Indonesian Journal of Electrical Engineering and Computer Science
Vol. 27, No. 1, July 2022, pp. 118~130
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v27.i1.pp118-130

Field programmable gate arrays implementation of different standard deviation estimation techniques

Serwan Ali Bamerni, Ahmed Kh. Al-Sulaifanie
Department of Electrical and Computer, College of Engineering, University of Duhok, Duhok, Iraq

Article Info

Article history:

Received Aug 30, 2021
Revised May 5, 2022
Accepted May 24, 2022

Keywords:

Field programmable gate arrays
Noise variance
Standard deviation
White Gaussian noise
Xilinx system generator

ABSTRACT

Additive white Gaussian noise level estimation has found its application in many fields such as biomedical signal processing, communication system, and image processing. Many methods have been proposed with different output accuracy, system complexity, power consumption, and speed. In this paper, three of the most well-known and largely used algorithms (median based, root mean square (RMS) based, and P84 based methods) have been implemented and investigated in a full comparison between them to find their advantage and disadvantage, and the suitability of each method for a specific application. The three designs are created using Xilinx system generator (XSG) and implemented on Xilinx field programmable gate arrays (FPGA) development board with Zynq series "XC7Z020-1CLG484", to evaluate the design's performance and the results are discussed in the paper.

Corresponding Author:

Serwan Ali Bamerni
Department of Electrical and Computer, College of Engineering, University of Duhok
Zakho Way, 42001, Duhok, Iraq
Email: serwan.mohammed@uod.ac

1. INTRODUCTION

Additive white Gaussian noise level estimation is a fundamental process in many fields, such as biomedical signal processing [1], [2], communication system [3], [4] and image processing [5], [6]. Removing this unavoidable noise is an essential step prior to any farther processing in these systems. Additive Gaussian noise is so named since it characterized as zero-mean Gaussian random variables distribution, which is continuous with a probability density function (pdf) given by (1):

$$f(x) = \mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}, -\infty < x < \infty \quad (1)$$

where μ is the mean of Gaussian distribution and σ^2 is its variance.

In the old systems, a fixed noise level was employed as a threshold in a variety of applications that needed to eliminate noise, but since noise levels are not constant and vary over time [7], [8], this technique becomes inapplicable. So, researchers concentrate on calculating an appropriate value for the threshold, with the aim of real-time estimate, acceptable power consumption, area occupancy, and accuracy.

Many algorithms and systems have been developed to perform real time standard deviation (SD) estimation. Donoho *et al*. [9] and Mallat *et al*. [10] showed that the SD of background noise can be estimated by dividing the median absolute deviation (MAD) by the 75th percentile of the standard normal distribution. Guillory and Norman [11] utilizing the traditional root mean square (RMS) based method to estimate the SD of background noises that is directly employing a programmable digital signal processing (DSP) chip for calculating the RMS value, the cost of employing such (DSP) units, limited a little bit the use of this

algorithm. Harrison and Charles [12] presented an analogue-based technique for estimating the SD of noise level adaptively. The circuit uses the output of the low-pass filter, which acts as an integrator, to calculate the duty cycle, or the time period the signal is above the threshold level over a specific time period, and compare it with the 15.9% of the initial threshold set at SD (so, it is also called P84). Then the SD value is continuously updated with a proportional-integrator (PI) control loop to meet the theoretical value. The simplicity of this method and its ability to be constructed using simple analog components give it an advantage over many of the other noise estimation methods. In contrast, the performance of this method decreases with the increase in signal-to-noise ratio (SNR). Despite that, this approach remains the most effective method in terms of circuit simplicity, power consumption, and area accupied [13]. Gagnon *et al.* [14] used sigma-delta in control loop to adjust the estimated value.

Other methods depending on first-order statistics have also been suggested for estimating noise level, such as [15] who proposed an algorithm known as cap-fitting, that considers only the cap, "the middle portion of the amplitude distribution", to estimate the noise level, and [16] which uses minimax spread (MMS) to calculate the noise SD. The main problem with these methods is the massive and complicated calculations required to produce a statistically meaningful characterization.

Recently, several of these algorithms have been used in conjunction with the wavelet transform to determine the SD of the noise, taking advantage of the capability of the wavelet transform in separating the energy of the signal into two components, the main signal energy is concentrated in the first band, while the second one includes the residual of the signal with the noise uniformly contributed over all the coefficients [9]. This technique can give better results, especially with higher SNR, with the help of the multiresolution decomposition property of the wavelet [17], [18]. The only problem with this method is the complex computation required for the wavelet algorithm to be implemented.

In this paper, the architectural design of three most commonly used techniques (RMS, median, and P84) based for noise level estimation are suggested and implemented in real time with the aid of Xilinx FPGA platform, and an extensive comparison is made in term of accuracy, die area, latency, and circuit complexity, to show the advantages and disadvantages of each method over the other and to enable the user to select the most suitable method for each specific application.

The rest of the paper is organized as shown in; section 2 gives a detailed description of the three tested algorithms with the background theory of each method, while in section 3, these methods are evaluated in terms of accuracy of the estimated noise, hardware resource utilization, power consumption, and system speed. Discussion and comparison of results are detailed in section 4. Finally, the conclusion of the research is given in section 5.

2. HARDWARE DESIGN

2.1. Median based standard deviation estimation

The idea behind this method is that the standard deviation σ cam be estimated by [9], [10]:

$$\sigma = \frac{median\,(|x_n|)}{0.6745}\,, n = 1, 2, \ldots N \quad (2)$$

where x_n is the data input, then the median value of an *N* data sample is calculated by first sorting the data in ascendingly then choosing the middle datum if *N* is odd, or determining the mean of the two middle data if *N* is even. Figure 1 shows the basic block diagram of the median based estimator. The main block of the algorithm is the sorting block, where the size of the sorting network decides the latency, size, and accuracy of the system.

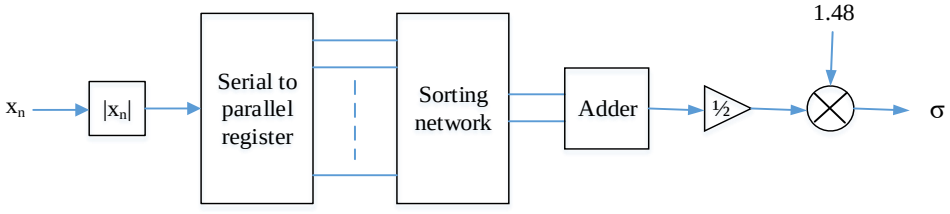

Figure 1. The median based block diagram for standard deviation estimation

An 8-sample parallel sorting network is shown in Figure 2. The sorting circuit could be scaled up to any size at the cost of increasing the complexity of the design [19], [20]. In this approach, a moving window is used for computing the median value; the window size determines the size of the sorting circuit; thus, for

each new sample, the window moves one sample ahead. As a result, the first accurate median value is calculated after the number of samples equals the size of the window.

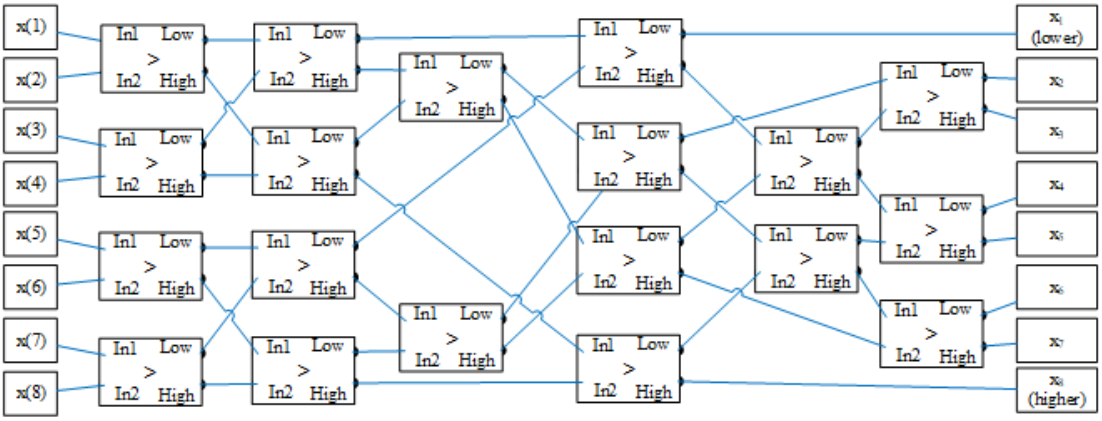

Figure 2. The parallel sorting network of eight samples

2.2. RMS based standard deviation estimation

Traditionally, the RMS based technique is employed to calculate the SD of the noise [11], where:

$$\sigma = \sqrt{\frac{1}{N}\sum_{1}^{N}(x_n)^2}\ , n = 1, 2, \dots N \tag{3}$$

where x_n is the data input, and N is the total number of input data samples. Figure 3 shows the basic block diagram for the RMS based estimator. It seems to be simpler than the median based construction, as shown below, accept its containing of the square root block (Sqrt block), which represents the stumbling block in the design that consumes the most power, limits speed, and occupies the most design area.

The concept behind this approach is to compute the RMS value for a moving window, where its size is given by N in Figure 3. The window moves each time a new sample enters the system. As in the previous method, the first accurate SD value is determined after N data samples are entered. while the window size may be easily modified by adjusting the N value in both the delay and gain blocks.

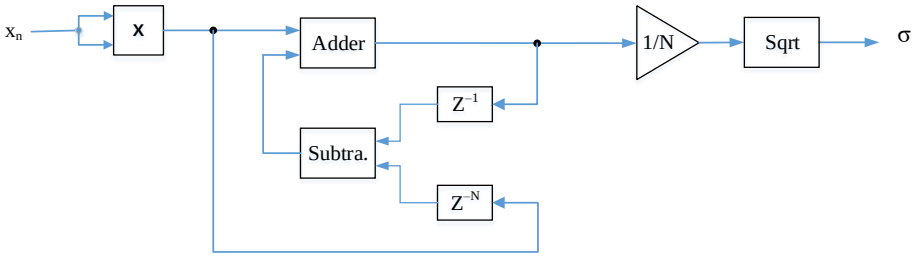

Figure 3. The RMS based structure for standard deviation estimation

2.3. P84 based standard deviation estimation

It is first proposed by Harrison and Charles [12] and since then, it has been used in a different applications [21]–[23] utilizing its capability to be implemented with analog components. Figure 4 shows the basic block diagram of this algorithm.

This method needs a bit more explanation to understand its principles. that the input Gaussian noise $x(t), \mathcal{N}(x; 0, \sigma)$ is first compared to a threshold voltage Th_r instantaneously using a comparator. The binary output waveform of the comparator, $x_c(t)$ is set high (A volt) any time the input Gaussian noise surpasses the threshold voltage Th_r otherwise, it set low (0 volt). Adding up the time duration, the output of the

comparator is set to high (t_k) of $x_c(t)$ and dividing by the length of measuring time T_m is identical of calculating the probability of the input $x(t)$ exceeding the threshold value, which, in turn, represents the average or DC value of the input signal;

$$p(x > Th_r) = \frac{1}{T_m} \int_0^{T_m} x_c(t)dt = A\frac{\sum_k t_k}{T_m} \quad (4)$$

where A is comparator pulse amplitude.

Then, with the aid of a simple first order low pass filter (one pole filter) of transfer function $H(s) = \frac{1}{1+T_L s}$ and time constant $T_L = Rc$ very larger than the time required for the input signal $x_C(t)$ to make an appreciable change, the circuit acts as an integrator and the output $y(t)$ will pass only the DC component of $x_c(t)$.

The time diagram for the comparator and leontief production function (LPF) output is shown in Figure 5, with the following parameter, $Th_r = \sigma = 1, A = 5$, LPF cutoff frequency $f_L = 1$ and bandwidth of the input signal $x_c(t)$ equal $f_C = 18\ KHz$.

As indicated perviously, one property of random Gaussian distribution is that the probability of a signal sample being above σ level in Gaussian noise during a predetermined period is 15.9 %, or $p(x > \sigma) = 0.159$. So, if the LPF output $y(t) = A\,p(x > Th_r)$ is subtracted from theoretical probability reference of $0.159A$, the error signal or the amount of difference between the current threshold from the real SD is obtained. The error signal at any instantons is

$$e(t) = p(x > Th_r)A - 0.159A \quad (5)$$

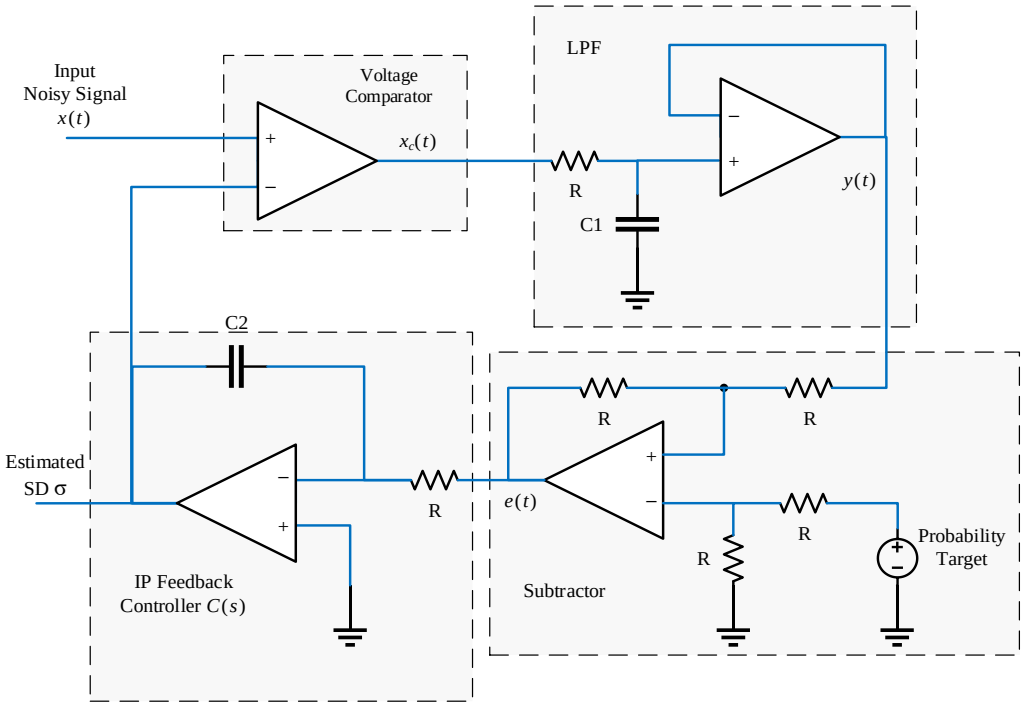

Figure 4. The P84 based circuit diagram for standard deviation estimation

The error signal $e(t)$ provides an appearance of the probability to be corrected by the feedback controller C(s). The estimated value by the closed loop controller Th_r equal the input Gaussian noise σ when $e(t) \approx 0$, (i.e. $Thr. = \sigma$). The digital implementation of this method is shown in Figure 6. The design can be summarized as follows:

- Counter 1 count up to M which represent the width of the window size. This means that standard deviation is measured and changed every M sample and it stays constant along this period.
- The comparator block compares the data stream with estimated standard deviation. Wherever the input data is greater than the estimated value, the output is logic 1, otherwise, it is logic 0.

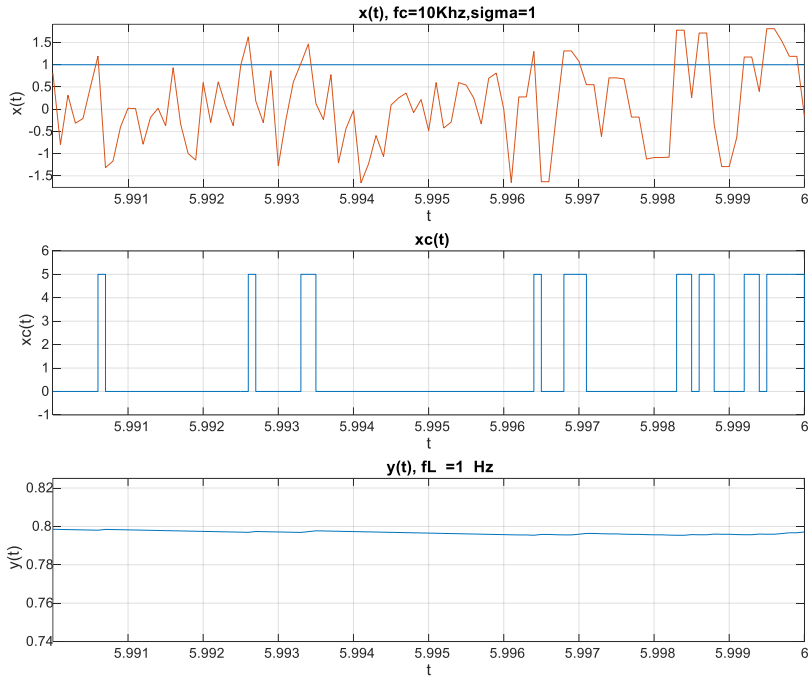

Figure 5. The comparator and LPF output waveform

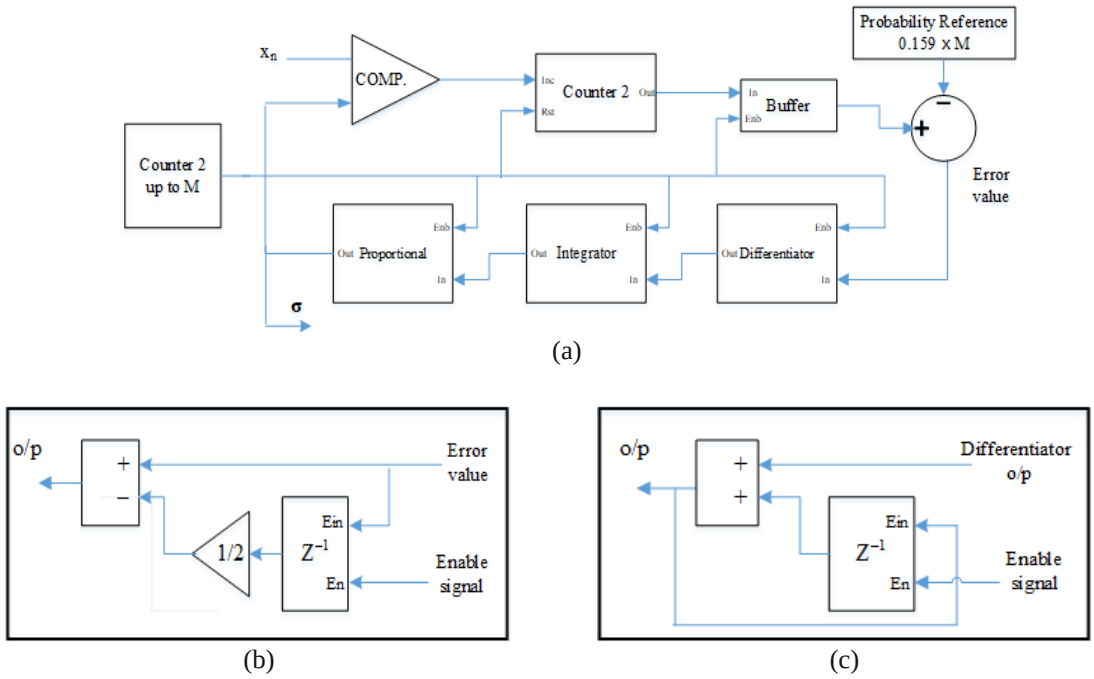

Figure 6. The P86 based block diagram (a) overall system, (b) integrator part and (c) differentiator part

- Counter 2 replace the LPF filter in the analog implementation of the design, this counter, count the number of samples that are greater than the estimated SD or the number of logic 1.
- The number of 1's (the output of Counter 2) is then compared with a constant percentage ($0.159 \times M$) to calculate the error. The resulted error value reflects the convergence or divergence from the exact value, where, as small as the resulted value, as near as the exact value.
- The feedback proportional–integral–derivative (PID) controller is used to continuously adjust the estimated value of SD. Figure 6(a), shows the series PID used, while Figure 6(b) shows the

implementation of the integral part of the controller, which is implemented by a simple cumulative addition, while the differential part is implemented by a difference between the previous value and the new value as shown in Figure 6(c). The proportional is implemented as a simple gain circuit with gain constant K, a small gain constant result a small output response or more time to reach the steady state, while high proportional gain results in a large change in the output and the system can become unstable.

Accordingly, there are many parameters that affect the accuracy of the estimated value like the M value, and the $C(s)$ parameters, which should be tuned for optimal performance of the system, in which the accuracy and the speed are the most important criteria.

3. PERFORMANCE EVALUATION

There are several criteria that could be used to assess the performance of the implemented structures. One of these criteria is to find the accuracy of the estimated value and the hardware resources needed to accomplish the structure, in addition to the consumed power and the speed of each structure. Studying these parameters helps one to select the most suitable structure.

3.1. Estimated value accuracy

For the purpose of accuracy evaluation, the three algorithms are used to estimate the noise level in four well-known benchmark signals shown in Figure 7(a)-(d), named, Blocks as in Figure 7(a), Bumps as in Figure 7(b), heavy sine as in Figure 7(c), and Doppler as in Figure 7(d), after corrupted with additive white Gaussian noise (AWGN) of various levels of standard deviation σ. These signals are first transformed to wavelet domain using Haar wavelet for the purpose of simplicity, and its details is then used as the input for these algorithms as described in [21]. This test is carried out to show the adaptivity of these algorithms to work with different signals and with different noise levels. The length of these signals is set to be (4069) samples. Two criteria are used to evaluate the output of these systems, the mean of the output SD, and mean square error (MSE) between the estimated SD and the likely SD values which can expressed as [24]:

$$MSE = \frac{\sum_{k=1}^{N}(X(k)-S(k))^2}{N} \quad (6)$$

where N is the length of input signal and $S(k)$ is the likely SD values. Tables 1 to 4 illustrate the results of these tests carried out by these test parameters, where the smaller value for MSE indicates best algorithm, and closer results of mean to the used SD is the more accurate results.

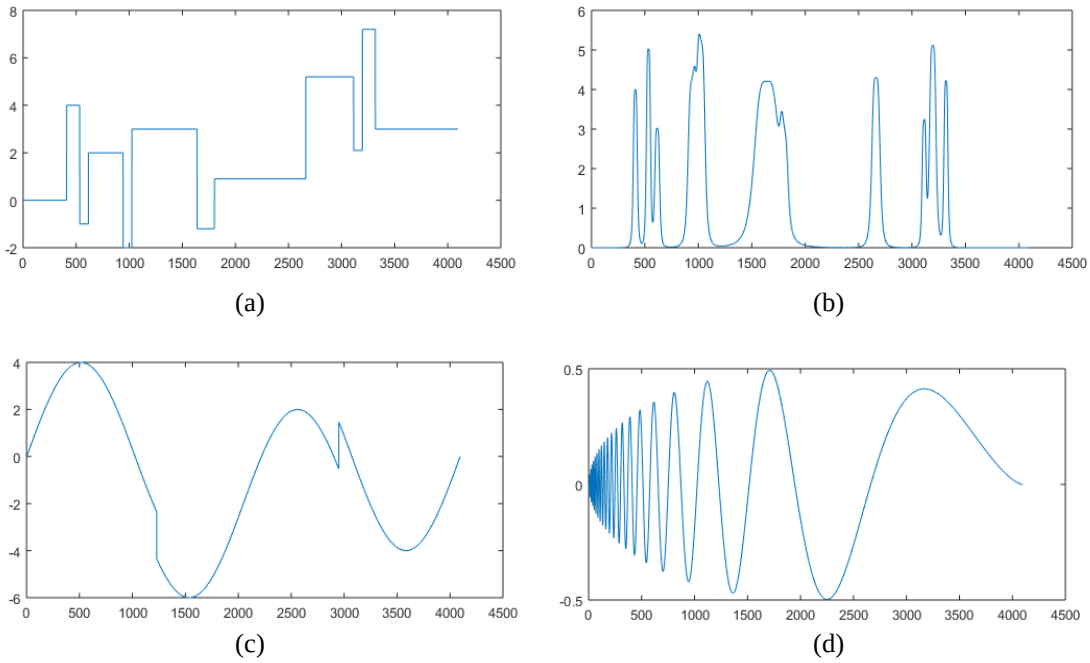

Figure 7. The testied signals (a) blocks, (b) bumps, (c) heavy sine, and (d) Doppler

Table 1. The accuracy evaluation of the SD estimator for the three methods using the Blocks signal

Method		Blocks signal corrupted with AWGN							
		σ = 0.5		σ = 0.8		σ = 1.2		σ = 1.5	
		Mean	MSE	Mean	MSE	Mean	MSE	Mean	MSE
Median based method	window size=16	0.5051	0.0185	0.8083	0.0474	1.2124	0.1067	1.5154	0.1664
RMS based method		0.5378	0.0271	0.8154	0.0306	1.1955	0.0525	1.4843	0.0779
P84 based method		0.4677	0.0080	0.7194	0.0304	1.0306	0.0974	1.2346	0.1882
Median based method	window size = 32	0.5001	0.0114	0.8003	0.0291	1.2004	0.0655	1.5005	0.1023
RMS based method		0.5439	0.0195	0.8195	0.0204	1.1999	0.0336	1.4895	0.0492
P84 based method		0.4609	0.0102	0.7053	0.0349	1.0079	0.1071	1.2077	0.2087
Median based method	window size=46	0.4926	0.0083	0.7876	0.0213	1.1817	0.0476	1.4774	0.0738
RMS based method		0.5019	0.0055	0.7903	0.0123	1.1786	0.0265	1.4708	0.0410
P84 based method		0.4513	0.0179	0.6937	0.0459	0.9908	0.1247	1.1839	0.2362
Median based method	window size =128	0.4869	0.0075	0.7791	0.0193	1.1687	0.0434	1.4608	0.0678
RMS based method		0.5370	0.0120	0.7998	0.0179	1.1679	0.0361	1.4490	0.0555
P84 based method		0.4996	0.0967	0.7245	0.0972	1.0023	0.1725	1.1950	0.2857
Median based method	window size =256	0.4754	0.0108	0.7607	0.0276	1.1411	0.0621	1.4264	0.0970
RMS based method		0.5184	0.0106	0.7696	0.0233	1.1227	0.0544	1.3926	0.0867
P84 based method		0.9577	1.4982	0.9600	0.8647	1.0947	0.6343	1.2460	0.6686

Table 2. The accuracy evaluation of the SD estimator for the three methods using the Bumps signal

Method		Bumps signal corrupted with AWGN							
		σ = 0.5		σ = 0.8		σ = 1.2		σ = 1.5	
		Mean	MSE	Mean	MSE	Mean	MSE	Mean	MSE
Median based method	window size = 16	0.5034	0.0187	0.8051	0.0477	1.2076	0.1071	1.5096	0.1670
RMS based method		0.5034	0.0098	0.7931	0.0226	1.1831	0.0491	1.4766	0.0758
P84 based method		0.4668	0.0086	0.7197	0.0303	1.0293	0.0968	1.2320	0.1920
Median based method	window size = 32	0.4988	0.0117	0.7976	0.0293	1.1962	0.0653	1.4953	0.1017
RMS based method		0.5052	0.0066	0.7959	0.0143	1.1873	0.0306	1.4818	0.0471
P84 based method		0.4566	0.0107	0.7077	0.0350	1.0076	0.1075	1.2136	0.2070
Median based method	window size = 46	0.4926	0.0083	0.7876	0.0213	1.1817	0.0476	1.4774	0.0738
RMS based method		0.5019	0.0055	0.7903	0.0123	1.1786	0.0265	1.4708	0.0410
P84 based method		0.4513	0.0179	0.6937	0.0459	0.9908	0.1247	1.1839	0.2362
Median based method	window size =128	0.4849	0.0078	0.7755	0.0197	1.1626	0.0439	1.4531	0.0684
RMS based method		0.4925	0.0064	0.7749	0.0157	1.1552	0.0351	1.4415	0.0548
P84 based method		0.4997	0.0954	0.7217	0.0975	1.1964	0.2875	1.1964	0.2875
Median based method	window size =256	0.4728	0.0109	0.7552	0.0280	1.1333	0.0626	1.4171	0.0976
RMS based method		0.4730	0.0094	0.7441	0.0248	1.1094	0.0566	1.3844	0.0890
P84 based method		0.9568	1.4789	0.9502	0.8451	1.0966	0.6405	1.2460	0.6681

Table 3. The accuracy evaluation of the SD estimator for the three methods using the Heavy sine signal

Method		Heavy sine signal corrupted with AWGN							
		σ = 0.5		σ = 0.8		σ = 1.2		σ = 1.5	
		Mean	MSE	Mean	MSE	Mean	MSE	Mean	MSE
Median based method	window size = 16	0.5044	0.0187	0.8071	0.0478	1.2106	0.1075	1.5130	0.1680
RMS based method		0.5010	0.0093	0.7932	0.0216	1.1846	0.0475	1.4786	0.0739
P84 based method		0.4669	0.0085	0.7175	0.0304	1.0246	0.0995	1.2279	0.1907
Median based method	window size = 32	0.5000	0.0113	0.8000	0.0290	1.2000	0.0652	1.4999	0.1019
RMS based method		0.5033	0.0055	0.7962	0.0130	1.1887	0.0290	1.4837	0.0452
P84 based method		0.4584	0.0105	0.7067	0.0346	1.0025	0.1099	1.2028	0.2105
Median based method	window size = 46	0.4950	0.0080	0.7917	0.0206	1.1872	0.0464	1.4835	0.0726
RMS based method		0.5000	0.0044	0.7904	0.0111	1.1798	0.0252	1.4725	0.0396
P84 based method		0.4496	0.0180	0.6932	0.0460	0.9903	0.1255	1.1829	0.2354
Median based method	window size =128	0.4871	0.0074	0.7792	0.0189	1.1687	0.0426	1.4602	0.0667
RMS based method		0.4900	0.0060	0.7744	0.0153	1.1559	0.0347	1.4427	0.0543
P84 based method		0.4981	0.0951	0.7144	0.0963	0.9961	0.1740	1.1873	0.2875
Median based method	window size =256	0.4749	0.0107	0.7598	0.0274	1.1397	0.0617	1.4238	0.0967
RMS based method		0.4708	0.0100	0.7441	0.0255	1.1106	0.0576	1.3861	0.0901
P84 based method		0.9531	1.4984	0.9560	0.8680	1.0932	0.6346	1.241	0.6705

Table 4. The accuracy evaluation of the SD estimator for the three methods using the Doppler signal

Method		Doppler signal corrupted with AWGN							
		σ = 0.5		σ = 0.8		σ = 1.2		σ = 1.5	
		Mean	MSE	Mean	MSE	Mean	MSE	Mean	MSE
Median based method	window size = 16	0.5032	0.0184	0.8051	0.0471	1.2075	0.1061	1.5094	0.1659
RMS based method		0.4911	0.0082	0.7855	0.0209	1.1782	0.0470	1.4728	0.0735
P84 based method		0.4630	0.0084	0.7181	0.0301	1.0218	0.0990	1.2287	0.1908
Median based method	window size = 32	0.4986	0.0112	0.7976	0.0287	1.1962	0.0644	1.4952	0.1007
RMS based method		0.4927	0.0051	0.7881	0.0130	1.1822	0.0291	1.4777	0.0455
P84 based method		0.4560	0.0105	0.7011	0.0352	1.0007	0.1104	1.2027	0.2118
Median based method	window size = 46	0.4933	0.0080	0.7891	0.0206	1.1836	0.0464	1.4793	0.0725
RMS based method		0.4889	0.0046	0.7820	0.0116	1.1731	0.0260	1.4664	0.0406
P84 based method		0.4519	0.0179	0.6957	0.0463	0.9908	0.1238	1.1800	0.2382
Median based method	window size =128	0.4858	0.0075	0.7770	0.0191	1.1654	0.0430	1.4566	0.0671
RMS based method		0.4789	0.0063	0.7661	0.0159	1.1492	0.0355	1.4366	0.0554
P84 based method		0.4998	0.1001	0.7201	0.0974	0.9932	0.1749	1.1885	0.2879
Median based method	window size= 256	0.4735	0.0107	0.7576	0.0275	1.1367	0.0618	1.4210	0.0966
RMS based method		0.46	0.0103	0.7358	0.0261	1.1037	0.0586	1.3798	0.0913
P84 based method		0.9576	1.4970	0.9584	0.8665	1.0946	0.6364	1.2460	0.6699

3.2. Utilized hardware resources

The hardware resources needed for the implementation of each algorithm with different size of the window also compared. The Zynq series "XC7Z020-1CLG484" FPGA development board is used to implement these systems. Table 5 illustrates the Vivado® Design Suite's estimation for the hardware resources. All design input is set to a fixed point of 15/8.

Table 5. Utilization of FPGA hardware resources for the three algorithms with different window size

Method		Slice LUTs		Slice Registers		DSPs		Block RAM Tile	
		Used	Utilize%	Used	Utilize%	Used	Utilize%	Used	Utilize%
Median based method	window	2074	3.09	240	0.23	0	0	0	0
RMS based method	size = 16	461	0.87	296	0.28	1	0.45	0	0
P84 based method		75	0.14	61	0.06	0	0	0	0
Median based method	window	6146	11.55	496	0.47	0	0	0	0
RMS based method	size = 32	468	0.88	296	0.28	1	0.45	0	0
P84 based method		72	0.14	61	0.06	0	0	0	0
Median based method	window	20097	37.78	1008	0.95	0	0	0	0
RMS based method	size = 46	485	0.91	314	0.30	1	0.45	0	0
P84 based method		72	0.14	61	0.06	0	0	0	0
Median based method	window	43201	81.20	2079	1.95	0	0	0	0
RMS based method	size =128	517	0.97	350	0.33	1	0.45	0	0
P84 based method		73	0.14	61	0.06	0	0	0	0
Median based method	window size =256	Synthesis and implementation failed: This design requires more Slice LUTs cells than are available in the target device.							
RMS based method		570	1.07	418	0.39	1	0.45	0	0
P84 based method		75	0.14	62	0.06	0	0	0	0

3.3. Consumed power

The consumed power for each system is affected by many aspects, including system frequency, density of resources, connection topology, and the level of supply voltage [25]. As in all embedded systems, the power consumption is made up of two parts, the dynamic power and static power. The dynamic power is entirely related to the implemented architecture and is calculated by adding the power of logic gates, signals, and the system clock. While the static power is practically unrelated to the architecture used and is mostly caused by the leakage current of the transistors when the system is powered on but not configured. Figures 8 shows the amount of the static and dynamic power consumed for the median based architecture, where Figure 8(a) shows the consumed power for the case of window size of 16 sample, Figure 8(b) for the window size of 32 sample, Figure 8(c) for the window size of 64, and Figure 8(d) for window size of 128, while the data for the design with 256 is not available because it couldn't be implemented with selected FPGA kit, Figure 9 shows the amount of the static and dynamic power consumed for the RMS based architecture, where Figure 9(a) shows the consumed power for the case of window size of 16 sample, Figure 9(b) for the window size of 32 sample, Figure 9(c) for the window size of 64, Figure 9(d) for window size of 128 and Figure 9(e)

for the window size of 64, also the consumed power for the P84 based architecture is shown in Figure 10, with widow size of 16 in Figure 10(a), window size of 32 in Figure 10(b), window size of 64 in Figure 10(c), window size of 128 in Figure 10(d), and window size of 256 in Figure 10(e). Table 6, summarize the consumed power for the three studied algorithms.

Table 6. The consumed power of the three tested algorthim with different window size

Window size	Power type	Median based architecture	RMS based architecture	P84 based architecture
16	Dynamic	0.110 W	0.017 W	0.001 W
	Static	0.121 W	0.118 W	0.118 W
32	Dynamic	0.324 W	0.017 W	0.001 W
	Static	0.122 W	0.118 W	0.118 W
64	Dynamic	1.182 W	0.018 W	0.001 W
	Static	0.137 W	0.118 W	0.118 W
128	Dynamic	2.905 W	0.019 W	0.001 W
	Static	0.187 W	0.118 W	0.118 W
256	Dynamic	Data unavailable	0.023 W	0.001 W
	Static		0.118 W	0.118 W

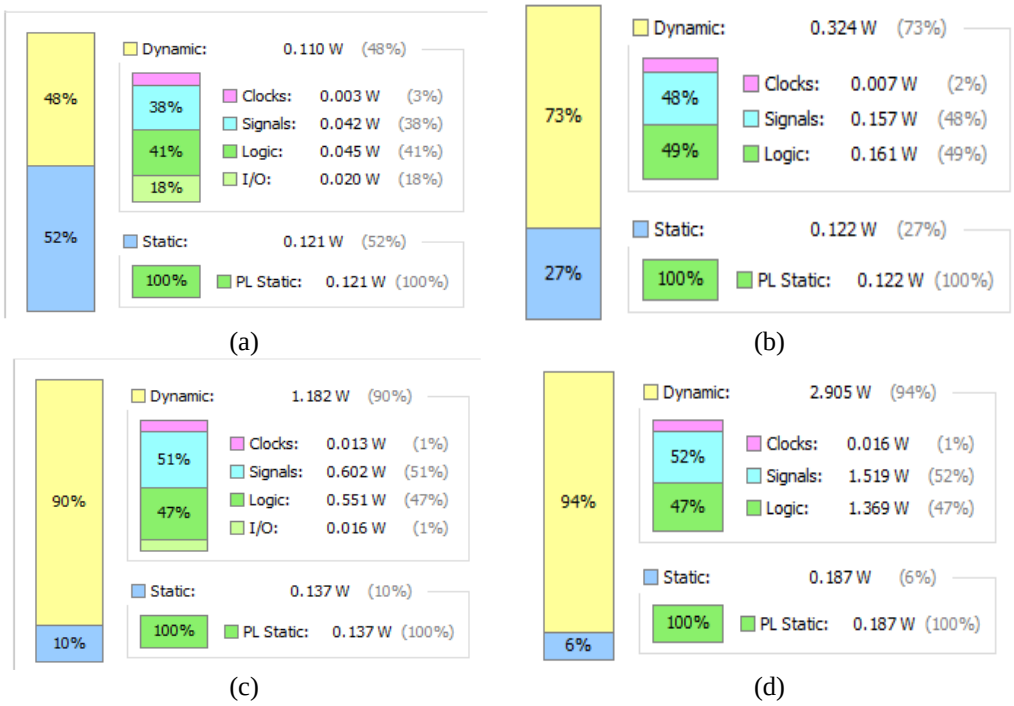

Figure 8. The consumed power by the architecture of median based method, (a) window size of 16, (b) window size of 32, (c) window size of 64, and (d) with window size of 128.

3.4. System speed

The other aspect to be considered in the comparison is the maximum system speed at which it can operate. The maximum speed is decided by the time of the longer (critical) path of the system. That is, samples from the shorter path arrived at a node first and couldn't propagate forward until the samples from the longer path arrived, limiting the system's speed.

The system clock is set to 100 MHz, in the development board used for the tests, (10 ns clock time (T_s)). The Vivado design suite used to synthesis and analysis the tested system didn't generate the maximum frequency directly, but it can be determined from the "Worst Negative Slack" (WNS) given in the timing report, as shown in (7) [26], [27]:

$$f_m = \frac{1}{T_s - WNS} \quad (7)$$

where f_m is the maximum frequency and T_s is the system clock time.
WNS: is the worst negative slack or worst path slack.

The WNS can be any value (positive or negative). if the value of WNS is positive, it indicates that the implemented system satisfies timing constraints. If WNS value is negative, it indicates the system cannot be synthesised. Table 7 stated the speed of all the three architects with the different window size.

Figure 9. The consumed power by the architecture of RMS based method, (a) window size of 16, (b) window size of 32, (c) window size of 64, (d) with window size of 128, and (e) with window size of 256.

Table 7. System speed for the three assessed method with different window size

Window size	Parameter	Median based architecture	RMS based architecture	P84 based architecture
16	WSN	7.662 ns	0.267 ns	2.197 ns
	Max. freq.	427 MHz	102 MHz	128 MHz
32	WSN	7.193 ns	0.518 ns	2.707 ns
	Max. freq.	356 MHz	105 MHz	137 MHz
64	WSN	7.424 ns	0.068 ns	2.343 ns
	Max. freq.	388 MHz	100 MHz	130 MHz
128	WSN	4.154 ns	0.484 ns	2.343 ns
	Max. freq.	171 MHz	105 MHz	130 MHz
256	WSN	Data unavailable	0.573 ns	2.484 ns
	Max. freq.		106 MHz	133 MHz

Figure 10. The consumed power by the architecture of P84 based method, (a) window size of 16, (b) window size of 32, (c) window size of 64, (d) with window size of 128 and (e) with window size of 256

4. COMPARISON AND DISCUSSION

In the previous section, a full comparison is made between these three techniques for different criteria. From that comparison, the following points can be noticed;

- For output accuracy, both Median based and RMS based methods show comparative results and outperform the P84 methods.
- Median based method shows better results with windows size of 32 for all signals tested with different noise levels, while in RMS based method the best performance fluctuates between window size 32 and 64 in most cases. So, it is clear that there is no much benefit in going for larger windows size. While in P84 based method, windows or frame size of 16 sample get the larger score, and that clear since the window or frame size should be changed more frequently to catch the changes in noise levels.
- The utilized hardware is lower in the P84 based method, while, median based method utilized the most resources to some extent, that couldn't be synthzed.
- It should be noticed that increase in window size for median based, caused the amount of hardware resources to increase exponentially, unlike the other two systems in which there is no noticeable increase in the hardware resources used.

- For the speed criteria, the median based is the fastest in speed among the other, since its structure containing only simple operation (comparator only).
- Even though a lot of advancement has been made in the design of FPGA chips in the last few decades, the process of multiplication remains difficult to implement and requires specific DSP units that fill a large silicon chip area. In the new FPGA devices, there are only a few of these DSP units, e.g., DSP48. Because the RMS based algorithm needs to figure out the square and square root of the input data, this method needs to utilize the DSP unit, which makes it the slowest of the systems we tried.
- In both RMS based and P84 techniques, the consumed power approximately independent on the size of the window, while in median based method, the power consumption increases exponentially as the size of the window increase, giving P84 a clear edge over the other two methods. Indeed, it is clear that P84 outperforms the other two methods in term of the consumed power.
- Another significant factor to consider is the ease with which the size of the window can be changed. Regarding the architectures of the three implemented algorithms, it is clear that the window size can easily be changed in the P84-based technique by varying the counter limit, while the entire sorting network needs to be changed in the median-based algorithm to change the window size, making it very difficult to change the size of the window. That gives an advantage to the P84 algorithm over the two other methods.

5. CONCLUSION

In this paper, we investigate the design and implementation of three methods that are most widely used in estimating the standard deviation in noisy signals. The comparison shows the advantage and disadvantage of each method, without deciding the favorability of method over other, because it is assumed that the application needs are the best decider for the most suitable, such as speed, low power or low resources is wanted. For example, the medical applications need low power device without concerning the speed, while in applications such as digital communication it desired speed over consuming power and resources used.

REFERENCES

[1] Z. Zhang and T. G. Constandinou, "Adaptive spike detection and hardware optimization towards autonomous, high-channel-count BMIs," *Journal of Neuroscience Methods*, vol. 354, p. 109103, Apr. 2021, doi: 10.1016/j.jneumeth.2021.109103.

[2] G. Wang *et al*., "ECG signal denoising based on deep factor analysis," Biomedical Signal Processing and Control, vol. 57, p. 101824, Mar. 2020, doi: 10.1016/j.bspc.2019.101824.

[3] P. Song, Y. Tan, X. Geng, and T. Zhao, "Noise reduction on received signals in wireless ultraviolet communications using wavelet transform," *IEEE Access*, vol. 8, pp. 131626–131635, 2020, doi: 10.1109/ACCESS.2020.3009944.

[4] J. Wang *et al*., "A novel underwater acoustic signal denoising algorithm for gaussian/non-gaussian impulsive noise," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 429–445, Jan. 2021, doi: 10.1109/TVT.2020.3044994.

[5] Z. Chen, Z. Zhou, and S. Adnan, "Joint low-rank prior and difference of Gaussian filter for magnetic resonance image denoising," *Medical and Biological Engineering and Computing*, vol. 59, no. 3, pp. 607–620, Mar. 2021, doi: 10.1007/s11517-020-02312-8.

[6] Y. E. Gökdağ, F. Şansal, and Y. D. Gökdel, "Image denoising using 2-D wavelet algorithm for Gaussian-corrupted confocal microscopy images," *Biomedical Signal Processing and Control*, vol. 54, p. 101594, Sep. 2019, doi: 10.1016/j.bspc.2019.101594.

[7] G. Baldazzi *et al*., "Systematic analysis of wavelet denoising methods for neural signal processing," *Journal of Neural Engineering*, vol. 17, no. 6, p. 066016, Dec. 2020, doi: 10.1088/1741-2552/abc741.

[8] A. Soleymankhani and V. Shalchyan, "A new spike sorting algorithm based on continuous wavelet transform and investigating its effect on improving neural decoding accuracy," *Neuroscience*, vol. 468, pp. 139–148, Aug. 2021, doi: 10.1016/j.neuroscience.2021.05.036.

[9] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard, "Wavelet Shrinkage: Asymptopia?," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 57, no. 2, pp. 301–337, Jul. 1995, doi: 10.1111/j.2517-6161.1995.tb02032.x.

[10] S. Mallat, *A Wavelet Tour of Signal Processing The Sparse Way*, Academia, vol. 59, no. 3. 2009.

[11] K. S. Guillory and R. A. Normann, "A 100-channel system for real time detection and storage of extracellular spike waveforms," *Journal of Neuroscience Methods*, vol. 91, no. 1–2, pp. 21–29, Sep. 1999, doi: 10.1016/S0165-0270(99)00076-X.

[12] R. R. Harrison and C. Charles, "A low-power low-noise CMOS amplifier for neural recording applications," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 6, pp. 958–965, Jun. 2003, doi: 10.1109/JSSC.2003.811979.

[13] F. Rummens, S. Ygorra, H. C. Mayiss Boussamba, S. Renaud, and N. Lewis, "Theoretical study and optimisation of a standard deviation estimator circuit for adaptive threshold spike detection," *International Journal of Circuit Theory and Applications*, vol. 44, no. 9, pp. 1742–1757, Sep. 2016, doi: 10.1002/cta.2192.

[14] G. G.-Turcotte and B. Gosselin, "Multichannel spike detector with an adaptive threshold based on a Sigma-delta control loop," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, Aug. 2015, vol. 2015-November, pp. 7123–7126, doi: 10.1109/EMBC.2015.7320034.

[15] P. H. Thakur, H. Lu, S. S. Hsiao, and K. O. Johnson, "Automated optimal detection and classification of neural action potentials in extra-cellular recordings," *Journal of Neuroscience Methods*, vol. 162, no. 1–2, pp. 364–376, May 2007, doi: 10.1016/j.jneumeth.2007.01.023.

[16] N. Li, M. Sawan, and L. Wang, "An efficient adaptive online neural spikes detection and classification engine based on Bayesian inference," in *2019 6th International Conference on Systems and Informatics, ICSAI 2019*, Nov. 2019, pp. 1100–1104, doi: 10.1109/ICSAI48974.2019.9010516.

[17] P. Goel, M. Chandra, A. Anand, and A. Kar, "An improved wavelet-based signal-denoising architecture with less hardware consumption," *Applied Acoustics*, vol. 156, pp. 120–127, Dec. 2019, doi: 10.1016/j.apacoust.2019.07.013.
[18] K. Naveed and N. U. Rehman, "Wavelet based multivariate signal denoising using mahalanobis distance and EDF statistics," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5997–6010, 2020, doi: 10.1109/TSP.2020.3029659.
[19] M. Codish, L. C.-Filipe, T. Ehlers, M. Müller, and P. S.-Kamp, "Sorting networks: To the end and back again," *Journal of Computer and System Sciences*, vol. 104, pp. 184–201, Sep. 2019, doi: 10.1016/j.jcss.2016.04.004.
[20] A. Norollah, D. Derafshi, H. Beitollahi, and M. Fazeli, "RTHS: A Low-cost high-performance real-time hardware sorter, using a multidimensional sorting algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 7, pp. 1601–1613, Jul. 2019, doi: 10.1109/TVLSI.2019.2912554.
[21] Y. Yang, S. Boling, and A. J. Mason, "A hardware-efficient scalable spike sorting neural signal processor module for implantable high-channel-count brain machine interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 4, pp. 743–754, Aug. 2017, doi: 10.1109/TBCAS.2017.2679032.
[22] F. Samann, S. A. Bamerni, J. A. Khorsheed, and A. K. Al-Sulaifanie, "Adaptive real-time wavelet denoising architecture based on feedback control loop," *Journal of Engineering Research (Kuwait*), vol. 9, pp. 1–8, 2021, doi: 10.36909/jer.v9iICRIE.11667.
[23] H. Hao, J. Chen, A. Richardson, J. Van Der Spiegel, and F. Aflatouni, "A 10.8 μw neural signal recorder and processor with unsupervised analog classifier for spike sorting," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 15, no. 2, pp. 351–364, Apr. 2021, doi: 10.1109/TBCAS.2021.3076147.
[24] L. Zhang, X. Xie, S. Feng, and M. Luo, "Heuristic dual-tree wavelet thresholding for infrared thermal image denoising of underground visual surveillance system," *Optical Engineering*, vol. 57, no. 08, p. 1, Aug. 2018, doi: 10.1117/1.OE.57.8.083102.
[25] S. A. Bamerni and A. K. Al-Sulaifanie, "An efficient non-separable architecture for Haar wavelet transform with lifting structure," *Microprocessors and Microsystems*, vol. 71, p. 102881, Nov. 2019, doi: 10.1016/j.micpro.2019.102881.
[26] L. Zhang, "System generator model-based FPGA design optimization and hardware co-simulation for Lorenz chaotic generator," in *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems, ACIRS 2017*, Jun. 2017, pp. 170–174, doi: 10.1109/ACIRS.2017.7986087.
[27] G. Haritha, K. Sundaramoorthy, and A. Sankar, "Xilinx system generator-based rapid prototyping of solid-state transformer for on-grid renewable energy integration," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 9, no. 2, pp. 1280–1289, Apr. 2021, doi: 10.1109/JESTPE.2019.2963277.

BIOGRAPHIES OF AUTHORS

Serwan Bamerni is lecturer at Electrical and computer Engineering, department, University of Duhok, Kurdistan region, Iraq. He received B.Sc. degree in Electrical Engineering from University of Baghdad and M.Sc. degree in solid state electronics from University of Technology, Baghdad, Iraq, in 1997 and 2002 respectively and Ph.D. in in Electrical and Computer Engineering at University of Duhok. His research field include VLSI architectures and algorithms for signal processing, and reconfigurable computing for multimedia systems. He can be contacted at email: serwan.mohammed@uod.ac.

Ahmed K. Al-Sulaifanie is Professor at Electrical and computer Engineering department, University of Duhok, Kurdistan region, Iraq. He received both his B.S. and M.Sc. degree in Electrical Engineering-Electronics and Communications College of Engineering at University of Mosul in 1977 and 1980 respectively and Ph.D. in Electrical Engineering-Computer Architectures at University of Mosul in 2003. His research interests include VLSI architectures and algorithms for signal processing data scheduling optimization, especially on multimedia applications. He can be contacted at email: ahmed.khorsheed@uod.ac.