# FPGA design for on-board measurement of intermittency from in-situ satellite data

**N. Deak[1], O. Creț[1], C. Munteanu[2] , E. Teodorescu[2], M. M. Echim[2,3]**

[1]Computer Science Department, Technical University Cluj-Napoca, România.

[2]Institute of Space Science, Măgurele, România.

[3]Royal Belgian Institute for Space Aeronomy, Brussels, Belgium.

Corresponding authors: Norbert Deak (deaknorbi93@yahoo.com), Marius Echim (echim@spacescience.ro)

**Key Points:**

- A solution based on the FPGA technology is designed and tested to compute the flatness parameter, a key measure of intermittency in space
- The design is optimized with respect to resource usage and can be deployed on space qualified FPGAs to be operated on-board spacecraft
- Tests with space-borne data give excellent results confirmed by independent scientific software tools applied on the same test data.

## Abstract

Intermittency is a fundamental property of space plasma dynamics, characterizing turbulent dynamical variables as well as passive scalars. Its qualitative and quantitative description from in-situ data requires an accurate estimation of the probability density functions (PDFs) of fluctuations and their moments, particularly the flatness, a normalized fourth order moment of the PDF. Such a statistical description needs a sufficiently large number of samples for the computation to be meaningful. Due to inherent technological limitations (e.g., limited telemetry bandwidth) not all samples collected on-board the spacecraft can be sent to the ground for further analysis. Therefore, a technology designed to process on-board the data and to compute the flatness is useful to fully exploit the capabilities of scientific instruments installed on robotic platforms, including nanosatellites. We designed, built and tested in laboratory such a technology based on Field Programable Gate Arrays (FPGA) . The building principle is the classical estimation of PDFs and their moments, based on normalized histograms of a measure. The technical design uses the FloPoCo framework with customized arithmetic operators; the computation block is a pipelined architecture which computes a new value of the flatness in each clock cycle. The design and implementation achieve optimization directives of the FPGA resources relevant for operation in space, like area, energy efficiency, and precision. The technology was tested in laboratory using Xilinx SRL16 or SRLC32 macros and provides correct results validated with test time series provided by magnetic field data collected in the solar wind by ULYSSES spacecraft. The characteristics and performance of the laboratory prototype pave the way for a space qualified version of the laboratory design.



## Plain Language Summary

The inherently limited resources onboard spacecraft (telemetry bandwidth, computing power and memory) allow that only a fraction of the scientific data are sent to the ground, thus available for scientific analysis. Complex strategies are put in place in order to select which fraction of data will be avalaible for scientist on grpound. Another approach is to perform the key data computations on-board the spacecraft and send the results to the ground. We designed a module able to perform such calculations to estimate the flatness parameter – a key statistical descriptor of data variability helping scientists to understand the turbulent dynamics of space plasmas.


## 1 Introduction

Space plasmas are natural laboratories where turbulent phenomena can be investigated in-situ at a level of detail not reachable on ground. Therefore, the investigation of space plasma turbulence has a many-folded impact. On the one hand, it helps understanding the inter-connections and the dynamical properties of the solar system plasma environment, with implications on the strategies to be developed in order to increase the resilience of space assets to natural solar-terrestrial hazards (or space weather). On the other hand it provides insight on the fundamental properties of turbulence as a universal phenomenon. The variability of data collected in turbulent space environments covers a large spectrum of spatio-temporal scales. A leading question in turbulence studies is how the energy is transferred between scales and is dissipated at the smallest ones.

Intermittency is a key feature of turbulence. There is no universally accepted definition of intermittency, nevertheless, we adopt here the point of view that intermittency is the dynamical

property of a turbulent system to exhibit a high degree of fragmentation in the physical and dynamical space (Chang, 2015, see also recent a recent review by Echim et al., 2020). In other words, intermittency is the dynamical property of a turbulent system to be controlled by "active" elements/structures covering a large spectrum of spatio-temporal scales, and whose structure/topology changes from scale to scale. The connection with the geometrical, or fractal, description of variability is straightforward. Indeed, intermittency is often considered to be a hallmark of the topological departure from self-similarity, thus susceptible to be described by multifractal analysis (Frisch, 1995, see also Wawraszek and Echim, 2020). The tools adapted to assess and characterize intermittency from observational in-situ data collected in space at limited resolution are not trivial.

We aim to develop such tools to help scientists to advance the current understanding of turbulence in magnetized collisionless plasmas, based on in-situ measurements performed by satellites. The plasma environments targeted by the technology discussed in this paper are the solar wind and the planetary magnetospheres, but can be expanded to other types of data, including ground based or Earth observation space missions. This study is part of a broader effort meant to build a semiautonomous device equipped with functional modules designed to perform on-board nonlinear analyses of turbulent fluctuations of plasma parameters. Based on its versatility and opportunities for optimization of resources we build a technology relying on Field Programable Gate Arrays (FPGA). Several modules of this device, the On-board Architecture for Nonlinear Analysis of data (OANA) are already released, like the IP cores modules devoted for the spectral and statistical analysis of fluctuations (Deak et al., 2018, Opincariu et al., 2019). Here we discuss a new feature added to OANA, namely the FPGA technology designed to compute the moments of the probability distribution functions (PDFs) of fluctuations, namely the flatness parameter (see. its definition in the next section). As it will be described below, although the mathematical algorithm to compute the flatness is a standard one, its implementation in FPGA is not trivial and requires advanced tools like the appropriate integration of floating point numbers.

Indeed, the specification of the format and behavior of the variables represented in floating-point strongly varies, depending on the application's characteristics. An important property is the representation range. The vast majority of the programming languages support the most common floating-point formats of the IEEE-754 standard: single and double precision floating-point, and some of the operations are directly supported in hardware, by the Floating-Point Unit (FPU) found on the target architecture, if available. There are various implementations of this standard for FPGAs: some of them are fixed-width, according to the IEEE-754 standard, while others are custom-width. The IEEE-754 standard for floating-point computation supports several formats of floating-point numbers; the most widely used ones are listed in Table 1.

*Table 1. IEEE-754 2008 binary floating-point format*

| **Common name** | $p$ **(significand digits or bits)** | $w_e$ **(exponent digits or bits)** | $e_{min}$ **(in decimal)** | $e_{MAX}$ **(in decimal)** | **Max FP** |
|---|---|---|---|---|---|
| Half precision | 11 | 5 | -14 | 15 | 65504 |
| Single precision | 24 | 8 | -126 | 127 | $3.4 \times 10^{38}$ |
| Double precision | 53 | 11 | -1022 | 1023 | $1.79 \times 10^{308}$ |
| Quadruple precision | 113 | 15 | -16382 | 16383 | $1.18 \times 10^{4932}$ |

The data to be processed by the FPGA technology presented in this study may be gathered by any type of sensor, although our primary goal is to use it in the context of solar system exploration missions. Thus, targeted data can be components of the magnetic and/or electric field,

plasma moments, etc., collected in-situ. The proposed architecture to compute the flatness is custom-width floating-point format created by means of the FloPoCo generator (De Dinechin and Pasca, 2011). The design of this architecture requires not only a set of custom floating-point hardware operators, but also an a priori analysis of the data to be acquired from the sensors in order to make decisions related to the size, latency, operating frequency, power consumption etc. of the system. All the design decisions related to the format of the floating-point numbers are presented hereinafter and supported by theoretical (mathematical) proofs. The paper is organized as follows: in section 2 we review briefly the main theoretical arguments and previous works in the field, in section 3 we describe the algorithm and the technical solution adopted to compute the flatness with an FPGA architecture. The paper concludes with a summary and perspective.

## 2. Theoretical background, computational building blocks

The statistical properties of turbulent fluctuations measured in-situ in space help unfolding the multi-scale structure of astrophysical plasma turbulence and intermittency. Traditionally, intermittency can be probed with four classes of methods: (1) estimating the anomalous scaling of the structure function, (2) searching for the non-Gaussianity features of the probability distribution functions and computing the flatness parameter, (3) determining a local intermittency measure from a wavelet representation of data and (4) from the multifractal spectrum (see, for instance, Wawrzaszek and Echim, 2020, for a recent review). Two analysis methods of this set provide a quantitative estimation of intermittency suitable for a semi-automatic algorithmic implementation: (i) computing the flatness (e.g., Bruno et al., 2003) and (ii) estimating the degree of multifractality (e.g., Wawrzaszek et al., 2015). The multifractal approach has a high level of complexity that requires computing resources not available for the space systems targeted by this study. Therefore, the quantitative estimator of intermittency adopted to be implemented in FPGA is the flatness. This is also a natural option as an FPGA solution is available for estimating the multi-scale probability distribution functions (Deak et al., 2018). However, the step from computing PDFs to estimating their moments with FPGA devices is not straightforward, as will be described below.

Intermittency is often linked to the non-Gaussianity of PDFs and formation of leptokurtic wings (Bruno et al., 2003, see also Wawrzaszek and Echim, 2020). The flatness is a measure of the departure of a probability distribution function from a normal (Gaussian) distribution. Formally, the flatness is derived from a normalization of the fourth order moment of the PDF (Frisch, 1995). In order to compute the PDFs one has to define a measure of variability of the physical variable; as an example, let the targeted variable be the magnetic field intensity, *B*. Let also be this variable measured in-situ at a cadence δt, resulting in time series of a total *N* samples. The measure is constructed from the incremental time differences of *B* computed from the respective time series at different time scales, τ:

$$\delta B(t,\tau) = B(t+\Delta) - B(t) = B(t+\tau\delta t) - B(t) \qquad (1)$$

The Probability Density Distribution of *B* at scale $\Delta = \tau\delta t$ (an integer multiple, $\tau$, of the measurement resolution $\delta t$) is estimated, in its simplest form, as a normalized histogram computed for $\delta B(t,\Delta)$ for the particular scale Δ. In the conventional approach, intermittency is investigated from the scaling behavior of the PDF moments, *q*, known as the structure functions (SF), defined as:

$$SF_q = \langle \delta B^q \rangle = \int_{-\delta X_{max}}^{+\delta X_{max}} |\delta B(\Delta)|^q P(\delta B, \Delta) d\delta B \qquad (2)$$

where the integration is carried over the entire range of incremental measures (δB) computed at scale Δ. The flatness parameter, *F*, is then calculated from the fourth order structure function:

$$F = \frac{SF_4}{(SF_2)^2} \qquad (3)$$

In practical applications one needs to carefully define the size of the binning used to compute the histograms at the basis of PDF computation. Indeed, each bin has to be "populated" by a sufficiently large number of samples (in some applications, see, e.g., Echim et al., 2007, a minimum value is considered to be equal to 100 samples per bin). The estimation of the moments of the PDF are also affected by errors when only a smaller number of samples are available from measurements (Dudok de Wit and Krasnosel'skikh, 1996). Data gaps also alter the estimation of the flatness. In the current design we assume the data are uniformly sampled, no gaps are considered. Strategies for restauration of data with gaps exist (see, e.g., Munteanu et al., 2015), however, they will be considered for later stages of development. The greatest challenges for a FPGA implementation of the flatness algorithm result from the need to treat accurately floating point operations required by steps (1)-(3) described above.

There are multiple implementations of floating-point arithmetic architectures on FPGAs. For example, VFLOAT (Fang and Lesser, 2016) is an open-source variable precision floating point library, which provides basic operators (adder, multiplier, divider, reciprocal and square root) up to double precision for the two major FPGA vendors, Altera and Xilinx. The main advantage consists in the flexibility, since there is no specific target architecture. Others focus on tradeoff analysis (Munoz et al., 2010), allowing the user to choose from multiple parameters, like bit-width, area cost, elapsed time and power consumption, while providing basic operators, where the division and square root are implemented by two different algorithms. In Govindu et al. (2004) optimization of computing performance is the most important factor, which is obtained by optimizing the number of pipeline stages to obtain the best throughput rate, up to 200 MHz for the double precision operators. There are many more implementations, but two of the most appreciated, complete and freely available IP cores are the Xilinx Floating-Point Operator core[1] and the FloPoCo framework (Pasca, 2011; Dinechin and B. Pasca, 2011).

The Xilinx FP core complies with most of the IEEE-754 standard, offering the most common floating point formats listed in Table 1; however, there are also some differences which appear because the aim is to provide a better balance between resource usage and functionality. One major feature is that the Xilinx FP core supports also non-standard floating-point formats. It can have at most 80 bits for a FP number, with the exponent width in the range of 4 to 16 bits, and the fraction part from 4 to 64 bits.

FloPoCo (Floating-Point Cores) is an arithmetic core generator for FPGAs. It aims at creating new high accuracy operators with less resource usage and top performance. It can generate fully parameterizable FP operators; the user must specify the format of the FP number (the width of the exponent and significand), and optionally the target frequency, target device family, and some additional implementation optimization options (e.g. using logic or digital signal processing resources for the implementation of a multiplier, using distributed or Block RAM memory blocks, etc.). It also has some differences compared to the IEEE-754 standard. First, it uses a different FP format, even if the user specifies the single or double precision standard formats as parameters. Exceptions (zero, infinity or Not a Number (NaN)) are encoded differently, especially using some

[1] Xilinx, "LogiCORE IP Floating-Point Operator v7.0," 2 April 2014. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/floating_point/v7_0/pg060-floating-point.pdf

additional bits, thus two more FP values are available for use, which are reserved in case of the IEEE-754 for these exceptions. Also, FloPoCo does not support subnormal numbers.

The above mentioned floating-point cores for FPGAs can be used to obtain accurate FP computations; the choice of the most appropriate one depends on the design specifications. However, since the aim of our design is to perform as many optimizations as possible ("*computing just right*", i.e. with the exact amount of computational resources necessary), FloPoCo is the chosen framework, because it offers more flexibility and optimization techniques than Xilinx FP core.

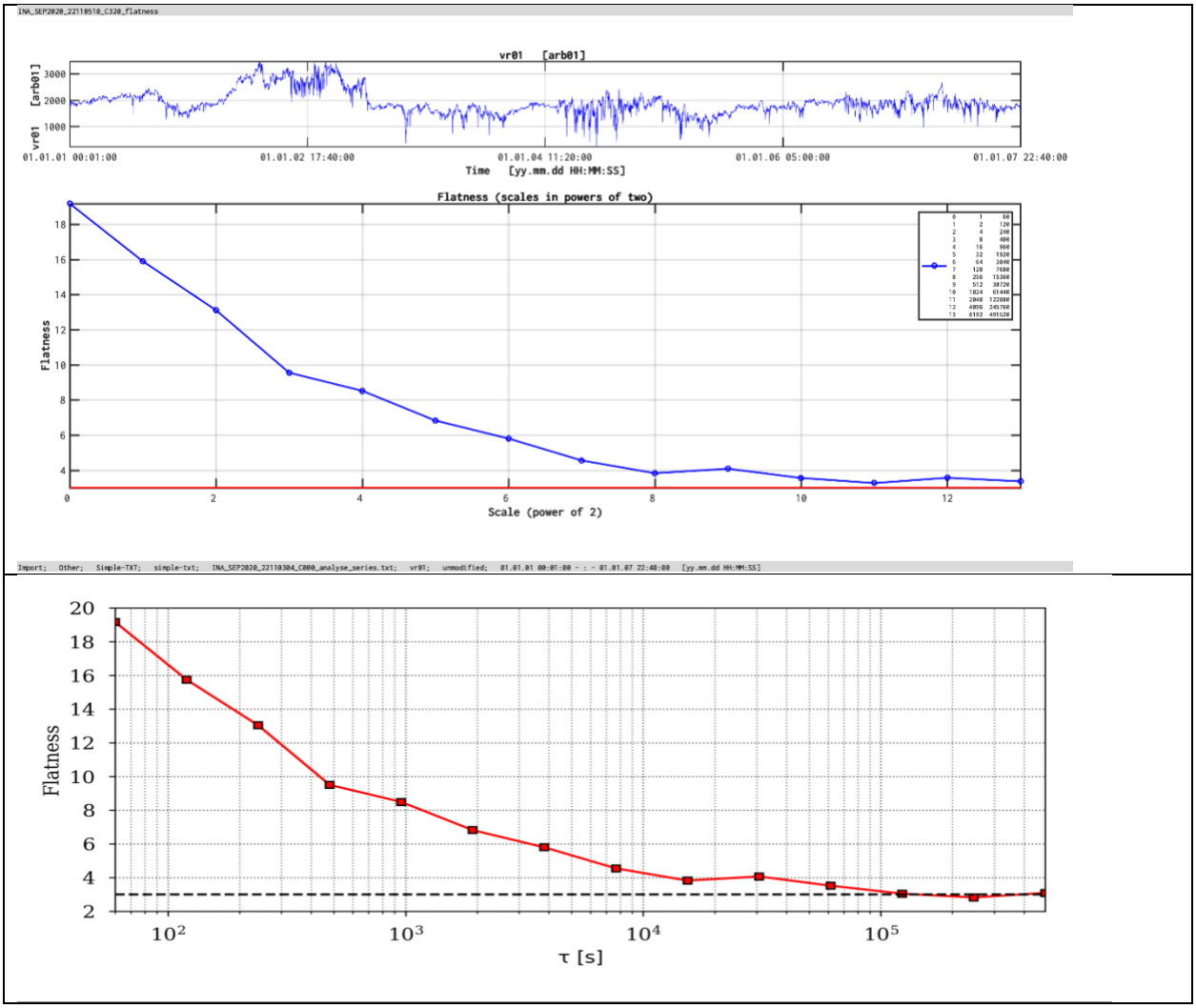


*Figure 1. Example of flatness calculation for a synthetic time series. The signal shown in the upper panel is derived by a multiplication by $10^3$ of the magnetic field measurements performed by ULYSSES in the solar wind. The middle panel show the flatness calculated by INA software (Munteanu, 2017, see also http://www.storm-fp7.eu/index.php/data-analysis-tools); the lower panel show the flatness calculated with ODYN software (Teodorescu and Echim, 2020).*

There are a few implementations of Flatness on FPGA devices, but none in the field of solar system exploration. Shyu and Li (2006) propose an algorithm to compute flatness and to measure non-Gaussianity in order to separate independent sources from their mixtures in the context of independent component analysis (ICA). The design implements its own FP arithmetic datapath to provide better accuracy and higher dynamic performance instead of using fixed-point

operations. However, the FP operations are simplified and instead of providing an exact value for the Flatness, the algorithm provides a value relative to the Gaussian variance to find the maximum non-Gaussianity. Quirós-Olozábal et al. (2016) compute the Spectral Kurtosis in order to detect the existence of low level harmonics in power distribution. This algorithm is implemented on a low-cost FPGA as a real-time analyzer. The authors designed the system mostly for smart grids to get faster responses regarding the power quality. The input for the Spectral Kurtosis processor is the output data from an FFT block, and it uses the Xilinx Floating-Point Core to compute the results in single-precision FP format.

In these previous works the flatness is estimated from equation (3) and work with a fixed floating point format, on 32-bits. All the previous architectures use either simplified FP operators, or the Xilinx FP Core. Therefore, their results have either a limited accuracy, or an increased resource usage, as it will be shown later. In space applications, however, the emphasis is on resources and not on execution speed. Indeed, the data acquisition rate from most of space sensors targeted by our application, namely in-situ measurements in the solar system plasmas, is of the order of hundreds of Hertz, while most FPGAs have an operational frequency of a few hundreds of MHz. Thus, the main optimization criterion to be fulfilled is the occupied area in the FPGA chip. By minimizing the resource usage, it becomes possible to implement more designs in the same FPGA chip, thus increasing the overall level of parallelism in the digital signal processing algorithms run on the satellite.

Various approaches to compute flatness on-ground are adopted by scientific software tools designed for data analysis. Two such solutions are provided by (i) the Integrated Nonlinear Analysis (INA) library (Munteanu, 2017, see also http://www.storm-fp7.eu] and (ii) the open source software ODYN (Teodorescu and Echim, 2020). These are publicly available software applications designed to extract various nonlinear data descriptors, including the flatness, from the analysis of time series available on ground, as higher order data products provided by space instrumentation. The mathematical kernels of the two software tools are similar, based on equation (3), and are the starting point for the development of the FPGA solution presented here.

The PDFs of the time series are computed for a number of $N$ scales; each scale comprises a number of points equal to $2^{\tau}$, where $\tau$ takes values between $\tau_{min}$, in general equal to 0, and ($\tau_{max}$-1), where $\tau_{max}$ is the smallest power of 2 for which $2^{\tau max}$ is still larger than the total length of the time series (Munteanu, 2017, Teodorescu and Echim, 2020). The PDFs are obtained by moving a sliding window of length $2^{\tau}$ over the entire time series and by taking the differences defined by (1). The window is displaced by one point at each step, thus consecutive windows overlap. The normalized histogram of the differences/increments gives the PDF at that scale. For each PDF, at each scale, the flatness is computed with formula (3) above. An example of such calculations performed with INA and ODYN is shown in Figure 1.

## 3 Design and Implementation of an FPGA solution to compute the flatness parameter

The FPGA design for flatness calculation is a general-purpose one, however, its primary application field is defined for space applications in the framework of exploration of solar system plasma. Thus, the system must work on-board a satellite and compute the Flatness value of the data samples gathered by on-board sensors. We consider a number of $N$ = 10000 sample points and we compute the Flatness value for a series of scales $\tau$ = {1, 2, 4, 8, 16, 32, 64, 128, 512, 1024, 2048, 4096, 8192}, so we need to compute the differences (1) for each scale $\tau$: $B(t) - B(t\text{-}\Delta)$, where $B(t)$ is the input data sample point at time $t$. Each incoming sample is represented in the *Two's*

*Complement* numbering system on 16 bits. The equation equation (3) can be rewritten as below, to simplify it for our design.

$$F = \frac{\frac{\sum|\Delta B|^4}{N-\tau}}{\left(\frac{\sum|\Delta B|^2}{N-\tau}\right)^2} \tag{4}$$

$$F = \frac{(N-\tau) \times \sum|\Delta B|^4}{(\sum|\Delta B|^2)^2} \tag{5}$$

The main steps of the computation are defined in Algorithm 1 that describes the implementation of equation (3) in a more procedural way.

**Algorithm 1.**
**Flatness computation of data variance**

1. $SumNominator \leftarrow 0,\ SumDenominator \leftarrow 0,\ Count \leftarrow 0$
2. **For each** $t$ **from** 0 **to** $N$
3. *Read* measured data sample $\boldsymbol{B}(t)$ at data acquisition rate ($CLK_A$)
4. *Save* it in the corresponding resource
5. **For each** $\tau$
6. $\Delta\boldsymbol{B}(t) \leftarrow \boldsymbol{B}(t) - \boldsymbol{B}(t-\Delta)$
7. $SumNominator \leftarrow SumNominator + |\Delta\boldsymbol{B}(t)|^4$
8. $SumDenominator \leftarrow SumDenominator + |\Delta\boldsymbol{B}(t)|^2$
9. $Count \leftarrow Count + 1$
10. $Flatness \leftarrow \frac{SumNominator \times Count}{SumDenominator^2}$
11. **End for**
12. **End for**

Steps 3 and 4 in Algorithm 1 present the data acquisition part (see Figure 2). They use the specific feature of Xilinx FPGAs allowing to store all the samples in slice look-up tables (LUTs), where each LUT is configured as a shift register using Xilinx SRL16 or SRLC32 macros. This way, by chaining them together, we obtain a large shift register, from where each required data sample is available, when computing the differences *ΔB*. This solution for Data Acquisition Block is similar to the one presented in Deak et al. (2018).

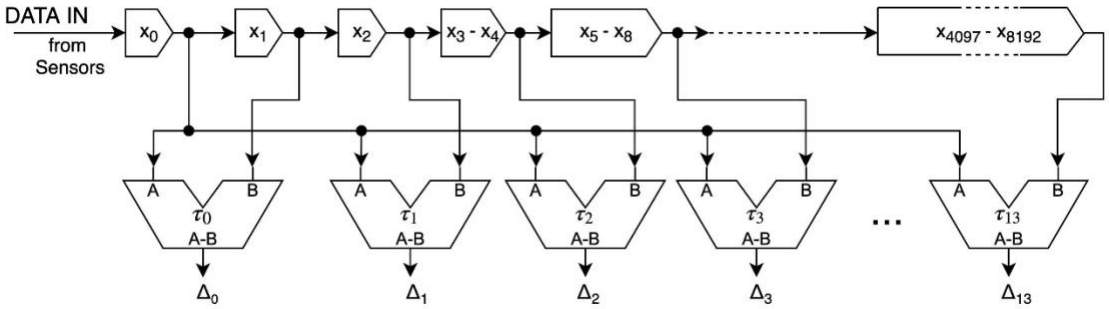


*Figure 2. FIFOs (LUTs configured as shift registers, SRL16 / SRLC32) providing access at the data samples with a step of* τ *(from Deak et al., 2018)*

## Flatness computation block

The flatness computation block consists of a hardware implementation of steps 6-10 described in Algorithm 1. The main design includes a Flatness computation block for each scale Δ. The Flatness value is a real number requiring floating-point arithmetic. However, some operations can be performed on integer arithmetic. In fact, the only operation where FP numbers are needed is the division; even the multiplication by the number of elements can be done in integer arithmetic. Thus, throughout the design, only three customized FP operators are needed: two for transforming the integers to FP numbers, and one for the division. The FloPoCo framework (Pasca, 2011) includes a special operator, called Fix2FP, which takes as input a fixed-point format number, in our case, an integer, and transforms it in a floating-point format. We have used two of these custom operators from the FloPoCo framework: one for the nominator and one for the denominator, which constitute the operands of the floating-point division operator (see equation 6 below), FPDiv (the latter being the third operator).

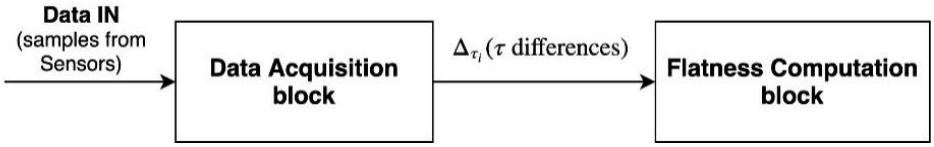


*Figure 3 . The main block diagram of the design*

These FP operators are generated by FloPoCo and are fully customizable, both the fixed-point ones (bit width and location of the decimal point) and the floating-point ones (bit width of mantissa and exponent, respectively). Thus, the specific FP format used in the design will always be the most efficient one. Figure 3 illustrates the block diagram of the Flatness computation module. The inputs to this block represent the number of elements already processed (*Count* from Algorithm 1) and the current data sample ($\Delta B(t)$). The output is the Flatness value in FP format.

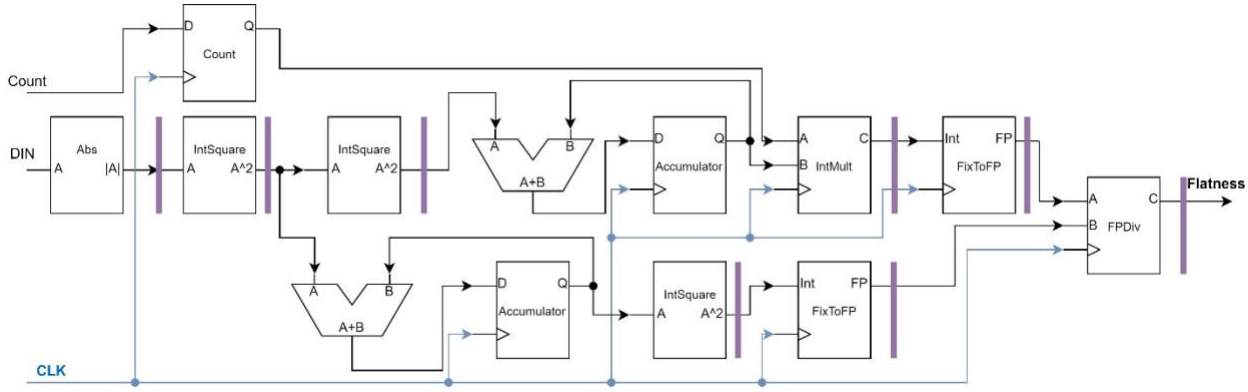


*Figure 4. The block diagram of the Flatness computation block. The thick vertical lines are the registers that delimit the stages of the pipeline*

The Flatness computation block is a pipelined architecture which computes a new value in each clock cycle, as illustrated by Figure 4. The FloPoCo framework automatically creates pipelined operators in order to divide the critical path delay and obtain higher speed, if not specified otherwise. The pipeline depth can differ inside each operator, depending on the complexity of the operation and the FP format (e.g. the FP division has a pipeline length of 12 for the IEEE-754 single precision format from Table 1, while transforming a 96 bit width fixed point to the same FP format, Fix2FP, takes 7 clock cycles). The FloPoCo operators may introduce

various delays, therefore additional pipeline registers are added in order to balance the various data paths of the data-flow graph. These pipeline registers are indicated in purple continuous lines in Figure . As the format of the floating-point numbers changes, the overall pipeline length of the Flatness computation block also changes: larger FP formats (more bits for the mantissa and exponent) increase the pipeline depth of the FloPoCo operators, thus it takes more clock cycles to obtain the final result. However, this is not a true limitation as the resolution of targeted sensors is well below the FPGA operational frequency.

## 4 Flatness calculation with FPGA: experimental tests and results

The technical approach adopted in our design is based on a bit-width evaluation of the flatness at hardware level, as detailed below.

### Bit-width computation

The FloPoCo framework allows the hardware designer to customize the most important features of the arithmetic operators. Thus, we can the minimize the required size of the FP numbers in our design, and this way the amount of resources is guaranteed to be minimal. First, since the input sample, $X_i$, is on 16 bits, we define:

$$X_i \in (-2^{15}, 2^{15}) \;\Rightarrow\; \Delta X \in (-2^{16}, 2^{16}) \;\Rightarrow\; |\Delta X| \leq 2^{16}. \tag{6}$$

Then, in case of the nominator:

$$\begin{gathered} x = RN(|\Delta X|) \\ \Rightarrow \quad x \leq 2^{16}, \quad x = M \cdot 2^{e-p+1} \\ \Rightarrow \quad x^4 \leq 2^{64} \\ \Rightarrow \quad N \times \sum x^4 \cong N \times 1000 \cdot 2^{64} \leq N \times 2^{74} \leq 2^{14} \times 2^{74} = 2^{88} \cong 3.095 \cdot 10^{26} \end{gathered} \tag{7}$$

As one can see, the nominator's value is of the order of $10^{26}$ in the worst case, and the denominator will be slightly smaller. Thus, the architecture that implements the Flatness computation needs a FP format capable of representing $10^{26}$ (or $2^{88}$) as the maximal value (otherwise overflow will occur). Table 1 illustrates the maximum FP number for each common FP format. For the case of the Flatness computation algorithm, half-precision is too small, and single-precision is too much. The best solution is to find a custom FP format, which is not necessarily an IEEE-754 standard, so that its precision is the minimum allowing a correct representation of the worst-case values of the nominator. The largest FP number can be written as in Muller et al. (2018) and it must be greater than $2^{88}$ in order to perform computations without overflow:

$$(2 - 2^{1-p}) \cdot 2^{e_{MAX}} \geq 2^{88} \tag{8}$$

Thus, this formula depends both on $p$ and $w_e$. But $2^{1-p}$ is very small, and it can be left out to obtain a lower bound on the exponent:

$$2^{e_{MAX}+1} \geq 2^{88} \Rightarrow e_{MAX} \geq 87 \tag{9}$$

From this, we can determine that $e_{MAX} = 87$, which means that we need 8 bits for the exponent.

Another similar computation would be done for determining the precision of the FP numbers (the number of bits after the decimal point). One way to constraint this is to decide how big the maximal allowed error is. That is, if $ROUND$ is one of the well-known rounding modes that are used in floating-point computations (*round towards negative* or *round down*, *round towards positive* or *round up*, *round towards zero* and *round to nearest*), and $x$ is a real number, the error $\epsilon$ is:

$$|ROUND(x) - x| = \epsilon \leq ulp(x). \tag{10}$$

where *ulp*(*x*) indicates the unit in the last place of number *x*.

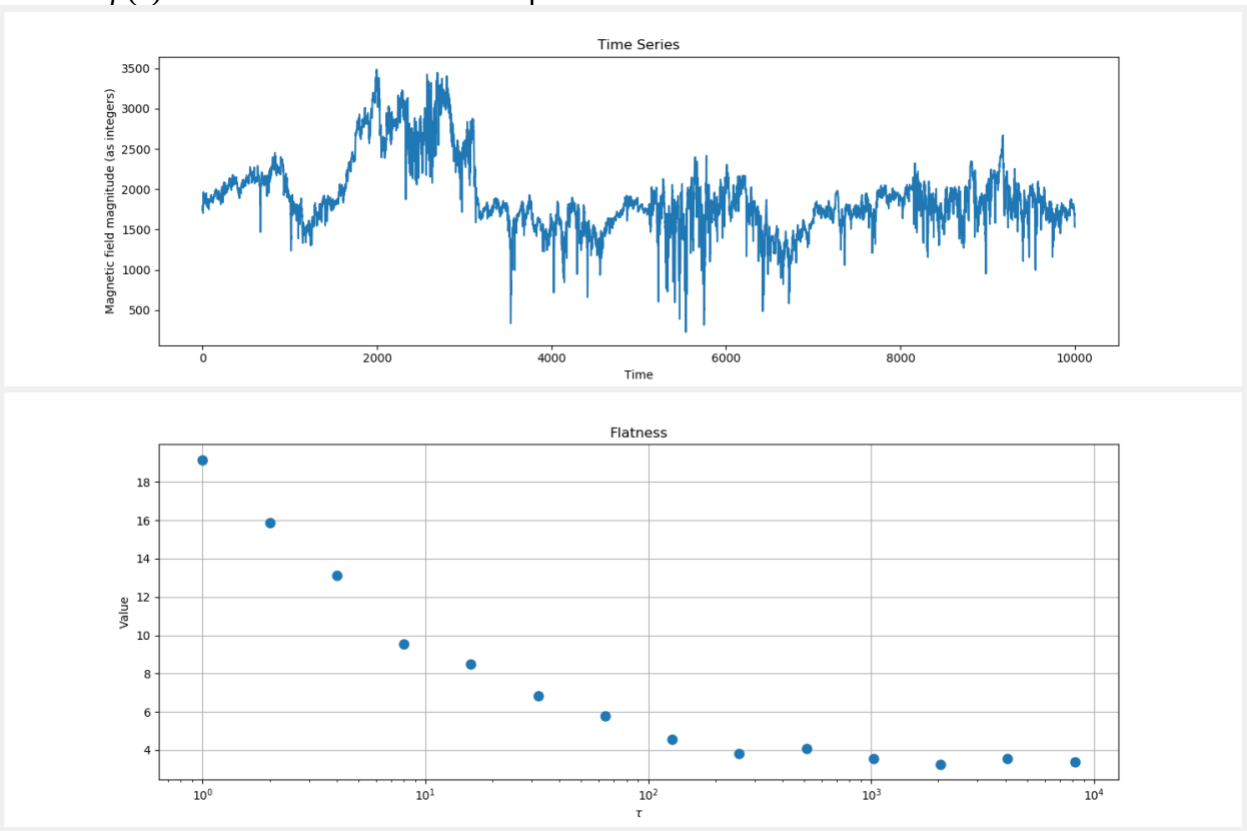


*Figure 5. Flatness for a sample data set for various values of τ. The analysis is performed in the same signal as shown in Figure 1 and gives similar results as the scientific software INA.*

The next step is to assume an upper bound for the Flatness value, for which we want $\epsilon$ to be smaller than a specified $\epsilon_{MAX}$. Since Flatness is used to quantify intermittency, thus departure from Gaussianity manifesting as deviations of the flatness from the reference value $F_G$=3 corresponding to a Gaussian PDF, we accept an upper bounding value of the flatness equal to $F_{max}$ = $2^4$. Flatness values greater than $F_{max}$ will then be cut-off to $F_{max}$ but this is not a problem in practical solar system plasma exploration where the flatness takes rarely such extreme values. Therefore, using the definition of $ulp$ applied to equation (4), we obtain:

$$x \in [2^3, 2^{3+1}) \Rightarrow ulp(x) = 2^{3-p+1} \leq \epsilon_{MAX} \tag{11}$$

For the $\epsilon_{MAX}$ we can define a small value, depending on how precise we want the result to be. For example, if we just want the result to have approximately 1 correct digit after the decimal point, then $\epsilon_{MAX} = 10^{-1} \leq 2^{-4}$, so we get:

$$2^{3-p+1} \leq 2^{-4} \Rightarrow p \geq 8. \quad (12)$$

We have obtained the minimum value for $p$, which guarantees that $\epsilon$ will always be smaller than $2^{-4}$.

Based on equations (9) and (12), we obtain the minimum format for our FP arithmetic: $(8, 8, -126, 127)$, which means that the significand $p$ is on 8 bits, the exponent is on 8 bits ($w_e$), $e_{min}$ is -126 and $e_{MAX}$ is +127 (see Table 1). So the exponent is on 8 bits, and the significand also on 8 bits. This way, we reduce the format of the FP representation of our operands to a total of 16 bits, with the desired precision. This is not a standardized format, but a custom one, made possible by the customization capabilities offered by FloPoCo. Due to the fact that the design is implemented in an FPGA device, we are able to "compute just right": there is no need to adjust existing FP operators, but we allocate and use the exact amount of hardware that is necessary to implement our task. By contrast, in a regular Central Processing Unit (CPU) these operators are fixed and one needs to use those that approximate the best the operators that are needed in a specific application, thus wasting a significant amount of logic resources.

## Results

To prove that the bit-width computation in our approach reduces the resource utilization, we have generated more results concerning the FPGA chip usage for various values of the resolution ($\epsilon_{MAX}$). The FPGA devices include general logic resources, *i.e.* slice Look-Up Tables (LUTs), Flip-Flops, and RAM memory blocks (BRAM), but also specific blocks which are dedicated to Digital Signal Processing (DSP blocks) and which include embedded adders and multipliers able to work at high speed. The resource allocation obtained for each of these designs is described in Table 2. While the logic resources and the flip-flops increase when $\epsilon_{MAX}$ decreases, the DSP blocks usage doesn't change, because DSP blocks are only used for integer multiplication.

*Table 2. Resource utilization for different $\epsilon_{MAX}$ values*

| $\epsilon_{MAX}$(if $x < 2^4$) | ~decimal | $p$ | Slice LUTs | Flip-Flops | BRAM blocks | DSP blocks |
|---|---|---|---|---|---|---|
| $2^{-4}$ | 0.0625 | 8 | 38177 | 41409 | 0 | 182 |
| $2^{-7}$ | 0.0078125 | 11 | 40968 | 42279 | 0 | 182 |
| $2^{-10}$ | 0.0009765625 | 14 | 44349 | 43526 | 0 | 182 |
| $2^{-14}$ | 0.000061035 | 18 | 48995 | 45152 | 0 | 182 |
| $2^{-20}$ | $9.53674316 \times 10^{-7}$ | 24 | 58427 | 49382 | 0 | 182 |

The technology was tested on a time series simulating data acquired by a space magnetometer. The data set included 10000 samples and is identical with the time series analyzed with INA (solar wind data from ULYSSES spacecraft, Balogh et al., 1992) and presented in Figure 1. This time series is an input into the data acquisition module (see Figure 2) and forward processed by the flatness computation module described above. The Flatness parameter was computed for the same scales considered by INA and ODYN, with $\tau \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192\}$. The results are shown in Figure 5 and indicate an intermittent behavior. Indeed, the flatness takes values around 3 for the largest scales, then starts to increase

with decreasing scales. These results are fully consistent with the results provided independently by INA and ODYN scientific software, which confirms the FPGA application provides correct results.

## Summary and perspective

Starting from a mathematical algorithm designed for space plasma time series analysis on-ground (available from the Integrated Nonlinear Analysis library as well as on open source technology, Teodorescu and Echim, 2020), we built IP cores for computing the flatness parameter for a signal that simulates data acquired on-board a spacecraft. The FPGA application is optimized and tested on XILINX FPGA board. The Flatness computation block is a pipelined architecture which computes a new value of the flatness parameter in each clock cycle. The application is optimized such that a minimum number of floating point operations are required.

In order to achieve a minimal use of resources, the maximum value of flatness is limited to $2^4$. Thus, larger values of the flatness are cut-off to this maximal limit. Nevertheless, since a signal is considered intermittent at a scale if the flatness at that scale is larger than 3, this limitation does not have a negative impact on the operational goal of the FPGA device. In addition the design capture the variation of flatness of scales for a wide range of values and scales. The application achieves its goal to identify intermittency based on flatness values. Numerical tests performed on relevant signals demonstrate the technology correctly detects departure of the flatness from the Gaussian value for a significant range of scales. The results are confirmed by an independent test by INA and ODYN scientific software applied on the same signal.

The release of an FPGA solution for computing the flatness is timely. Indeed, its usefulness goes beyond the goal of studying fragmentation in the phase space of a turbulent dynamic system, like solar system plasmas. Due to its simplicity and versatility the solution can be adapted for various other autonomous data analysis contexts where departure from “normal” behavior, in the sense of Gaussian statistics, is of interest.

### Acknowledgments, Samples, and Data

This work was supported by the Romanian Space Agency (Project STAR OANA), the Ministry of Research and Innovation via a PCCDI grant (18PCCDI/2018) and Program NUCLEU LAPLAS. ME acknowledges support from the Belgian Solar Terrestrial Center of Excellence (STCE). The datasets used to test the IP cores are magnetic field records (Balogh et al., 1992) provided by ULYSSES spacecraft; these data are publicly available from the Ulysses Final Archive (http://ufa.esac.esa.int/ufa/) maintained by the European Space Agency.

### References

Balogh, A.; Beek, T. J.; Forsyth, R. J.; Hedgecock, P. C.; Marquedant, R. J.; Smith, E. J.; Southwood, D. J.; Tsurutani, B. T. (1992), The magnetic field investigation on the ULYSSES mission - Instrumentation and preliminary scientific results, Astronomy and Astrophysics Supplement Series, vol. 92, no. 2, p. 221-236

Bruno R, Carbone V, Sorriso-Valvo L, Bavassano B. (2003), Radial evolution of solar wind intermittency in the inner heliosphere. Journal of Geophysical Research (Space Physics) 108 1130. doi:10.1029/372 2002JA009615

Chang, T. (2015), *An Introduction to Space Plasma Complexity*, Cambridge University Press

De Dinechin F., and B. Pasca, (2011) "Designing custom arithmetic data paths with FloPoCo," *IEEE Design & Test of Computers*, vol. 28, no. 4, pp. 18-27

Deak, N.; Creţ, O.; Echim, M.; Teodorescu, E.; Negrea, C.; Văcariu, L.; Munteanu, C.; Hângan, A. (2018), "Edge computing for space applications: Field programmable gate array-based implementation of multiscale probability distribution functions," *Review of Scientific Instruments*, vol. 89, no. 12, p. 125005

Echim, M. M., Lamy, H., and Chang, T. (2007),Multi-point observations of intermittency in the cusp regions, *Nonlin. Processes Geophys*., 14, 525–534, doi:10.5194/npg-14-525-2007

Echim, M., T. Chang, P. Kovacs, A. Wawrzaszek, E. Yordanova, Y. Narita, Z. Vörös, R. Bruno, W. Macek, K. Mursula, and G. Consolini (2021), "Turbulence and Complexity of Magnetospheric Plasmas," chapter 5 of *Space Physics and Aeronomy Collection Volume 2: Magnetospheres in the Solar System*, Geophysical Monograph 259, First Edition. Edited by Romain Maggiolo, Nicolas André, Hiroshi Hasegawa, and Daniel T. Welling. © 2021 American Geophysical Union. Published 2021 by John Wiley & Sons, Inc., DOI: 10.1002/9781119507512, pp. 63–88

Fang, X. and M. Leeser (2016), "Open-source variable-precision floating-point library for major commercial fpgas," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 9, no. 3, p. 20

Frish, U., *Turbulence: The Legacy of A. N. Kolmogorov*. Cambridge: Cambridge University Press, Cambridge, UK., 1995

Govindu, G., Zhuo, V, Choi, S. and V. Prasanna (2004), "Analysis of high-performance floating-point arithmetic on FPGAs," *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, p. 149

Muller, J.-M., N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol and S. Torres (2018), Handbook of Floating-point Arithmetic (2nd edition), Birkhäuser Basel

Muñoz, D. M., Sanchez, D. F., Llanos, C. H. and M. Ayala-Rincón (2010), "Tradeoff of FPGA design of a floating-point library for arithmetic operators," *Journal of Integrated Circuits and Systems*, vol. 5, no. 1, pp. 42-52

Munteanu, C., "Turbulent fluctuations and discontinuities in the solar wind: statistical properties and possible effects on the terrestrial plasma environment", Ph.D. Thesis, UNIVERSITY OF BUCHAREST, Faculty of Physics, Doctoral School of Physics, Bucharest, 2017

Opincariu, L., Deak N., Creţ O., Echim M., Munteanu, C. and L. Văcariu (2019), "Edge computing in space: Field programmable gate array-based solutions for spectral and probabilistic analysis of time series," in Review of Scientific Instruments, 90, 114501, https://doi.org/10.1063/1.5119231

Pasca, P.M., *High-performance floating-point computing on reconfigurable circuits,* Ph.D. Theis Lyon: Ecole normale superieure, 2011

Quirós-Olozábal, Á, González-de-la-Rosa, J.-J., Cifredo-Chacón M.-Á. and J.-M. Sierra-Fernández (2016), "A novel FPGA-based system for real-time calculation of the Spectral Kurtosis: A prospective application to harmonic detection," *Measurement,* vol. 86, pp. 101-113

Shyu K-K and Li M-H (2006), "FPGA Implementation of FastICA based on Floating-Point Arithmetic Design for Real-Time Blind Source Separation," The 2006 IEEE International Joint Conference on Neural Network Proceedings, Vancouver, BC, pp. 2785-2792, doi: 10.1109/IJCNN.2006.247185.

Teodorescu, E.; Echim, M. M. (2020), Open-Source Software Analysis Tool to Investigate Space Plasma Turbulence and Nonlinear DYNamics (ODYN), Earth and Space Science, Volume 7, Issue 4, article id. e01004

Wawrzaszek A, Echim M, Macek WM, Bruno R. (2015) , Evolution of intermittency in the slow and fast solar wind beyond the ecliptic plane. The Astrophysical Journal Letters, 814, L19

Wawrzaszek, A., Echim, M. (2021), On the Variation of Intermittency of Fast and Slow Solar Wind With Radial Distance, Heliospheric Latitude, and Solar Cycle, *Forntiers in Astronomy and Space Science*, 13 January 2021 https://doi.org/10.3389/fspas.2020.617113