	32	83%
GIBIS img. - 7135_SM1_6	3.48	3.39	**3.59**	3.77	16	96%
GIBIS img. - 5291_RVS1_5	1.94	2.02	**2.61**	2.69	32	97%
Misc img. - stellar_field	1.12	2.14	**2.63**	5.14	64	56%
Misc img. - noisy_source	1.01	1.03	**1.15**	1.19	32	97%
Misc img. - galaxy	1.14	2.17	**2.58**	4.84	64	59%
Misc img. - ground_2	**1.92**	1.86	1.83	1.81	8	106%
VPU TM - SM_L90b40	2.21	**2.39**	2.38	2.53	16	94%
VPU TM - AF_L10b70	1.68	1.69	**1.69**	1.73	16	98%
VPU TM - BP_L170b60	3.56	3.52	**3.58**	3.68	16	97%
VPU TM - RVS_L1b1	2.15	2.19	**2.21**	2.35	16	95%
Raw GPS - global_S1	**2.30**	2.25	2.29	2.35	16	98%
Raw GPS - global_L1	1.57	1.64	**1.67**	1.76	24	95%
Treated GPS - is07_lat	3.61	3.56	**3.68**	4.40	16	86%
LPF - kp30_row2	3.86	3.85	**4.00**	4.12	24	97%
LPF - kp30_row10	12.83	**15.34**	13.76	20.35	24	75%
Spectra - Observ_irrad	**2.79**	2.75	2.72	2.68	16	104%
Spectra - er_spec	**1.62**	1.61	1.61	1.63	24	99%

In some cases, large symbol sizes are used in the data files, such as 32 or even 64 bits per sample. The implementations used in the tests of the entropy coders described here can handle up to 28 bits per sample, and thus a *sample splitting* process has been applied. That is, 32-bit samples have been split into two, alternatively compressing the most and least significant 16-bit words of the prediction errors. Similarly, in case of 64-bit samples, four 16-bit words have been used.

The effect on the coders is that the compressor receives values with very different statistics. The most significant portions of the samples appear as very low-entropy values, while the least significant portions resemble uniformly distributed noise and thus appear as outliers. In other words, huge amounts of outliers appear in the data. This operation has interesting effects on the results, since subexponential and FAPEC are able to deal with this situation without significantly affecting the overall performance. Table 2 summarizes the most relevant tests, also indicating the Shannon Limit (S_L) and the word length or sample size in bits (W_L). The best result for each file is highlighted in bold face, and its corresponding compression efficiency is shown in the last column.

As we can see, the optimality of each coder depends on the kind of data, but generally speaking the CCSDS 121.0 recommendation is beaten by the outlier-resilient entropy coders presented here. FAPEC typically offers the best results. In the worst of the cases it offers ratios 5% smaller than CCSDS 121.0, while in the best of the cases in can double the ratios obtained with such system. Such extremely good results are due to the sample splitting procedure previously mentioned, leading to a situation equivalent to having even more than 50% of outliers. The worst efficiency found in our tests is 46% of the Shannon limit, but in that case the CCSDS 121.0 is only able to perform at 27%. Regarding the adaptive subexponential coder, it also uses to outperform the current standard, although the improvement uses to be smaller than with FAPEC. Finally, it is worth highlighting the results beyond the Shannon limit. Those have been achieved owing to the low-entropy extensions existing in both the CCSDS 121.0 framework and in FAPEC, which can generate single small codes for sequences of zeroes.

VI. HARDWARE IMPLEMENTATION OF FAPEC

The tests described in the previous section have been done with highly optimized software implementations, which reveal very similar processing requirements for the three coders – even slightly lower for the case of FAPEC. The lossless operation has obviously been assessed. Being FAPEC the coder offering the best results, we have considered mandatory to demonstrate its in-flight feasibility by implementing it on an FPGA device [22]. More specifically, an ACTEL PROASIC3L development kit has been used, which is similar enough to the radiation-hardened RTAX antifuse devices of the same manufacturer that can be used in satellite payloads.. The kit features an M1A3P1000L FPGA with 1 Mgate and a 48 MHz clock, as well as 1 MByte SRAM and 16 MByte Flash in the board.

The implementation of the FAPEC compressor, including a simple pre-processing stage, has been done in VHDL without any non-standard function. A modular approach has been adopted to simplify the validation of the prototype. Minor modifications were required on the algorithm in order be more hardware-friendly, although its effect on the compression results has been assessed to be negligible. Regarding the hardware performance, the power consumption for the FPGA alone is estimated in just 35 mW when working at 2 Msample/s – that is, receiving an input of 32 Mbps. The total cell usage of the complete system is just 12% of the selected device, with PEC as the largest contributor (with 4% of the FPGA cells). The typical latency is less than 137 μs for compression ratios around 2.2. Considering the low cell usage, up to 7 FPGA cores could be implemented in parallel in this device, thus leading to a theoretical throughput of 224 Mbps.

VII. SUMMARY AND CONCLUSIONS

In this paper we have discussed the importance of an adequate entropy coder in order to build an efficient data compression solution, either for simple and generic compressors, or for specific and elaborated systems such as imaging, hyperspectroscopy or various instrumental data. Some models and metrics for evaluating an entropy coder have been presented and applied to the current lossless compression standard for space (CCSDS 121.0), to a brand new solution with promising results (FAPEC), and to a simple modification to the current standard also giving good results. The availability of a data compression corpus has made possible to evaluate the several coders under real conditions. The compression ratios achieved with these tests reveal that the new FAPEC algorithm is a reliable alternative to the CCSDS 121.0 recommendation. Its software implementation has been evaluated, indicating very similar processing requirements than those for CCSDS 121.0. Additionally, a hardware prototype implemented on an FPGA device is available, which assesses the applicability of FAPEC in space missions with very modest requirements. Alternatively, the CCSDS 121.0 recommendation could be improved by substituting the Rice-Golomb codes by outlier-resilient codes, such as a subexponential coder. Although the results with such system are often slightly worse than those of FAPEC, they are resilient enough in front of outliers in the data, and thus appear as another reliable alternative.

In general, it is highly recommended to use outlier-resilient entropy coders when data from space-borne instrumentation is to be compressed. While guaranteeing almost the same compression ratios as the current standards under any situation, they can take much more advantage of the data redundancies when large amounts of outliers appear – such as those caused by Prompt Particle Events or artifacts in the data or in the instruments.

REFERENCES

[1] J. Portell, E. García-Berro, X. Luri, and A. G. Villafranca, “Tailored data compression using stream partitioning and prediction: application to Gaia,” *Experimental Astronomy* 21, 125–149 (2006).

[2] CCSDS-101.0-B-5 Blue Book, *Telemetry channel coding*, 2001.

[3] CCSDS-121.0-B-1 Blue Book, *Lossless data compression*, 1993.

[4] CCSDS-120.0-G-2 Informational Report, *Lossless data compression*, 2006.

[5] R. F. Rice, “Some practical universal noiseless coding techniques,” *JPL Technical Report* 79-22 (1979).

[6] S. W. Golomb, “Run-lengths encodings,” *IEEE Transactions on Information Theory* 12, 399–401 (1966).

[7] P.-S. Yeh, “Implementation of CCSDS lossless data compression for space and data archive applications,” *Proc. CCSDS Space Operations Conf.* , 60–69, 2002.

[8] P.-S. Yeh, P. Armbruster, A. Kiely, B. Masschelein, G. Moury, C. Schaefer, and C. Thiebaut, “The new CCSDS image compression recommendation,” *IEEE Aerospace Conf.*, 4138–4145, 2005.

[9] M. Clotet, J. Portell, A. G. Villafranca, and E. García-Berro, “Simple resiliency improvement of the CCSDS standard for lossless data compression,” *Proc. SPIE 7810*, 2010.

[10] J. Portell, A. G. Villafranca, and E. García–Berro, “Designing optimum solutions for lossless data compression in space,” *Proc. ESA On-Board Payload Data Compression Workshop*, 35–44, 2008.

[11] J. Portell, A. G. Villafranca, and E. García-Berro, “A resilient and quick data compression method of prediction errors for space missions,” *Proc. SPIE 7455*, 2009.

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

[13] P.-S. Yeh, R. Rice, and W. Miller, “On the optimality of code options for a universal noiseless coder,” *JPL Technical Report* 91-2 (1991).

[14] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Communicat. ACM* 30, 520–540 (1987).

[15] D. Solomon, *Data Compression. The complete reference*, Springer, 2004.

[16] Howard, P. and Vitter, J., “Fast progressive lossless image compression,” in *Image and Video Compression Conference*, SPIE, 98-109 (1994).

[17] Nieto-Santisteban, M. A., Fixsen, D. J., Offenberg, J. D., Hanisch, R. J. & Stockman, H. S., “Data Compression for NGST”, in *Astronomical Data Analysis Software and Systems VIII*, vol. 172 of Astronomical Society of the Pacific Conference Series, 137–140 (1999).

[18] F. Murtagh and R. H.Warmels, “Test image descriptions,” in *Proc. 1st ESO/ST-ECF Data Analysis Workshop*, 17(6), 8–19 (1989).

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

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

[21] A. G. Villafranca, I. Mora, P. Ruiz-Rodríguez, J. Portell, and E. García-Berro, “Optimizing GPS data transmission using entropy coding compression”, *Proc. SPIE 7810*, 2010.

[22] A. G. Villafranca, S. Mignot, J. Portell, and E. García-Berro, “Hardware implementation of the FAPEC lossless data compressor for space”, *Proc. NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2010.