ACOR: on the design of energy-efficient autocorrelation for emerging edge applications

Eleftheriadis, C., & Karakonstantis, G. (2023). ACOR: on the design of energy-efficient autocorrelation for emerging edge applications. In *2023 IEEE/ACM International Conference On Computer Aided Design (ICCAD): proceedings* (IEEE/ACM ICCAD Proceedings). Institute of Electrical and Electronics Engineers Inc.. https://doi.org/10.1109/ICCAD57390.2023.10323733

Published in:
2023 IEEE/ACM International Conference On Computer Aided Design (ICCAD): proceedings

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

ACOR: On the Design of Energy-Efficient Autocorrelation for Emerging Edge Applications

Charalampos Eleftheriadis, Georgios Karakonstantis

Institute of Electronics, Communications and Information Technology (ECIT), Queen's University Belfast.

Email: C.Eleftheriadis@qub.ac.uk, G.Karakonstantis@qub.ac.uk

Abstract—The identification of patterns and changes in time-series using the autocorrelation function (ACF) is traditionally used in several applications from communications, multimedia to remote health monitoring. Existing ACF implementations have tried to meet the throughput requirements of specific domains by mainly using time-domain approaches, however such techniques require several costly multiplications, which hinder their use in power-constrained devices, essential in emerging ACF-based edge applications. Frequency-domain (FD-ACF) approaches could reduce the computational complexity of the ACF calculation, but their use is limited in specific domains, leaving room for further power-aware algorithmic and architectural optimizations. This paper presents a framework, named ACOR, for the design of energy-efficient pipelined ACF architectures under various settings, throughput and energy requirements that vary across ACF-based applications. The proposed framework allows the quick exploration of ACF architectures for different sampling window sizes, window overlapping ratios, number of lags, and precision levels, which is impossible with the existing scattered domain-specific works. Our experimental results show that when compared with existing ACF architectures used in bio-signal analysis, linear predictive coding and telecommunications our proposed framework achieves up to 27.18%, and 51.47% reduction in the circuit area and energy consumption, respectively, with a slight throughput reduction of 8%.

Index Terms—Fast autocorrelation calculation (FACF), fast Fourier transform (FFT), pipelined autocorrelation architecture.

I. Introduction

In recent years, various signal processing applications, are increasingly executed on energy-constrained devices. These applications generally require numerous operations, such as multiplications, to analyze and compare signals in real-time. This often involves computationally intensive workloads, including convolutions and correlations. Among them, the autocorrelation function (ACF) represents a special case of correlation in which a single input signal is compared with a time-shifted version of itself.

The ACF is commonly employed in various algorithms, such as autoregressive modeling, power spectral analysis (PSA) , linear predictive coding (LPC), and signal synchronization which are pivotal for commonly used signal processing applications, like bio-signal analysis [1–3], telecommunications [4], lidar systems [5] and audio processing/compression algorithms[6, 7]. While providing key insights, the ACF calculation is considered to be one of the foremost energy-consuming parts of these algorithms. Indeed, most existing approaches [1–7], utilize only a predetermined number of lags p of the ACF out of the N total window samples, as the ACF computational complexity of the re-

quired multiplications is $O(pN)$. Therefore, various pipeline dedicated circuits have been proposed to efficiently perform the required operations while also achieving high processing throughput.

To reduce the energy dissipation of the power-dominant arithmetic operations (i.e. multiplications), several time-domain-based (TD-ACF) architectures have been devised. These architectures, based on either multiply-accumulate (MAC) [1, 2, 5, 6] or systolic architectures [3, 8], perform the required multiplications in a pipelined manner. While the TD-ACF architectures are relatively simple and straightforward, they present high energy consumption due to their quadratic computational complexity $\big(O(pN)\big)$. To this end, frequency-domain (FD-ACF) algorithms [9–11], based on the low-complexity fast Fourier transform (FFT), can be applied to achieve a loglinear computational complexity of $O(N \log_2 N)$. As in many applications, the number of utilized lags p for the ACF calculation is significantly smaller than the sampling window size N, these full-sized FFT-based FD-ACF methods [9–11] not only require more multiplications, but also result in significantly more hardware units compared with the TD-ACF methods. This is attributed to the complex multiplications (rotations) of the FFT algortihm [12]. Finally in many applications in order to extract further information [13] of a given input signal, an overlapping ratio R resulting in $Q = \frac{1}{1-R}$ distinct input window channels is employed, resulting to further energy/hardware overhead.

Given the vast number of application domains that utilize the ACF and the scattered, non-optimized architectures that the existing works have employed, it is currently impossible for designers to identify the best choices that meet their requirements/specifications. Therefore, this paper introduces a novel FD-ACF architecture generation framework named ACOR, that based on the specifications of an application can provide an energy-efficient ACF architecture for a given size p, N, window overlapping ratio (R), input processing throughput T and precision level. The contributions of this paper can be summarized as follows:

1) Reformulate the conventional full FFT-based FD-ACF achitectures based on a revised fast autocorrelation algorithm [14] that requires smaller p sized FFTs instead of full N sized FFTs. Therefore, a computational complexity of $O(N \log_2 p)$ is achieved, resulting in a drastic reduction of hardware units.

2) Adaptively generate optimized ACF architectures for a given input processing throughput T and window-

overlapping ratio R. This allows us the trade-off throughput for area and power consumption while facilitating the calculation of multiple overlapping windows. Our hardware savings are even higher if a multi-channel architecture is employed.

3) Redesign the FFT complex multipliers (rotators) by introducing a novel multiplierless technique using only a few adders, multiplexers, and bit-shifters. Unlike the existing ones [15–18], this technique jointly minimizes the employed adders and multiplexers, resulting in significant power improvements compared to generic complex multipliers and other optimizations methods.

4) Apply architecture level optimizations on the proposed ACOR generated ACF architectures based on the input data $x[n]$. If $x[n] \in \mathbb{R}$ then several optimizations in terms of the required number of multiplier and hardware units can be performed due to the underlying conjugate symmetries that arise at the frequency domain [19].

5) Demonstrate the efficacy of our proposed approach by performing an architecture exploration in designing ACF calculation units of various throughputs, overlapping ratios and lag samples that are suitable for specific applications with particular requirements. Our experimental results indicate that when the resultant ACF architectures, are synthesized in $45\,\mathrm{nm}$, up-to 27.18% and 51.47% savings are achieved in terms of the area, and energy consumption, respectively with a throughput penalty of around 8%.

The rest of the paper is organized as follows. Section II discusses the background and challenges of the existing ACF methods. Section III presents the proposed approach and the design steps of ACOR. Section IV presents a complexity analysis of the produced architectures and our experimental results. Section V draws the final conclusions.

II. BACKGROUND AND CHALLENGES

This section introduces some basic background regarding the existing ACF calculation algorithms and their computational complexity in terms of the number of operations. Proceeding that, previously proposed ACF hardware architectures are presented, along with some challenges that remained addressed.

A. Autocorrelation function (ACF) calculation

The ACF for a discrete windowed time-series $x[n]$ can be computed according to:

$$c[n] = \sum_{k=0}^{N-n-1} x[k]x^*[k+n] \mid n \in [0, p-1] \qquad (1)$$

where N is the number of input window samples and p the number of required lags and $x^*[n]$ is the complex conjugate of $x[n]$ (if $x[n] \in \mathbb{R}$ then $x^*[n] = x[n]$). Computing directly, at the time-domain (TD-ACF), the p first coefficients $c[n]$ of (1) results in $pN - \frac{p(p-1)}{2}$ total multiplications which comprise of

---

Algorithm 1 Rader ACF algorithm [14]

1: $X_{old}[k] \leftarrow 0$
2: $C[k] \leftarrow 0$ where $k \in [0, 2p-1]$
3: **for** $i = 0 : N/p - 1$ **do** ▷ Subsequencies
4: $\quad X_{new}[k] \leftarrow \mathcal{F}\{x[ip : (i+1)p-1]\|\vec{0}_p\}$
5: $\quad C[k] \leftarrow C[k] + X_{new}^*[k]\big[X_{new}[k] + (-1)^k X_{old}[k]\big]$
6: $\quad X_{old}[k] \leftarrow X_{new}[k]$
7: **end for**
8: $c[n] \leftarrow (1/N)\mathcal{F}^{-1}\{C[k]\}$

---

the main processing overhead of several applications [1, 2, 5–7].

Due to the $O(pN)$ complexity of the TD-ACF methods an FFT-based algorithm can be employed to reduce the computaional complexity to loglinear order. So, according to the Wiener-Khinchin theorem, (1) can be computed with two FFTs as [9–11]:

$$c[n] = \mathcal{F}^{-1}\left\{\left|\mathcal{F}\{x[n]\|\vec{0}_N\}\right|^2\right\} \qquad (2)$$

where $x[n]\|\vec{0}_N$ is the zero-padded input signal with N additional samples ($\vec{0}_N$) to avoid the frequency-domain aliasing and $\mathcal{F}, \mathcal{F}^{-1}$ are the oversampled $2N$ forward and inverse FFT algorithms, respectively. Finally, only the p first ACF coefficents are utilized. The computational complexity of (2) is $O(N \log_2 N)$ and the total number of operations depend on the implementation of the forward and inverse FFT ($\mathcal{F}, \mathcal{F}^{-1}$) and the squaring operation ($|\cdot|^2$).

While (2) may have reduced the computational complexity of the TD-ACF calculation, it is clear that for small values of p, the direct method (1) outperforms the FFT-based one (2). To this end, Rader in 1970 [14] proposed an improved autocorrelation algorithm which by segementing the original sequence in smaller ones and perfoming smaller sized FFTs and by leveraging the circular shift properties of the FFT achieved a computational complexity of $O(N \log_2 p)$. Particularly, as show in line 4 of Alg. 1 the input sequence $x[n]$ is seperated in N/p parts of size p and which are then zero-padded with p additional samples ($\vec{0}_p$). Following that, the $2p$ sized FFTs of each subsequence is computed in order to compute the ACF at the frequency-domain (line 5). Finally a scaled IFFT of size $2p$ is performed (line 8), while keeping the p first output ACF coefficients. Regarding the computational overhead of Alg. 1, there are $N/p + 1$ FFTs of size $2p$ (lines 4, 8) and $N/p - 1$ complex multiplications (line 5), which account for a combined computational complexity of $O(N \log_2 p)$.

B. Overlapping windows

In many PSA and LPC applications [1, 6] a window function $\phi[n]$ (i.e. hanning, hamming, gaussian, etc.) is utilized to limit the spectral leakage of the side lobes of the conventional rectangular window [13]. Whereas very beneficial, as the values of $\phi[n]$ approach to 0 near the start and the end of the time-window, information of the original signal at these regions is lost. To compensate for this loss overlapping windows are used as shown in Fig. 1. In this case the overlapping ratio is $R = 0.5$ since half of the input data reutilized. As in [20] the
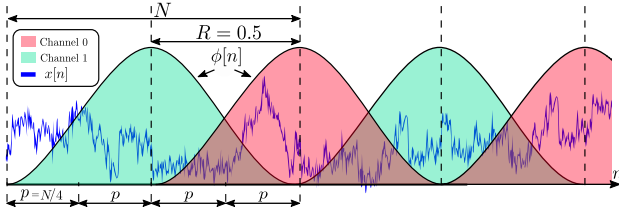
Fig. 1: Multi-channel windowing for $Q = 2$.

Fig. 3: Conventional FD-ACF architecture with full-sized FFTs.

time-windows can be seperated in $Q = 2$ channels (channel 0, channel 1), thus the computational workload is doubled for a single input sample. Furthermore, if $R = 0.75$ then $Q = 4$ channels are need, hence the computational workloaded is quadripled for a single input sample. Hence, for the rest of the paper we distiguish the explored architectures in single-channel (SC) and multi-channel (i.e. MCQ2, MCQ4). Also we define a multi-channel architecture of $T = 1$ if it facilitates the processing of all the Q total channels simultaneously.

C. Autocorrelation function (ACF) architectures

TD-ACF architectures: To efficiently compute the ACF coefficients of (1), in recent works several TD-ACF hardware architectures have been proposed, which can be distinguisted in two types: the multiply-accumulate (MAC) [1, 2, 5, 6] and systolic-based pipelined architectures [3, 8]. As shown Fig. 2, both TD-ACF architectues utilize p generic multipliers, to perform the multiplications of (1). In case of the MAC-based approach (Fig. 2a) the multipliers are connected in parallel, while for the systolic based approach (Fig. 2b) they are serially connected. Furthermore, both these architectures proccess one sample per clock, thus the achieved processing throughput is $T = 1$. Also in both cases, the number of required mutlipliers increases proportionally to the total required lags p in order to maintain $T = 1$ (unrolled case), while utilizing $p/2$ mutlipliers will result in half the area and power and $T = 0.5$ (folded case). Consequently, there is linear relationship between the number of multiplication units, the area and power and an inverse-linear relationship with the achieved processing throughput as also mentioned in [1].

FD-ACF architectures: As discussed in Section II-A existing FD-ACF approaches [9, 10] require the utilization of 2
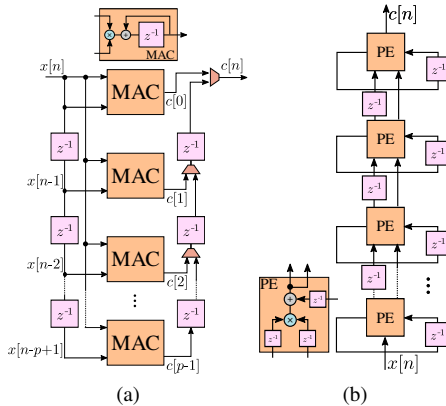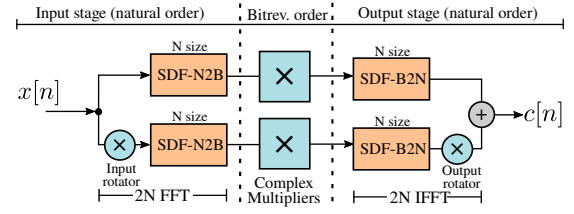
FFTs ($\mathcal{F}, \mathcal{F}^{-1}$) of full size $2N$ and a complex multiplication unit. Note that the IFFT can be implemented with a properly scaled forward FFT as $\mathcal{F}^{-1} = \frac{1}{2N}\mathcal{F}$ [21]. Therefore, in Fig. 3 a pipelined implementation for $T = 1$ is presented (unrolled case). This architecture is comprised of 4 SDF radix-2^2 [12] FFTs and a complex multiplication unit which computes the magnitude of the output of the two forward FFTs ($\mathcal{F}$). Also it is important to note that the input samples $x[n]$ are in natural order and the outputs $X[k]$ are in bit-reverse order (N2B). Thus to compensate for this scambling the two inverse FFTs ($\mathcal{F}^{-1}$) are designed for bit-reverse inputs and natural outputs (B2N), respectively, so the final ACF coefficients $c[n]$ are in correct order. Eventually, such a pipelined FD-ACF architecture of Fig. 3 requires $6\log_2 N - 2$ real multipliers as each N sized SDF archituecture consist of $\log_2 N/2 - 1$ FFT rotators [12] (3 real multipliers per FFT rotator are utilized).

D. Challenges

Although the existing TD-ACF architectures have been adopted in many signal processing applications and the full FFT-based FD-ACF approach seems promising, there are some clear design challenges that can be summarized as follows:

1) The MAC-based and systolic-based achitectures Fig. 2 scatterly proposed in [1–3, 5, 6, 8], are based on the direct TD-ACF computation of (1), presenting high energy consumption due to the the p total multipliers that are required in order to achieve $T = 1$. Futhermore, their computational complexity is $O(pN)$ thus these architectures are ideal for small values of p.

2) The full FFT-based architecture of Fig. 3 has a loglinear computational complexity $O(N\log_2 N)$ but it is outperformed by its TD-ACF counterparts for small values of p. Furthermore the multiplicands of FFT rotators are a-priori know, thus developing optimized multiplierless techniques [17, 18], is crucial for further energy savings.

3) The low-complexity ACF calculation algorithm, proposed in [14] may have reduced the required number of operation ($O(N\log_2 P)$) but it have not efficiently mapped in a hardware architecture that leverages the partitioning of a original sequence $x[n]$ into smaller subsequences of size p. Also the original algorithm targeted non-overlapping windows neglecting the case of multi-channel windowing (overlapping windows). Furthermore, additional microarchitecture optimizations can be achieved if the input signal $x[n] \in \mathbb{R}$.

4) All the previously proposed architectures where applied on a specific case study. However, exploring the options and identifying the best of the particular domain require-

Fig. 2: Existing TD-ACF architectures. (a) MAC based TD-ACF architecture. (b) Systolic based TD-ACF architecture.

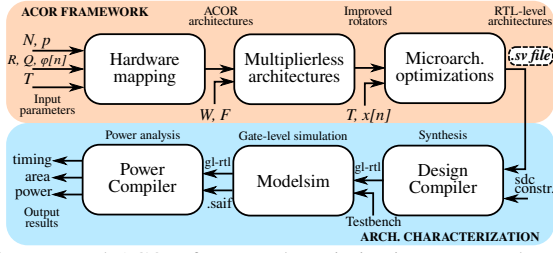Fig. 4: Proposed ACOR framework optimization steps and architecture characterization.

ments such as: the size N, p, the overlapping ratio R (or channels Q) and the processing throughput T is challenging and is not yet done in literature. This is especially needed now as several autocorrelation based applications are quickly growing and need to be integrated in portable devices with high energy efficiency.

III. PROPOSED APPROACH

To address the challenges discussed in the preceding section, the ACOR design framework is introduced. This framework enables the generation of optimized energy-efficient FD-ACF architectures by taking into account the required input specifications, including N, p, R, Q, T, and accuracy level (W: word length, F: fractional part) as shown in Fig. 4. The ACOR framework comprises several optimization steps, beginning with algorithmic optimizations derived from [14]. Following that, the employed multiplierless architectures are introduced for the design energy efficient FFT rotator. Eventually, based on the required throughput, windowing scheme and input signal $x[n]$ the various architecture optimizations are presented.

A. Algorithmic reformulation & Hardware mapping

Single-channel rectangular windowing: As previously mentioned in Section II-A, Alg. 1 was designed to achieve a computational complexity of $O(N \log_2 p)$. Furthermore, this algorithm have been conceived [14] for a single-path non-overlapping rectangular window. Thus, the root architecture (SC-unrolled) of the proposed ACOR framework is based on [14], for $T = 1$ is illustrated in Fig. 5a.

The proposed SC-unrolled architecture can be divided into three stages, the input, intermediate and output stages. The first and last stages are similar to the ones used in the conventional

---

Algorithm 2 Proposed multi-channel ACF algorithm

1: **Inputs:** R, N, p, T_{win}
2: $Q \leftarrow \frac{1}{1-R}$
3: **for** $q = 0 : Q - 1$ **do** ▷ Channels
4:     **for** $t = 0 : T_{win} - 1$ **do** ▷ Windows/channel
5:         $\hat{x}[n] \leftarrow \phi[n] \cdot x(\frac{qN}{Q} + tN : \frac{qN}{Q} + (t+1)N - 1)$
6:         $X_{old}[k] \leftarrow 0$
7:         $C[k] \leftarrow 0$ where $k \in [0, 2p-1]$
8:         **for** $i = 0 : N/p - 1$ **do** ▷ Subsequencies
9:             $X_{new}[k] \leftarrow \mathcal{F}\{\hat{x}[ip : (i+1)p - 1] \| \vec{0}_p\}$
10:            $C[k] \leftarrow C[k] + X_{new}^*[k]\big[X_{new}[k] + (-1)^k X_{old}[k]\big]$
11:            $X_{old}[k] \leftarrow X_{new}[k]$
12:         **end for**
13:         $c[q, t, n] \leftarrow (1/N)\mathcal{F}^{-1}\{C[k]\}$
14:     **end for**
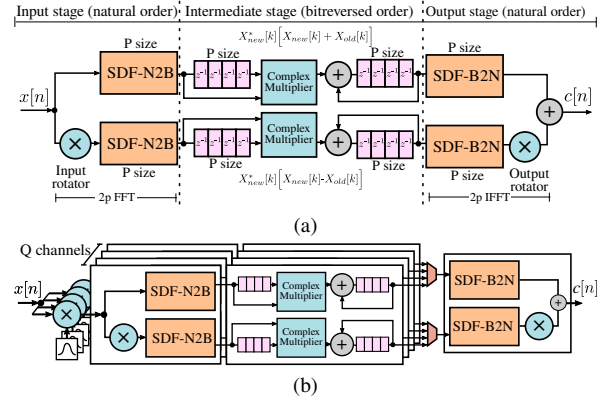15: **end for**

(a)

(b)

Fig. 5: (a) Root SC-unrolled ACOR architecture. (b) MCQ-unrolled ACOR architecture.

full FFT-based FD-ACF approach shown in Fig. 3, with the only difference that p-sized FFTs are used instead of N-sized ones. The input FFTs are used to computed the $2p$-sized forward FFT, stated in line 4 of Alg. 1, while its output counterpart is utilized of the final $2p$-sized IFFT, shown in line 8. Additionally, at the intermediate stage, a p-sized shift-register is placed after the outputs of the forward FFTs to save the values X_{old}. As the values X_{new} and X_{old} are in bitreversed order the top complex multiplier computes $X_{new}^*[k]\big[X_{new}[k] + X_{old}[k]\big]$, whereas the bottom one computes $X_{new}^*[k]\big[X_{new}[k] - X_{old}[k]\big]$. After that, a p-sized circular shift register and an adder are employed to compute the summation of line 5. Finally the outputs of the IFFTs are combined and the ACF coefficients $c[n]$ are computed in natural order.

Multi-channel windowing: In order to generalize and expand the SC-unrolled architecture (Fig. 5a) to the multi-channel case, as discussed in Section II-B a modified ACF algorithm is proposed in Alg. 2. The total number of channels is computed as $Q = \frac{1}{1-R}$, where R is the overlapping ratio, and T_{win} are the total time-windows per channel. So for each channel q (line 3) and time-window per channel t (line 4), a N-sized windowed sequence $\hat{x}[n]$ is constructed (line 5). Then, a single channel ACF calculation is performed (lines 6-12), and the final ACF coefficients $c[q, t, n]$ of channel q and time-window t are computed. As previously, mentioned we define $T = 1$ for processing one input sample $x[n]$ per clock cycle for all Q channels.

The proposed multi-channel architecture (MCQ-unrolled), for $T = 1$, is presented in Fig. 5b. It consist of four stages: windowing stage, input stage, intemediate stage and output stage. As Q total channels are needed, Q windowing multipliers (windowing stage) are employed along with an equal amount of input and intermediate stages, described in the SC-unrolled architecture. Regarding the output stage as shown in line 13 of Alg. 2, the IFFT is active once the summation is $C[k]$ (line 10) is computed for all the $\frac{N}{p}$ subsequences of $\hat{x}[n]$ (line 8). Furthermore, if $Q \leq \frac{N}{p}$ then the active time-periods of the output stage of each channel are distict and do not coincide, thus a single output stage can be shared for all channels. For instance if $N = 128$, $p = 16$ then the output stage can be shared for up-to $Q = \frac{N}{p} = 8$ channels.

B. Multiplierless architectures

The energy-dissipation of existing TD-ACF architectures [1, 2, 5–7], is mainly attributed to the p generic multipliers (Fig. 2). However, the multiplications of the FD-ACF methods (Fig. 3, 5) can be further optimized. Specifically, the twiddle factors of the SDF-FFT rotators lay on the unit circle and depending on the SDF stage they are a-priori known. Thus there have been many approaches [17, 18] aiming to achieve energy savings by designing simpler rotators.

Rotator design: The rotator is a hardware unit widely employed in pipelined FFT architectures [12, 17]. Essentially, a FFT rotator performs, during a butterfly operation, a complex multiplication on a complex input $z = x + jy$ with a set of constants $c_i + js_i | i \in [0, ..., N - 1]$ as stated in (3).

$$z(c_i+js_i) = (x+jy)(c_i+js_i) = x(c_i+js_i)+y(jc_i-s_i) \quad (3)$$

Regarding the ACOR framework, the multiplications $x(c_i + js_i)$ and $y(jc_i - s_i)$ are conducted by designing single input dual output (SIDO) multipliers. Thus the real part of the output is computed as $c_i x - s_i y$ and the imaginary part as $s_i x + c_i y$, as shown in Fig. 6. Also due to the underlining symmetry over the unit circle, additional input/output multiplexers, that interchange the real and imaginary parts, can be employed to perform rotations over all four quadrants, with minimal hardware overhead.

Designing SIDO multipliers has been extensively studied in previous works [17, 18] and various techiques have been employed, with the most promising being the use of the time-multiplexed direct acyclic graph (TM-DAG) method [15, 16, 22]. A TM-DAG is network of interconnected adders and multiplexers, that can produce different intermediate and output multiplication products based on the select signals of the multiplexers. Note that the multiplication/division by a multple of 2 corresponds to a no-cost left/right bit-shift. For instance in Fig. 7c a SIDO multiplier employed in a W_{16} rotator is presented. It is designed to multiply an input signal with the complex twiddle factors $8027, 7416 + j3072$ and $5676+j5676$ which are unity scaled by a factor of 8027.05677 [17] (for unity scale the first real twiddle factor must a multiple of 2).

SIDO multiplier design: In order to design energy-efficient TM-DAGs, both the number adders and multiplexers and must be minimized. Existing SIDO design methods have only focused on the total adders [17, 18] neglecting the multiplexers, leading to suboptimimal architecures. While in
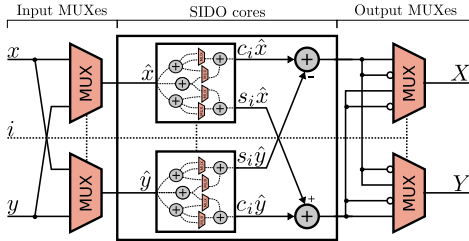
[15, 16] the authors attempted to jointly minimize the multiplexers alongside the adders for the single input single output (SISO) case, the proposed optimizations produced *normalized* TM-DAGs by exploring graphs with only odd intermediate values. Furthermore the applied optimizations depended on random DAG permutation-based algorithms and multiplexed minimization heuristics that are applied at the last stages of the TM-DAG generation algorithm, leading to an excessive number of multiplexers.

To this end a new optimization method have been developed and integrated in the ACOR framework to produce enegry-efficient SIDO multipliers with the least number of multiplexers and adders given a twiddle factor set. Thus, the proposed optimization based on the MAG2 algorithm [22] initialy produces the all the *normalized* DAG representations for twiddle factor sets proposed in [17]. After that we relax the condition of *normalized* DAGs which imposes that one node input should not be bit-shifted [15], by allowing more admissable bit-shifts in order to increase the probability of common edge sharing, during the DAG merging stage. This is achieved by applying an edge weight transform (EWT) on a *normalized* DAG which basically reallocates a bit-shift for the output edges of a node, to both inputs (or vice versa) as shown in Fig. 7a, 7b for 8027. Consequently, for each *normalized* DAG, that correspond to a coefficient, several *non-normalized* ones are derived which result in high quality TM-DAGs, in terms of the utilized mutliplexers.

Following the graph denormalization, the time-multiplexing procedure is conducted. Initially, we perform a node matching followed by a common edge merging check, similarly to the multiplierless method, i.e. DAG FUSION [15]. Furthermore, for our approach instead of using a random permutation based merging heuristic (as in DAG FUSION), we have implemented a depth first search (DFS) combined with a threshold metric, which is based on the number of 2-1 multiplexers. As shown in Fig. 8 all the possible graph merging can be depicted with a N level connected super-graph, where G_{S_i} are all the DAGs of the coefficient $c_i | i \in 0, 1, ..., N - 1$ ($\cup$ stands for the time-multiplexing operation). Initially we perform a DFS from level
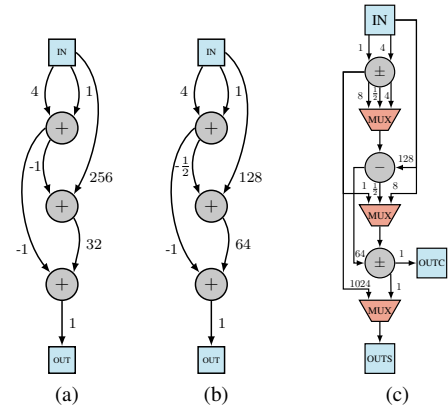

Fig. 7: (a) *Normalized* DAG of 8027 (b) *Non-normalized* DAG of 8027. (c) Proposed SIDO TM-DAG for 8027, 7416 + j3072 and 5676+j5676 with 3 less multiplexers compared to [17]

Fig. 6: Microarchitecture of a FFT rotator with two SIDO multiplierless cores and I/O multiplexers.
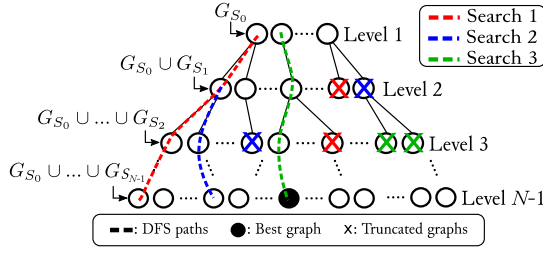
Fig. 8: DFS based search DAG merging algorithm.

1 to level $N-1$ highlighted as search 1 with the red path in Fig. 8. After that an estimation of the threshold metric is evaluated (in our case the total number of required multiplexers) for the candidate $G_{S_0} \cup ... \cup G_{S_{N-1}}$ merging. Thus, during the next searches we can truncate all the graph merges (note as colored $\times$) at $level < N-1$ that violate the initial estimation of the threshold metric and we continue with the second DFS in order to find a better metric estimation (e.g. a solution that utilizes less multiplexers). So in general an exhaustive search is guaranteed if the condition of (4) is met:

$$
\begin{aligned}
&\text{If } best\_est = METRIC(\textstyle\bigcup_{i=0}^{N-1} G_{S_i}) \\
&\text{Discard all mergings such that:} \\
&METRIC(\textstyle\bigcup_{i=0}^{k} G_{S_i}) \leq best\_est \\
&\text{where } k < N-1 \text{ is the current search level}
\end{aligned}
\tag{4}
$$

If the admissible TM-DAGs meet the required the number of multiplexers $\#MUXes$, the RTL of the TM-DAGs is produced, written in SystemVerilog. We applied this TM-DAG generation algorithm for the design of SIDO multipliers used several FFT rotators proposed in [17, 18] for various precision levels and we incorporated them in the pipelined SDF-FFT architectures (Fig. 5) used in the ACOR framework.

C. Architecture optimizations

As previously discussed, depending on the input data (complex or real), the input processing throughput and windowing requirements the ACOR framework can adaptively construct an ACF architecture based on the SC-unrolled root architecture of Fig. 5a. Therefore in this section we elaborate on the different optimizations that are enabled by the ACOR framework, suitable for various applications with different specifications.

Reconfigurable throughput: The SC-unrolled root architecture depicted in Fig. 5a consists of two parallel datapaths (top and bottom) that operate in tandem to achieve a throughput $T = 1$. However, for applications with a lower processing rate, such as single lead ECGs [1], a datapath (DP) folded architecture is preferable to minimize power consumption. In this regard, in Fig. 9a, a modified version of the SC-unrolled architecture is illustrated (SC-DP-folded), that can achieve a throughput of $T = 0.5$ by utilizing a single datapath along with two p-sized register files inserted at the input and output of the architecture. So this design reduces the number of hardware units by nearly half and results in lower power consumption.

In the case of a multi-channel ACF (Fig. 5b), it is possible to adjust the throughput by effectively reusing the input and intermediate stages. In Fig. 9b, a $Q = 2$ channel architecture
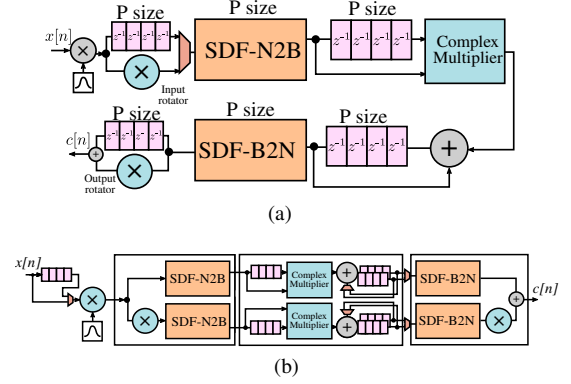

Fig. 9: (a) SC-DP-folded ACOR architecture. (b) MCQ2-channel-folded ACOR architecture.

$(R = 0.5)$ is presented (MCQ2-channel-folded), achieving $T = 0.5$. To accomplish this, three p-sized shift registers are introduced: one at the input of the architecture to perform the initial multiplication with the window function and its delayed version, and two at the end of the intermediate stage to store the required $C[k]$ values for both channels. This approach can be extended to any number of channels Q by folding the architecture in Fig. 5b λ times to achieve a throughput of $T = 1/\lambda$, resulting in approximately λ times less hardware and power consumption. Additionally, the throughput of the architecture in Fig. 9b can be further reduced by a factor of two $(T = 0.25)$ by folding the two parallel datapaths as in Fig. 5a (MCQ2-fully-folded).

Microarchitecture optimizations: To achieve further energy savings, several microarchitecture optimizations can be applied to reduce the number of needed hardware units. In this regard, we propose three additional (Fig. 10) real-valued optimizations $(x[n] \in \mathbb{R})$, and one final optimization that can be applied when the output stage of an ACOR ACF architecture is 50% active. Indeed, if $x[n] \in \mathbb{R}$, the half outputs of the initial two SDF-B2N FFT are needed due to the conjugate symmetry, thus only one complex multiplier is needed to perform half of the multiplications for each datapath per clock cycle interchangeably. Additionally, the register files of the intermediate stage, which consist of p total complex memory cells, can be reduced by a factor of two due to the underlying conjugate symmetry by replacing them with a modified dual-
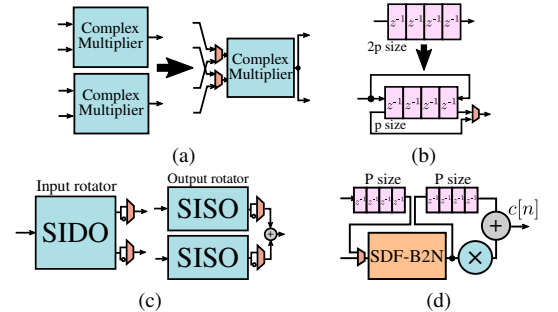

Fig. 10: (a) Complex multiplier optimization (b) Dual-port memory optimization (c) I/O rotators optimization (d) Output stage FFTs optimization.

TABLE I: Hardware complexity of windowed: (a) single-channel ACF architectures. (b) multi-channel ACF architectures.

(a)

Architecture	Multipliers	Adders	Memory	Throughput (T)
TD-ACF MAC [5–7]	$p+1$	p	$3p-2$	1
TD-ACF Systolic [3, 8]	$p+1$	p	$5p-1$	1
FD-ACF FFT-based [9, 10]	$2\log_2 N$-2 rot. +7 mul.	$16\log_2 N + 12$	$8N-8$	1
ACOR SC-unrolled	$\frac{3}{2}\log_2 p$-1 rot. +4 mul.	$12\log_2 p + 11$	$10p-8$	1
ACOR SC-DP-folded	$\log_2 p$ rot. +4 mul.	$8\log_2 p + 8$	$8p-8$	0.5

(b)

Architecture	Multipliers	Adders	Memory	Throughput (T)
TD-ACF MAC [5–7]	$Q(p+1)$	Qp	$Q(3p-2)$	1
TD-ACF Systolic [3, 8]	$Q(p+1)$	Qp	$Q(5p-1)$	1
FD-ACF FFT-based [9, 10]	$Q(2\log_2 N$-2) rot. +7Q mul.	$Q(16\log_2 N + 12)$	$Q(8N-8)$	1
ACOR MCQ-unrolled	$2Q\log_2 p$-1 rot. +4Q mul.	$\approx (10Q+2)\log_2 p + 9Q$	$\approx 8pQ$	1
ACOR MCQ-channel folded	$\frac{Q}{\lambda}\log_2 p$-1 rot. +4$\frac{Q}{\lambda}$ mul.	$\approx (10\frac{Q}{\lambda}+2)\log_2 p + 9\frac{Q}{\lambda}$	$\approx 6\frac{pQ}{\lambda}$	$1/\lambda$
ACOR MCQ-fully folded	$\log_2 p1$ rot. +4 mul.	$8\log_2 p$ rot. +8	$\approx 8pQ$	$1/2Q$

TABLE II: Individual component hardware resources.

Component p=16 W=32, F=12 @1.1V, 45nm	Area (μm^2)	Power (mW)	Max delay (ns)
Real multiplier	7522	2.69	6.53
Complex multiplier	24631	8.9	7.11
DC rotator W_{16}	19289	4.55	8.10
DC input rotator W_{32}	11665	5.45	7.32
DC output rotator W_{32}	12526	5.31	6.92
Proposed rotator W_{16}	7797	3.24	7.14
Proposed input rotator W_{32}	6309	4.61	6.31
Proposed output rotator W_{32}	10463	3.8	6.93
Shift register	4075	0.16	-

port memory buffer similar to that used in [19]. Moreover, the input and output rotators in Fig. 5a can be constructed by a single SIDO and two SISO architectures as $x[n], c[n] \in \mathbb{R}$. Finally, if the output stage is underutilized, a single SDF-B2N FFT can be employed, combined with two p-sized memories.

IV. RESULTS

To demonstrate the efficacy of the proposed ACOR framework we perfomed a hardware complexity analysis for several ACF architectures. After that we implemented some of the generated architectures and individual components in $45\,nm$ technology using Design Compiler, and compared them with the existing TD-ACF architectures in terms of area, energy-consumption expressed as power-delay-product (PDP) and throughput for various cases studies, as shown in Fig. 4.

A. Complexity Analysis

Table I presents a comparison of the hardware complexity of the existing TD-ACF [3, 5–8] and FD-ACF FFT-based architectures [9–11] with those generated by the proposed ACOR framework for the windowed single and multi-channel cases. Notably, these estimations are based on the assumption that $x[n] \in \mathbb{R}$ thus the real-valued microarchitecture optimizations presented in Section III-C were applied. Additionally, for the single-channel case, the output stage optimization was also applied, i.e., one output SDF-B2N FFT. The SDF-FFT radix 2^2 architecture [12] was utilized for the FFT implementations. Moreover, it is important to distinguish the FFT rotators from the generic multipliers since the ACOR framework utilizes multiplierless architectures for their implementation.

Therefore, in Table Ia, we observe that the proposed ACOR architectures utilize $\frac{3}{2}\log_2 p$-1 and $\log_2 p$ rotators plus 4 generic multipliers for the SC-unrolled and SC-DP-folded cases, respectively, while the full FFT-based FD-ACF approach requires $2\log_2 N$-2 rotators and 7 generic multipliers, and the TD-ACF approaches require $p + 1$ multipliers. It is evident that the ACOR architectures exhibit superior scaling ($O(\log_2 p)$) with respect to the number of multipliers, compared to the other approaches. For instance, for $N = 256$ and $p = 32$, assuming a rotator is constructed with 3 generic multipliers, the SC-unrolled ($T = 1$) and SC-DP-folded ($T = 0.5$)

ACOR architectures achieve savings of 37.5% and 73.68% compare with the MAC/systolic approaches ($T = 1$). The ACOR architectures can achieve additional savings (discussed in Section IV-B) by utilizing multiplierless rotators. For the multi-channel case (Table Ib), all the existing approaches require Q parallel single-channel architectures to achieve $T = 1$, while the MCQ-unrolled ACOR architecture requires Q parallel input and intermediate stages and a single output stage as shown in Fig. 5b, resulting in even further savings in terms of the number of multipliers compared to the single-channel case. Finally, the MCQ-channel folded architecture reduces the required hardware units and processing throuput by a factor of λ, while for the MCQ-fully folded architecture a single datapath is used as the the SC-DP-folded architecture and the achieved througput is $1/2Q$.

In terms of adders and memory cells, it is observed that for small values of p, the ACOR architectures are surpassed by the simpler TD-ACF ones. However, in the multi-channel case, the overhead of adders and memory cells is lower compared to the single-channel case. It should be noted that the savings and optimizations achieved in multipliers, as discussed in the following section, compensate for the overhead in adders and memory. This leads to lower power consumption for a given throughput and, in some cases, lower area as well.

B. Measurements

To evaluate the achieve area, energy/power consumption and processing throughput of all the proposed circuits, we synthesized several ACF architectures and individual hardware components targeting maximum clock frequency.

Component-level: In Table II, a comparison is made between the hardware components utilized in the TD-ACF architectures [3, 5–8] and the proposed ACOR architectures in terms of area, power, and maximum delay for $p = 16$. Note that for the implementation of all the rotators, multiplierless architecture have been generated with our optimized framework (Section III-B) for a W=32 bit datapth, F=12 fractional bits and they were compared with the same coefficient rotators produced by Design Complier optimizations. Particularly, the complex multiplier is found to require approximately 3.2 times more area and power compared to the real multiplier, while having slightly more maximum delay (8.9%). The W_{16} rotator (Fig. 6) exhibits a 68.36% reduction in area and requires 16.98% less power compared to the complex multiplier, while also having a shorter maximum delay by 8.52%. Similarly, the W_{32} input rotator (Fig. 10c left) displays a 74.38% reduction in area and requires 41.47% less power compared to the complex multiplier, albeit having a slightly longer maximum delay by 3.49%. The W_{32} output rotator (Fig. 10c right)
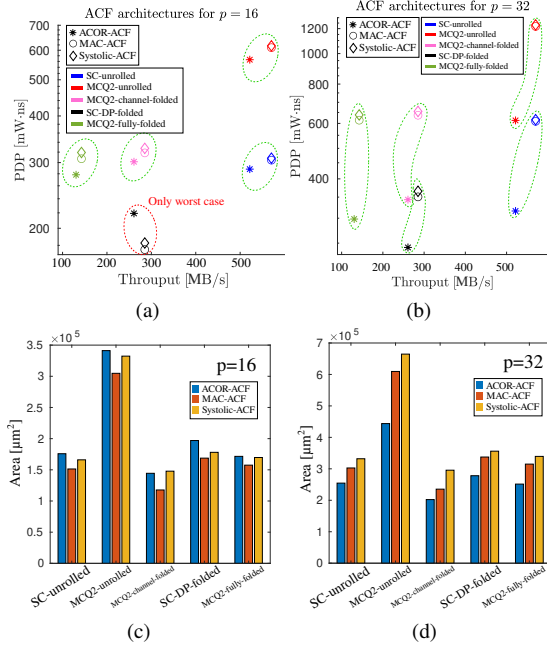
Fig. 11: (a)-(b) PDP vs Throughput comparison of existing TD-ACF and ACOR-ACF architectures. (c)-(d) Area comparison of existing TD-ACF and ACOR-ACF architectures.

exhibits a 57.61% reduction in area and requires 29.21% less power compared to the complex multiplier, while also having a slightly shorter maximum delay by 5.77%. Comparing our proposed mutliplierless rotators with the ones produced by Design Compiler optimizations, it is evident that we have achieve better area, power and max delay by up-to 59.57%, 40.43% and 16% respectively. Furthermore, the power consumption of a $p = 16$ sized shift register is only $0.16\,\mathrm{mW}$, which is significantly smaller than that of multiplier units. The observed slight increase in the maximum delay of the proposed W_{16} rotator can be attributed to the longer critical path of the three adders and multiplexers connected in series, compared to a generic multiplier used by Design Compiler.

Architecture-level: In Fig. 11a-b, a comparison between the achieved energy efficiency, expressed as PDP, and the input processing throughput is presented for various ACF architectures with p=16 and p=32 for a W=32 and F=12 datapath. The tested architectures achieve are designed to achieve three input proccesing throughputs of approximately $130\,\mathrm{MB/s}$, $260\,\mathrm{MB/s}$ and $520\,\mathrm{MB/s}$ which meet the requirements for emerging low-power applications. Furthermore, the tested ACF architectures are categorized based on their implementation, including the proposed ACOR, the MAC-based, and the systolic-based, with distinct symbols ($*$, $\circ$, $\diamond$) used to represent each. Additionally, the architectures are distinguished based on the number of channels and the folding method, with 5 different colors: blue for the single-channel unrolled case shown in Fig. 5a (SC-unrolled), black for the single-channel with folded datapath case shown in Fig. 9a (SC-DP-folded), red for the dual-channel (Q=2) unrolled case shown in Fig. 5b (MCQ2-unrolled), pink for the dual-channel

channel folded case shown in Fig. 9b (MCQ2-channel-folded), and green for the dual-channel channel and data folded case (MCQ2-fully-folded). Note that channel/datapath folding for the MAC and systolic architectures is achieved by using half the number of multipliers for each folding in order to match the input processing throughput ($T = 1$, $T = 0.5$ and $T = 0.25$) of their ACOR-based counterparts.

In all test cases, for both $p = 16$ and $p = 32$, the ACOR-ACF approach consistently produced more energy-efficient architectures, except for one case when $p = 16$ for the SC-DP-folded architecture which was outperformed by the MAC-based approach by 25.14% in terms of PDP as p is relatively small. For the remaining test scenarios, the ACOR-ACF approach achieved energy savings ranging from 6.55% to 14.56% for $p = 16$ compared with the MAC-based and systolic-based approaches. For $p = 32$, the energy savings were even greater due to the $O(\log_2 p)$ scaling of the multipliers, ranging from 30.91% to 51.47% compared with the MAC-based and systolic-based approaches. Also for all test scenarios, the ACOR-ACF approach achieved a slightly smaller throughput, by approximately 8%, due to the larger maximum delay of the multiplierless rotators used in our proposed architectures.

In Fig. 11c-d, we present a comparison of the achieved area for the three ACF architecture implementations (ACF, MAC, systolic) across 5 different number of channel and folding methods for $p = 16$ and $p = 32$. As depicted in Fig. 11c, for $p = 16$, the proposed ACOR-ACF method exhibits a slightly higher area requirement, ranging from 8.94% to 22.71%, compared to the best-performing MAC-based approach. This additional area overhead is primarily attributed to the utilization of extra memory cells in the ACOR approach but as previously discussed (Fig. 11a) the ACOR approach requires less energy for all cases except one (SC-DP-folded). However, when considering $p = 32$ (Fig. 11d), the ACOR approach clearly outperforms the MAC-based approach due to the advantageous $O(\log_2 p)$ scaling of the multipliers, achieving improvements of up to 27.18% for the MCQ2-unrolled case.

V. CONCLUSION

This paper presents ACOR, a framework that enables the adaptive generation of ACF architectures based on the specific requirements of a target application. Utilizing a modified ACF algorithm, the proposed framework optimizes the architecture mapping for any desired number of lags, sampling window size and function and throughput. This allows designers to explore various design options and select the most suitable architecture. In addition, several architecture optimizations have been proposed within the framework, including the utilization of multiplierless architectures and microarchitecture enhancements, aimed at minimizing energy dissipation typically associated with power-hungry multipliers commonly found in existing approaches. By incorporating these optimizations, the proposed framework achieves significant energy savings while maintaining high throughput, due to the pipelining.

REFERENCES

[1] S. Yoshida *et al.*, "Energy-efficient spectral analysis method using autoregressive model-based approach for internet of things," *IEEE Trans. Circuits Syst. I*, vol. 66, no. 10, pp. 3896–3905, 2019.

[2] K. Kajihara *et al.*, "Hardware implementation of autoregressive model estimation using burg's method for low-energy spectral analysis," in *IEEE SIPS.*, 2018, pp. 199–204.

[3] N. Alwan, "Systolic parallel architecture for brute-force autoregressive signal modeling," *Computers & Electrical Engineering*, vol. 39, no. 4, pp. 1358–1366, 2013.

[4] J. Cho, Y. Cho, M. R. Islam, J. Kim, and W.-K. Cho, "Hardware-efficient auto-correlation for synchronization of mimo-ofdm wlan systems," in *2009 International SoC Design Conference (ISOCC)*, 2009, pp. 560–563.

[5] S. Abdelazim, D. Santoro, M. Arend, F. Moshary, and S. Ahmed, "A hardware implemented autocorrelation technique for estimating power spectral density for processing signals from a doppler wind lidar system," *Sensors*, vol. 18, no. 12, 2018.

[6] W. E. M. Atri, F. Sayadi and R. Tourki, "Efficient hardware/software implementation of lpc algorithm in speech coding applications," *Journal of Signal and Information Processing*, vol. 3, no. 1, pp. 122–129, 2012.

[7] B. Fazlali and M. Eshghi, "A pipeline design for implementation of lpc feature extraction system based on levinson-durbin algorithm," in *2011 19th Iranian Conference on Electrical Engineering*, 2011, pp. 1–5.

[8] N. Alwan, "Systematic design of systolic correlators with application to parallel blackman-tukey spectral estimation," *Computing*, vol. 66, pp. 395–412, 06 2001.

[9] B. Apicella, A. Bruno, X. Wang, and N. Spinelli, "Fast fourier transform and autocorrelation function for the analysis of complex mass spectra," *International Journal of Mass Spectrometry*, vol. 338, pp. 30–38, 2013.

[10] Y. Xu, D. Zhen, J. X. Gu, K. Rabeyee, F. Chu, F. Gu, and A. D. Ball, "Autocorrelated envelopes for early fault detection of rolling bearings," *Mechanical Systems and Signal Processing*, vol. 146, p. 106990, 2021.

[11] K. Vos, "A fast implementation of burg's method."

[12] M. Garrido, "A survey on pipelined fft hardware architectures," *Journal of Signal Processing Systems*, vol. 94, 07 2021.

[13] A. V. Oppenheim and R. W. Schafer, *Discrete-time signal processing, 3rd edition*, 2009.

[14] C. Rader, "An improved algorithm for high speed autocorrelation with applications to spectral estimation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 18, no. 4, pp. 439–441, 1970.

[15] P. Tummeltshammer, J. C. Hoe, and M. Puschel, "Time-Multiplexed Multiple-Constant Multiplication," *IEEE TCAD*, vol. 26, no. 9, 2007.

[16] K. Möller, M. Kumm, M. Garrido, and P. Zipf, "Optimal shift reassignment in reconfigurable constant multiplication circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 3, pp. 710–714, 2018.

[17] M. Garrido, F. Qureshi, and O. Gustafsson, "Low-Complexity Multiplierless Constant Rotators Based on Combined Coefficient Selection and Shift-and-Add Implementation (CCSSI)," *IEEE Trans. Circuits Syst. I*, vol. 61, no. 7, pp. 2002–2012, 2014.

[18] R. Andersson and M. Garrido, "Using rotator transformations to simplify FFT hardware architectures," *IEEE Trans. Circuits Syst. I*, vol. 67, no. 12, pp. 4784–4793, 2020.

[19] C. Eleftheriadis and G. Karakonstantis, "Energy-Efficient Fast Fourier Transform for Real-Valued Applications," *IEEE Trans. Circuits Syst. II*, vol. 69, no. 5, pp. 2458–2462, 2022.

[20] H. Jeon, Y. Jung, S. Lee, and Y. Jung, "Area-efficient short-time fourier transform processor for time–frequency analysis of non-stationary signals," *Applied Sciences*, vol. 10, no. 20, 2020.

[21] P. Duhamel *et al.*, "On computing the inverse dft," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 2, pp. 285–286, 1988.

[22] A. Dempster and M. Macleod, "Multiplication by two integers using the minimum number of adders," in *IEEE ISCAS.*, 2005, pp. 1814–1817 Vol. 2.