IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 10 June 2024, accepted 25 June 2024, date of publication 3 July 2024, date of current version 12 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3422802

RESEARCH ARTICLE

A Novel Digital Equalizer Based on RF Sampling Beyond GHz

LORENZO CANESE[1], GIAN CARLO CARDARILLI[1], (Life Member, IEEE), RICCARDO LA CESA[1], LUCA DI NUNZIO[1], (Member, IEEE), ROCCO FAZZOLARI[1], DANIELE GIARDINO[2], MARCO RE[1], (Member, IEEE), AND SERGIO SPANÒ[1]

[1]Department of Electronic Engineering, Tor Vergata University of Rome, 00133 Rome, Italy
[2]IT Systems Srl, 00173 Rome, Italy

Corresponding author: Riccardo La Cesa (la.cesa@ing.uniroma2.it)

This work was supported by European Union—NextGenerationEU by Rome Technopole—CUP B83C22002820006, NRP Mission 4 Component 2 Investment 1.5, under Project ECS 0000024.

ABSTRACT Hardware implementations represent the major challenges when digital signal processors for ultra-wideband (UWB) signals must be developed. Due to the limitation of the maximum clock rate in digital devices, systems with high sampling rates (above GHz) cannot easily be implemented. In the literature, several works propose parallel architectures for the implementation of UWB. They are implemented on Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs) and try to overcome the limitation on the maximum clock frequency. In this work, a novel parallel architecture for digital UWB equalizer and an optimized version of the Least Mean Square Block (LMS) based on the Fast FIR Algorithms (FFA) are presented. Circuit simulations show that the proposed equalizer can process a UWB signal with a bandwidth reaching several GHz, using the typical clock frequencies available in FPGAs. The proposed version of the Block LMS is compared with the Fast Block LMS in terms of computational complexity. It exhibits better results and greater hardware design flexibility. Finally, the hardware implementation based on a Xilinx Kintex Ultrascale to process a UWB signal sampled at 1.6 GHz is described.

INDEX TERMS RF sampling, equalization, fast FIR algorithm (FFA), fast block least mean square (LMS), channel estimation.

I. INTRODUCTION

In the last decades, Analog-to-Digital Converters (ADC) and Digital-to-Analog Converters (DAC) have become capable of processing Radio Frequency (RF) signals with a bandwidth over the GHz. These converters are based on new technologies and are referred to as RF-Sampling, allow for the replacement of the Intermediate Frequency (IF) analog subsystem with a digital subsystem [1], reducing material costs, design time,despiteer. Furthermore, the flexibility of RF converters allows for a radio system design suitable for various applications, such as Software Defined Radio (SDR), wideband communication, and radar. Notable applications include positioning in new 6G technology, penetrating radar and high-resolution radar, used in fields like through-wall tracing, anti-terrorism stabilization, earthquake and disaster relief, archaeological discoveries, geological surveys and life sign detection [2], [3], [4], [5], [6], [7]. As discussed below, these RF converters allow for the implementation digital systems with very high bandwidth and throughput.

When the bandwidth of a signal is very wide, it can be considered as an Ultra-Wideband (UWB) signal. We can use the term UWB when the signal satisfies one of the two following conditions [8]: (1) the signal bandwidth is greater than 20% of its carrier frequency, or (2) the signal bandwidth is greater than 0.5 GHz.

UWB systems can take advantage of the ability of RF converters to digitize broadband signals [9]. Making a UWB analogue front-end, which is characterized by a good

The associate editor coordinating the review of this manuscript and approving it for publication was Li Zhang.

frequency response both for magnitude and phase, is becoming increasingly difficult. Therefore, an equalizer system is often necessary to mitigate signal distortions, compensate for channel frequency-dependent losses, and optimize channel overall performance [10], [11]. Typically, equalizers can be implemented by using digital devices based on Digital Pre-Distortion (DPD). They use different algorithms such as the Recursive Prediction Error Method (RPEM) [12] and the Least Mean Square (LMS) approach [13]. These algorithms can estimate a transfer function capable of compensating for the distortions of an unknown system. Generally, the unknown system is an analog component, and the transfer function obtained from the above algorithms is an estimate of the inverse of the unknown system transfer function. Equalizers also involve the use of a digital filter based on Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) architectures that implement the estimated transfer function [14], [15].

The implementation of digital equalizers involves the use of digital devices such as Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs), but the hardware implementation becomes a critical aspect when UWB signals are involved. This is because the devices cannot reach the clock frequency required to process a high sample rate beyond GHz [16]. To overcome this limitation, conventional systems must perform parallel processing through the design of parallel digital architectures, as shown in [17] and [18]. Therefore, modern converters use a high clock frequency to sample the UWB signal and a low clock frequency for digital devices to process the signal. Devices receive the signal represented as a burst of L samples at each clock cycle and, in this way, they can process the signal through parallel architectures, preserving the high sample rate of the signal. Consequently, the sampled signal is treated in the digital devices as an L-block of samples, and the digital equalizer must be designed through a parallel architecture to process the signal in real-time operation. Since the design of an analog front-end for a UWB application, which guarantees good performances in terms of magnitude and phase responses, is not easily achievable, this solution for the design of a digital equalizer for UWB signals that compensates the magnitude and phase distortions caused by the analog front-end.

To design a parallel architecture for a digital equalizer, several block-based algorithms such as the Block LMS and the Fast Block LMS [14] are considered. These algorithms can treat the signal as a block of L samples [14], [19], [20]. Consequently, they are useful to estimate the inverse transfer function of the analog component and to implement a digital equalizer for UWB signals.

In this work, we present a novel parallel architecture for digital UWB equalizer named Parallel Indirect Learning Architecture (PILA), based on the Indirect Learning Architecture (ILA) [21], which is used to compensate for the distortions of an analog component by estimating its inverse transfer function. Moreover, within the digital equalizer, we present the Parallel LMS (PLMS), an optimized version of the Block LMS algorithm. We highlight that PLMS is investigated considering complex data numbers, if compared to the literature where conventional block algorithms are analyzed considering real data [14], [19], [20]. Unlike traditional block-based algorithms, PLMS is based on the Fast FIR filtering Algorithms (FFA) that allow to increase the parallel computation and to reduce the computational complexity of the algorithms in terms of multiplications and sums [22], [23]. We prove that the PLMS exhibits better performance in terms of complexity, if compared to the Block LMS algorithm and the Fast Block LMS algorithm, based on time and frequency domain approaches, respectively [14], [20], and [24]. Finally, the hardware implementation of the PILA, composed of a parallel filter and the PLMS, shows that a digital system for UWB applications needs large resource utilization, which can be reduced by the FFA. Despite a thorough review of the literature on LMS-based equalizers, none of the recent work has addressed the problem of digital equalization operating with UWB signals considering the algorithms we reviewed. This emphasizes the innovation of our approach and the results obtained.

A. PAPER ORGANIZATION

The paper is organized as follows.

Section II gives an overview of the digital equalization based on the ILA architecture and the Block LMS algorithm in the time domain.

Section III shows the parallel filtering and the FFA in terms of architectures and computational complexity.

Section IV describes PILA and PLMS.

Section V shows the results of the numerical simulation, the computational complexity, and the hardware implementation.

Section VI draws the conclusions of this work.

II. EQUALIZATION AND CHANNEL ESTIMATION

A. INDIRECT ARCHITECTURE (ILA)

Digital equalization is a process used to compensate the signal distortions introduced by the analog front-end such as the power amplifier. For the sake of simplicity, the analog component will be treated as an unknown system. Several equalizer architectures have been proposed in the literature, providing different advantages in terms of hardware implementation and equalization performance. One of the most widely used equalizers is ILA [21]. To equalize an unknown system, the ILA employs an estimator block and a predistorter digital filter, as illustrated in Fig. 1.

The estimator block is implemented through an algorithm that evaluates the inverse transfer function of the unknown system represented by the set of coefficients $\hat{\mathbf{w}}[n]$. Estimation is performed by observing the input $d[n]$ and the output $u[n]$ of the unknown system. Therefore, the algorithm finds a set of coefficients related to the inverse transfer function, which depends on the digital filter structure (FIR or IIR).

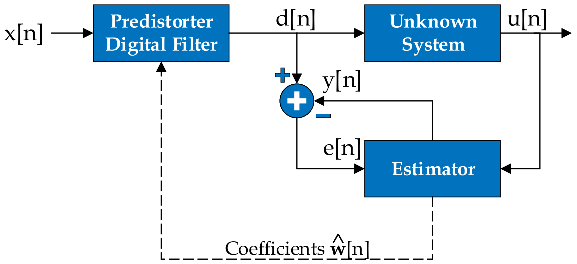


FIGURE 1. Indirect learning architecture.

The estimator generates $y[n]$, which is an estimation of $d[n]$, using the signal $u[n]$ and the estimated coefficients $\hat{\mathbf{w}}[n]$. Consequently, the error signal $e[n]$, obtained by the difference of the signal $d[n]$ with $y[n]$, is used to adjust the computed coefficients. The error is minimized at each iteration of the system and a copy of the estimated inverse transfer function is shared with the digital filter predistorter [25]. The sharing allows to compensate for the distortions introduced by the unknown system by applying a predistortion to the signal $x[n]$. This improves the performance of the channel. Finally, $u[n]$ will be a copy of the input signal $x[n]$.

A critical aspect of the architecture is the estimator because it estimates the coefficients $\hat{\mathbf{w}}[n]$. This block can be implemented through several algorithms as the LMS algorithm treated in the next section.

B. LMS ALGORITHM

LMS algorithm is based on the Method of Stochastic Gradient Descent (SGD) [14], a method used to minimize a cost function typically defined as the mean square error $J[n] = \mathbb{E}\{|e[n]|^2\}$, where $\mathbb{E}$ is the statistical expectation operator and $e[n]$ is an error signal. The SGD algorithm uses an FIR filter to model the unknown transfer function and is summarized as follows [26]:

$$\begin{aligned} y[n] &= \hat{\mathbf{w}}^T[n]\,\mathbf{u}[n] \\ e[n] &= d[n] - y[n] \\ \hat{\mathbf{w}}[n+1] &= \hat{\mathbf{w}}[n] - \mu \nabla J[n]. \end{aligned} \tag{1}$$

where $\hat{\mathbf{w}}[n] = [\hat{w}_0[n], \ldots, \hat{w}_{N-1}[n]]^T$ is the set of the estimated coefficients of the N-length FIR filter, $\mathbf{u}[n] = [u[n], \ldots, u[n-(N-1)]]^T$ is the input vector (regressor), $d[n]$ is the desired signal that is delayed of $D = \lfloor N/2 \rfloor$ samples to consider the fixed latency of the unknown system represented as a FIR filter, $y[n]$ is the estimation of the desired signal, μ is the step-size parameter, and $\nabla J[n]$ is the gradient of the cost function. SGD updates the coefficients $\hat{\mathbf{w}}[n]$ until $\nabla J[n]$ reaches a minimum value.

The gradient of the cost function $\nabla J[n]$ can be solved with respect to the complex variables $w_i[n]$, as shown in [26]:

$$\begin{aligned} \nabla J[n] &= \nabla \mathbb{E}\{|e[n]|^2\} = \mathbb{E}\{\nabla |e[n]|^2\} \\ &= \mathbb{E}\{e[n]\nabla e^*[n]\} = -\mathbb{E}\{e[n]\mathbf{u}^*[n]\}. \end{aligned} \tag{2}$$

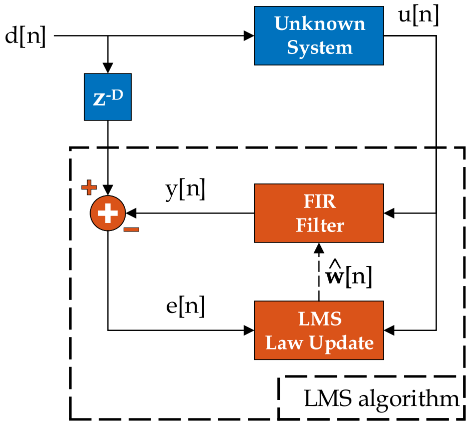


FIGURE 2. Architecture of the LMS algorithm.

Generally, the expectation $\mathbb{E}\{e[n]\mathbf{u}^*[n]\}$ is not known and is replaced by a mean [26]:

$$\hat{\mathbb{E}}\{e[n]\mathbf{u}^*[n]\} = \frac{1}{K}\sum_{i=0}^{K-1} e[n-i]\mathbf{u}^*[n-i] \tag{3}$$

where K is the number of samples used for the estimation. Using the expectation (3) in (2) for the simple case $K = 1$, equation (1) becomes the LMS algorithm:

$$\begin{aligned} y[n] &= \hat{\mathbf{w}}^T[n]\,\mathbf{u}[n] \\ e[n] &= d[n] - y[n] \\ \hat{\mathbf{w}}[n+1] &= \hat{\mathbf{w}}[n] + \mu e[n]\mathbf{u}^*[n] \end{aligned} \tag{4}$$

which is represented by the architecture shown in Fig. 2, where LMS Law Update is the block that estimates the new FIR filter coefficients $\hat{\mathbf{w}}[n+1]$.

C. BLOCK LMS ALGORITHM

The conventional LMS cannot be used when the signal is represented as a block of L samples. Consequently, we need to introduce the Block LMS algorithm to process a block of L samples at each clock cycle [14], [19], [20]. As follows, the Block LMS algorithm is obtained by manipulating the equations from (1) to (4).

First, SGD (1) must be represented using the time domain parallel notation to process a block of L samples. Filtering between the regressor $\mathbf{u}[n]$ and the estimated coefficients $\hat{\mathbf{w}}[n]$ becomes

$$Y[n] = \hat{\mathbf{w}}[n] \circledast U[n] \tag{5}$$

where $U[n]$ and $Y[n]$ are the block representations of $u[n]$ and $y[n]$ respectively:

$$\begin{aligned} U[n] &= [u[kL], u[kL+1], \ldots, u[kL+L-1]] \\ Y[n] &= [y[kL], y[kL+1], \ldots, y[kL+L-1]] \end{aligned} \tag{6}$$

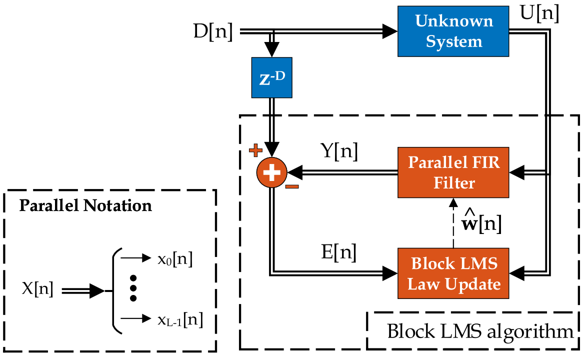


FIGURE 3. Block LMS architecture.

considering k as the block index related to the original sample n:

$$n = kL + i, \qquad i = 0, 1, \ldots, L-1$$
$$k = 0, 1, 2, \ldots. \tag{7}$$

We want to highlight and to anticipate that the time domain parallel filtering is shown in detail in the next sections because it allows for the optimization of the computational complexity of the filtering.

The error signal represented with a block of L samples becomes:

$$E[n] = D[n] - Y[n] \tag{8}$$

where $D[n]$ and $E[n]$ are the block representations of $d[n]$ and $e[n]$ respectively:

$$\begin{aligned} D[n] &= [d[kL], d[kL+1], \ldots, d[kL+L-1]] \\ E[n] &= [e[kL], e[kL+1], \ldots, e[kL+L-1]]. \end{aligned} \tag{9}$$

Subsequently, the gradient of the cost function in (2) can be replaced using (3) with $K = L$ and considering the block notation:

$$\begin{aligned} \hat{\mathbb{E}}\{e[n]\mathbf{u}^*[n]\} &= \frac{1}{L}\sum_{i=0}^{L-1} e[n-i]\mathbf{u}^*[n-i] \\ &= \frac{1}{L}\mathbf{A}^H[n]\,E[n] \end{aligned} \tag{10}$$

where $(.)^H$ represents the conjugate transpose. $\mathbf{A}^H[n]$ has dimensions $N \times L$ and is defined as:

$$\mathbf{A}^T = [\mathbf{u}[kL], \mathbf{u}[kL+1], \ldots, \mathbf{u}[kL+L-1]]. \tag{11}$$

Finally, the block diagram of the Block LMS, composed of the parallel Fir filter and the Block LMS Law Update, is shown in Fig. 3 and summarized as follows:

$$\begin{aligned} Y[n] &= \hat{\mathbf{w}}[n] \circledast U[n] \\ E[n] &= D[n] - Y[n] \\ \hat{\mathbf{w}}[n+1] &= \hat{\mathbf{w}}[n] + \frac{\mu}{L}\mathbf{A}^H[n]E[n]. \end{aligned} \tag{12}$$

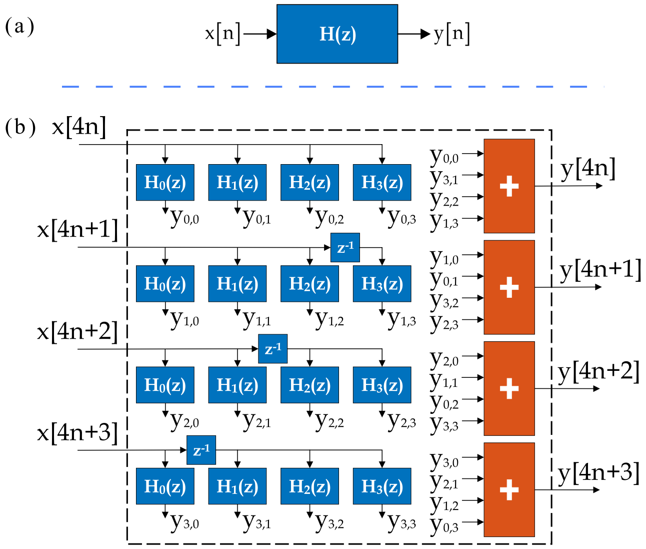


FIGURE 4. Comparison between a conventional FIR filter (a), and its 4-parallel version (b).

III. PARALLEL FILTER STRUCTURES

A. TIME DOMAIN PARALLEL FILTERING

In the literature, time domain parallel filtering is treated in several works [27], [28] showing different parallel architectures to process a block of L samples. Generally, a parallel filter involves a high number of multiplications equal to $N \times L$, where N is the number of the coefficients of the FIR filter, and L is the parallel factor that corresponds to the number of the polyphase components of $x[n]$. In this work, we consider the parallel factor and the block size L of the input signal as the same thing.

To explain the difference between a conventional FIR filter structure and its parallel structure, we show in Fig. 4 (a) an example of a conventional FIR filter and its 4-parallel version.

The inputs $x[4n+i]$ and outputs $y[4n+i]$ of Fig. 4 (b) are obtained by the polyphase decomposition of $x[n]$ and $y[n]$ of Fig. 4 (a). Using the same approach, the coefficients of $H_i(z)$ of Fig.4 (b) are obtained by the polyphase decomposition of the prototype filter $H(z)$ [29].

From here, the polyphase components of the inputs and outputs are represented by the notation (13) and the related Z transform shown in (14):

$$\begin{aligned} x[Ln+i] &= x_i[n] \\ y[Ln+i] &= y_i[n] \end{aligned} \tag{13}$$

$$\begin{aligned} X_i(z) &= Z\{x_i[n]\} \\ Y_i(z) &= Z\{y_i[n]\} \end{aligned} \tag{14}$$

where $i \in [0, L-1]$.

The Z transform equations are introduced because optimized filter structures are investigated using the Z transform [22], [23], [27]. Finally, we show the general form of the parallel filtering represented by the matrix notation:

$$\begin{bmatrix} Y_0(z) \\ Y_1(z) \\ \vdots \\ Y_{L-1}(z) \end{bmatrix} == \begin{bmatrix} H_0(z) & H_{L-1}(z)z^{-1} & \dots H_1(z)z^{-1} \\ H_1(z) & H_0(z) & \dots H_2(z)z^{-1} \\ \vdots & \vdots & \vdots \\ H_{L-1}(z) & H_{L-2}(z) & \dots \quad H_0(z) \end{bmatrix} \begin{bmatrix} X_0(z) \\ X_1(z) \\ \vdots \\ X_{L-1}(z) \end{bmatrix} \tag{15}$$

where the coefficients $H_i(z)$ are obtained by the polyphase decomposition of the prototype filter $H(z)$.

B. FAST FIR FILTERING ALGORITHM

The parallel filter shown in (15) involves a high number of multiplications equal to $N \times L$, where N is the length of $H(z)$ which is supposed to be a FIR filter. Several works based on Fast FIR Filtering Algorithms (FFA) or Fast FIR Algorithms [22], [23], [30] take into account the matrix product (15) to improve the computational complexity by reducing the number of FIR filters of Fig. 4 (b). This analysis allows one to reduce the number of the required multiplications. Since the computational cost of the multiplication is more expensive than the sum [31], we want to investigate this algorithm to improve the Block LMS equations in terms of computational and hardware complexity. There are also other works that analyze the symmetry of the impulse response of $H(z)$ to obtain optimized architectures [32], [33]. These structures cannot be used in an equalizer because the estimator algorithm does not guarantee the symmetry of the estimated coefficients. Consequently, we show how to manipulate (15) to obtain an optimized parallel filter architecture and, in the following sections, the same approach will be used to reduce the computational complexity of the Block LMS shown in (12).

As follows, the FFA for $L = 2$ is shown. There are several algorithms to implement an efficient structure and, generally, the FFA based on the Winograd algorithm is the most used [34]. In this section, the Winograd algorithm is described using the matrix representation since it allows for a better understanding of the PLMS analysis shown in the next sections.

We want to anticipate that, for PLMS, the filtering $Y[n]$ and the matrix product $\mathbf{A}^H[n]E[n]$ of (12) are improved in terms of computational complexity by using the Winograd algorithm. Typically, FFA algorithms are analyzed in the literature to optimize parallel filtering, but they are not investigated to improve a matrix product, as shown in the PLMS section.

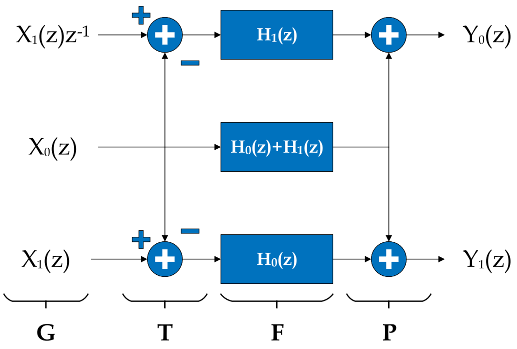


FIGURE 5. **Optimized 2-Parallel Filter. The architecture is arranged to highlight the matrices of (16).**

The Winograd algorithm for $L = 2$ is shown in (16):

$$\begin{aligned} \begin{bmatrix} s_0 \\ s_1 \end{bmatrix} &= \begin{bmatrix} g_1 & g_0 \\ g_2 & g_1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \end{bmatrix} = \\ &= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_1 & 0 & 0 \\ 0 & f_0 + f_1 & 0 \\ 0 & 0 & f_0 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \\ &= \mathbf{P} \cdot \mathbf{F} \cdot \mathbf{T} \cdot \mathbf{G} \end{aligned} \tag{16}$$

where $\mathbf{P}$ and $\mathbf{T}$ are called pre-processing matrix and post-processing matrix respectively, while $\mathbf{F}$ and $\mathbf{G}$ contain the values of the original matrix.

The Winograd algorithm can be used by writing the matrix equation (15) for $L = 2$ as follow:

$$\begin{aligned} \begin{bmatrix} Y_0(z) \\ Y_1(z) \end{bmatrix} &= \begin{bmatrix} H_0(z) & H_1(z)z^{-1} \\ H_1(z) & H_0(z) \end{bmatrix} \begin{bmatrix} X_0(z) \\ X_1(z) \end{bmatrix} = \\ &= \begin{bmatrix} X_0(z) & X_1(z)z^{-1} \\ X_1(z) & X_0(z) \end{bmatrix} \begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix}. \end{aligned} \tag{17}$$

By comparing (16) and (17), we obtain:

$$\begin{bmatrix} Y_0(z) \\ Y_1(z) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} H_1(z) & 0 & 0 \\ 0 & H_0(z) + H_1(z) & 0 \\ 0 & 0 & H_0(z) \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} X_1(z)z^{-1} \\ X_0(z) \\ X_1(z) \end{bmatrix} \tag{18}$$

by assuming $[s_0, s_1] = [Y_0(z), Y_1(z)]$, $[f_0, f_1] = [H_0(z), H_1(z)]$ and $[g_0, g_1, g_2] = [X_1z^{-1}, X_0, X_1]$. The optimized parallel filter for $L = 2$ is shown in Fig. 5.

The optimized structure of Fig. 5 is composed of only 3 subfilters $H_i(z)$ rather than 4 subfilters, as shown in equation (17). In this way, we reduce the multiplications related to one subfilter, and also the sums related to the same subfilter at the cost of an increase of 4 adders of the pre-processing and post-processing matrices. When the coefficients of $H(z)$ and the components $x_i[n]$ are complex signals, this optimization saves a lot of resources that can be used to extend the length of the filter $H(z)$ improving the performance of the equalizer.

C. ITERATIVE METHOD

The L-parallel filter structures can be designed using several algorithms such as Cook-Toom and Winograd [34]. When the parallel factor L is very large, the structures obtained by these algorithms are not well optimized. As shown in (16), the pre-processing and post-processing matrices involve only sums for $L = 2$. For large values of L, the $\mathbf{P}$ and $\mathbf{T}$ matrices also involve several constant multiplications. Therefore, a better filter structure that involves only sums in the pre-processing and post-processing matrices, can be obtained by iterating the Winograd approach for $L = 2$ [22].

In this approach, the optimized 2-parallel filter is considered as the building block. We can iterate the block to obtain a L-parallel filter where the matrix $\mathbf{P}$ and $\mathbf{T}$ involve only sums. An example is the case $L = 4$:

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} g_3 & g_2 & g_1 & g_0 \\ g_4 & g_3 & g_2 & g_1 \\ g_5 & g_4 & g_3 & g_2 \\ g_6 & g_5 & g_4 & g_3 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} \tag{19}$$

We can partition (19) into 2-by-2 blocks to obtain a similar form of (16):

$$\begin{bmatrix} \mathbf{s_0} \\ \mathbf{s_1} \end{bmatrix} = \begin{bmatrix} \mathbf{g_1} & \mathbf{g_0} \\ \mathbf{g_2} & \mathbf{g_1} \end{bmatrix} \begin{bmatrix} \mathbf{f_0} \\ \mathbf{f_1} \end{bmatrix} = \\ = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{f_0}+\mathbf{f_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{f_0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ \mathbf{g_2} \end{bmatrix} \tag{20}$$

where

$$\mathbf{g_0} = \begin{bmatrix} g_1 & g_0 \\ g_2 & g_1 \end{bmatrix} \quad \mathbf{g_1} = \begin{bmatrix} g_3 & g_2 \\ g_4 & g_3 \end{bmatrix} \quad \mathbf{g_2} = \begin{bmatrix} g_5 & g_4 \\ g_6 & g_5 \end{bmatrix} \\ \mathbf{s_0} = \begin{bmatrix} s_0 & s_1 \end{bmatrix}^T \quad \mathbf{s_1} = \begin{bmatrix} s_2 & s_3 \end{bmatrix}^T \\ \mathbf{f_0} = \begin{bmatrix} f_0 & f_1 \end{bmatrix}^T \quad \mathbf{f_1} = \begin{bmatrix} f_2 & f_3 \end{bmatrix}^T. \tag{21}$$

By using the same approach of $L = 2$, we obtain for $L = 4$:

$$\begin{bmatrix} \mathbf{Y_0}(z) \\ \mathbf{Y_1}(z) \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{H_1}(z) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H_0}(z)+\mathbf{H_1}(z) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{H_0}(z) \end{bmatrix} \\ \begin{bmatrix} \mathbf{I} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{X_1}z^{-1} \\ \mathbf{X_0} \\ \mathbf{X_1} \end{bmatrix} \tag{22}$$

where:

$$\mathbf{H_0}(z) = \begin{bmatrix} H_0(z) & H_3(z)z^{-1} \\ H_1(z) & H_0(z) \end{bmatrix} \quad \mathbf{H_1}(z) = \begin{bmatrix} H_2(z) & H_1(z) \\ H_3(z) & H_2(z) \end{bmatrix} \\ \mathbf{Y_0}(z) = \begin{bmatrix} Y_0(z) & Y_1(z) \end{bmatrix}^T \quad \mathbf{Y_1}(z) = \begin{bmatrix} Y_2(z) & Y_3(z) \end{bmatrix}^T \\ \mathbf{X_0}(z) = \begin{bmatrix} X_0(z) & X_1(z) \end{bmatrix}^T \quad \mathbf{X_1}(z) = \begin{bmatrix} X_2(z) & X_3(z) \end{bmatrix}^T. \tag{23}$$

Each matrix transfer function can be implemented by the optimized structure in Fig. 5, which involves three scalar transfer functions, for a total of 9 scalar transfer functions for the 4-parallel filter [22].

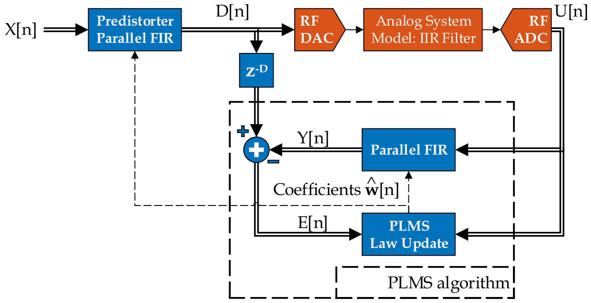


FIGURE 6. Parallel Indirect Learning Architecture (PILA).

As shown in [22], the iterative approach can be used for the general case $L = 2^m$ for m integer and allows one reduce the number of scalar transfer functions from $L \times L$ to 3^m. The number of multiplications and sums related to the transfer functions is reduced thet a cost of a small increase of the sums for the pre-processing and post-processing matrices.

IV. PROPOSED ARCHITECTURE

A. PARALLEL INDIRECT LEARNING ARCHITECTURE (PILA)

ILA is an often used equalizer and is obtained by merging the analog front-end with a digital device. When a UWB signal is involved, digital devices such as the FPGAs cannot be used to implement a conventional equalizer architectures with a clock frequency beyond GHz. For this reason, the architecture of Fig. 1 must be converted to a parallel architecture that allows for the parallel processing of UWB signals. This solution allows for the use of digital devices such as FPGA and ASIC for the equalization process. To preserve the high sample rate of the UWB signal, the digital device treats the signal as a block of L samples with a clock frequency of $F_{clk} = F_s/L$, where F_s is the sampling frequency used to sample the UWB signal by the RF converters. Therefore, we present PILA, the parallel version of ILA, that performs the parallel processing as shown in Fig. 6.

Compared to Fig. 1, we highlight that the unknown system is an analog component, and we introduce the RF converters that represent the conversion from the digital domain to the analog domain and vice versa. The meaning of the signals is the same as in Fig. 1 and the difference consists of their parallel representation. The digital filter is replaced with a parallel FIR filter optimized by FFA, while the estimator is the critical block of the architecture and it is implemented by the PLMS as shown in the next section.

B. PARALLEL LMS (PLMS)

PLMS is an optimized algorithm of the Block LMS summarized in (12) and shown in Fig. 3. The computational complexity of the Block LMS is improved by the FFA that reduces the number of required multiplications involving an improvement in terms of hardware implementation.

The first improvement concerns the filtering of (5) and is obtained by the FFA which reduces the number of transfer functions implemented, as explained in the previous section and in [22], [23], [27], and [30].

The second improvement is less obvious and is related to the matrix product $\mathbf{A}^H[n]E[n]$ of (12). To prove that FFA can be used, we consider the Z transform of the matrix product for the case $N = 2$ and $N = 4$ with $L = 2$, where N is the number of the coefficients $\hat{\mathbf{w}}[n]$.

The first example is the case $L = 2$ and $N = 2$. First, the matrix $\mathbf{A}^H[n]$ and the vector $E[n]$ must be represented by the Z transform. Since $L = 2$, the signal $u[n]$ is treated as a block composed of 2 samples, we consider two input signals $u_0[n]$ and $u_1[n]$. Consequently, also the vector $E[n]$ is composed of 2 error signals $e_0[n]$ and $e_1[n]$. The Z-trasforms of the signals are $\mathbb{Z}\{u_i[n]\} = U_i(z)$ and $\mathbb{Z}\{e_i[n]\} = E_i(z)$ for $i \in [0, 1]$, they can be used to consider the Z transform of the matrix product. We highlight that the matrix $\mathbf{A}^H[n]$ is composed of the elements $[u[n], u[n-1], u[n+1]]$, and their Z transforms corresponds to $\mathbb{Z}\{u[n]\} = U_0(z)$, $\mathbb{Z}\{u[n+1]\} = U_1(z)$ and $\mathbb{Z}\{u[n-1]\} = U_1(z)z^{-1}$. At this point, if (12) is compared with (16), we obtain:

$$\mathbf{A}^H(z)E(z) = \begin{bmatrix} U_0^*(z) & U_1^*(z) \\ U_1^*(z)z^{-1} & U_0^*(z) \end{bmatrix} \begin{bmatrix} E_0(z) \\ E_1(z) \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} E_1(z) & 0 & 0 \\ 0 & E_0(z)+E_1(z) & 0 \\ 0 & 0 & E_0(z) \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} U_1^*(z) \\ U_0^*(z) \\ U_1^*(z)z^{-1} \end{bmatrix} \tag{24}$$

where $[g_0, g_1, g_2] = [U_1^*(z), U_0^*(z), U_1^*(z)z^{-1}]$ and $[f_0, f_1] = [E_0(z), E_1(z)]$. In this case, we highlight that $E_i(z)$ are scalar values and are not transfer functions that involve several multiplications. Consequently, the multiplications are reduced from 4 to 3.

The second example is the case $L = 2$ and $N = 4$ and is less obvious than the case $N = 2$:

$$\mathbf{A}^H(z)E(z) = \begin{bmatrix} U_0^*(z) & U_1^*(z) \\ U_1^*(z)z^{-1} & U_0^*(z) \\ U_0^*(z)z^{-1} & U_1^*(z)z^{-1} \\ U_1^*(z)z^{-2} & U_0^*(z)z^{-1} \end{bmatrix} \begin{bmatrix} E_0(z) \\ E_1(z) \end{bmatrix}. \tag{25}$$

The matrix $\mathbf{A}^H(z)$ can be partitioned into 2-by-2 blocks to obtain a similar form of (24) allowing the use of the FFA:

$$\mathbf{A}^H(z)E(z) = \begin{bmatrix} \mathbf{A_0}^H(z) \\ \mathbf{A_1}^H(z) \end{bmatrix} \begin{bmatrix} E_0(z) \\ E_1(z) \end{bmatrix} \tag{26}$$

where:

$$\mathbf{A_0}^H(z) = \begin{bmatrix} U_0^*(z) & U_1^*(z) \\ U_1^*(z)z^{-1} & U_0^*(z) \end{bmatrix}$$

$$\mathbf{A_1}^H(z) = \begin{bmatrix} U_0^*(z)z^{-1} & U_1^*(z)z^{-1} \\ U_1^*(z)z^{-2} & U_0^*(z)z^{-1} \end{bmatrix}. \tag{27}$$

By comparing (26) with (24), we see that FFA can be applied to $\mathbf{A_0}^H(z)$ and $\mathbf{A_1}^H(z)$ optimizing the matrix product (25).

For the general case $L = 2^m$, the matrix product $\mathbf{A}^H(z)E(z)$ can be optimized by partioning the $\mathbf{A}^H(z)$ into L-by-L blocks $\mathbf{A}_i^H(z)$ where $i \in [0, \ldots, \lceil N/L \rceil - 1]$, and applying the FFA for each block.

V. RESULTS

The performance of the proposed equalizer is investigated by Fixed-Point (FXP) simulations. The purpose of the analysis consists to show the operation of the digital equalizer composed on PILA and the PLMS shown in Fig. 6.

Several FXP simulations are performed by using Simulink and considering an analog component characterized by an IIR filter. The simulated analog component represents any cascade of analog components that lie between the Analog Digital Converter (ADC) and the antennas. The simulation is thus aimed at demonstrating the system's effectiveness in real-world scenarios. By the simulation we highlight: (a) the error reduction of the LMS algorithm, (b) the compensations obtained by the equalizer, and (c) the quality improvement of the transmitted signal supposing to have a 16-QAM (Quadrature Amplitude Modulation) signal.

The Simulink results are also validated through the Mathworks FPGA-In-The-Loop approach that provides the same results as the Simulink FXP simulations.

A. NUMERICAL SIMULATIONS

The PILA and PLMS performances are analyzed by several fixed-point (FXP) simulations in Simulink environment. The experimental setup is a transmitter based on Fig. 6 where the RF DAC and RF ADC are replaced by a parallel-to-serial and a serial-to-parallel blocks, respectively. The transmitter is designed with a parallel factor of $L = 8$ and a clock frequency of 200; [MHz] to accommodate a UWB signal sampled by the converters at a sample frequency of 1.6; [GHz]. $X[n]$ is a 16-QAM signal generated by a binary source with a bit-rate of 3.2; [Gbps], shaped by a Raised Cosine Filter with a roll-off of 0.25, and represented by 2 samples per symbol. The analog component is modeled as a channel composed of a cascade of an IIR filter, which introduces amplitude and phase distortions, and a noise generator, which sets a signal-to-noise ratio (SNR) of 50; [dB]. The Signal-to-Noise Ratio (SNR) value is very high because the noise introduced by the analog feedback path can be supposed to be very small for a transmitter. The estimator is based on the PLMS shown in the previous sections, and itestimates the inverse transfer function of the IIR filter using the coefficients 32.

FXP simulations are performed for different step-size values. We evaluated the performance in terms of the error $E[n]$ and the Modulation Error Ratio (MER) since the modulation type is a 16-QAM. The results are shown in Fig. 7 and 8.

The plots show that the convergence time is shorter for large values μ and that performance is improved for each iteration. Since the SNR is very high, the curves converge at the same value with increasing time for both figures. Consequently, a large μ can improve the analog channel estimation exhibiting a small convergence time, a small error and a large MER. Additionally, we evaluated the performance

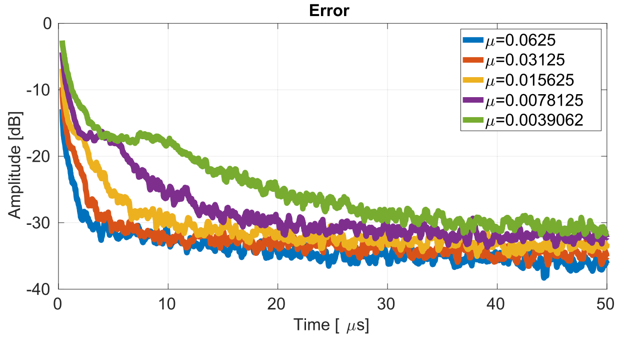


FIGURE 7. Experimental error for several μ values.

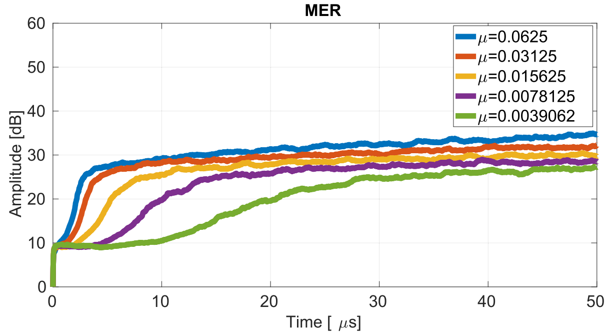


FIGURE 8. Experimental MER for several μ values.

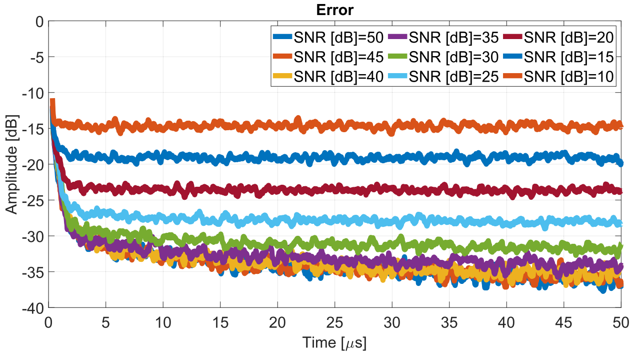


FIGURE 9. Experimental error for several SNR values.

in terms of the error $E[n]$ for different SNR values. The results shown in Fig. 9 illustrate how the error decreases as the signal-to-noise ratio increases.

To prove the correct operation of the digital equalizer, we show the constellations of the unequalized and equalized signal $U[n]$ in Fig. 10 and 11, evaluated for $\mu = 0.0625$ and a simulation time of 100 [μs]. In both figures, the red crosses represent the reference constellation, and the blue points represent the constellation obtained by the signal $U[n]$ of Fig. 6. In Fig. 10, the blue constellation is strongly distorted and also rotated compared to the reference constellation. In Fig. 11, the distortions are compensated and the blue constellation matches with the reference constellation.

The same simulation is performed to evaluate the cascade of the estimated filters with the IIR approach. We show the frequency response to evaluate the magnitude distortion in Fig. 12 and 13, and the group delay response to evaluate the phase distortion in Fig. 14 and 15.

Fig. 12 shows that the frequency response of the cascade (orange line) of the predistorter (Parallel FIR) with the IIR filter matches with the response of the analog component. Fig. 13 shows the amplitude ranging from -1 to $+1$ dB of the Fig. 12. The magnitude ripple of the analog component (blue line) is compensated by the digital equalizer exhibiting

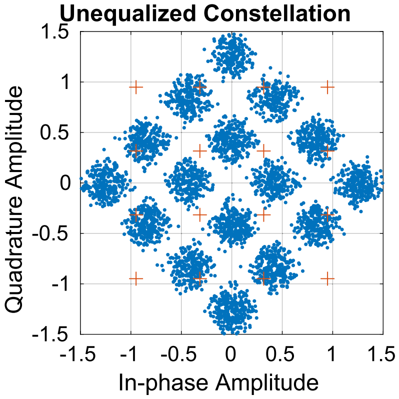


FIGURE 10. Experimental unequalized constellation.

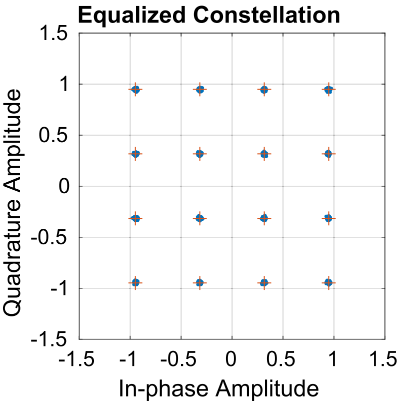


FIGURE 11. Experimental equalized constellation evaluated for $\mu = 0.0625$ and a simulation time of 100 [μs].

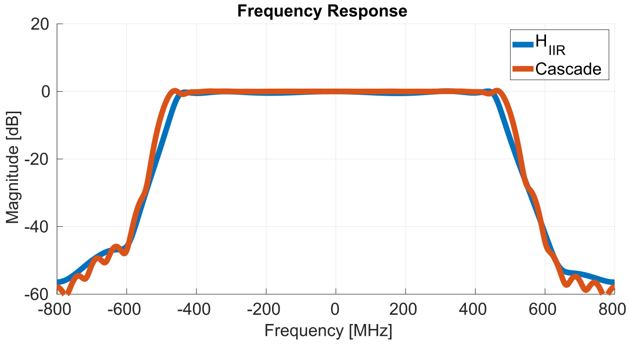


FIGURE 12. Comparison of the frequency response between the IIR filter (blue line) and the filters cascade (red line) evaluated for $\mu = 0.0625$ and a simulation time of 100 [μs].

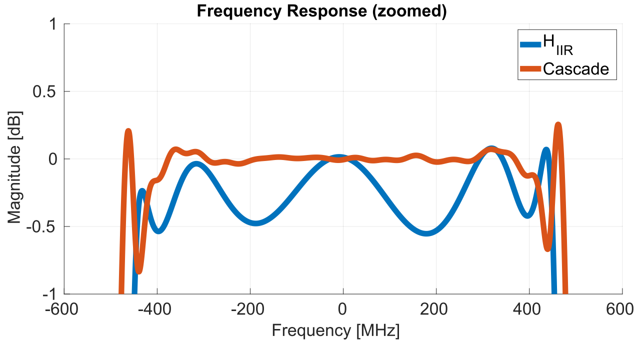


FIGURE 13. Zoomed frequency response comparison between the IIR filter (blue line) and the filters cascade (red line) evaluated for $\mu = 0.0625$ and a simulation time of 100 [μs].

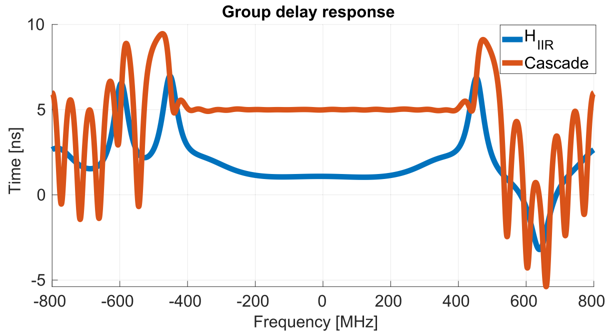


FIGURE 14. Group delay response comparison between the IIR filter (blue line) and the filters cascade (red line) evaluated for $\mu = 0.0625$ and a simulation time of 100 [μs].

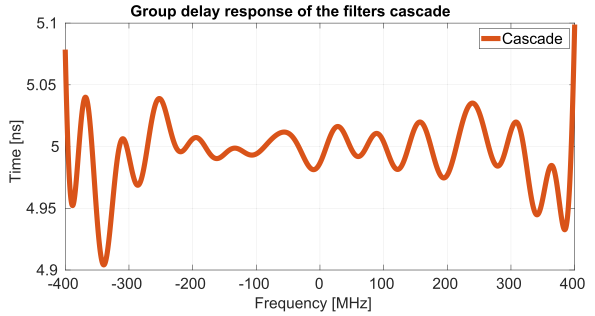


FIGURE 15. Zoomed group delay response of the filters cascade (blue line) evaluated for $\mu = 0.0625$ and a simulation time of 100 [μs].

an almost flat response (red line) obtained by the cascade of the filters.

Regarding phase distortions, Fig. 14 analyzes the phase represented by its group delay. As can be observed, the response of the analog component (blue line) is compensated by the predistorter exhibiting a flat response (organge line). Finally, Fig. 15 shows the group delay response obtained by the cascade of the filters, which exhibits a ripple less than 100 [ps] in the frequency range between −400 [MHz] and +400 [MHz] that corresponds to the symbol rate of 800 [MSps].

At increasing time, the system tends to improve performance by reducing the ripples of the magnitude and the group delay responses of Fig. 13 and 15.

B. COMPUTATIONAL COMPLEXITY

PILA shows a reduced computational complexity in terms of multiplications due to the FFA. The analysis is performed for the general case $L = 2^m$ and $N = 2^k$ for m and k integers with $k = m$. We highlight that the computational complexity in the literature is analyzed in terms of real multiplications considering real data, but, in this analysis, we consider the complex multiplications because complex data are involved in real world use cases. The analysis is partitioned into two parts: (a) the predistorter parallel filter and (b) the comparison between the PLMS, the Block LMS and the Fast Block LMS [14].

The predistorter parallel FIR filter is improved by FFA which reduces the required multiplications from $N \times L$ to $N/L \times 3^m$, where N is the number of coefficients, L is the block size, N/L is the number of coefficients of the scalar transfer functions, and 3^m is the number of scalar transfer functions.

The computational complexity of the PLMS can be analyzed considering the required multiplications of the L-parallel filter and the matrix product $\mathbf{A}^H[n]E[n]$. The L-parallel filter is performed by $N/L \times 3^m$ complex multiplications, and the matrix product can be split into L-by-L blocks $\mathbf{A}_i^H[n]$. For the general case analyzed in this work, the matrix product is composed of N/L blocks, and 3^m complex multiplications are performed for each block. Consequently, the complex multiplications required by the PLMS are:

$$\frac{N}{L}3^m + \frac{N}{L}3^m = 2\frac{N}{L}3^m \tag{28}$$

To compare the PLMS with the Block LMS and the Fast Block LMS, we have to consider a block size of $L = N$ because the Block LMS and the Fast Block LMS are analyzed in the literature considering the multiplications required to process a block of N samples. Consequently, PLMS requires a number of complex multiplications equal to:

$$\begin{aligned} 2\frac{N}{L}3^m &= 2\frac{N}{L}3^{log_2(L)} \\ &= 2\frac{N}{N}3^{log_2(N)} = 2N^{log_2(3)} \end{aligned} \tag{29}$$

The Block LMS is shown in (12) and requires $2N^2$ complex multiplications: N^2 for the parallel filter and N^2 for the matrix product.

Fast Block LMS is analyzed in several works considering real data and its architecture is shown in [14]. The architecture is composed of 5 M-point Fast Fourier Transforms (FFTs) and $2M$ complex multiplications performed in the frequency domain for the filtering and the gradient estimation. Typically, a M-point FFT is performed by $M\, log_2(M)$ complex multiplications [29], and considering the overlap-and-save method, the required multiplications become $2N\, log_2(2N)$

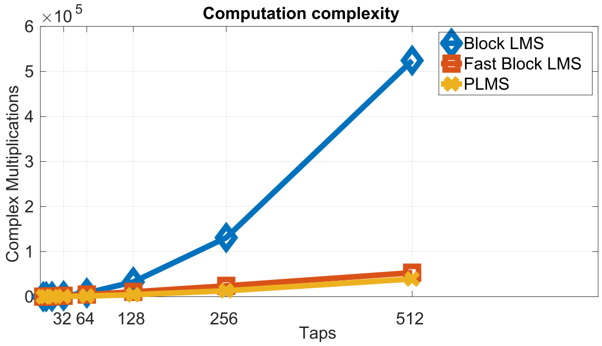


FIGURE 16. Comparison between the Block LMS, the Fast Block LMS and the PLMS.

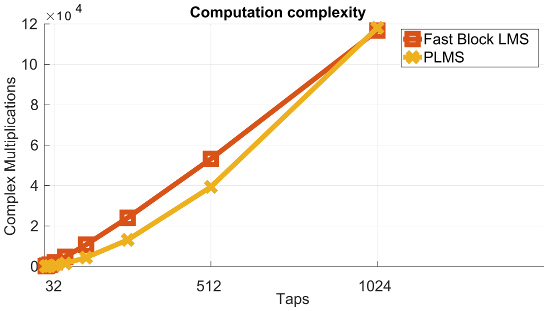


FIGURE 17. Comparison between the Fast Block LMS and the PLMS.

where $M = 2N$ because the algorithm considers the 50% overlap to obtain a most efficient structure [14]. By considering the 5 FFTs, the whole computational complexity becomes:

$$5M\, log_2(M) + 2M = 10N\, log_2(2N) + 4N \qquad (30)$$

The comparison of the 3 algorithms is shown in Fig. 16.

As is well known in the literature, the Fast block LMS is better than the Block LMS as it requires a lower computational complexity. However, the PLMS exhibits a better computational complexity than the Fast Block LMS as shown in Fig. 17.

For $N < 1024$, PLMS requires fewer multiplications then the Fast Block LMS. For a digital equalizer implemented on a digital device as an FPGA or ASIC, the hardware resources usage is a critical aspect and, consequently, a parallel architecture of an equalizer characterized by a lot of coefficients cannot be implemented. The results of Fig. 17 are also summarized in table 1.

Finally, the PLMS is better as it requires fewer multiplications for a block size of $L = N$ and, consequently, it requires less hardware resources. Another aspect of PLMS is the flexibility of the design, because it allows the implementation of an equalizer to estimate the coefficients N with a parallel factor $L < N$. This advantage allows one to design an equalizer with a reduced number of the implemented multipliers, and, consequently, the hardware complexity can be improved in terms of resources, area, and power. In addition, the Fast Block LMS can be implemented with a parallel factor $L < N$, but the hardware design of the FFTs is not easily achievable in this case and involves an increase in the design difficulty.

TABLE 1. Complex multiplications estimation of the fast block LMS and the PLMS.

Coefficients	FLMS	PLMS
4	136	18
8	352	54
16	864	162
32	2048	486
64	4736	1458
128	10752	4374
256	24064	13122
512	53248	39366
1024	116736	118098

TABLE 2. Resource utilization and dynamic power consumption of the 8-parallel filter optimized by the FFA.

Resource	Used	Available	Utilization %
LUT	12027	331680	3.63
FF	9484	663360	1.43
DSP48E2	432	2760	15.65
Dynamic Power	1.357 (W)		

C. HARDWARE IMPLEMENTATION

The hardware complexity of PILA shown in Fig. 6 is investigated considering a parallel factor of $L = 8$, a FPGA clock frequency of 200 [MHz], a set of $N = 32$ coefficients, and a Fixed-Point (FXP) dynamic of 14 bits. The parallel factor and the FPGA clock frequency allows to process a signal sampled at 1.6 [GHz]. The above parameters were chosen to maximize the use of available FPGA hardware resources by selecting a degree of parallelism that maximizes the speed of sample processing, thereby making maximum use of the available DSPs within the FPGA.

The architecture is partitioned into two blocks that are analyzed separately: (a) the parallel FIR optimized by the FFA and (b) the PLMS. For both analyses, the blocks are coded in VHDL, at the Register Transfer Level (RTL), and synthesized using Xilinx Vivado 2019.1 on a Virtex Ultrascale xcku060-ffva1517. The VHDL code is generated using the HDL Coder toolbox from MathWorks. This toolbox allows for the generation of portable and synthesizable Verilog and VHDL code from Simulink RTL models. The generated HDL code can be used for both FPGA programming and ASIC development. Although the code is suitable for ASIC implementations, the results presented are specific to FPGA resources for the purpose of fast prototyping. In Tables 2 and 3 resource utilization and dynamic power consumption are shown.

As expected from the computational complexity analysis of the previous section, the tables show that the (DSPs)employed to implement the multipliers and adders are the most used resources and the major contributors to power consumption. Considering a possible design requirement

TABLE 3. Resource utilization and dynamic power consumption of the PLMS.

Resource	Used	Available	Utilization %
LUT	67561	331680	20.37
LUTRAM	9372	146880	6.38
FF	44097	663360	6.65
DSP48E2	2160	2760	78.26
Dynamic Power	12.170 (W)		

for complex applications, where a more judicious use of FPGA resources for other tasks is necessary, it is possible to leverage the proposed architecture by decreasing the degree of parallelism and selecting a different length for the filter coefficients. Although we thoroughly reviewed the literature on LMS-based equalizers, recent works have yet to address the issue of digital equalization with UWB signals in the context of the algorithms we examined. This subtly underscores the novelty of our approach and the significance of our findings.

FFA has reduced resource utilization, allowing for an optimal FPGA implementation, and showing better hardware design flexibility than the Fast Block LMS. Generally, the M-point FFT implementation for the Fast Block LMS is investigated as a fully parallel architecture with M inputs and M outputs. The M-point FFT can also be implemented by a hybrid architecture that involves a number of inputs and outputs less than M, but this solution is not easily implemented due to the increase in the complexity of the hardware design. These problems highlight the advantages of PLMS in terms of hardware resources and design complexity.

VI. CONCLUSION

In this paper, a novel parallel architecture for digital UWB equalizer and an optimized Block LMS have been proposed. The UWB equalizer, namely PILA, shows high design flexibility due to the FFA that allows for the making of flexible parallel architectures. PILA is the parallel version of ILA, an architecture widely used for the equalization process, and allows for the digital signal processing for UWB signals by a parallel filter and the PLMS algorithm whose architectures are optimized by FFA. PLMS is the main block of PILA and it is an optimized version, obtained through mathematical manipulations of the algorithm, of the Block LMS. The computational complexity between the PLMS and the Fast Block LMS, the improved version of the Block LMS introduced in the literature, is analyzed considering complex data. As result, the analysis shows that the PLMS algorithm exhibits a lower computational complexity. Despite a thorough review of the literature on LMS-based equalizers, none of the recent work has addressed the problem of digital equalization operating with UWB signals considering the algorithms we reviewed. This emphasizes the innovation of our approach and the results obtained. PLMS is also validated by FPGA implementation considering a 16-QAM signal with a bit-rate of 3.2 [Gbps], and a sampling frequency of 1.6 [GHz] for the RF converters. The results show that a generic UWB system involves high resource utilization that can be reduced by using several algorithms such as the FFA.

Future research will focus on exploring advanced algorithms to reduce complexity, improve efficiency, and achieve less resource-intensive implementations. Emphasis will also be placed on ensuring scalability and flexibility for diverse applications, and conducting extensive real-world testing to validate and refine performance.

ACKNOWLEDGMENT

The authors would like to thank Advanced Micro Devices (AMD) Inc., for providing the FPGA hardware and software tools with the AMD-Xilinx University Program.

REFERENCES

[1] B. Farley, J. McGrath, and C. Erdmann, "An all-programmable 16-nm RFSoC for digital-RF communications," *IEEE Micro*, vol. 38, no. 2, pp. 61–71, Mar. 2018.

[2] R. Giuliano, "From 5G-advanced to 6G in 2030: New services, 3GPP advances, and enabling technologies," *IEEE Access*, vol. 12, pp. 63238–63270, 2024.

[3] D. Giardino, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, A. Nannarelli, M. Re, and S. Spanò, "M-PSK demodulator with joint carrier and timing recovery," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 6, pp. 1912–1916, Jun. 2021.

[4] D. J. Daniels, *Ground Penetrating Radar*, vol. 1. U.K.: IET, 2004.

[5] A. Florio, N. Bnilam, C. Talarico, P. Crosta, G. Avitabile, and G. Coviello, "LEO-based coarse positioning through angle-of-arrival estimation of signals of opportunity," *IEEE Access*, vol. 12, pp. 17446–17459, 2024.

[6] L. Anitori, A. de Jong, and F. Nennie, "FMCW radar for life-sign detection," in *Proc. IEEE Radar Conf.*, May 2009, pp. 1–6.

[7] G. Ciccarella, R. Giuliano, F. Mazzenga, F. Vatalaro, and A. Vizzarri, "Edge cloud computing in telecommunications: Case studies on performance improvement and TCO saving," in *Proc. 4th Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Jun. 2019, pp. 113–120.

[8] B. Wang, H. Song, W. Rhee, and Z. Wang, "Overview of ultra-wideband transceivers—System architectures and applications," *Tsinghua Sci. Technol.*, vol. 27, no. 3, pp. 481–494, Jun. 2022.

[9] E. Grayver, B. Davidson, and C. Lee, "Ultrawideband (5 GHz) mixed-signal front end and recorder for software defined radio," in *Proc. IEEE Aerosp. Conf.*, Mar. 2018, pp. 1–8.

[10] M. Verplaetse, T. De Keulenaer, A. Vyncke, R. Pierco, R. Vaernewyck, J. Van Kerrebrouck, J. Bauwelinck, and G. Torfs, "Adaptive transmit-side equalization for serial electrical interconnects at 100 Gb/s using duobinary," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 7, pp. 1865–1876, Jul. 2017.

[11] Y.-H. Song and S. Palermo, "A 6-Gbit/s hybrid voltage-mode transmitter with current-mode equalization in 90-nm CMOS," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 8, pp. 491–495, Aug. 2012.

[12] E. Abd-Elrady and L. Gan, "Adaptive predistortion of Hammerstein systems based on indirect learning architecture and prediction error method," in *Proc. Int. Conf. Signals Electron. Syst.*, 2008, pp. 389–392.

[13] Q. You, L. Gu, and J. Liu, "Improved variable step-size lms for digital predistortion in wideband power amplifiers," in *Proc. 8th Int. Conf. Electron. Inf. Emergency Commun. (ICEIEC)*, 2018, pp. 116–119.

[14] S. S. Haykin, *Adaptive Filter Theory*. London, U.K.: Pearson, 2005.

[15] A. Florio, G. Avitabile, C. Talarico, and G. Coviello, "A reconfigurable full-digital architecture for angle of arrival estimation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 71, no. 3, pp. 1443–1455, Mar. 2024.

[16] L. Canese, G. C. Cardarilli, L. D. Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Efficient digital implementation of a multirate-based variable fractional delay filter for wideband beamforming," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 6, pp. 2231–2235, Jan. 2023.

[17] C. Lin, J. Zhang, and B. Shao, "A multi-gigabit parallel demodulator and its FPGA implementation," *IEICE Trans. Fundamentals Electron., Commun. Comput. Sci.*, vol. 95, no. 8, pp. 1412–1415, 2012.

[18] X. Hao, Q. Wu, Z. Wang, and C. Lin, "Parallel timing synchronization algorithm and its implementation in high speed wireless communication systems," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2019, pp. 1–6.

[19] Y. Liu, R. Ranganathan, M. T. Hunter, and W. B. Mikhael, "Complex adaptive LMS algorithm employing the conjugate gradient principle for channel estimation and equalization," *Circuits, Syst., Signal Process.*, vol. 31, no. 3, pp. 1067–1087, Jun. 2012.

[20] G. Clark, S. Mitra, and S. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-29, no. 3, pp. 744–752, Jun. 1981.

[21] C. Eun and E. J. Powers, "A new Volterra predistorter based on the indirect learning architecture," *IEEE Trans. Signal Process.*, vol. 45, no. 1, pp. 223–227, Jan. 1997.

[22] J. I. Acha, "Computational structures for fast implementation of L-path and L-block digital filters," *IEEE Trans. Circuits Syst.*, vol. 36, no. 6, pp. 805–812, Jun. 1989.

[23] I.-S. Lin and S. K. Mitra, "Overlapped block digital filtering," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 8, pp. 586–596, Aug. 1996.

[24] E. Ferrara, "Fast implementations of LMS adaptive filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-28, no. 4, pp. 474–475, Aug. 1980.

[25] J. Harmon and S. G. Wilson, "Iterative approach to the indirect learning architecture for baseband digital predistortion," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2010, pp. 1–5.

[26] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. Hoboken, NJ, USA: Wiley, 2009.

[27] Z.-J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast nonrecursive filtering," *IEEE Trans. Signal Process.*, vol. 39, no. 6, pp. 1322–1332, Jun. 1991.

[28] G. C. Cardarilli, L. D. Nunzio, R. Fazzolari, D. Giardino, M. Matta, M. Re, S. Spano, and L. Simone, "Efficient FPGA implementation of high speed digital delay for wideband beamforming using parallel architectures," *Bull. Electr. Eng. Informat.*, vol. 8, no. 2, pp. 422–427, Jun. 2019.

[29] S. K. Mitra and Y. Kuo, *Digital Signal Processing: A Computer-Based Approach*, vol. 2. New York, NY, USA: McGraw-Hill, 2006.

[30] Z. J. Mou and P. Duhamel, "Fast FIR filtering: Algorithms and implementations," *Signal Process.*, vol. 13, no. 4, pp. 377–384, Dec. 1987.

[31] N. H. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. London, U.K.: Pearson, 2015.

[32] Y.-C. Tsao and K. Choi, "Area-efficient parallel FIR digital filter structures for symmetric convolutions based on fast FIR algorithm," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 2, pp. 366–371, Feb. 2012.

[33] J. Tian, G. Li, and Q. Li, "Hardware-efficient parallel structures for linear-phase FIR digital filter," in *Proc. IEEE 56th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2013, pp. 995–998.

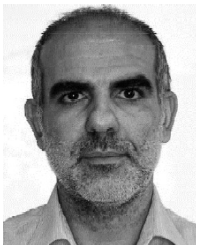
[34] S. Winograd, *Arithmetic Complexity of Computations*, vol. 33. Philadelphia, PA, USA: SIAM, 1980.

LORENZO CANESE received the M.S. degree (summa cum laude) in electronic engineering from the Tor Vergata University of Rome, in 2020, where he is currently pursuing the Ph.D. degree in electronic engineering. His research interests include machine learning, swarm intelligence, ASIC/FPGA hardware design, and the design and digital implementation of multi-agent reinforcement learning algorithms.

GIAN CARLO CARDARILLI (Life Member, IEEE) was born in Rome, Italy. He received the Laurea degree (summa cum laude) from Sapienza Universitã di Roma, in 1981. Since 1984, he has been with the Tor Vergata University of Rome, where he is currently a Full Professor of digital electronics and electronics for communication systems. From 1992 to 1994, he was with the University of L'Aquila. From 1987 to 1988, he was with the Circuits and Systems Team, EPFL, Lausanne, Switzerland. He works in the field of computer arithmetic and its application to the design of fast signal digital processors. He has also regular cooperation with companies, such as Alcatel Alenia Space, Italy; STM, Agrate Brianza, Italy; Micron, Italy; and Selex S.I., Italy. His research interests include VLSI architectures for signal processing and IC design and the design of special architectures for signal processing. In this field, he has published more than 160 papers in international journals and conferences.

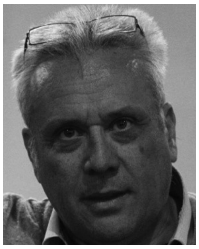
RICCARDO LA CESA received the M.S. degree (summa cum laude) in electronic engineering from the Tor Vergata University of Rome, Italy, in 2023, where he is currently pursuing the Ph.D. degree in electronic engineering. His research interests include digital signal processing (DSP) for wideband signals, RF sampling, telecommunications, RADAR, and the design and digital implementation of MIMO systems focused on calibration and synchronization techniques of phased array systems.

LUCA DI NUNZIO (Member, IEEE) received the master's degree (summa cum laude) in electronics engineering and the Ph.D. degree in systems and technologies for space from the Tor Vergata University of Rome, in 2006 and 2010, respectively. He is currently an Adjunct Professor with the Digital Electronics Laboratory, Tor Vergata University of Rome, and an Adjunct Professor of digital electronics with Guglielmo Marconi University. He has experience with several companies in the fields of electronics and communications. His research interests include reconfigurable computing, communication circuits, digital signal processing, and machine learning.



ROCCO FAZZOLARI received the master's degree in electronic engineering and the Ph.D. degree in space systems and technologies from the Tor Vergata University of Rome, Italy, in May 2009 and June 2013, respectively. He is currently a Postdoctoral Fellow and an Assistant Professor with the Department of Electronic Engineering, Tor Vergata University of Rome. He works on hardware implementation of high-speed systems for digital signals processing, machine learning, the array of wireless sensor networks, and systems for data analysis of acoustic emission (AE) sensors (based on ultrasonic waves).

DANIELE GIARDINO received the M.S. degree in electronic engineering and the Ph.D. degree in space systems and technologies from the Tor Vergata University of Rome, Italy, in 2017 and 2021, respectively. His topics regard digital signal processing for wideband signals, RF sampling, telecommunications, and machine learning. In specific, he is focused on the digital implementation of MIMO systems for wideband signals sampled by RF sampling. Currently, he is a DSP and Hardware Designer with IT Systems Srl Company.



MARCO RE (Member, IEEE) received the Ph.D. degree in microelectronics. He is currently an Associate Professor with the University of Rome Tor Vergata, where he teaches digital electronics and hardware architectures for DSP. He is also the Director of the Master of Audio Engineering with the Department of Electronic Engineering, Tor Vergata University of Rome. He was awarded two NATO fellowships with the Cadence Berkeley Laboratories, University of California at Berkeley, as a Visiting Scientist. He has been awarded the Otto Moensted Fellowship as a Visiting Professor with the Technical University of Denmark. He collaborates in many research projects with different companies in the field of DSP architectures and algorithms. He is the author of about 200 papers in international journals and international conferences. His research interests include low-power DSP algorithms architectures, hardware-software codesign, fuzzy logic and neural hardware architectures, low-power digital implementations based on non-traditional number systems, computer arithmetic, and CAD tools for DSP. He is a member of the Audio Engineering Society (AES).

SERGIO SPANÒ received the bachelor's, master's, and Ph.D. degrees (summa cum laude) in electronic engineering from the Tor Vergata University of Rome, in 2015 and 2018, respectively. Since 2022, he has been an Adjunct Professor with the Tor Vergata University of Rome, where he is currently a Postdoctoral Research Fellow. He is also an Adjunct Professor with the Guglielmo Marconi University of Rome. He has several industrial work experiences in the fields of space and telecommunications. His research interests include digital signal processing, machine learning, the IoT, the development of telecommunication systems, and the implementation of machine learning accelerators for embedded and low-power systems.

Open Access funding provided by 'Università degli Studi di Roma "Tor Vergata"' within the CRUI CARE Agreement