Hardware Implementation of the Quasi Maximum Likelihood Estimator Core for Polynomial Phase Signals

Nevena R. Brnović [1], Igor Djurović [1,2], Veselin N. Ivanović [1*], Marko Simeunović [2,3]

[1] Electrical Engineering Department, University of Montenegro, Cetinjski put bb, 81000 Podgorica, Montenegro
[2] Institute for Cutting Edge Information and Communication Technologies, Džordža Vašingtona 66/354, 81000 Podgorica, Montenegro
[3] Faculty for Information Systems and Technologies, University of Donja Gorica, Oktoih 1, 81000 Podgorica, Montenegro
[*] very@ac.me

Abstract: Flexible, multiple-clock-cycle, hardware design for the quasi maximum likelihood (QML) algorithm core realization for the polynomial phase signals (PPSs) estimation is proposed. The QML algorithm significantly outperforms existing PPS estimators in terms of accuracy. However, its practical applications require efficient software and hardware systems. The main challenges in the proposed hardware development with respect to existing systems for time-frequency analysis are realization of time-frequency (TF) representation based instantaneous frequency (IF) estimator, the polynomial regression, and phase extraction. The developed design is tested on a PPS corrupted by a white Gaussian noise and verified by a field programmable gate array (FPGA) circuit design. All implementation and verification details are provided along with the comparison of the results achieved by hardware and software implementations.

1. Introduction

The polynomial phase signal (PPS) is the standard model of the nonstationary signals motivated by the Weierstrass theorem that states that any continuous time function can be modeled as a polynomial. This model has been studied for more than three decades with numerous techniques designed for parameter estimation. Three main groups of the parameter estimators are [1], [2]:

- Maximum likelihood (ML) estimators (accurate but with unacceptable complexity for PPS of order higher than 3);
- Phase unwrapping (PU) estimators (accurate only for narrowband signals and high signal-to-noise ratio (SNR) but inaccurate for wideband signals and/or more emphatic noise) [3];
- Phase differentiation (PD) estimators (efficient techniques but with limited accuracy especially for higher-order PPSs) [4]-[21].

For a long time due to efficiency reasons, the PD estimators were a primary research topic. They are reviewed in [1], [2]. However, their performance for higher-order PPS is not satisfactory. Recently an alternative approach has demonstrated excellent accuracy significantly improving results with respect to all three groups of estimators [22]-[29]. It is essentially a combination of the PU and the ML estimators with pre-processing. In the pre-processing stage, the rough estimate of the signal parameters is obtained by polynomial regression of the instantaneous frequency (IF) estimates. The IF estimates are obtained from the position of the short-time Fourier transform (STFT) maxima [22], [30], [31]. Note that there are special purpose hardware systems with different configurations for the STFT realization [32]-[39]. The STFT realizations are commonly used, as an initial stage, in more sophisticated systems dedicated for the realization of the advanced time-frequency (TF) representations [40]-[45], as well as for realization of the advanced time-varying and space-varying filtering systems, [46]-[49]. However, multiple STFTs are evaluated in the QML what is one of the main difficulties in both software and hardware realizations of this algorithm. After initial/rough PPS coefficients' estimates are obtained by polynomial regression of the IF estimates, the noisy signal is demodulated using signal reconstructed by roughly estimated PPS coefficients. The resulting signal is the PPS of the same order as initial one but it is narrowband and low-pass suitable for the PU estimation. This signal is filtered with low-pass moving average filter reducing noise influence without impact to the signal content. In the fine stage, the phase coefficients are estimated from the unwrapped phase of demodulated and filtered signal. This process is referred as the O'Shea refinement strategy [18]. The fine estimate is an aggregation of results from the rough and fine stages. The final algorithm output is selected based on the ML criterion from estimates calculated with multiple STFTs for various window widths. In this case, the search is performed over the single parameter, i.e., window width in the STFT, and not as in traditional ML algorithm realization where the search is performed over all phase parameters that is infeasible for higher-order PPSs.

Excellent results of the described technique qualify it for many practical applications, such as the radar/sonar systems, communications, sensor networks, or biomedical signals and systems. However, this technique also requires high calculation complexity that seriously reduces its applicability in real-time applications. Nevertheless, hardware implementation, if possible, can help in over-

coming of this nuisance. Therefore, it is an urgent need to develop the hardware for the QML algorithm realization.

The QML algorithm realization is challenging with evaluation of the STFT, IF estimation (the maxima STFT detection), polynomial regression of the IF and phase, demodulation, and filtering. A mitigating circumstance is the fact that all the above listed procedures are repeated in the same manner but for the different STFT window width. Hence, a hardware implementation of these procedures for a particular STFT window width is the key task that will be solved and presented in this paper.

The paper is organized as follows. The QML algorithm is presented in Section 2. Hardware implementation of the algorithm, for a particular STFT window width, is developed and described in Section 3. In Section 4, the developed design is tested on a noisy PPS and verified through the implementation on a field programmable gate array (FPGA) device. Finally, corresponding conclusions are derived in Section 5.

2. PPS Estimation and QML Algorithm

The *M*th order PPS can be described as

$$x(n) = f(n) + v(n), \ n \in [-S/2, S/2),$$
$$f(n) = A\exp(j\phi(n)) = A\exp\left(j\sum_{i=0}^{M} a_i (n\Delta)^i\right), \quad (1)$$

where $v(n)$ is complex zero-mean white Gaussian noise with variance σ^2, A the signal amplitude, $\phi(n)$ the signal phase, $a_i, i = 0,...,M$, the phase parameters, and Δ the sampling interval; S is a number of available samples. The IF of signal $f(n)$ is given as:

$$\omega(n) = \phi'(n) = \sum_{i=1}^{M} i\Delta a_i (n\Delta)^{i-1}, \quad (2)$$

where the first derivative is with respect to *n*. The problem of interest is to estimate signal parameters $\{A; a_i, i = 0,...,M\}$ from noisy observation $x(n)$.

The QML algorithm can be summarized with the following steps.

---

1. Calculate the STFT of $x(n)$ given by (1) with different window widths $STFT_h(n,\omega)$, $h \in H$:
$$STFT_h(n,\omega) = \sum_k x(n+k) w_h(k) \exp(-j\omega k\Delta),$$
$$k \in [-S/2 + h/2\Delta, S/2 - h/2\Delta), \quad (3)$$
where window function $w_h(k)$ is given as $w_h(k) = 1$ for $k \in [-h/2\Delta, h/2\Delta)$ and $w_h(k) = 0$ elsewhere; H is the set of window widths and S is the number of signal samples.

2. Estimate IFs for each window from the set $h \in H$:
$$\hat{\omega}_h(n) = \arg\max_{\omega} |STFT_h(n,\omega)|. \quad (4)$$

3. Perform polynomial interpolation of IF estimates to obtain a rough estimate of the phase parameters $\hat{\hat{\mathbf{a}}}_h = [\hat{a}_{h,1}, \hat{a}_{h,2}, ..., \hat{a}_{h,M}]$:
$$\hat{\hat{\mathbf{a}}}_h = (\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{\Gamma}^T\mathbf{\Omega}_h, \quad (5)$$
where index h denotes estimates obtained based on the window width h in the STFT. Matrix $\mathbf{\Gamma}$ is $S \times M$ with elements $\xi_{i,p} = pi^{p-1}$, where i are instants, $p \in [1, M]$, and $\mathbf{\Omega}_h$ is a column vector of IF estimates (4).

4. Perform dechirping, filtering, and phase unwrapping. Dechirping is done using unit amplitude signal obtained based on phase parameters estimates from the rough estimation stage:
$$\tilde{x}(n) = x(n)\exp\left(-j\sum_{i=1}^{M} \hat{a}_{h,i}(n\Delta)^i\right). \quad (6)$$
Noise influence is attenuated with the moving average filter:
$$\hat{x}(n) = \frac{1}{P}\sum_{l=-(P-1)/2}^{(P-1)/2} \tilde{x}(n+l) \quad (7)$$
where filter length is P (assumed odd number). Then the phase of dechirped (low-pass) and filtered signal $\hat{x}(n)$ is extracted and unwrapped
$$v(n) = \text{unwrap}(\text{phase}(\hat{x}(n))). \quad (8)$$

5. Now, residual phase parameters $a_i' = a_i - \hat{a}_i, i \in [1, M]$, can be estimated from $v(n)$ using polynomial regression:
$$\delta\hat{\hat{\mathbf{a}}}_h = (\mathbf{\Xi}^T\mathbf{\Xi})^{-1}\mathbf{\Xi}^T\mathbf{v}, \quad (9)$$
where elements of $(M+1) \times S$ matrix $\mathbf{\Xi}$ are $\xi_{i,p+1} = i^p$, i are instants, $p \in [0, M]$ are exponents corresponding to the phase polynomial coefficients, $\mathbf{v}$ is a column vector of phase samples $v(n)$. The fine estimate is equal to $\hat{\mathbf{a}}_h = \hat{\hat{\mathbf{a}}}_h + \delta\hat{\hat{\mathbf{a}}}_h$ with elements dependent on the window width $\hat{a}_{i,h}^f$, $h \in H$.

6. The matched correlator (ML criterion function) is used to determine the optimal window width
$$\hat{h} = \arg\max_h J(h), \quad (10)$$
$$J(h) = \left|\sum_n x(n)\exp\left(-j\sum_{i=0}^{M} \hat{a}_{i,h}(n\Delta)^i\right)\right|. \quad (11)$$

7. The optimal window width determine final estimate from the ensemble of estimates
$$\hat{a}_i = \hat{a}_{i,\hat{h}}, \ i \in [0, M]. \quad (12)$$

From the perspective of hardware implementation, the algorithm steps 1 to 5 are the essential ones, and they represent its core. Namely, those steps are repeating for each STFT window width and, therefore, require the same hardware implementation. Based on the simple criterion (11), it is decided which results (for what STFT window width realization) are the most accurate ones and they are chosen as the final estimates, (12). Having in mind the

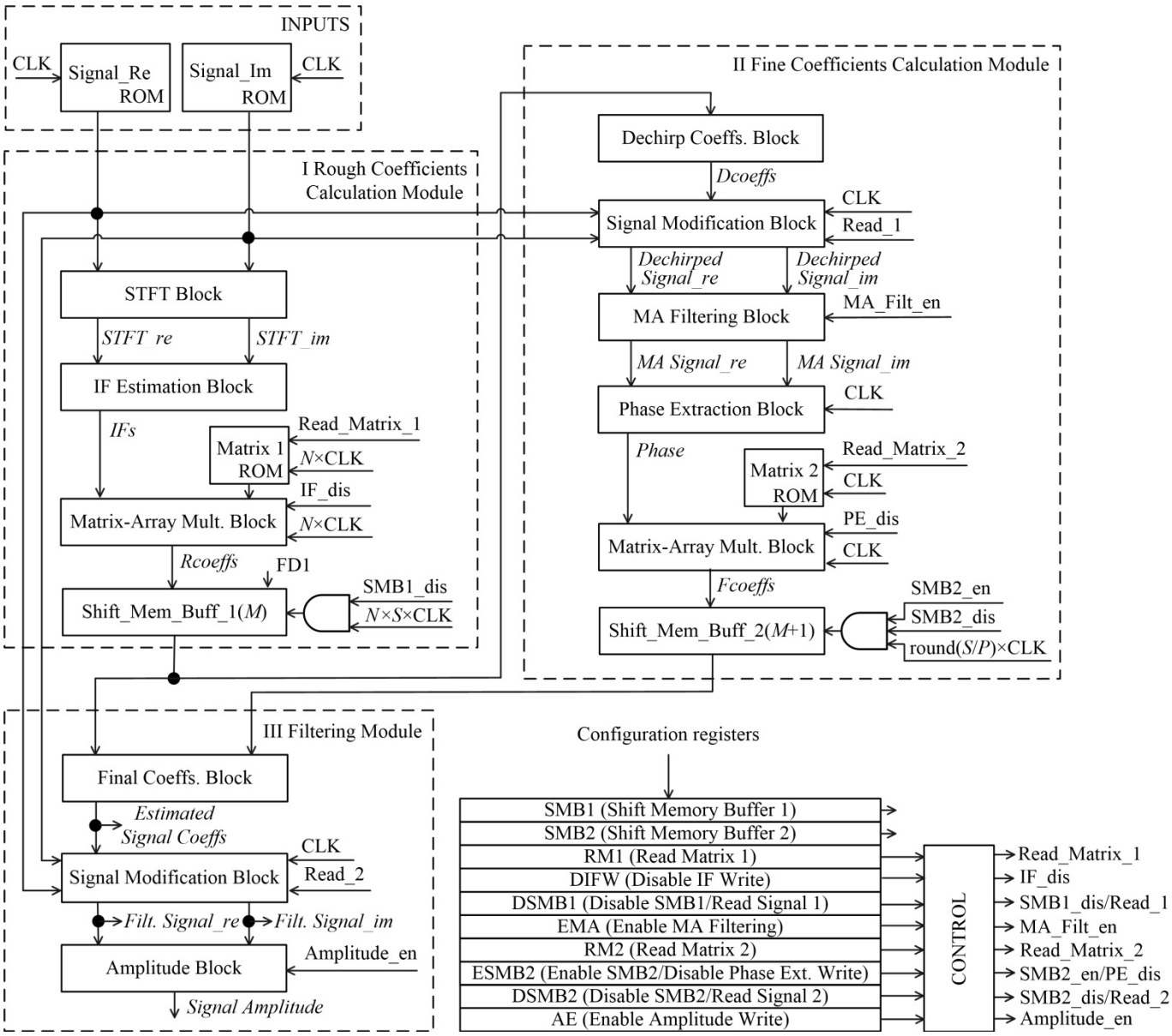

Figure 1. *Hardware design of the QML algorithm-based estimator core for the polynomial phase signals. The stored parameters' names and their description are given into configuration registers. M is the order of the observed signal, N is the width of the STFT lag window, S is the length of the observed signal, and P is the moving average filter length.*

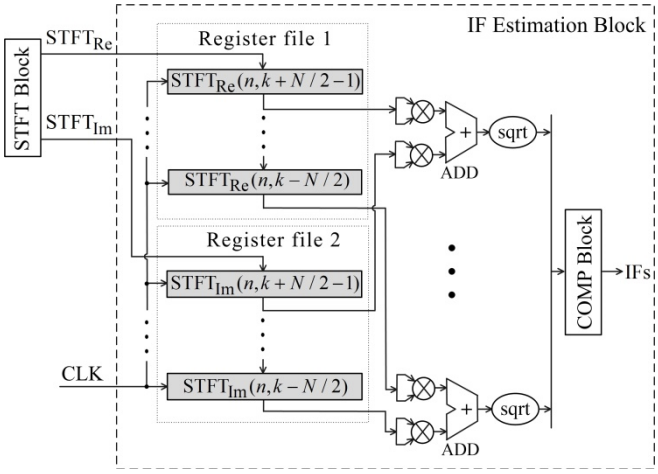

Figure 2. *IF estimation block from Fig. 1. Unit denoted by sqrt performs square root operation.*

above elaboration, a hardware implementation of the QML algorithm core presented in the next section is the main contribution of this paper.

3. Hardware Implementation

Hardware implementation capable to provide the QML algorithm-based estimation of the PPSs is given in Figs. 1-5 and Table 1. It has three main functional modules. In accordance with their associated names, the first two modules provide the calculation of the rough and fine coefficients of the QML algorithm, respectively. The third module creates the final outputs of the implementation based on the input signal and the outputs of the first two modules (rough and fine coefficients). The final outputs are estimated signal coefficients, filtered signal, as well as estimated signal amplitude.

The STFT block from Fig. 1 represents one of the well-known and already available modules that are used to

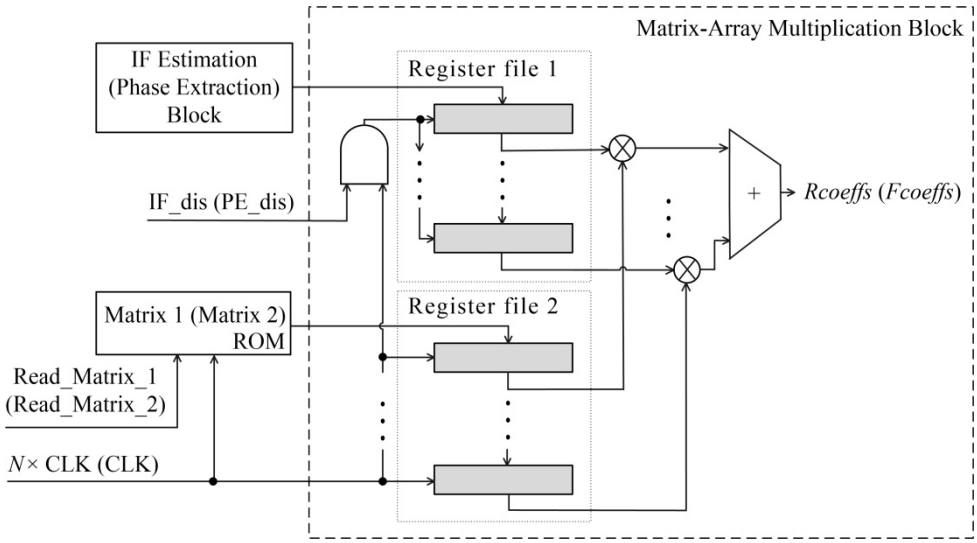

Figure 3. *Matrix-Array Multiplication Block from Fig. 1. Notation given outside the brackets corresponds to the Matrix-Array Multiplication Block from the Rough Coefficients Calculation Module, whereas the notation given inside the brackets corresponds to the Matrix-Array Multiplication Block from the Fine Coefficients Calculation Module.*

Table 1 Parameters from the Configuration registers, Fig. 1, and their corresponding values. Parameters' values are expressed by the number of needed CLK cycles.

Parameters	**Parameters' values**
SMB1	M
SMB2	M+1
RM1	N
DIFW	$N\times S$
DSMB1	$N\times S\times M$
EMA	$N\times S\times M+S$
RM2	$N\times S\times M+S+1$
ESMB2	$N\times S\times M+S+$round(S/P)+1
DSMB2	$N\times S\times M+S+(M+1)$round($S/P$)+1
AE	$N\times S\times M+2S+(M+1)$round($S/P$)+2

provide the TF signal representation based on the linear STFT. These modules are implemented either by using the available FFT chips, [32], [33], or by using approaches based on the recursive algorithms, [34]-[39]. Based on the calculated STFT samples and the peak detection along the frequency direction, [50]-[52], the IF Estimation Block (see eq. (4)), given in Fig. 2, recognizes the instantaneous frequencies (IFs) of the analyzed signal. The COMP Block from the IF Estimation Block compresses set of 2-input comparators combined with the basic logic gates to perform the peak detection between the frequency-only-dependent STFT samples and, accordingly, to determine IF of the input signal in the TF point correspondent (in frequency) to the detected peak STFT sample. Finally, the Matrix-Array multiplication block, given in Fig. 3, generates (at the output and based on the estimated IFs and Matrix 1 elements) the rough coefficients *Rcoeffs*. Note that Matrix 1 represents the matrix used in polynomial interpolation of the IF estimates (5). Its elements, located in ROM, are the outcome of the product $(\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{\Gamma}^T$ (eq. (5)).

The Dechirp Coefficients Block together with Signal Modification Block, Fig. 4, provides a calculation of the dechirped signal, according to (6). The outputs of the Dechirp Coefficients Block, named *Dcoeffs* in Fig. 1, are dechirp coefficients $\sum_{i=1}^{M}\hat{a}_{h,i}(n\Delta)^i$ from (6). Together with the analyzed PPS samples, these coefficients represent inputs of the Signal Modification Block. After calculation in the Signal Modification Block, the dechirped signal passes through the moving average filter (MA Filtering Block), (7). Extraction (and unwrapping) (8) is performed in the Phase Extraction Block, using the filtered, dechirped signal samples from the MA Filtering Block as the inputs. Matrix-Array Multiplication Block, Fig. 3, is used in this functional module as well. Its architecture is the same as in the case of the Rough Coefficients Calculation Block. However, inputs of the Matrix-Array Multiplication Block from the Fine Coefficients Calculation Block are different. This block serves for multiplication of the extracted (and unwrapped) phase with the elements from Matrix 2. Note that Matrix 2 represents the matrix used in polynomial regression, (9), and its elements, located in ROM, are the outcome of the product: $(\mathbf{\Xi}^T\mathbf{\Xi})^{-1}\mathbf{\Xi}^T$ (eq. (9)). In this way, we are calculating the fine coefficients *Fcoeffs*.

Results of two previously described functional modules (*Rcoeffs* from Rough Coefficients Calculation Module and *Fcoeffs* from the Fine Coefficients Calculation Module) are inputs of the Final Coefficients Block, the input block of the third functional module named the Filtering Module. *Estimated Signal Coeffs* are calculated in this block according to $\hat{\mathbf{a}}_h=\hat{\hat{\mathbf{a}}}_h+\delta\hat{\hat{\mathbf{a}}}_h$ (step 5 of the QML algorithm, described in the previous Section). Signal Modification Block, Fig. 4, is used in this functional module as well. Its architecture is the same as in the case of the second functional module, but with analyzed PPS samples and *Estimated Signal Coeffs* as the inputs. The output of the Signal Modification Block is the filtered PPS. The last calculation block in the developed design named the

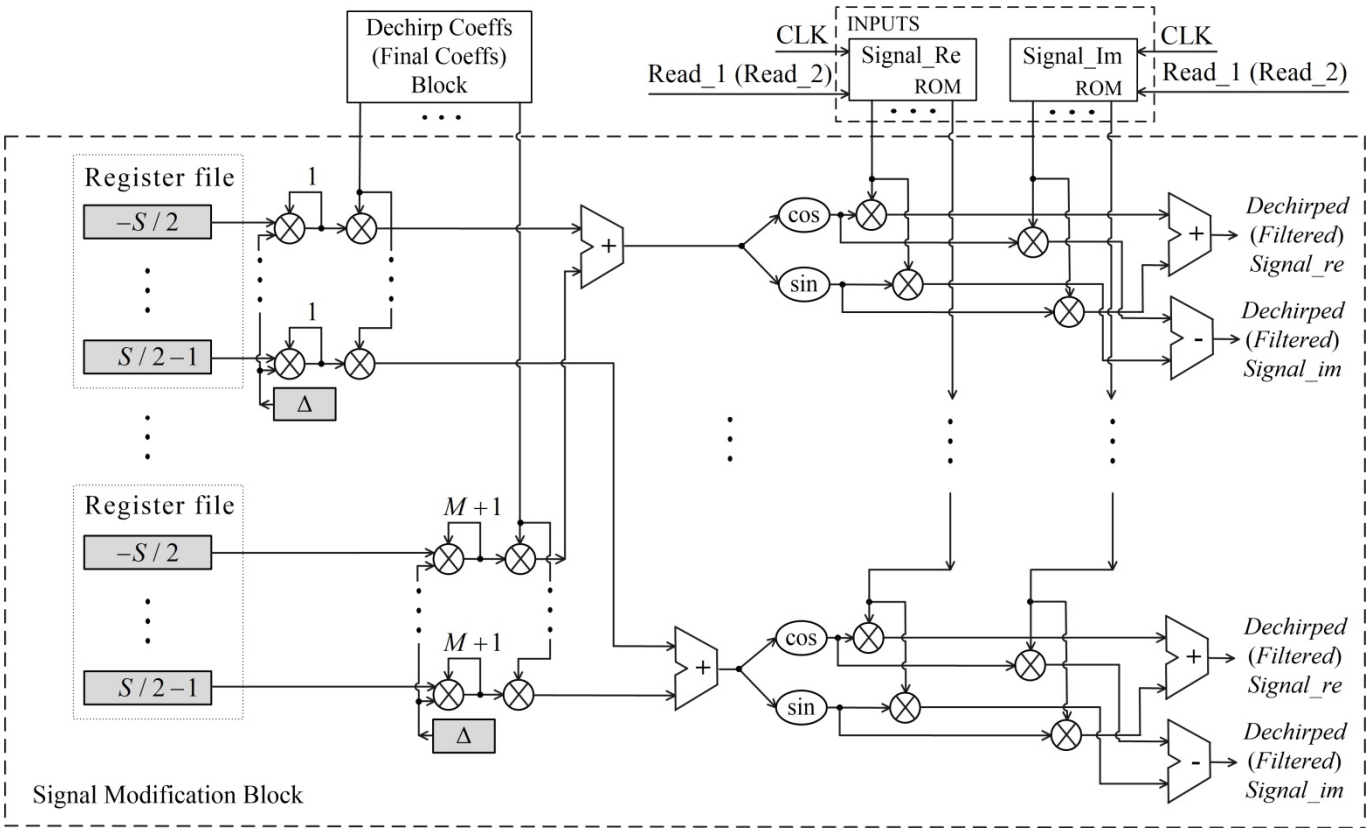

Figure 4. *Signal Modification Block from Fig. 1. Notation given outside the brackets corresponds to the Signal Modification Block from the Fine Coefficients Calculation Module, whereas the notation given inside the brackets corresponds to the Signal Modification Block from the Filtering Module. Value* Δ *is the sampling rate of the analyzed signal. Within the system initialization, the memory contents of register files are automatically loaded from outside* (*by using general purpose microcontroller or PC*)*. Within the execution, these registers can only be read, their contents cannot be changed and, therefore, they should not be managed by control signals. Feedbacks on the multipliers denote execution of the exponentiation functions (number written on the feedback denotes exponent value). Units denoted by sin and cos perform sine and cosine operations, respectively and are implemented as a sum of the first two terms of the corresponding Taylor series.*

Amplitude Block uses filtered PPS samples and calculates the signal amplitude.

Control logic from Fig. 1 manages the execution. It generates control signals based on the parameters from the Configuration registers, Table 1. To this end, the control logic groups modules that consist of the variable length up/down binary counters and the binary magnitude comparators whose references are parameters from Configuration registers. In this way, each of these modules creates the control signal corresponding to the parameter from the Configuration registers.

Within the output signal calculation, the proposed implementation takes multiple and fixed number of clock cycles (CLKs). Calculations of the rough coefficients *Rcoeffs* and the fine coefficients *Fcoeffs* take $N\times S\times M$ CLKs and $S+(M+1)\text{round}(S/P)+1$ CLKs, respectively, where operator round(x) denotes rounding of the real variable x to the nearest integer, whereas the Filtering Module requires $S+1$ CLKs to complete calculation of the output. Note that number of taken CLKs depends on the known algorithm parameters N, M, S, and P, which means that execution time of the proposed hardware implementation can be calculated in advance that can be of great importance in many practical applications.

Real and imaginary parts of input STFT data, numerically calculated in STFT Block, are imported to the proposed implementation on every CLK and are stored in Register file 1 and Register file 2 of the IF Estimation Block, Fig. 2. After N CLKs, these register files contain all frequency-only-dependent STFT samples from the observed time instant. Over the absolute $STFT(n,k), k=-N/2,...,N/2-1$ values, implemented as $|STFT(n,k)|=\sqrt{STFT_{\text{Re}}^2(n,k)+STFT_{\text{Im}}^2(n,k)}$, the COMP Block performs, in the already described manner, the peak detection. According to (4), IF is detected in the TF point correspondent in frequency to the detected peak STFT sample. Hence, the IF Estimation Block takes $N\times S$ CLKs to complete the IFs estimation for the windowed PPS.

For each $N\times$CLK cycle, the estimated IFs are imported to the Register file 1 of the Matrix-Array Multiplication Block, Fig. 3. Simultaneously, on every $N\times$CLK cycle, and controlled by the Read_Matrix_1 signal, the Matrix 1 elements are imported to the Register file 2 of the same block. After $N\times S$ CLKs, all the estimated IFs are imported to the Register file 1, but only one Matrix 1 column is imported into the Register file 2. At that instant,

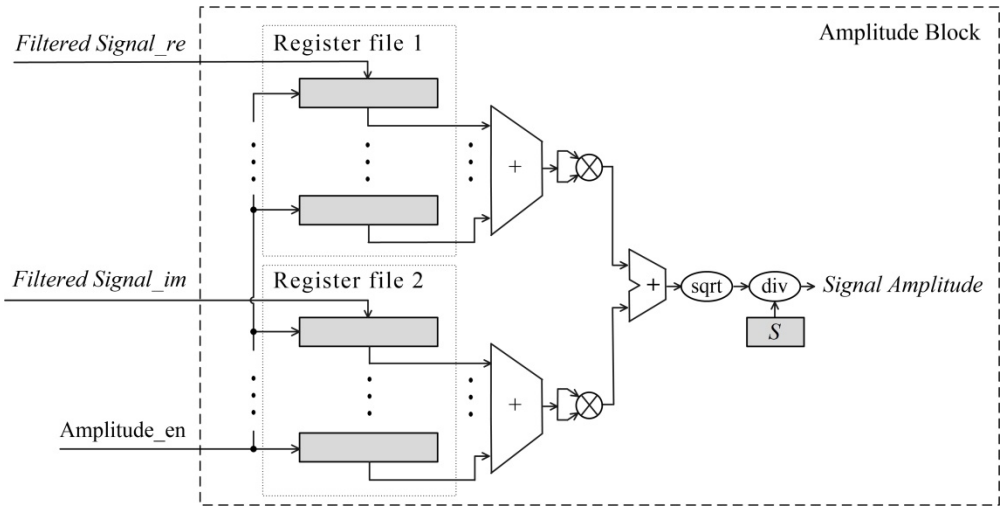

Figure 5. *Amplitude Calculation Block from Fig. 1. Unit denoted by div performs division operation between the sqrt unit output and the length S of the observed signal.*

by the multiplication of the corresponding elements from the Register file 1 and Register file 2, and afterward by the summation of the calculated products, one rough coefficient is produced at the output of the Matrix-Array Multiplication Block. This procedure is repeated M times (one time per a Matrix 1 column). As a result, after $N\times S\times M$ CLKs all rough coefficients *Rcoeffs* are calculated and imported to the Shift_Memory_Buffer_1 block. Note that, controlled by the IF_dis signal, when all estimated IFs are imported to the Register file 1, registers from this file do not change their values. In the same manner, the SMB1_dis signal disables the Shift_Memory_Buffer_1 block to change its content after $N\times S\times M$ CLKs, which allows the design to have the *Rcoeffs* (in the Shift_Memory_Buffer_1 block) until completion of the execution.

Implementing (6) based on the calculated *Rcoeffs*, the Dechirp Coefficients Block (the simple combinational logic containing a set of the two-input dividers) performs the calculation of the dechirp coefficients *Dcoeffs* simultaneously imported to the Signal Modification Block. At the same time, on every CLK cycle and controlled by the Read_1 signal, the estimated signal samples (both their real and imaginary parts) are imported to the Signal Modification Block of the Fine Coefficients Calculation Module. After S CLKs, all signal samples are imported to this block, providing a base for the execution of the necessary operations (exponentiation, multiplication, summation, sine, and cosine). As a result, real and imaginary parts of the dechirped signal (*Dechirped Signal_re* and *Dechirped Signal_im*) are produced at the output.

Controlled by the MA_Filt_en signal, real and imaginary parts of the calculated dechirped signal are separately imported to the MA Filtering Block. To create a base for the moving average filtering (7) with no overlapping windows, the imported samples are grouped in the P length register files. Filtering (7) is implemented by the simple combinational logic containing a set of the P-input adders and two-input dividers. The *MA_Signal_re* and *MA_Signal_im* samples, produced at the output of MA Filtering Block, are imported to the combinational Phase Extraction Block, which performs the arcus tangent and the phase unwrapping function (8). This block requires only one CLK to finish the calculation and to produce the *Phase* signal at its output.

On every CLK, the calculated *Phases* are imported to the Register file 1 of the Matrix-Array Multiplication Block, shown in Fig. 3. Simultaneously, on every CLK, and controlled by the Read_Matrix_2 signal, the Matrix 2 elements are imported to the Register file 2 of the same block. After round(S/P) CLKs, all the calculated *Phases* are imported to the Register file 1, but only one Matrix 2 column is imported into the Register file 2. At that instant, by the multiplication of the corresponding elements from the Register file 1 and Register file 2, and afterward by the summation of the calculated products, one fine coefficient is produced at the output of the Matrix-Array Multiplication Block. This procedure is repeated (M+1) times (one time per a Matrix 2 column). As a result, after round(S/P)(M+1) CLKs all fine coefficients *Fcoeffs* are calculated and imported to the Shift_Memory_Buffer_2 block. Note that, controlled by the PE_dis signal, when all the estimated *Phases* are imported to the Register file 1, registers from this file do not change their values. In the same manner, the SMB2_dis signal disables the Shift_Memory_Buffer_2 block to change its content after ($N\times S\times M+S+(M+1)\times$ round(S/P)+1) CLKs, which allows the design to have the *Fcoeffs* (in the Shift_Memory_Buffer_2 block) until completion of the execution.

The *Estimated Signal Coefficients* are calculated inside the Final Coefficients Block (the simple combinational logic containing a set of the two-input multipliers and a set of the two-input dividers), based on the already calculated rough and fine coefficients *Rcoeffs* and *Fcoeffs*. Simultaneously with the calculation, the *Estimated Signal Coefficients* are imported to the Signal Modification Block. At the same time, on every CLK cycle and controlled by the Read_2 signal, the estimated signal samples (both their real and imaginary parts) are imported to the Signal Modification Block. After S CLKs, all signal samples are imported to this block, providing a base for the execution of

Table 2 Summarized resource utilization of the proposed design implemented on the Cyclone III EP3C16E144C7 FPGA device and determined by N=128, M=6, S=256, P=5, and input data length l=32. *Note that a total number of pins used by the complete design correspond to the sum of input pins of the first module and output pins of the third module. **Note that a total number of memory bits used by the complete design correspond to the sum of a total number of memory bits used by each module separately increased for a total number of memory bits necessary to locate input signal samples, Fig. 1.

Resource utilization	*I Rough Coeffs Calculation Module*	*II Fine Coeffs Calculation Module*	*III Filtering Module*	*In total (complete design)*
Total Logic Elements	2,956/15,408 (19%)	2,106/15,408 (14%)	7,234/15,408 (47%)	12,296/15,408 (80%)
Combinational Functions	2,956/15,408 (19%)	2,106/15,408 (14%)	3,376/15,408 (22%)	8,438/15,408 (55%)
Dedicated Logic Registers	2,467/15,408 (16%)	2,051/15,408 (13%)	4,383/15,408 (28%)	8,901/15,408 (58%)
Total Pins	33/85 (39%)	65/85 (76%)	80/85 (94%)	33/85 (39%)*
Total Virtual Pins	0	0	0	0
Total Memory Bits	26,600/516,096 (5%)	3,856/516,096 (7%)	2,048/516,096 (4%)	48,888/516,096 (9%)**
Embedded Multiplier 9-bit elements	6/112 (5%)	16/112 (14%)	42/112 (38%)	64/112 (57%)
Total PLLs	0/4 (0%)	0/4 (0%)	0/4 (0%)	0/4 (0%)

the necessary operations (exponentiation, multiplication, summation, sine, and cosine). As a result, real and imaginary parts of the filtered signal (*Filtered Signal_re* and *Filtered Signal_im*) are produced at the output.

Controlled by the Amplitude_en signal, real and imaginary parts of the filtered signal are separately imported to the Register file 1 and the Register file 2 of the Amplitude Block, as shown in Fig. 5. Calculation of the *Signal Amplitude* signal (including summation, multiplication, dividing, and square root operation performed inside this block) requires only one CLK (the last CLK taken within the execution).

4. Testing and verification

To verify the developed hardware approach for the QML algorithm-based estimation of PPS, it is first implemented on an FPGA device and is tested, after that, on the PPS (1) of the sixth-order, M=6, within the time interval [–1,1), and with the sampling interval of Δ=1/128. Signal (1) is corrupted by the additive white noise. The implementation is done by considering different input SNR values (from 0 dB to 20 dB in a stepwise of 2 dB). Within the implementation, the Cyclone III EP3C16E144C7 device has been selected in accordance with the optimal resource occupation, Table 2. Besides, as a key implementation detail, the maximum CLK rate of 50 MHz is achieved within the execution. Samples of the observed noisy signal are written in the signed 32-bit fixed-point notation including 8-bit fraction, as well as the numerically calculated and imported STFTs. Within the STFT numerical calculation, the rectangular lag-window width of N=128 is selected. For the moving average filter length of P=5 is chosen.

Results of the real-time implementation for different input SNRs are given in Table 3 (rows named Hardware). In addition, to check the accuracy of the achieved results, numerical results obtained from MATLAB simulation are also presented in Table 3 (rows named Software 1). To provide a fair comparison, numerical results are obtained for the same input noisy PPS and for the same numerically calculated STFTs, as in the hardware implementation case. The proposed hardware implementation provides high quality estimation and only small differences in the parameter estimation can be noticed. Typically, differences are from 0.03% (in the case of parameter a_0 estimation and for SNR_{in}=20dB) up to 2.66% (in the case of parameter a_3 estimation and for SNR_{in}=12dB). The inaccuracy is mainly caused by the approximate realization of some basic functions within the hardware implementation. For example, the trigonometric functions realization is based on Taylor series representation and within hardware implementation, only first two terms of the corresponding Taylor series are taken into account. On the other hand, calculation of these functions in MATLAB uses, by default, 250 terms of the Taylor series. To confirm this statement, default trigonometric functions realizations have been replaced in MATLAB by the corresponding Taylor series representations, but with the same accuracy as in the hardware implementation case (only first two terms of the corresponding Taylor series have been taken into account). Results achieved in this way are also presented in Table 3 (rows named Software 2), for each particular input SNR value. Now, comparing the corresponding Hardware and Software 2 rows from Table 2, the accuracy of the proposed hardware implementation can easily be checked. Note that, in this case, differences in the parameters estimation (from 0.012% in the case of parameter a_5 estimation and for SNR_{in}=20dB up to 0.51% in the case of parameter a_1 estimation and for SNR_{in}=8dB) are caused by the finite register length influence [53] and are significantly smaller than differences noted within the comparison of the proposed hardware implementation and the MATLAB implementation which uses default realizations of the trigonometric functions.

Note that, if it would be required, the accuracy of the proposed hardware implementation can be improved by using more precise mathematical formulas for the mentioned basic functions realization. However, in this way, hardware complexity would be significantly increased, together with the increase in execution time. Therefore, the hardware

Table 3 Results of the observed PPS (1) parameters estimation. Results, achieved by the developed hardware design (Hardware), by the MATLAB simulation (Software 1), and by the MATLAB simulation with reduced accuracy (Software 2), are given for different input SNR values (from 0 dB to 20 dB in a stepwise of 2 dB).

Parameters		A	a_6	a_5	a_4	a_3	a_2	a_1	a_0
SNR_{in}=20dB	Hardware	0.730	18.506	62.829	-57.198	67.611	-82.090	-30.576	65.559
	Software 1	0.734	18.566	62.810	-57.107	67.657	-82.150	-30.274	65.538
	Software 2	0.733	18.530	62.821	-57.158	67.629	-82.019	-30.525	65.540
SNR_{in}=18dB	Hardware	0.737	18.575	63.122	-57.433	67.797	-82.239	-30.524	65.992
	Software 1	0.734	18.562	62.808	-56.978	67.730	-81.912	-30.282	65.469
	Software 2	0.735	18.570	62.997	-57.231	67.750	-82.014	-30.432	65.689
SNR_{in}=16dB	Hardware	0.738	18.516	63.131	-56.540	67.862	-83.950	-30.541	66.089
	Software 1	0.732	18.530	62.817	-56.667	67.595	-83.359	-30.299	66.410
	Software 2	0.735	18.509	63.030	-56.501	67.849	-83.914	-30.432	66.209
SNR_{in}=14dB	Hardware	0.736	18.658	62.995	-57.257	67.869	-84.133	-30.569	66.952
	Software 1	0.731	18.529	62.807	-57.142	67.599	-83.466	-30.357	66.619
	Software 2	0.734	18.598	62.871	-57.193	67.702	-83.890	-30.418	66.876
SNR_{in}=12dB	Hardware	0.734	18.431	62.885	-56.513	65.902	-83.029	-30.666	66.735
	Software 1	0.730	18.528	62.805	-56.614	67.707	-83.720	-30.359	66.830
	Software 2	0.732	18.445	62.877	-56.583	66.003	-83.121	-30.561	66.782
SNR_{in}=10dB	Hardware	0.715	18.954	63.430	-58.765	68.267	-82.339	-29.296	63.608
	Software 1	0.709	18.767	62.865	-57.897	67.726	-81.767	-29.180	63.355
	Software 2	0.712	18.899	63.210	-58.555	67.997	-81.987	-29.201	63.594
SNR_{in}=8dB	Hardware	0.707	18.307	62.821	-58.608	69.253	-81.199	-27.149	63.001
	Software 1	0.701	18.300	62.911	-58.028	69.538	-80.660	-27.652	63.077
	Software 2	0.705	18.302	62.859	-58.493	69.349	-81.018	-27.290	63.022
SNR_{in}=6dB	Hardware	0.797	18.327	64.101	-58.657	68.956	-85.086	33.124	62.748
	Software 1	0.791	18.200	63.846	-58.077	69.653	-85.343	33.258	62.250
	Software 2	0.795	18.250	64.009	-58.463	69.211	-85.219	33.198	62.560
SNR_{in}=4dB	Hardware	0.812	16.400	63.222	-57.179	68.499	-88.089	26.459	58.547
	Software 1	0.817	16.304	63.917	-58.370	69.983	-88.170	26.355	58.490
	Software 2	0.814	16.352	63.359	-57.274	68.509	-88.199	26.397	58.501
SNR_{in}=2dB	Hardware	0.849	16.158	64.867	-58.106	70.463	-88.743	34.373	58.197
	Software 1	0.843	15.999	64.225	-58.516	70.323	-88.566	34.546	57.645
	Software 2	0.846	16.104	64.632	-58.244	70.347	-88.621	34.445	58.003
SNR_{in}=0dB	Hardware	0.620	13.234	64.070	-64.688	71.120	-76.402	-36.469	54.690
	Software 1	0.602	13.531	64.718	-64.985	71.715	-76.195	-36.516	54.861
	Software 2	0.611	13.262	64.285	-64.707	71.150	-76.300	-36.497	54.701

implementation should be developed by making a trade-off between required precision from one side, and hardware complexity and execution time from the other side, as performed here.

To complete estimation of the observed 6-th order PPS, the developed hardware design requires a period of time depending on the number of taken CLKs, discussed for each used module in Section 3, and on the single CLK duration, where the single CLK duration of 20ns corresponds to the maximum CLK rate of 50 MHz. This period of time (about 3.95 ms) is about 10 times smaller than the period required by the MATLAB simulation, performed on a high performance computer (24GB RAM, i7 Intel processor).

5. Conclusion

Efficient multiple-clock-cycle hardware implementation of the QML estimator core, the recently proposed powerful alternative to the state-of-the-art PPS estimators, is developed, tested, and verified. Based on the fact that PPSs have great importance in many practical applications (radars, sonars, sensor networks, biomedicine, and communications), as well as that the QML significantly outperforms other corresponding algorithms (in terms of both characteristics: the SNR and the mean square error), the efficient hardware implementation of such estimation obviously is of great benefit and importance. In that way, this accurate and useful estimation algorithm will get its full practical valorization through possible real-time applications. The developed design is verified by implementation on an FPGA device and is tested on the real, noisy PPS, whereas the achieved very high accuracy is proven by comparison with the results obtained by the MATLAB simulation. It is shown that deviations in the estimation of signal coefficients are mainly caused by the limited precision in the realization of some basic mathematical functions, but also that their precision can be increased if required. A future research could be focused on the parallelization of the necessary number of QML algorithm cores, along with the

implementation of conditional steps of that algorithm. However, due to the expected extensive increase in calculation complexity (that is already high in the realization of the single algorithm core), advanced methods for optimization should be required.

6. Reference

[1] I. Djurović, M. Simeunović, B. Lutovac, "Are genetic algorithms useful for the parameter estimation of FM signals," *Digital Signal Processing*, vol. 22, no. 6, Nov. 2012, pp. 1137-1144.

[2] I. Djurović, M. Simeunović, P. Wang, "Cubic phase function: A simple solution for polynomial phase signal analysis," *Signal Processing*, vol. 135, June 2017, pp. 48-66.

[3] R. McKilliam, B. Quinn, I. Clarkson, B. Moran, B. Vellambi, "Polynomial phase estimation by least squares phase unwrapping," *IEEE Transactions on Signal Processing*, vol. 62, no. 8, August 2014, pp. 1962-1975.

[4] B. Friedlander, J. M. Francos, "Estimation of amplitudes and phase parameters of multicomponent signals," *IEEE Transactions on Signal Processing*, vol. 43, no. 4, Apr. 1995, pp. 917-926.

[5] S. Peleg, B. Porat, "Estimation and classification of polynomial phase signals," *IEEE Transactions on Information Theory*, vol. 37, no. 2, Mar. 1991, pp. 422-430.

[6] S. Barbarossa, V. Petrone, "Analysis of polynomial-phase signals by the integrated generalized ambiguity function," *IEEE Transactions on Signal Processing*, vol. 45, no. 2, Feb. 1997, pp. 316-327.

[7] D. S. Pham, A. M. Zoubir, "Analysis of multicomponent polynomial phase signals," *IEEE Transactions on Signal Processing*, vol. 55, no. 1, Jan. 2007, pp. 56-65.

[8] B. Porat, B. Friedlander, "Asymptotic statistical analysis of the high-order ambiguity function for parameter estimation of polynomial-phase signals," *IEEE Transactions on Information Theory*, vol. 42, no. 3, May 1996, pp. 995-1001.

[9] S. Barbarossa, A. Scaglione, G. B. Giannakis, "Product high-order ambiguity function for multicomponent polynomial phase signal modeling," *IEEE Transactions on Signal Processing*, vol. 46, no. 3, Mar. 1998, pp. 691-708.

[10] P. O'Shea, "A new technique for instantaneous frequency rate estimation," *IEEE Signal Processing Letters*, vol. 9, no. 8, Aug. 2002, pp. 251-252.

[11] T. G. Zhou, Y. Wang, "Exploring lag diversity in the high-order ambiguity function for polynomial phase signals," *IEEE Signal Processing Letters*, vol. 4, no. 8, August 1997, pp. 240-242.

[12] P. O'Shea, "A fast algorithm for estimating the parameters of a quadratic FM signal," *IEEE Trans. Sig. Proc.*, vol. 52, no. 2, Feb. 2004, pp. 385-393.

[13] M. Farquharson, P. O'Shea, "Extending the performance of the cubic phase function algorithm," *IEEE Tran. Sig. Proc.*, vol. 55, no. 10, Oct. 2007, pp. 4767 - 4774.

[14] I. Djurović, M. Simeunović, S. Djukanović, P. Wang, "A hybrid CPF-HAF estimation of polynomial-phase signals: detailed statistical analysis," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, Oct. 2012, pp. 5010-5023.

[15] M. Simeunović, I. Djurović, "CPF-HAF estimator of polynomial-phase signals," *IET Electronics Letters*, vol. 47, no. 17, Aug. 2011, pp. 965-966.

[16] P. O'Shea, "An iterative algorithm for estimating the parameters of polynomial phase signals," in *Proc. of ISSPA*, vol. 2, August 1996, pp. 730-731.

[17] I. Djurović, P. Wang, C. Ioana, "Parameter estimation of 2-D polynomial cubic signals using cubic phase function with genetic algorithms," *Signal Processing*, vol. 90, no. 9, Sep. 2010, pp. 2698-2707.

[18] P. O'Shea, "On refining polynomial phase signal parameter estimates," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 46, no. 3 July 2010, pp. 978-987.

[19] B. Ristić, B. Boashash, "Comments on "The Cramer-Rao lower bounds for signals with constant amplitude and polynomial phase"," *IEEE Transactions on Signal Processing*, vol. 46, no. 6, June 1998, pp. 1708-1709.

[20] F. Gini, G. B. Giannakis, "Hybrid FM-polynomial phase signal modeling: parameter estimation and Cramer-Rao bounds," *IEEE Transactions on Signal Processing*, vol. 47, no. 2, Feb. 1999, pp. 363-377.

[21] P. Wang, P. V. Orlik, K. Sadamoto, W. Tsujita, F. Gini, "Parameter estimation of hybrid sinusoidal FM-polynomial phase signal," *IEEE Signal Processing Letters*, vol. 24, no. 1, Jan. 2017, pp. 66-70.

[22] I. Djurović, LJ. Stanković, "Quasi maximum likelihood estimator of polynomial phase signals," *IET Signal Processing*, vol. 13, no. 4, June 2014, pp. 347-359.

[23] I. Djurović, "On parameters of the QML PPS estimator", *Signal Processing*, vol. 116, Nov. 2015, pp. 1-6.

[24] I. Djurović, LJ. Stanković, "STFT-based estimator of polynomial phase signals",*Signal Processing*, vol. 92, no. 11, Nov. 2012, pp. 2769-2774.

[25] I. Djurović, "Quasi ML algorithm for 2-D PPS estimation," *Multidimensional Signals and Systems*, vol. 28, no. 2, Apr. 2017, pp. 371-389.

[26] I. Djurović, "High precision technique for PPS estimation in impulsive noise environment," *Signal Processing*, vol. 127, Oct. 2016, pp. 151-155.

[27] I. Djurović, "QML-RANSAC: PPS and FM signals estimation in heavy noise environments," *Signal Processing*, vol, 130, Jan. 2017, pp. 142-151.

[28] I. Djurović, M. Simeunović, "Resolving aliasing effect in the QML estimation of PPSs," *IEEE Transaction on Aerospace and Electronic Systems*, vol. 52, no. 3, June 2016, pp. 1494-1499.

[29] I. Djurović, V. Popović-Bugarin, M. Simeunović, "The STFT-based estimator of micro-Doppler parameters," *IEEE Transactions on Aerospace and Electronics Systems*, vol. 53, no. 3, June 2017, pp. 1273-1283.

[30] LJ. Stanković, I. Djurović, S. Stanković, M. Simeunović, M. Daković, "Instantaneous frequency in time-frequency analysis: Enhanced concepts and performance of estimation algorithms," *Digital Signal Processing*, vol. 35, Dec. 2014, pp. 1-13.

[31] J. Lerga, V. Sučić, "Nonlinear IF estimation based on the pseudo WVD adapted using the improved sliding

pairwise ICI rule," *IEEE Signal Processing Letters*, vol. 16, no. 11, Nov. 2009, pp. 953-956.
[32] A. Papoulis, *Signal Analysis*, McGraw-Hill, New York, USA, 1997.
[33] A. Oppenheim, R.W. Schafer, *Discrete-Time Signal Processing*, 3rd edition, Prentice-Hall,2009.
[34] S. Stanković, LJ. Stanković, V.N. Ivanović, R. Stojanović, "An architecture for the VLSI design of systems for time-frequency analysis and time-varying filtering", *Annales des Telecommunications*, vol. 57, no. 9/10, Sept./Oct. 2002, pp. 974-995.
[35] K. J. R. Liu, "Novel parallel architectures for Short-time Fourier transform", *IEEE Trans. on Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 40, no 12, Dec. 1993, pp. 786-790.
[36] M. G. Amin, "A new approach to recursive Fourier transform", Proc. of the IEEE, vol. 75, no. 11, 1987, pp. 1537-1538.
[37] M. Unser, "Recursion in short time signal analysis", *Signal Processing*, vol. 5, no 5, 1983, pp. 229-240.
[38] M. G. Amin, "Spectral smoothing and recursion based on the nonstationarity of the autocorrelation function", *IEEE Trans. on Signal Processing*, vol. 41, no 2, Feb. 1993, pp. 930-934.
[39] M. G. Amin, K .D. Feng, "Short time Fourier transform using cascade filter structures", *IEEE Trans. on Circuits and Systems–II: Analog and Digital Signal Processing*, vol. 42, no 10, Oct. 1995, pp. 631-641.
[40] B. Boashash, *Time frequency signal analysis and processing: a comprehensive reference*, Second edition, Amsterdam: Academic Press, Elsevier, 2016.
[41] E. Sejdić, I. Djurović, J. Jiang, "Time-frequency feature representation using energy concentration: An overview of recent advances," *Digital Signal Processing*, vol. 19, no. 1, Jan. 2009, pp. 153-183.
[42] V. N. Ivanović, R. Stojanović, LJ. Stanković, "Multiple clock cycle architecture for the VLSI design of a system for time-frequency analysis," *EURASIP Journal on Applied Signal Processing, Special Issue on Design Methods for DSP Systems*, vol. 2006, pp. 1-18.
[43] V. N. Ivanović, R. Stojanović, "An efficient hardware design of the flexible 2-D system for space/spatial-frequency signal analysis," *IEEE Trans. on Signal Processing*, vol. 55, no. 6, June 2007, pp. 3116-3125.
[44] V. N. Ivanović, S. Jovanovski, "Signal adaptive system for time–frequency analysis," *Electronics Letters*, vol. 44, no. 21, Oct. 2008, pp. 1279-1280.
[45] V. N. Ivanović, S. Jovanovski, "Signal adaptive system for space/spatial–frequency analysis," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1-16.
[46] S. Jovanovski, V. N. Ivanović, "Signal adaptive pipelined hardware design of time-varying optimal filter for highly nonstationary FM signal estimation," *Journal of Signal Processing Systems,* vol. 62, no. 3, 2011, pp. 287-300.
[47] V. N. Ivanović, S. Jovanovski, N. Radović, "Superior execution time design of optimal (Wiener) time-frequency filter," *Electronics Letters*, vol. 52, no. 17, Aug. 2016, pp. 1440-1442.
[48] V. N. Ivanović, N. Radović, S. Jovanovski, "Real–time design of space/spatial–frequency optimal filter," *Electronics Letters*, vol. 46, no. 25, Dec. 2010, pp. 1696-1697.
[49] V. N. Ivanović, N. Radović, "Signal adaptive hardware implementation of a system for highly nonstationary two-dimensional FM signal estimation," *AEUE – International Journal of Electronics and Communications*, vol. 69, no. 12, Dec. 2015, pp. 1854–1867.
[50] V. N. Ivanović, M. Daković, LJ. Stanković, "Performances of quadratic time-frequency distributions as instantaneous frequency estimators", *IEEE Transactions on Signal Processing*, vol. 51, no. 1, Jan. 2003, pp. 77-89.
[51] LJ. Stanković, M. Daković, V. N. Ivanović, "Performances of spectrogram as an IF estimator", *Electronics Letters*, vol. 37, no. 12, June 2001, pp. 797–799.
[52] V. N. Ivanović, M. Daković, I. Đurović, LJ. Stanković, "Instantaneous frequency estimation by using time-frequency distributions", in *Proc. of IEEE ICASSP 2001*, Salt Lake City, May 2001, pp. 3521-3524.
[53] V.N. Ivanović, LJ. Stanković, D. Petranović, "Finite word-length effects in implementation of algorithms for time-frequency signal analysis", *IEEE Trans. on Signal Processing*, vol.46, no.7, July 1998, pp.2035-2041.