A Suitable Hardware Realization for the Cohen Class Distributions

Irena Orović, *Member, IEEE*, Srdjan Stanković, *Senior member, IEEE* and Branka Jokanović

***Abstract*— A multi-distribution hardware for realization of Cohen class distributions is proposed. This approach is based on the modified form of kernel function that is computationally efficient and suitable for real-time implementation. The proposed solution includes various kernels based on the exponential and sine function, with possibility to choose different kernel parameters. The proposed architecture is implemented using the FPGA and can be used in high-speed real-time applications.**

***Index Terms*— ambiguity function, Cohen class of distributions, flexible hardware realizations, kernel function**

I. Introduction

NOWADAYS, the time-frequency (TF) distributions have been used as a powerful tool for the analysis of signals with time-varying spectral content [1]-[5]. Some interesting applications based on TF signal processing include analysis of 3G telecommunication signals, detection of echoes, diagnostics and telemedicine using biomedical signal analysis (arrhythmia detection, swallowing difficulties, etc), [3],[6],[7]. A large number of TF distributions have been proposed in order to deal with different kinds of signals. Namely, most of the TF distributions are appropriate only for specific signal type, while they may exhibit significant drawbacks in other cases. For example, the spectrogram has a low TF resolution, WD produces cross-terms for multicomponent signals, while the S-method does not satisfy marginal conditions. In order to overcome the aforementioned problems, the Cohen class distributions are commonly employed. They are based on the advantages of using the ambiguity domain, where the auto–terms are concentrated around the origin, while the cross-terms are dislocated away. Then the 2D low pass filter function called kernel is used for cross-terms suppression. The kernel type specifies a particular distribution from the Cohen class and determines its properties. The class includes variety of distributions, and even the spectrogram and the WD belong to the general Cohen class as simplified cases.

Although the Cohen class distributions are very important in practical applications, the related multi-distribution hardware solution has not been considered yet (to the best of our knowledge). The main reason is that the Cohen class includes large number of kernels having different forms and parameters, whose implementation would require high computational complexity and significant hardware resources. Hence, most of the existing systems for TF analysis are based on individual simple forms such as the spectrogram, the WD or the S-method [6]-[13]. In this paper we propose a modified formulation of kernels which leads to efficient multi-distribution hardware solution for Cohen class implementation. It allows us to choose various distributions with different parameters for each particular kernel. Unlike other systems that are designed for specific signal characteristics, the proposed one provides flexibility to work with different types of signals within different applications, i.e., it is independent on signal and application type.

The paper is organized as follows. The theoretical background is given in Section II. The hardware realization of Cohen class distributions is presented in Section III. The FPGA implementation is given in Section IV, while Section V contains comparison and discussion about achieved performance. Concluding remarks are given in Section VI.

II. Theoretical Background and Discussion

The Cohen class of distributions in the discrete form can be defined as, [1]:

$$CD(n,k) = 2\sum_{p=-N_1}^{N_1}\sum_{m=-N_2}^{N_2} c(p,m)A(p,m)e^{-j\frac{2\pi}{N}(-pn+\frac{N}{N_p}km)},$$
$$A(p,m) = \sum_{u=-N_1}^{N_1} x(u+m)x^*(u-m)e^{-j\frac{2\pi}{N}pu}. \quad (1)$$

where $c(p,m)$ and $A(p,m)$ are kernel and ambiguity function, respectively, while $N=2N_1+1$ and $Np=2N_2+1$. The realization of the Cohen class distribution can be summarized as follows:

1. Calculation of the auto-correlation function;
2. Ambiguity function and kernel computation;
3. Filtering in the ambiguity domain by using the kernel;
4. Applying inverse 2D Fourier transform to the filtered ambiguity function.

The most challenging step is realization of the kernel function. In this paper we will consider kernels based on exponential and sine function, since they are majority in the Cohen class (e.g., Choi-Williams, Gauss, Born-Jordan) [1]. Regarding the realization possibilities, we will discuss two conventional methods: Lookup tables (LUTs) and Taylor series expansion for direct implementation of these kernel functions (kernel argument denoted as $\psi(p,m)$).

The LUTs could be a simple solution for a certain kernel function, but only if we use fixed shape and parameters values. However, due to variety of kernels and different parameters that should be used for different applications, the LUTs based approach would require large number of memory units.

The authors are with the University of Montenegro, Faculty of Electrical Engineering, 20000 Podgorica, Montenegro. The corresponding author email is: irenao@ac.me

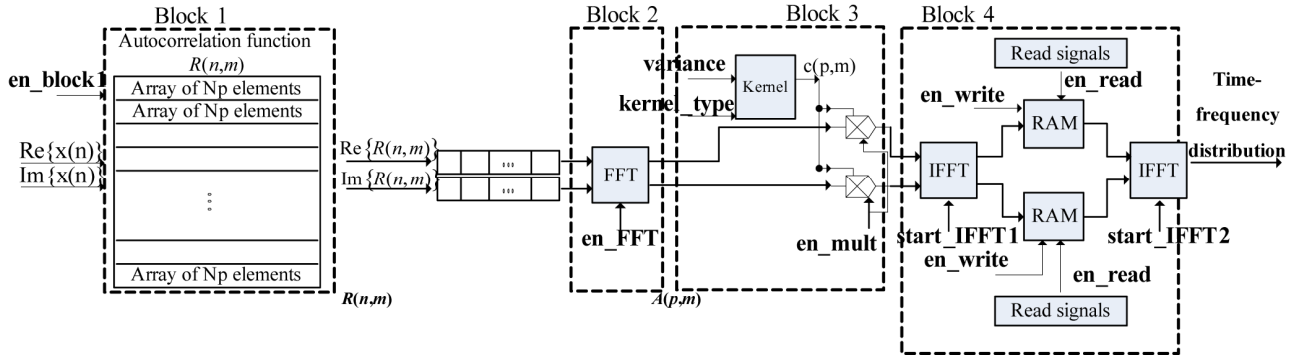

Fig. 1. Serial configuration for the realization of the Cohen class distributions

As an illustration, consider a kernel slope as an important parameter, which should be adapted to each particular signal case. Hence, if we use M=12 bits precision for slope values, then 2^{12} matrices will be needed for implementation (one matrix for each slope value). Then the memory size will be $2^M \cdot N_b \cdot (N - N_p + 1) \cdot N_p$, where N is signal length, N_b is the number of bits used for signal representation, while N_p is the window size. Now if we have only five different kernels, while N=256, N_p=128 and N_b=32, the memory size becomes approximately 10.65 Gb, which is large and cannot be stored on standard FPGA components. Also, a specific control circuit would be required for memory searching, which would be exhaustive anyway. Therefore, instead of the LUTs, we need a more appropriate approach for realization of kernel function.

Another possible method for kernel implementation could be based on the Taylor series expansion. Observe four expansion terms for argument in the range (-1,1). If $\psi(p,m)$ is outside this range, Taylor series converges slowly, which means that higher order terms should be included. More expansion terms cause higher computational time. For example, if the argument has a value $\psi(p,m)$=2 and if four expansion terms are used, then $e^{\psi(p,m)}$= 6.3333, while the correct result is 7.3891. Further, to obtain result close to the correct one we need 11 terms, which require large number of operational units that introduce significant latency. The similar analysis holds for the kernels based on the sine function.

Note that the presented methods are either limited by their calculation precision or they are hardly feasible due to the large number of operations required. Hence, a more optimal solution will be proposed in the following section.

III. Hardware Realization

The block scheme for serial realization of Cohen class distributions is proposed. It is designed according to the steps given at the beginning of Section II.

*A. Hardware for ambiguity function – **Block 1 and Block 2***

The real and imaginary parts of complex signal $x(n)$ are fed to the input of the system. ***Block 1*** is used for calculation of auto–correlation function. The calculation of the local auto-correlation function $R(n,m)$ at a time instant n is based on multiplication of $x(n+m)$ and $x^*(n-m)$, for $m \in [-N_2, N_2]$. In matrix form $R(n,m)$ can be written as:

$$\mathbf{R} = \begin{bmatrix} x(1)x^*(Np) & \cdots & x(Np)x^*(1) \\ \vdots & \vdots & \vdots \\ x(N-Np+1)x^*(N) & \ldots & x(N)x^*(N-Np+1) \end{bmatrix}. \quad (2)$$

Each output (for real and imaginary part of $R(n,m)$) requires one adder/subtractor and two multipliers. In the serial configuration, the columns of **R** are sequentially obtained at the output of ***Block 1***. In order to compute ambiguity function, ***Block 2*** performs the fast Fourier transform (FFT) over the columns of **R** (one FFT block is required). The ambiguity function is obtained at the output of ***Block 2***. Note that the size of **R** ($N_p \times (N-N_p)$) depends on the window width N_p and data length N. Increasing the window width will increase the number of FFT points, while the number of sequential FFT operations depends on the signal length.

*B. A form of the Cohen class distributions suitable for hardware realization – **Block 3***

First, we will introduce a form of the Cohen class distributions with exponential kernels (as the dominant case) suitable for hardware realization, as follows:

$$CD(n,k) = 2 \sum_{p=-N_1}^{N_1} \sum_{m=-N_2}^{N_2} (1 + \psi_{M/2}(p,m)) \prod_{i=0}^{M/2-1} c_{p,m}(i) \times \times A(p,m) e^{-j\frac{2\pi}{N}pn - j\frac{2\pi}{N_p}km}, \quad (3)$$

where $c_{p,m}(i)$ and $\psi_{M/2}(p,m)$, for a given point (p,m), are constants calculated using an iterative procedure, described below. For kernels based on the sine function, the proposed form (3) is slightly modified as:

$$CD(n,k) = 2 \sum_{p} \sum_{m} \mathrm{Im}\left((1 + j\psi_{M/2}(p,m)) \frac{\prod_{i=0}^{M/2-1} c_{p,m}(i)}{\alpha(p,m)}\right) \times \times A(p,m) e^{-j\frac{2\pi}{N}pn - j\frac{2\pi}{N_p}km}. \quad (4)$$

For instance, $\alpha(p,m) = pm/2$ for the Born-Jordan kernel. The constants are given in the following form [14]:

$$c_{p,m}(i) = \begin{cases} 1 + a(i)2^{-i}, & a(i) \in \{0,1\}, \text{for the exp. function,} \\ 1 + jb(i)2^{-i}, & b(i) \in \{-1,1\}, \text{for the sine function.} \end{cases} \quad (5)$$

The values $a(i)$ (or $b(i)$ for sine function) are calculated within $M/2$ iterations, where M is the number of bits that defines the computational precision. The sequence of binary values $a(i)$ (or $b(i)$) determines $\prod_{i=0}^{M/2-1} c_{p,m}(i)$. Hence, there are $2^{M/2}$ constants. In order to optimize hardware resources, the

constants are pre-calculated for all binary combinations **a**:{$a(i)$, $i\in[0,M/2\text{-}1]$} (**b**:{$b(i)$, $i\in[0,M/2\text{-}1]$}) and stored in the memory. The sequence **a** (or **b**) is the memory input and determines which address will be read. It is important to emphasize that the same memory unit is employed for all samples of kernel function. Fig. 2 shows how the constants in memory are determined by the sequence **a**.

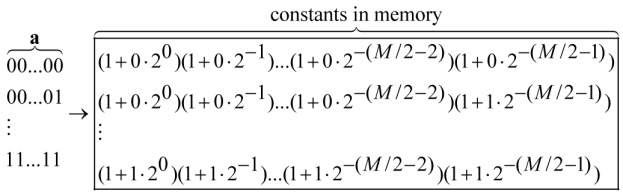

Fig. 2. LUT for the exponential function: the sequence **a** determines the address of the corresponding constant in memory

The value $\psi_{M/2}(p,m)$ is automatically obtained as the residual after $M/2$ iterations. The iterative procedure can be summarized through the following steps:

Step 1: $\psi_0(p,m)=\psi(p,m)$,

Step 2: $$a(i)=\begin{cases}1, if\ \psi_i(p,m)-\ln(1+2^{-i})\geq 0,\\ 0, otherwise,\end{cases}$$

$$b(i)=\begin{cases}1, if\ \psi_i(p,m)\geq 0,\\ -1, otherwise,\end{cases},$$

Step 3: $\psi_{i+1}(p,m)=\psi_i(p,m)-\ln(1+a(i)2^{-i})$, for exp. func.

$\psi_{i+1}(p,m)=\psi_i(p,m)-b(i)\tan^{-1}(2^{-i})$, for sine func.

Step 4: If $i=M/2-1$ holds, the procedure is finished, otherwise go to **Step 2**.

Note that $\psi_{M/2}(p,m)$ is obtained in **Step 3** for the last iteration $i=M/2\text{-}1$. More details about the exponential and sine function decompositions are given in the Appendix. In the considered procedure the argument of exponential function is limited to the range [0,1.56], [14]. However, the argument $\psi(p,m)$ of the kernel function may have an arbitrary value. We can overcome this limitation by writing $\psi(p,m)$ as:

$$\psi(p,m)=(I+F)\ln 2, \quad (6)$$

where I is the integer part, while F is the fractional part of $\psi(p,m)\log_2 e$ ($0\leq F<1$). The exponential function becomes:

$$e^{\psi(p,m)}=e^{(I+F)\ln 2}=2^{I}e^{F\ln 2}. \quad (7)$$

Now, the argument of exponential function is $F\ln 2$ and it is within the range [0, 0.69). Multiplication of $e^{F\ln 2}$ with 2^I is simply performed by adding I to the exponent of floating point number $e^{F\ln 2}$. For the sine function the previously described mathematical operations are not needed.

Block3: The realization of kernel and its multiplication with the ambiguity function is done within the ***Block 3*** (Fig. 1). The pins ***kernel_type*** and ***variance***, used to compute the argument of kernel function, are at the input of ***Block 3***. In the case of exponential function, the argument range is firstly scaled to [0,0.69) and then it is fed to the part of the system performing Steps 2 and 3 of iterative procedure (Fig. 3, blocks B_1,..., $B_{M/2}$). Each block performs comparison (Step 2) and subtraction (Step 3), as well as the setting of **a**(i), $i\in[1,M/2]$. As a result, the address ***a*** is fed to the ROM (LUT in Fig. 2), while $\psi_{M/2}(p,m)$ is increased by 1 and multiplied by the scaling factor 2^I (Fig. 3). After multiplication with the output form the ROM, the point (p,m) of kernel function is obtained. Thus, the kernel design includes $M/2$ subtractors, $M/2$ comparators, one adder, two multipliers and one ROM. At the output of Block 3, the product of ambiguity and kernel function is obtained.

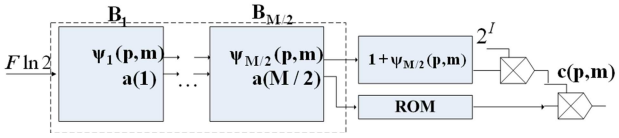

Fig. 3. The block scheme for the realization of kernel

C. Hardware for the inverse 2D FFT – *Block 4*

The 2D inverse IFFT is realized in ***Block 4***. The operation can be written as a composition of the 1D IFFT: $IFFT_2(\mathbf{B})=IFFT(IFFT(\mathbf{B})^T)^T$, where **B** represents an arbitrary matrix. In our case, **B** is a product of kernel and ambiguity function. The rows of matrix **B** are fed to the input of the IFFT block. The output of IFFT block is a matrix of size $(N\text{-}Np+1)\times Np$ which is stored in RAM (Fig. 4). The next step is the realization of a circuit which provides that the transpose matrix is obtained at the RAM output. This circuit is simply realized as a counter (Fig. 4) which generates the addresses for reading the elements over matrix columns. The elements at the output of RAM appear in the following order: 1, Np+1, 2Np+1, ..., 2, Np+2, 2Np+2, ..., until the last column is read.

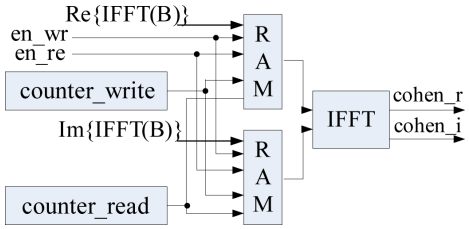

Fig. 4. The part of block that performs 2D IFFT

IV. FPGA REALIZATION OF THE PROPOSED SYSTEM

Having in mind that the FPGA offers reprogrammability, high degree of parallelism with lower cost compared to ASIC and higher speed compared to DSP, we choose FPGA for implementation of the proposed system. The simulation is performed using Quartus II v8.0 platform on 64-bit Windows 7 (Intel Core2Duo 3 GHz, 8 GB RAM). Here we describe the serial FPGA realization using device from Stratix III family.

The implementation of the proposed system is based on the architecture design given in Section III. In this realization N=256 samples of an 8-bit signed signal are fed to the input of the system where Np=128. This results in autocorrelation matrix of size 128x128. Note that the 8-bit representation is used since it is usually sufficient for various types of signals including 3G communication signals [6], but we can easily increase the number of bits at inputs. At the outputs of the blocks for computing autocorrelation function, 16-bit signals are obtained. To achieve higher precision, these signals are represented in the floating point arithmetic. The columns of autocorrelation matrix are fed to the FFT block (inbuilt FPGA

component), to calculate the ambiguity function. The choice of kernel function is done by using the input pin ***kernel_type***. Without loss of generality, here we consider three specific kernels: Choi-Williams, Gauss and Radial Gauss. Hence, the input pin ***variance*** controls the slope of the considered kernels. In our realization, the high calculation precision is achieved by 12 iterations, i.e., $M/2=12$.

The utilization of chip characteristics (EP3SL150F1152I3) obtained for the entire system is shown in Table I. The combinational *ALUTs* are used to perform arithmetic and logic operations. The *block memory bits* and *dedicated logic registers* are used to store different constants (for kernels, FFT and other calculations). Table II shows the logic utilization for different blocks used in the realization, i.e., their breakdown area for the specified chip. Note that the proposed redesigned multikernels solution – Block 3, requires the smallest percentage of logic utilization (11%). Blocks 1 and 2 with 36% of logic utilization corresponds to the complexity level of the Wigner distribution. The most demanding is Block 4, which is 2D IFFT block. As an advanced solution, one might consider, the DFT architecture which avoids large stride memory access and transpose operations [15], or alternatively new form of FFT chips based on fused floating-point operations [16]. The throughput of the proposed system is 1.6 Gbps, while the latency at the output is 0.35ms. Clock cycle is 50 MHz. The most demanding part is FFT (regarding operational clock frequency that allows correct functional behavior of the system), especially due to complex multipliers. Throughput is calculated as the number of bits obtained at the output in 1s, for single-precision floating-point arithmetic and for specified clock cycle. Note that this implementation applies block-level pipelining, based on simultaneous calculation of the ambiguity function and kernel. Finally, we can conclude that the proposed design is suitable for real time application, where the speed is of MHz order. The control signals (Fig.1) and number of clock cycle, when they become active are:

en_block1: 1	**en_FFT: 134**	**kernel_type: 581**
variance:581	**en_mult: 652**	**enIFFT1: 658**
en_write: 1045	**en_read: 17429**	**enIFFT2: 17430**

TABLE I. CHARACTERISTICS FOR THE CHIP IN SERIAL CONFIGURATION

EP3SL150F1152I3	**Available**	**Utilized**
No. of pins	744	171(23%)
Combinational ALUTs	113,600	76,047(67%)
Total block memory bits	5,630,976	2,538,193(45%)
Dedicated logic registers	113,600	65,203(57%)

TABLE II. LOGIC UTILIZATION FOR DIFFERENT BLOCKS IN SERIAL CONFIGURATION

	Block 1	Block 2	Block 3	Block 4
Logic utilization	12%	24%	11%	48%

V. COMPARISON AND DISCUSSION

The goal of the proposed solution is to overcome the drawbacks that limit the applicability of some existing hardware implementations of TF distributions, such as of the spectrogram, the WD and the S-method [8]-[10]. Namely, the spectrogram produces quite low TF resolution, the WD suffers from strong cross-terms in the case of multicomponent signals, while the S-method does not satisfy desirable marginal conditions. On the other hand, the proposed multi-distribution hardware is able to solve the above issues and to provide an adequate TF representation for various signals. For instance, let us consider a signal composed of two chirps. The output results for the spectrogram, WD and Cohen class distribution with Gaussian kernel are shown in Fig. 5. The spectrogram has poor TF resolution, while the WD is seriously affected by the cross-terms. In the case of multi-distribution, we have flexibility to choose different kernel parameters. Thus, we can adjust kernels to each particular signal case (Fig. 5, lower row). In the considered example, the proposed solution provides an optimal compromise between the concentration and cross-terms elimination, leading to a cross-terms free TF distribution. Furthermore, this hardware includes also the spectrogram and WD (with simplified kernel equal to 1) as special cases. Consequently, the hardware complexity of multi-distribution implementation cannot be compared on a fair level with any of the simplified individual solutions. The increased hardware complexity is justified by its comprehensiveness, flexibility and adjustability. At the same time, we would like to emphasize that it is designed to provide real-time processing, and thus to be appropriate for real-world applications.

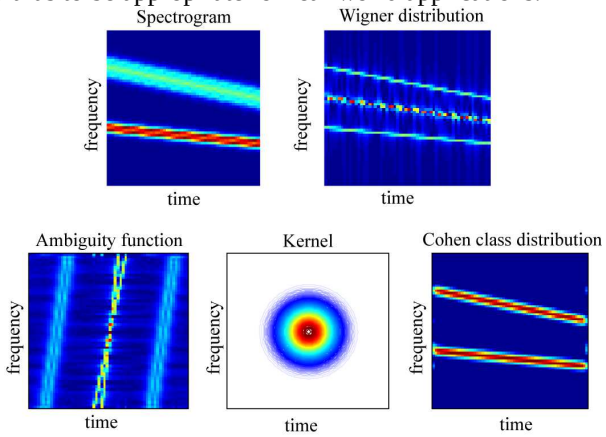

Fig. 5. Upper row: Spectrogram and WD; Lower row: Ambiguity function, Gauss kernel, Cohen class distribution at the system output

Furthermore, we will also compare the performance of the proposed realization with some conventional non-optimized approaches (discussed in Section II). Therefore, in Table III we provide the number of required logic and memory units for different kernel realizations: LUT, Taylor expansion and proposed approach. It can be seen that LUT based approach requires large memory, and as such cannot be stored on available FPGA platforms. Regarding the memory utilization, the proposed approach requires small, almost negligible, amount of memory, as well as Taylor series expansion with 0 bytes memory. However, Taylor expansion requires significantly higher number of multipliers/dividers. These components introduce significant latency and area requirements, especially for considered floating-point arithmetic. For example, in Table III, we consider 11 terms within the Taylor series. Hence, for the highest order term, we need 9 serial multipliers (no adders or dividers are counted) causing 45 clock cycles delay, or ~1μs, which is significant comparing to the total latency of the system which is 0.35ms.

Therefore, from the overview in Table III, we can conclude that the proposed realization is the most appropriate concerning both hardware resources and low-latency requirements. Additionally, we consider the calculation precision in terms of the mean squared error (MSE). The MSE introduced by using fixed point representation at the input of the proposed system, with $M/2=12$, is 0.0358% of the maximal signal value. In the case of Taylor expansion, the MSE can be up to 50 times higher for a certain value of the exponent.

TABLE III. COMPARISON OF LOGIC AND MEMORY UTILIZATION FOR DIFFERENT KERNEL REALIZATIONS

LUT	0 adders, 0 mult/dividers, 10.65 Gb of memory
Taylor (11 terms)	10 adders, 18 mult/dividers, 0 b of memory
Proposed (M/2=12)	13 adders, 2 mult/dividers, 128Kb of memory

Finally, the proposed system can be also realized using parallel architecture. The main benefit of parallel realization is lower latency (approximately N_p times less that in the serial realization), since each windowed part of input signal is independently processed. On the other side, hardware resources are approximately N_p times higher. It is important to note that kernel realization is the same in both architectures.

VI. CONTRIBUTIONS AND CONCLUSIONS

The performance of the Cohen class brought benefits for applications dealing with nonstationary signals. However, its real-time implementation was a challenging task, and to the best of our knowledge, has not been tackled so far. The main reason is that for each particular signal, we need to change either the kernel type or shape or parameters, because the results are very dependent on signal-kernel compatibility. Each kernel is a separate mathematical function, which would require separate hardware. In that sense, one of the major contributions of this paper is the unification of kernels definitions, which resulted in the modified Cohen class form suitable for hardware implementation. The kernels are re-designed to be uniformly represented using a set of constants (iteratively calculated), while maintaining high calculation precision as in the case of original definitions. In this way, we designed the multi-distribution comprehensive TF hardware that provides flexibility and adjustability to any signal or application. Furthermore, it provides high throughput, low output latency and high calculation precision.

ACKNOWLEDGEMENT

The authors are very thankful to the reviewers for comments that helped to improve the paper. We also thank to Dr Nikola Žarić for his help during the work on this paper.

APPENDIX

In this part, we will show that the kernel function can be calculated as the product of M constants. Let us consider the exponential function with positive argument $\psi(p,m)$ which is scaled to the range [0, 0.69). The idea is to write the argument $\psi(p,m)$ using M positive numbers:

$$\psi_1(p,m) \geq \ldots \psi_i(p,m) \geq \psi_{i+1}(p,m) \geq \ldots \psi_M(p,m) \rightarrow 0$$

Starting from $\psi_0(p,m)=\psi(p,m)$ and using relation (as in [12]):

$$\psi_{i+1}(p,m) = \psi_i(p,m) - \ln c_{p,m}(i), \quad (8)$$

we obtain the following system of equations:

$$e^{\psi_i(p,m)} = c_{p,m}(i) e^{\psi_{i+1}(p,m)}, i \in [0, M-1]. \quad (9)$$

The constants $c(p,m)$ are given in the form (5). Since, $\psi_0(p,m)=\psi(p,m)$ and $\psi_M(p,m)=0$ holds:

$$e^{\psi(p,m)} = c_{p,m}(0)...c_{p,m}(M-1) e^{\psi_M(p,m)} = \prod_{i=0}^{M-1} c_{p,m}(i). \quad (10)$$

Further, we show that the calculations can be reduced by using the partial product of constants as in (3). Namely, we calculate the product of $M/2$ constants by using (10). Then for the remaining $M/2$ constants, the approximation $\ln(1+2^{-i}) \approx 2^{-i}$ is used. In Step 2 of the iterative procedure $a(i)$ is calculated as:

$$a(i)=1, \text{ if } \psi_i(p,m)-2^{-i} \geq 0, \text{ otherwise } a(i)=0.$$

Namely, $a(i)$ has the value '1' if the i-th bit of $\psi_i(p,m)$ is '1', and vice versa. In other words, we do not need to calculate $a(i)$ for $i \geq M/2$, since they are contained within $\psi_{M/2}(p,m)$:

$$\prod_{i=M/2}^{M-1} (1+a(i)2^{-i}) \approx 1+\psi_{M/2}(p,m). \quad (11)$$

REFERENCES

[1] B. Boashash, *Time-Frequency Signal Analysis and Processing*, Elsevier, 2003, pp.1-29, 59-83, 231-241.

[2] Y. Zhang, W. Mu and M.G. Amin, "Subspace analysis of spatial time-frequency distribution matrices," *IEEE Trans. Signal Processing*, vol. 49, no. 4, pp. 747-759, Apr. 2001.

[3] N. Josso, C. Ioana, J. I. Mars and C. Gervaise, "Source motion detection, estimation and compensation for underwater acoustics inversion by wideband ambiguity lag-Doppler filtering," *Journal of the Acoustical Society of America*, vol. 128, pp. 3416-3425, 2010.

[4] P. Honeine, C. Richard, P. Flandrin, and J-B. Pothin: "Optimal selection of time-frequency representations for signal classification: A kernel target alignment approach," *ICASSP*, pp. 476–479, May 2006.

[5] I. Orovic, S. Stankovic, "A Class of Highly Concentrated Time-Frequency Distributions Based on the Ambiguity Domain Representation and Complex-Lag Moment," *EURASIP Journal on Advances in Signal Proc.*, vol. 2009, ID 935314, 9 pages

[6] L. D. Vito, S. Rapuano and G. Truglia, "A 3-D baseband signal analyzer prototype for 3G mobile telecommunication System," *IEEE Trans. Instrumentation and Measur.*, vol. 54, no.4, pp.1444-1452, Avg. 2005.

[7] A. Dliou, R. Latif, M. Laaboubi and F. M. R. Maoulainine, "Arrhythmia ECG signal analysis using non parametric time-frequency techniques," *Int. Journal of Computer Applications*, vol. 41, no.4, Mar. 2012.

[8] D. Petranovic, S. Stankovic, LJ. Stankovic, "Special purpose hardware for time frequency analysis," *Elect. Lett*, vol.33, no.6, pp.464-466, 1997

[9] S. Stankovic and LJ. Stankovic, "An architecture for the realization of a system for time-frequency analysis," *IEEE Trans. Circuits and Syst. II*, vol. 44, no.7, pp.600-604, Feb. 1997.

[10] M. G. Amin and K. D. Feng, "Short-time Fourier transform using cascade filter structures," *IEEE Trans. Circuits and Syst.*, vol.42, no. 10, pp.631-641, Oct. 1995.

[11] N. Harte, N. Hurley, C. Fearon and S. Rickard, "Towards a hardware realization of time-frequency source separation of speech," *European Conference on Circuit Theory and Design*, vol.1, pp. I/71 - I/74, 2005.

[12] N. Zaric, I. Orovic and S. Stankovic, "Hardware realization of generalized time-frequency distribution with complex-lag argument," *EURASIP Journal on Advances in Signal Proc.*, 2009.

[13] N. Zaric, N. Lekic and S. Stankovic, "An implementation of the L-estimate distributions for analysis of signals in heavy-tailed noise," *IEEE Trans. On Circuits and Syst. II*, vol. 58, no. 7, pp. 427-431, 2011.

[14] I. Koren, *Computer Arithmetic Algorithms*, Peters, 2002, pp. 225-247.

[15] C.L. Yu, K. Irick, C. Chakrabarti, V. Narayanan, "Multidimensional DFT IP Generator for FPGA Platforms," *IEEE Trans. on Circ. and Syst.*—I, vol. 58, no. 4, 2011, pp. 755-764

[16] E. E. Swartzlander, H. H.M. Saleh, "FFT Implementation with Fused Floating-Point Operations" *IEEE Trans. on Computers*, vol. 61, no. 2, 2012, pp. 284-288