Fast Hardware Design of $1/f$ Fractional Brownian Noise Generator

G. Loss and R. Coelho

***Abstract*—This paper proposes the design of a fast and accurate $1/f$ fractional Brownian motion (fBm) noise generator. The solution is based on the successive random addition algorithm using the midpoint displacement (SRMD) technique. The implementation was performed on a high-speed field-programmable gate array (FPGA) Development Kit. It enabled the generation of 150 million of 16-bit noise samples per second. The $1/f$ noise generator achieved 7.4% of the logic elements, 52% of the RAM memory and 1.6% of the ROM memory. The accuracy of the $1/f$ noise samples was evaluated for different sequence size and confidence intervals. The noise samples pattern was examined by the probability density (PDF) and the heavy-tail distribution (HTD) functions. The results also include the auto-correlation function (ACF) to show the low-frequency statistics of the $1/f$ noise samples. For the investigation it was used the real $1/f$ *speech-babble* acoustic noise parameters. The results showed that the proposed $1/f$ noise generator design can be promising to the performance evaluation of communications channels with low bit error rate (BER) values. It can also be interesting for signal processing applications and other areas concerned with noisy conditions.**

***Index Terms*—$1/f$ noise, fractional Brownian noise, low-frequency statistics, field-programmable gate array.**

I. Introduction

THE provision of fast and accurate noise generators has become an important issue for digital communications systems which attained reliable and high-speed channels. Moreover, $1/f$ noise [1] has been observed in several physical systems such as, communication channels [2], electronic components and semiconductors devices [1] [3], natural phenomena (e.g., rivers, ocean flows, average seasonal temperature, rainfall) [4], financial markets [5], image [6], acoustics and music [7]. The $1/f$ noise is a random process characterized by its low-frequency or scaling invariance over a wide range of timescales [8]. White noise represented by the Gaussian distribution, is widely employed since it enables tractable analysis. But, since the presence of $1/f$ noise is a reality, its investigation became a key issue. The fractional Brownian motion (fBm) [8] [9] [10] leads to an efficient representation of the $1/f$ or fractional noise samples with low-frequency statistics or scaling invariance degree represented by the Hurst ($0 < H < 1$) parameter [4].

Therefore, the main contribution of this paper is the proposal of a fast design of $1/f$ noise generator based on the fractional Brownian motion. This enables more realistic and accurate estimation of low bit error rate (BER) (10^{-9} to 10^{-12}) in communications channels. It should be also applied to signal processing and other scientific areas concerned with noise issues. Hardware architectures for white Gaussian noise generators [11] [12] [13] have been reported in the literature. However, from the authors knowledge, the proposition of a fast hardware design of $1/f$ fBm noise generator is addressed for the first time in this work. This $1/f$ noise solution describes a hardware design for the successive random addition method using the midpoint displacement (SRMD) technique introduced by Mandelbrot [9]. In a preliminary study [14] the SRMD design was proposed and evaluated for the generation of optical signal samples. The results showed to be very promising.

The fractional noise generator was implemented on Altera Stratix EP1S25 FPGA Development Kit using the Quartus II v5.0 suite for Linux. The design proposed in this work also considered the trade-off between the device resources usage (e.g., memory, logic elements and embedded multipliers) to produce fast and accurate $1/f$ 16-bit noise samples. The design also includes a Gaussian random variables (RVs) block generator. The Box-Muller [15] method is the most important approach used for the generation of Gaussian RVs. The generation is based on transformation functions performed on Uniform variables. In [11] the authors proposed a hardware architecture based on the Box-Muller algorithm and the Central Limit Theorem (CLT) [16]. This solution was also performed by Xilinx [17] and evaluated under different FPGA devices by the authors in [12] [13]. The results demonstrated that the implementation based on both Box-Muller algorithm and the CLT achieves accurate and fast Gaussian RVs. Hence, this method was used for the Gaussian samples generation. Nevertheless, the solution presented in this work introduces a memory optimization for the Box-Muller functions evaluation. The Tckacik algorithm [18] was applied for the Uniform RVs generation. This technique achieved fast generation, good randomness and few device resources requirements.

The proposed design performance was examined in terms of its device resources requirements and the representation of the pattern and low-frequency statistics (i.e., evaluations of the probability density (PDF), heavy-tail distribution (HTD) and auto-correlation coefficient (ACF) functions) of the $1/f$ noise samples. For the validation experiments, it was considered the mean (m), standard-deviation (σ) and scaling parameter (H) estimated from the real $1/f$ *factory2* acoustic noise of the NOISEX-92 [19] database. The implementation also included a pure-Brownian noise, i.e., with $m = 0$, $\sigma = 1$ and $H = 1/2$. The results accuracy is examined and presented considering different samples sequence sizes and confidence intervals (CI) [20]. For the low-frequency statistics (H parameter) estimation it was used the wavelet-based method [21] considering

G. Loss and R. Coelho are with the Electrical Engineering Department, Military Institute of Engineering (IME), Rio de Janeiro, Brazil, e-mail:{gloss,coelho}@ime.eb.br.

Daubechies filters with 12 coefficients.

II. 1/f FRACTIONAL BROWNIAN MOTION NOISE

Generally, noises can be represented by the power spectral density fluctuations per a frequency interval. Or, similarly, by its variations over a time scale of order $1/f^{\beta}$ ($0 \leq \beta \leq 2$). In the literature, white, red/brow and pink noises are also referred as $1/f^0$, $1/f^2$ and $1/f$ noises, respectively.

The $1/f$ noise is a random process with power spectral density ($S(f)$) defined as $S(f) = \frac{1}{f^{\beta}}$ where $0 \leq \beta \leq 2$. Mandelbrot showed that $1/f$ fractional noises have $S(f)$ defined by $S(f) = f^{1-2H}$ with H parameter values frequently close to 1. Thus, H parameter is the exponent of the power spectral density of the $1/f$ fractional noises. The H parameter expresses the low-frequency statistics, i.e., the scaling or time-invariance degree, of a random process. It can also be defined by the decaying rate of the auto-correlation coefficient function $\rho(k)$ ($-1 < \rho(k) < 1$) as $k \to \infty$. A random process can be classified by its H ($0 < H < 1$) parameter as

- Anti-persistent process ($0 < H < \frac{1}{2}$): The ACF varies rapidly and $\sum_{k=-\infty}^{\infty} \rho(k) = 0$.
- Short-range dependence process (SRD) ($H = \frac{1}{2}$): The ACF $\rho(k)$ exhibits an exponential decay to zero, such that $\sum_{k=-\infty}^{\infty} \rho(k) = c$, where $c > 0$ is a finite constant.
- Long-range dependence process (LRD) ($\frac{1}{2} < H < 1$): The ACF $\rho(k)$ is a slowly-vanishing function. It has slow variations meaning a strong dependence degree between samples that are far apart or $\sum_{k=-\infty}^{\infty} \rho(k) = \infty$. Thus, the $S(f)$ is concentrated in the low-frequency spectrum.

Harold Edwin Hurst [4] found the presence of low-frequency or LRD statistics ($H = 0.72 \pm 0.006$) in several observations of different natural phenomena. Brownian motion is a well known example of a SRD process while $1/f$ fractional noises, by definition, are LRD random processes [9].

The fractional Brownian motion [8] [9] is a random process ($X_H(t)$) with Gaussian independent increments, indexed by the single scalar H parameter defined in $\Re$ with zero mean and continuous sample path (null at origin). The variance of the increments is proportional to its time interval as $Var[X(t_2) - X(t_1)]\ \alpha\ |t_2 - t_1|^{2H}$ for $0 \leq t_1 \leq t_2$. For all t_1 and t_2 instants it follows that: $X_H(t)$ has stationary increments; $X_H(0) = 0$ and $E[X_H(t)] = 0$ for any instant t ; and $X_H(t)$ presents continuous sample paths. This means that fBm is a self-similar random process, i.e., its statistical characteristics hold for any time scale. For any τ and $r > 0$, it follows that $[X_H(t + \tau) - X_H(t)_{\tau \leq 0}] \overset{d}{\cong} r^{-H}[X_H(t + r\tau) - X_H(t)_{\tau \leq 0}]$, where r is the scaling factor and $\overset{d}{\cong}$ means similar in distribution. Thus, $X_H(t)$ is a random process completely characterized by its mean (null), variance (σ^2) and H parameter.

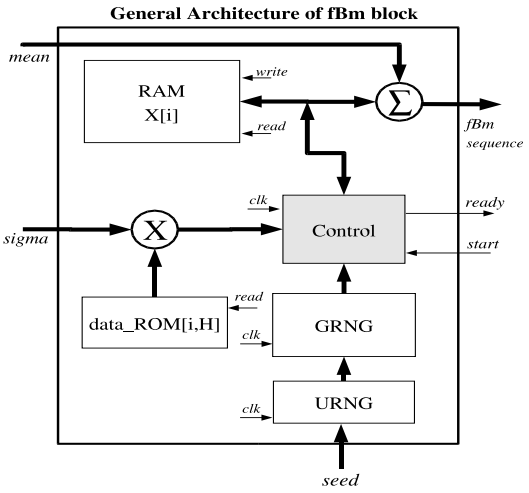
III. SRMD ALGORITHM AND THE 1/f FBM-BLOCK DESIGN

Fig. 1 shows the SRMD pseudo-code and the block diagram of the proposed $1/f$ fBm noise generator. Consider a time index t defined at the interval $[0, 1]$. Setting $X(0) = 0$ and $X(1)$ as a Gaussian RV with zero-mean and variance σ^2

```
begin

    for i:=1 to maxlevel do
        delta[i]:= sigma * power(0.5,i*H) * sqrt(0.5) *
                          * sqrt( 1-power(2,2*H-2) )
    end for
    N:= power(2,maxlevel)
    X[0] := 0
    X[N]:= sigma * Gauss(• )
    D:= N
    d := D/2
    level:= 1
    while (level<=maxlevel) do
       for i:= d to N-d step D do
            X[i]:= 0.5 * (X[i-d] + X[i+d])
       end for
       for i:=0 to N step d do
            X[i]:= X[i] + delta[level] * Gauss(• )
       end for
       D:= D/2
       d := d/2
       level:= level+1
    end while
end
```

(a)



(b)

Fig. 1. $1/f$ Noise Generator: (a) SRMD Algorithm [9], (b) $1/f$ fBm-block Design.

then, $Var[X(1) - X(0)] = \sigma^2$ and $Var[X(t_2) - X(t_1)] = |t_2 - t_1|\sigma^2$ for $0 \leq t_1 \leq t_2 \leq 1$. To achieve this property a random offset displacement (D_i) with mean zero and variance δ_i^2 must be added to the samples where $\delta_i^2 = 1/2^{-(i+1)}\sigma^2$. For example, the $X(1/2)$ value is obtained by the interpolation of $X(0)$ and $X(1)$ with variance $\delta^2/2^{2H+1}$. Then, several iterations are proceeded to compose the fBm random sequence. In order to provide stationary increments, after the midpoints interpolation, a D_i of a certain variance ($\propto (r^n)^{2H}$; where r is the scaling factor.) is applied to all points (time increments) and not just the midpoints. The $X[i]$ noise sequence has $i = N + 1$ time increments, where $N = 2^{maxlevel}$ and *maxlevel* is the maximal number of iterations of the SMRD algorithm (Fig. 1(a)). The fBm statistics are generally achieved for values in the interval [0,16]. However, the *maxlevel* value of the SMRD is not limited to 16. The other SRMD inputs are the standard-deviation (*sigma*) and the *H* parameter. The Gaussian function (refer to *Gauss(.)* in Fig. 1.(a)) provides the mean value. d, D and $level$ are counters variables.

The $1/f$ fBm architecture (Fig. 1(b)) is composed by the following main blocks: Uniform random number (URNG); Gaussian random number (GRNG); data_ROM$[i, H]$; Control

and $X[i]$ RAM memory. All functions performed by the fBm block were coded as very-high-speed integrated circuit hardware description language (VHDL) entries. In this work, the Uniform random numbers generator (URNG) implementation is based on a combination of multiple-bit linear feedback shift registers (LFSR) and cellular automata shift registers (CASR) method proposed in [18]. This method generates a random sample by using XORs with the LFSRs and CASRs registers. This combination leads to a good randomness and its implementation requires few device resources. The linear function was implemented by a XOR of 43-bit position values. The cellular automata consists of a circular register with K cells, each having a value a_i equal to 0 or 1. These values are synchronously updated in discrete time steps according to $\grave{a_i} = a_{i-1}\ XOR\ (a_i\ OR\ a_{i+1})$ or, equivalently, $\grave{a_i} = (a_{i-1} + a_i + a_{i+1} + a_i a_{i+1})\ mod\ 2$. The $a^{(t)}$ values attained by a particular cell through time serve as the random sequence. The URNG block was coded to produce 32-bit uniformly distributed samples with periodicity 10^{10}. The LSFRs are started with a random *seed*.

The Gaussian random number generator (GRNG) block (Fig. 1(b)) performs the $Gauss(\cdot)$ function (Fig. 1(a)) of the SRMD algorithm. The Box-Muller method shows that, given u_1 and u_2 obtained from Uniform RVs distributed over the interval [0,1], a set of intermediate functions f, g_1, g_2 such that

$$f(u_1) = \sqrt{-ln(u_1)} \tag{1}$$

$$g_1(u_2) = \sqrt{2} sin(2\pi\, u_2) \tag{2}$$

$$g_2(u_2) = \sqrt{2} cos(2\pi\, u_2), \tag{3}$$

and the following x_1 and x_2 RVs

$$x_1 = f(u_1)\, g_1(u_2) \tag{4}$$

$$x_2 = f(u_1)\, g_2(u_2). \tag{5}$$

Then, x_1 and x_2 are independent Gaussian distributed RVs with zero mean and unit variance. Linear segmentation approximation is then applied to determine the f, g_1 and g_2 functions of the Box-Muller algorithm. The CLT is used to reduce the nonlinearities of these functions for values close to the interval limits, i.e., 0 and 1. Thus, improving the representation accuracy of the Gaussian RVs. The output of the URNG block provides both u_1 and u_2 to obtain the x_1 and x_2 (Eqs 2 and 3) in only one clock cycle. Look-up tables (LUT) are largely used to store the linear segmentation outputs of the Box-Muller functions (f, g_1 and g_2) in a few clock cycles. But, in these solutions, large LUTs sizes are required to store all the resulting linear approximation segments of the Box-Muller functions. In the proposed design the linear approximation of Box-Muller functions were performed by using *multipliers*, *adders* and a LUT to store each line segment. Since 32-bits were considered for u_1 coding, it is possible to have a representation range of 6.7σ values for each GRNG sample. Similar representation range was found by the Xilinx implementation of [17] of the Boutillon *et al.* design. Boutillon *et al.* [11] reported a 4σ range for each Gaussian representation with 0.02% accuracy. The following step of the Box-Muller algorithm is the sum of the two registered values of the x_1 and x_2 functions. It takes 3 clock cycles: one cycle to store each x_1 and x_2 value and a second cycle for the sum of these values. It can be seen from Eq. 2 and Eq. 3 that both g_1 and g_2 functions are indexed by u_2. Thus, their implementation were *pipelined* to improve the block performance.

TABLE I
BOX-MULLER GRNG BLOCK

Compilation	Without Memory usage	With Memory usage
Logic elements	193	158
ROM Memory (bits)	0	382
Embedded Multipliers	2	2
Max Clock Freq.	56 MHz	168 MHz

In this work, two different approaches were evaluated for the implementation of the GRNG block. Firstly, the GRNG block was coded in VHDL using only the available standard logic elements. In a second approach, a set of Altera Stratix Device Family's memory blocks was defined to deal with the LUT size issue. Table I shows the devices resources needed for both GRNG implementations. The second strategy achieved an improvement of 18% in the logic elements usage. It can also be noted that its overall performance was increased, e.g., a maximum clock frequency of 168 MHz (improvement of 300%) when compared to the first implementation approach. This solution also uses around 0.03 % of the ROM memory. Hence, this approach was chosen as the default implementation of the GRNG block.

The data_ROM block performs the computation of the $delta[i]$ vector of the SRMD algorithm. The data_ROM is indexed by i and H and it is defined by

$$data\_ROM[i,H] := \frac{delta[i]}{sigma} = (\frac{1}{2})^{iH} \sqrt{\frac{1}{2}} \sqrt{1 - 2^{2H-2}}. \tag{6}$$

It can be seen from Eq. 6, that $data\_ROM[i,H]$ uses some complex calculations to compute the $delta[i]$ vector. The *power* and *square-root* functions evaluated by LUTs, usually take several clock cycles to be solved. In the proposed approach all $delta[i]$ values are stored and addressed by the *i* and *H* indexes. This would be prohibitive due to the large amount of memory resources needs for storing a wide range of $delta[i]$ values. However, H values can be represented with only 2-digits after the decimal point. Thus, 1,000 H values are necessary for each iteration of the SRMD algorithm. Since $0 \leq maxlevel \leq 16$, $delta[i]$ vector can have a maximum of 16,000 elements. Hence, 1.6% of the ROM memory ressource was needed to store the $delta[i]$ vector. In fact, $1/f$ noises have $1/2 < H < 1$ and the memory needs can be even reduced. This second memory block is used to store the output sample vector ($X[i]$). The binary representation of each $X[i]$ noise output was truncated to 16-bit wide. For example, for *maxlevel*$= 16$ it will be necessary $16 * 2^{16} \approx 1.0$ Mbits of a RAM resource.

The main functions of the control block are:

1) Read the GRNG block output (Gaussian samples);
2) Read the data_ROM values according to the selected values indexed by i and H;

TABLE II
DEVICE RESOURCES USAGE OF THE $1/f$ FBM BLOCK DESIGN

maxlevel	14	15	16
Logic elements	1595 (6.1%)	1723 (6.9%)	1848 (7.4%)
RAM memory (bits)	261,144 (13%)	542,288 (27%)	1,048,576 (52%)
Embedded multipliers	10	10	12
Initializing time (cycles)	344064	688128	1376256
Max. Clock Freq. (MHz)	176	173	152

TABLE III
STATISTICS ESTIMATION: *factory2* NOISE PARAMETERS.

parameters (frame=16ms)	m (bits/frame)	σ (bits/frame)	H
factory2	639.50	369.65	0.990
fBm block (CI 95%)	642.03±3.432	370.75±5.683	0.989±0.00357
fBm block (CI 99%)	638.18±4.747	368.83±6.911	0.987±0.00732

3) Evaluate $delta$ by multiplying previous data_ROM data by $sigma$;
4) Fill the initial values of the $X[i]$ vector with the computed $sigma * Gauss$ values;
5) Perform the *loops* iterations (one *while* and two *for*'s);
6) Update the d, D and $level$ counters;
7) Read fBm output samples from $X[i]$ vector.

The EP1S25 FPGA kit used for the $1/f$ noise block implementation has approximately 25,000 logic elements, 4 RAM blocks of 500 Kbits each, ROM of 1 Mbits, 10 digital signal processor (*dsp*) blocks and 6 phase locked loop (PLL) used for clock generation. The *multstyle* attribute was set for using the dsp blocks. Altera's Stratix devices have a fixed number of dsp blocks which include a fixed number of embedded multipliers. The *multstyle* attribute specifies the implementation style for multiplication operations (*) in the VHDL source code. It also defines if the compiler shall implement a multiplication operation in general logic or in dedicated hardware. The $D/2$ and $d/2$ division operations of the SRMD algorithm were implemented using shift registers.

IV. RESULTS

This section presents the main results obtained from the evaluation of the proposed $1/f$ fBm noise generator. For the tests it was used the m, σ^2 and H statistics estimated from the real $1/f$ *factory2* acoustic noise. This acoustic noise was sampled at 16kHz and 16 ms frame length considering Hamming window. Additionally, a pure-Brownian generation was performed with statistics $m = 0$, $\sigma^2 = 1$ and $H = 1/2$. Five independent fBm noise sequences were generated of 10^6, 10^9, 10^{10}, 10^{11}, 10^{12} bit samples sizes. For the *H* parameter estimation it was used the wavelet-based method [10] with 12 Daubechies filters with the 3-12 scale ranges.

The main results presented in this section include:

- Device resources usage;
- Estimation of the statistical parameters: *m*, σ, H;
- $1/f$ fBm noise samples representation: PDF, (log-log) HTD and ACF plots.

The device resources usage was evaluated for different *maxlevel* values, i.e., the number of increments of the $1/f$ fBm noise sequence. This report is presented in Tab. II. It can be noted that for the maximum number of SRMD iterations, i.e., largest $1/f$ generated noise sequence, the implementation used 7.4% of logic elements, 52% of RAM memory and 1.6% of ROM memory of the FPGA main resources.

TABLE IV
STATISTICS ESTIMATION: PURE $1/f$ NOISE PARAMETERS.

parameters	m	σ	H
pure 1/f noise	0.000010	1.00	0.50
fBm block (CI 95%)	0.000011	1.00±0.00001	0.51±0.00099
fBm block (CI 99%)	0.000011	1.00±0.00001	0.51±0.00112

The $delta[i]$ of fBm block solves two *power* and two *square-root* functions (using the LUT method), hence it is necessary a total of $152 * 2^{maxlevel}$ clock cycles. Since the SRMD algorithm performs one $while$ and two for loops, the $X[i]$ vector spends an initial output delay t_{init}. This initial delay depends on the $maxlevel$ variable and is equal to

$$\begin{aligned} t_{init} &= (2^{maxlevel} - 1) + (2^{maxlevel+2} + \\ &\quad + maxlevel - 2) * t_{gauss} \\ &\approx (1 + 4 * t_{gauss}) * 2^{maxlevel}\ [cycles], \end{aligned} \tag{7}$$

where t_{gauss} is the total clock cycles to generate a Gaussian sample. However, after this initial time, only one clock cycle is needed to read a sample value from the $X[i]$ vector.

Tables III and IV show the statistics results obtained with the $1/f$ fBm noise samples using the *factory2* and the pure-Brownian noises parameters. The m, σ and H results accuracy was obtained considering the *t-student* method [20] for the 95% and 99% confidence intervals. Considering n independent experiments it follows that $\hat{X_n} = \bar{X}_n \pm t_{n-1,1-\alpha}\sqrt{\frac{\sigma_n^2}{n}}$, where t_n is a *t-student* RV with $n-1$ degrees of freedom and an exact $100(1-\alpha)$ percent confidence interval. These results show that the samples generated by the $1/f$ fBm blocks achieved close statistics to real *factory2* and the pure-Brownian noises. The scaling degrees accuracy was achieved for both noise generation experiments. Particularly, the H results obtained from the $1/f$ fBm samples, were very interesting since they indicate the good representation of the low-frequency ($H > 1/2$) characteristics of the $1/f$ noise.

Fig. 2 presents the PDF, HTD and ACF curves obtained from the fBm noise samples. These plots were evaluated considering the real *factory2* parameters since it is a $1/f$ noise ($H = 0.75$). The results obtained from the noise samples generated from the hardware fBm block were compared to the referred *ideal* curve in which the fBm samples were obtained from a C++ language implementation of the SRMD algorithm considering 10^6 (*maxlevel*$= 16$) samples. The PDF plots (Fig. 2(a) were evaluated by the approximation of the noise samples histograms. Note that the noise samples pattern distribution obtained from the hardware implementation is quite similar to the one obtained from software-based generation.

The log-log HTD ($P[X > x]$) curves were very important to show the *tail* distribution of the noise samples. For example, pure-white-Gaussian distributed samples have Exponential decaying *tails*. Differently, $1/f$ samples with scaling invariance degrees ($H > 1/2$) have slow or heavy decaying tail behavior. As expected, the generated fBm samples ($H \pm 0.75$) presented heavy tails (Fig. 2(b)).

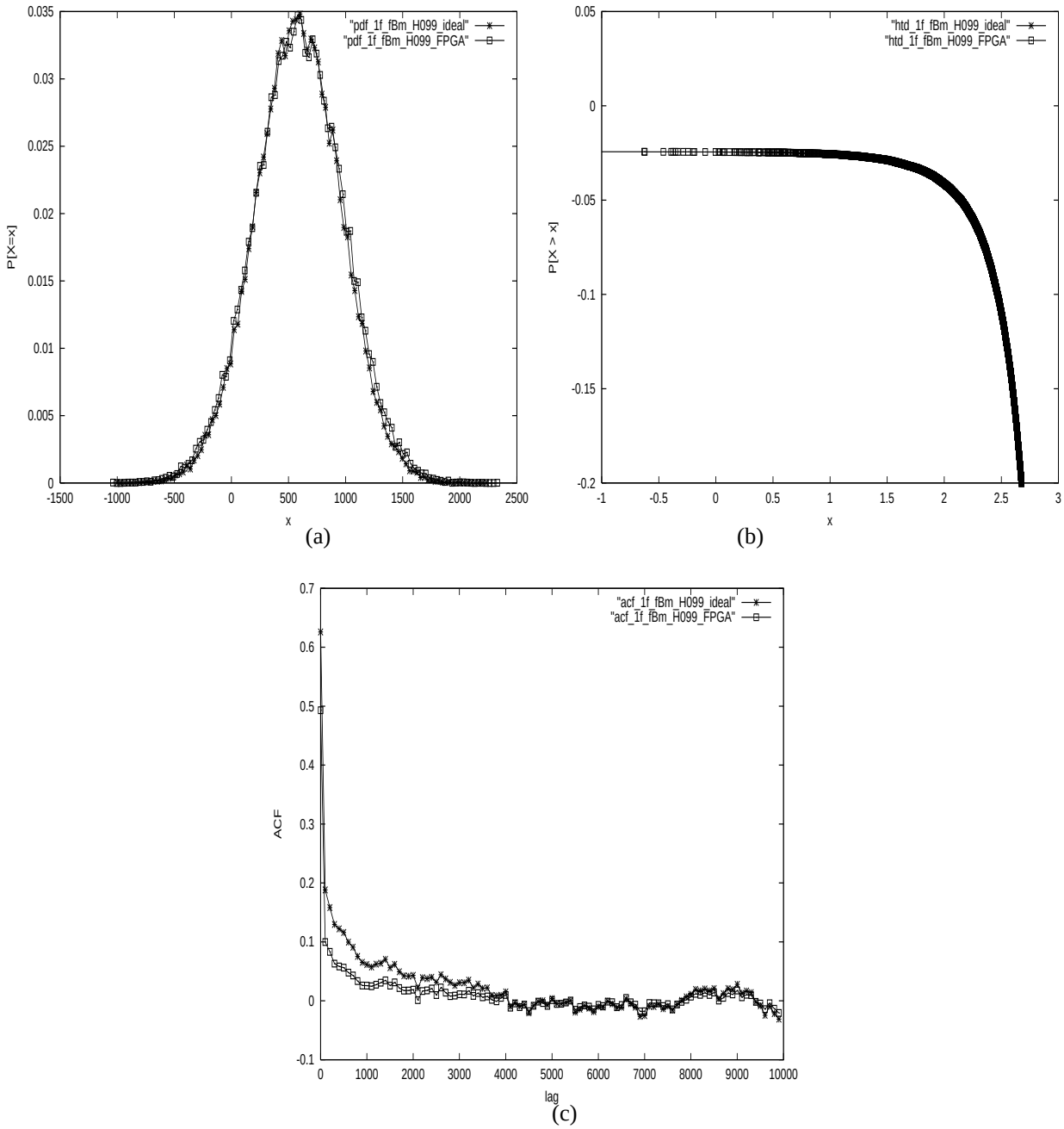


Fig. 2. $1/f$ fBm Noise Design Validation Results: (a) PDF, (b) log-log HTD, (c) ACF.

From the ACF graphics (Fig.2(c)) it can be inferred that they follow a slowly-vanishing function with slow variations to infinity (i.e., lag $k \rightarrow \infty$). This also demonstrates that the noise samples have $S(f)$ concentrated in low-frequency range.

V. Conclusion

This paper proposed a fast and accurate hardware design of $1/f$ noise generator based on the fractional Brownian motion. The noise generator is described for the successive random addition midpoint displacement method. The solution enabled the generation of 150 million of 16-bit noise samples per second. The $1/f$ noise generator achieved for the largest sample size generation 7.4% of logic elements, 52% of RAM memory and 1.6% of ROM memory. The estimation results showed that $1/f$ fBm noise generator accurately achieved the mean, variance and the low-frequency statistics and the noise samples pattern. Moreover, the auto-correlation function of the generated noise samples confirmed the expected slow variations of the real $1/f$ *factory2* acoustic noise. The proposed design showed to be very promising to evaluate the performance of communications channels with very low BER values. It can also be interesting for signal processing applications (e.g., noise extraction) that deals with noisy conditions such as automatic speaker recognition [22]. Other algorithms for the generation of $1/f$ fBm samples [9] [6] and true random number [23] can be considered as future

work. The proposed design will also be applied to generate impulsive or alpha-stable noises [24].

Acknowledgments: This work was partially supported by the APQ1/FAPERJ (E26/170.518/2007) and Universal/CNPq (472461/2009-5) research grants.

REFERENCES

[1] M. Keshner, "1/f noise," *Proceedings of the IEEE*, vol. 70, pp. 212–218, March 1982.
[2] G. Wornell, "Communication over fractal channels," *International Conference on Acoustics, Speech and Signal Processing (ICASSP91*, vol. 3, pp. 1945–1948, 1991.
[3] M. Fukuda, T. Hirono, T. Kurosaki, and F. Kano, "1/f noise behavior in semiconductor laser degradation," *IEEE Photonics Technology Letters*, vol. 10, pp. 1165–1167, October 1993.
[4] E. Hurst, "Methods of using long–term storage in reservoirs," *Proc. of Inst. Civil Engs*, pp. 519–543, 1956.
[5] B. Mandelbrot, *Fractals and Scaling in Finance*. Springer-Verlag, 1997.
[6] D. Saupe, "Random fractals in image synthesis," *Fractals and Chaos*, pp. 89–118, Springer-Verlag 1991.
[7] R. Voss and J. Clarke, "1/f noise in music: Music from 1/f noise," *Journal of Accoustic Society Am.*, vol. 63, pp. 258–263, 1978.
[8] B. Mandelbrot and J. Van Ness, "Fractional brownian motions, fractional noises and applications," *SIAM Review*, vol. 10, pp. 422–437, October 1968.
[9] M. Barnsley et al, *The Science of Fractal Images*. USA: Springer-Verlag New York Inc., 1988.
[10] P. Flandrin, "Wavelet analysis and synthesis of fractional brownian motion," *IEEE Transactions on Information Theory*, vol. 38, pp. 910–917, March 1992.
[11] E. Boutillon, J.-L. Danger, and A. Ghazel, "Design of high speed awgn communication channel emulator," *Analog Integrated Circuits and Signal Processing*, vol. 34, pp. 133–142, 2003.
[12] D. Lee, W. Luk, J. Villasenor, and P. Cheung, "A gaussian noise generator for hardware-based simulations," *IEEE Transactions on Computers*, vol. 53, pp. 1523–1532, December 2004.
[13] D. Lee, J. Villasenor, W. Luk, and P. Cheung, "A hardware gaussian noise generator using the box-muller method and its error analysis," *IEEE Transactions on Computers*, vol. 55, pp. 659–671, June 2006.
[14] G. Loss and R. Coelho, "Fpga-based fractional brownian motion signal patterns for optical packet generation," *Proceedings of the International Telecommunications Symposium (ITS2006)*, vol. 1, pp. 1–6, September 2006.
[15] G. Box and M. Muller, "A note on the generation of random normal deviates," *Annals Math. and Statistics*, vol. 29, pp. 610–611, 1958.
[16] A. Papoulis, *Probability, Random Variables and Stochastic Processes*. New York: McGraw-Hill, 1965.
[17] Xilinx, "Additive white gaussian noise (awgn) core v1.0," *On line*, Available at http://www.xilinx.com 2002.
[18] T. Tkacik, "A hardware random number generator," *Proceeding of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*, vol. 2523, pp. 450–453, Lecture Notes in Computer Science 2003.
[19] A. Varga, H. Steeneken, M. Tomlinson, and D. Jones, "The noisex-92 study on the effect of additive noise on automatic speech recognition," *Technical Report of Defence Evaluation and Research Agency*, Available at http://spib.rice.edu/spib/ 1992.
[20] A. Law and W. Kelton, *Simulation Modeling and Analysis*. USA: McGraw-Hill Book Company, 1982.
[21] D. Veith and P. Abry, "A wavelet-based joint estimator of the parameters of long-range dependence," *IEEE Trans. Information Theory*, vol. 45, pp. 878–897, Available at http://www.emulab.ee.mu.oz.au/darryl 1999.
[22] J. Ming, T. Hazen, J. Glass, and D. Reynolds, "Robust speaker recognition in noisy conditions," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 15, pp. 1711–1723, July 2007.
[23] G. Balachandran and R. Barnett, "A 440-na true random number generator for passive rfid tags," *IEEE Transactions on Circuits and Systems-I*, vol. 55, pp. 3723–3732, December 2008.
[24] M. Shao and C. Nikias, "Signal processing with fractional lower order moments: Stable processes and their applications," *Proceedings of the IEEE*, vol. 81, pp. 986–1010, July 1993.

Gustavo Lima Loss received the B.S. and M.Sc. degrees in Electrical Engineering from the Instituto Militar de Engenharia (Military Institute of Engineering-IME) of Rio de Janeiro in 2006 and 2000, respectively.
In 2006, he joined the Military Material Industry of the Brazilian Army. His main research interests include VLSI architectures for digital signal processing and communications, signal and noise generation, and embedded systems with reconfigurable devices such as field-programmable gate array.

Rosângela Fernandes Coelho received the Ph.D. degree from the Ecole Nationale Supérieure des Télécommunications (ENST-Paris) in 1995 and the M.Sc. degree from the Pontifícia Universidade Católica of Rio de Janeiro (PUC-Rio) in 1991, both in Electrical Engineering.
She joined the Instituto Militar de Engenharia (Military Institute of Engineering-IME) of Rio de Janeiro, in 2002, where she is Associate Professor at the Electrical Engineering Department. Prof. Coelho also founded and heads the Acoustic Signal Processing Laboratory. In 2003, she received the University Research Program research grant from CISCO/USA. She also served as editorial board member of the IEEE Communications Surveys and Tutorials from 1999-2007. Since 2008, she is responsable for the International Scientific Collaboration IME-ParisTech that includes 10 french engineering schools. Prof. Coelho was President-Adjoint of the Brazilian Telecommunications Society (SBrT) from 2008-2010. In 2011, Prof. Rosângela Coelho received the USPTO patent of an automatic speaker recognition method based on a new speech feature and speaker classifer. Her main research interests include acoustic signal processing, speech and speaker recognition, acoustic emotion and noise detection and representation, acoustic speech features, time-frequency analysis, speech enhancement, acoustic signal and noise generation, non-stationary noise, wavelets estimation, and statistical signal processing.