High-Throughput and Configurable Preprocessor for ICA-based Self-interference Cancellation

Hsi-Hung Lu, Chung-An Shen, Mohammed E. Fouda, and Ahmed M. Eltawil

Abstract—Independent component analysis (ICA) has been used in many applications, including self-interference cancellation in in-band full-duplex wireless communication systems. This paper presents a high-throughput and highly efficient configurable preprocessor for the ICA algorithm. The proposed ICA preprocessor has three major components for centering, for computing the covariance matrix, and for eigenvalue decomposition (EVD). The circuit structures and operational flows for these components are designed both in individual and in joint sense. Specifically, the proposed preprocessor is based on a high-performance matrix multiplication array (MMA) that is presented in this paper. The proposed MMA architecture uses time-multiplexed processing so that the efficiency of hardware utilization is greatly enhanced. Furthermore, the novel processing flow of the proposed preprocessor is highly optimized, so that the centering, the calculation of the covariance matrix, and EVD are conducted in parallel and are pipelined. Thus, the processing throughput is maximized while the required number of hardware elements can be minimized. The proposed ICA preprocessor is designed and implemented with a circuit design flow, and performance estimates based on the post-layout evaluations are presented in this paper. This paper shows that the proposed preprocessor achieves a throughput of 40.7 kMatrices per second for a complexity of 73.3 kGE. Compared with prior work, the proposed preprocessor achieves the highest processing throughput and best efficiency.

I. INTRODUCTION

The blind source separation (BSS) problem lies in the separation of a mixed signal which is a combination of many independent sources. For instance, separating the voice signal that is the mixed of many people speaking at the same time is the most well-known BSS problem. In the BSS problem, the mixed signal is constructed by mixing independent source signals through an unknown channel. Without prior knowledge of the mixing channel, independent source signals can be separated only by observing the mixed received signals [1].

The independent component analysis (ICA) algorithm is the most widely used approach for separating source signals in BSS problems [1]–[5]. The ICA algorithm separates the mixed signal by taking advantage of the statistical independence and non-Gaussian properties of source signals [1]. The ICA algorithm has been applied in many applications. For example, it is proposed in [4], [5] that the self interference cancellation (SIC) in the In-Band Full-Duplex (IBFD) wireless communication system can be considered as a BSS problem. The ICA algorithm is applied to separate the self-interference signal and the signal of interest this system. Furthermore, the ICA algorithm is applied to separate multiple superimposed signals in order to identify the cyclic features of each transmitted signal on a cognitive radio (CR) system [6]. Recently, it is proposed in [7], [8] that the ICA is used in a temporal-critical industrial internet of things (IoT) application for anomaly detection or identification. The source signals are expected to be extracted with real-time in these various using scenarios.

The ICA algorithm has two main stages: the preprocessing stage and the iterative stage [1], [3], [9]. ICA preprocessing aims to whiten the received signal so that the correlation between the signals is reduced. The whitened received signal is sent to the iterative stage where the demixing matrix is estimated and the mixed signals are separated. The preprocessing stage is an essential step in ICA, as it enhances the quality of the separated signals and accelerates the convergence in the iterative stage. However, ICA preprocessing requires a number of highly complex vector and matrix computations, including the calculation of the covariance matrix and the eigenvalue decomposition (EVD). Therefore, designing an efficient preprocessor is a great challenge in realizing efficient ICA circuits and systems.

Several VLSI architecture designs for ICA preprocessors have been reported in the literature. Reference [10] proposed an ICA-based signal separation architecture for a biomedical application. The preprocessor in this design is based on a very serial processing flow, which results in a very low processing speed. Reference [11] designed a low-power ICA processor and improved the processing throughput of the EVD in the ICA preprocessor. However, the performance of this design is limited by the low operating frequency. Furthermore, [12] and [13] reported that they had improved the speed of ICA architectures by increasing the parallelism. However, the processing flow of the preprocessor is still sequential in nature and thus, cannot achieve high throughput. For instance, a low-complexity singular value decomposition (SVD) engine with high processing latency is proposed in [14]. The authors in [15] proposed a high-performance SVD architecture with multiple processing elements (PEs). In short, the ICA preprocessors described in the literature support only real-valued signals and operate at low processing speeds. However, other applications require higher data rates such as modern wireless communication systems that could have a data rate of hundreds of megabits per second [16]–[18] with complex-valued signals or time-sensitive applications that require ultra-low latency such as IoT require strict real-time processing [7], [8]. Therefore, it is crucial to design a high-throughput ICA preprocessor.
that can be used in modern wireless communication or IoT systems.

This work presents a high-throughput and highly efficient ICA preprocessor. The main contributions of this paper are summarized as follows:

- This paper presents an ICA preprocessor containing centering, covariance, and EVD units. The proposed preprocessor supports complex-valued signals whereas the number of received signals and sample lengths are configurable. The preprocessor has a highly improved processing flow, so that the hardware utilization is maximized. Therefore, the proposed preprocessor achieves a high processing throughput with the minimum consumption of hardware resources.
- A high-performance matrix multiplication array (MMA) is presented in this paper. The proposed MMA architecture uses time-multiplexed processing, so that the hardware utilization is greatly enhanced. Therefore, compared to prior designs, the proposed MMA unit achieves high processing throughput with low area complexity.
- A low-latency centering unit and covariance unit based on the proposed MMA are presented. The operating flow is highly improved, so that the centering and calculating the covariance matrix can be performed in parallel.
- A highly efficient EVD unit is proposed in this paper. The designed EVD engine is optimized for processing Hermitian matrices and has a low processing latency.
- The experimental results show that the proposed preprocessor outperforms the prior designs. It achieves a high throughput with a minimized area complexity. The proposed design also achieves the best efficiency compared to the prior ICA preprocessor.

The rest of this paper is organized as follows. The background of ICA and preprocessing for ICA are introduced in Section II. The proposed centering and covariance units are presented in Section III, and the proposed EVD unit is described in Section IV. The experimental results and comparisons with prior designs are given in Section V. The conclusions are in Section VI.

II. BACKGROUND AND RELATED WORK

This section introduces the basic concepts of the ICA. The state-of-the-art ICA designs with focus on preprocessors are also reviewed.

A. The Basic Concept of ICA and the ICA Preprocessing

Considering the BSS problem, the received signal is constructed by mixing independent source signals through an unknown channel. Without prior knowledge of the mixing channel, independent source signals can be separated only by observing the mixed received signals \( Y \). The ICA algorithm is the most widely used approach for separating source signals in BSS problems [1]–[3]. For example, it is proposed in [4] that SIC in IBFD systems can be considered as a BSS problem where the ICA algorithm is applied to separate the self-interference signal and the signal-of-interest in the received signal.

In the ICA algorithm, the mixed signals are separated by utilizing the statistical independence and non-Gaussianity of the source signals. Furthermore, the number of observed signals must be equal to the number of source signals, so in the generic ICA representation the mixed received signals and the source signals can be expressed in matrix form as follows:

\[
Y = \begin{bmatrix}
Y^1 \\
Y^2 \\
\vdots \\
Y^N
\end{bmatrix} = \begin{bmatrix}
h_{11} & h_{12} & \cdots & h_{1N} \\
h_{21} & h_{22} & \cdots & h_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
h_{N1} & h_{N2} & \cdots & h_{NN}
\end{bmatrix}\begin{bmatrix}
X^1 \\
X^2 \\
\vdots \\
X^N
\end{bmatrix} = HX
\]

(1)

where \( Y \) and \( X \) are the mixed received signals and the source signals, respectively. \( Y^1 \) and \( X^1 \) are both row vectors with \( M \) elements where \( M \) is the sample length. The matrix \( H \) has size \( N \times N \), where \( N \) is the number of signals. After receiving the mixed received signals \( Y \), the ICA estimates a demixing matrix \( W \) and uses it to separate the received signals. The separated signals \( \hat{X} \) can be defined as follows:

\[
\hat{X} = \begin{bmatrix}
\hat{X}^1 \\
\hat{X}^2 \\
\vdots \\
\hat{X}^N
\end{bmatrix} = \begin{bmatrix}
w_{11} & w_{12} & \cdots & w_{1N} \\
w_{21} & w_{22} & \cdots & w_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
w_{N1} & w_{N2} & \cdots & w_{NN}
\end{bmatrix}\begin{bmatrix}
Y^1 \\
Y^2 \\
\vdots \\
Y^N
\end{bmatrix} = WY
\]

(2)

where the demixing matrix \( W \) is the inverse of the mixing matrix, assuming the estimation is perfect.

In ICA, the demixing matrix is estimated by searching for orthogonal vectors iteratively. Works in the literature [3] show that including a preprocessing stage before searching for the orthogonal vectors is crucial for reducing the number of iterations required and for improving the quality of the estimated demixing matrix.

The main purpose of ICA preprocessing is to whiten the received signal and the operational flow is shown in Fig. 1. In the first step of ICA preprocessing, the received signals are centered by subtracting the mean of signals, which is shown as follows:

\[
\hat{Y}^i = Y^i - E\{Y^i\}
\]

(3)
Specifically, \( \bar{Y}^i \) and \( E\{Y^i\} \) are the \( i \)-th centered mixed received signal and its mean, respectively. The second step of preprocessing stage is whitening where the centered signals are whitened to have unit variance. In order to whiten the centered signals, the covariance matrix is calculated firstly as follows:

\[
Y_c = E\{\bar{Y}\bar{Y}^H\}
\]  

(4)

where \( Y_c \) is the covariance matrix of the centered signals, which is also a \( N \times N \) Hermitian matrix. In the following, the EVD is performed on the covariance matrix as follows:

\[
Y_c = EDE^H
\]  

(5)

where \( D \) is a diagonal matrix composed of \( N \) eigenvalues and \( E \) is an orthogonal matrix with \( N \) eigenvectors. The matrices \( D \) and \( E \) are applied to whiten \( \bar{Y} \). It can be seen that a number of matrix and vector operations are required in the ICA preprocessing and thus, the complexity is high and the throughput is degraded. Therefore, it is a great challenge to design a high-throughput and highly efficient configurable ICA preprocessor.

B. Related Work

The VLSI architectures for the ICA and ICA preprocessor are reported in the literature. Reference [10] proposes a real-valued signal separation architecture based on the FastICA algorithm. The preprocessor in this architecture has a centering unit, a covariance unit, and an EVD engine. Since the units in this preprocessor operate in a very sequential manner, the latency is extended. Specifically, the centering unit and covariance unit sequentially access the same memory several times, leading to very long delays. According to our analysis, for \( N \) complex-valued received signals and a sample length of \( M \), a total of \( 8 \times N \times M + 6 \times M \times N \times N \) cycles are consumed by the centering and covariance units of [10]. Furthermore, to increase the throughput, parallelism is introduced in the preprocessor in [19]. A MMA based on a systolic array structure is proposed in this design for calculating the covariance matrix. This MMA uses a real-valued matrix with a size of \( 3 \times 64 \). It takes three cycles to multiply a column vector and a row vector. However, the utilization of the MMA is not optimized, especially for complex-valued matrices. In addition, in [12] and [13], the preprocessor has to share computing units. The operating flow of these designs was not fully optimized, resulting in excessive processing delays.

Moreover, the EVD engine is one of the core units in the ICA preprocessor and has a high degree of complexity. An EVD unit in the preprocessor based on the approximate Jacobi algorithm was presented in [11]. The aim was to realize a low-power ICA system. However, since the critical path of the EVD unit was very long, the operating frequency was very low and the processing throughput was heavily degraded. A configurable SVD unit based on the CORDIC operation and folded systolic array structure was described in [14]. Although these authors produced a low-complexity circuit, the processing had high latency and was slow. Furthermore, a sorting network was proposed in [15] to enhance the efficiency of memory accesses in the EVD architecture. However, a matrix multiplier was employed in this design to perform matrix rotations, which resulted in high area complexity.

Modern wireless communication systems operate at a data rate of hundreds of megabits per second [16]–[18] with complex-valued signals. Furthermore, IoT applications usually require strict real-time processing [7], [8]. Therefore, it is crucial to design a high-throughput ICA preprocessor for these systems. No relevant improvements for the operational flow and architecture of ICA preprocessors have been reported in literature. To achieve a high-throughput and highly efficient ICA preprocessor, in this work, we design an architecture by jointly considering the operational flow and circuit structure for the centering, covariance, and EVD units. The proposed centering and covariance units are operated in parallel, so that the processing throughput is much higher. Furthermore, the covariance unit is based on a novel highly efficient MMA circuit. To reduce the amount of computation, a novel diagonalization process for a Hermitian matrix was designed for the EVD.

III. THE PROPOSED CENTERING AND COVARIANCE UNITS

A. The Proposed Efficient MMA Architecture

The centering and covariance units in the preprocessor use the proposed efficient complex-valued MMA (ECMMA). The architecture is shown in Fig. 2(a) for a \( 3 \times 64 \) matrix. The proposed ECMMA contains three complex-valued multipliers and one column of PEs. These PEs process the real and imaginary parts of the covariance matrix. Note that the MMA reported in [19] is a \( 3 \times 3 \) array. Figure 2(b) shows the operational flows for a \( 3 \times 64 \) matrix for the proposed MMA and for the MMA reported in [19]. Each PE in the proposed MMA is utilized to store the multiply-and-accumulate (MAC) for different columns of data using time-multiplexing. Moreover,
if there are submatrices and each submatrix is processed independently. 

\[ Y \]

\( \text{denotes the } i \text{ matrix } \)

\( Y \)

\( \text{highlighted with solid lines, such as example, the lower-triangular elements highlighted in red in generated at different stages by different approaches. For has three different parts, and the elements in each part are those highlighted with dotted lines, are decomposed as another centered signals, such as \( \bar{Y} \) concurrently with the operation of the centering unit while so that the elements highlighted with solid lines are computed \( \bar{Y} \) computed once the centered signals \( \bar{Y} \) is consumed by the PE in the accumulation of the real and imaginary parts of the element. Due to this time-multiplexed utilization of the PE, the efficiency of the proposed MMA is much higher.

Table 1 compares the number of hardware elements in the proposed efficient MMA and the MMA in [19] for an \( m \times n \) complex-valued matrix. This table shows that, although additional multiplexers are needed, the number of adders is much lower. Moreover, the number of adders required for the MMA of [19] increases exponentially with the size of the matrix, whereas the number of adders for the proposed efficient MMA increases linearly.

**B. The Proposed Centering and Covariance Units**

As mentioned in Section III-A, the prior centering and covariance units need to sequentially access the memory multiple times and thus, they incur excessive delays. Furthermore, the centering and covariance matrix operations are performed serially in these designs, so that the processing throughput is degraded. To achieve high-throughput preprocessing for the ICA, this work optimizes the operational flow so that the centering unit and the covariance unit run in parallel. Furthermore, the covariance unit is based on the proposed efficient MMA architecture, which strikes a balance between high speed and low complexity. If there are \( N \) received signals, the covariance matrix has a size of \( N \times N \) and must be a Hermitian matrix. Figure 3 is an example of an \( 8 \times 8 \) covariance matrix \( Y_c \) assuming eight received signals. The notation \( \bar{Y}_i \) denotes the \( i \)-th row vector in the centered received signals \( \bar{Y} \).

Because it is a Hermitian matrix, the covariance matrix has three different parts, and the elements in each part are generated at different stages by different approaches. For example, the lower-triangular elements highlighted in red in Fig. 3 do not need to be calculated. Furthermore, the elements highlighted with solid lines, such as \( Y_{11}^{\text{c}}, Y_{12}^{\text{c}}, \) and \( Y_{22}^{\text{c}} \), can be computed once the centered signals \( \bar{Y}_1 \) and \( \bar{Y}_2 \) are generated. Therefore, in this design, a novel processing flow is proposed so that the elements highlighted with solid lines are computed concurrently with the operation of the centering unit while other centered signals, such as \( \bar{Y}_3 \) and \( \bar{Y}_4 \), are being generated. Finally, the remaining upper-triangular elements, such as those highlighted with dotted lines, are decomposed as \( 2 \times 2 \) submatrices and each submatrix is processed independently. Note that the proposed processing flow is applied to any covariance matrix with an even number of received signals. If there are \( N \) received signals, then the number of \( 2 \times 2 \) submatrices is \( (N^2 - 2 \times N)/8 \). In other words, this design based on the proposed processing flow contains a high degree of regularity and is highly reconfigurable.

The centering and covariance units were designed and implemented based on the proposed processing flow. The architecture of these units is presented in Fig. 4. Figure 5 is the timing diagram.

Figure 4 shows that, for a \( 2 \times 2 \) matrix, the centering unit has four centering components (CCs) and the covariance unit has one diagonal submatrix component (DSC) and one ECMMA. The DSC computes those elements shown in solid lines in Fig. 3 whereas the ECMMA unit computes the elements shown in dashed lines. Furthermore, the four CCs compute two complex-valued signals concurrently. Each CC sequentially accumulates data from the input signal and calculates the average value. The same signal is then read by the next CC in the sequence and the average value is subtracted to give the centered signal. The centered signal is output in sequence and is both written into the memory and sent to the DSC in the covariance unit. Therefore, based on the proposed processing flow, the centering unit outputs the centered data in sequence while the DSC in the covariance unit calculates elements of the covariance matrix at the same time. After all the received signals have been centered and the elements in each L-shaped region marked by a solid line in Fig. 5 have been calculated by the DSC, the elements in the squares marked with a dotted line continue to be calculated through the ECMMA in the covariance unit. The ECMMA unit executes \( (N^2 - 2 \times N)/8 \) times to calculate \( (N^2 - 2 \times N)/8 \) submatrices in sequence, if there are \( N \) received signals. Note that, based on the proposed flow and architecture, the centering operation and the covariance operation overlap in time and the elements of covariance matrix are generated as soon as possible. Therefore, the throughput of the overall architecture is greatly enhanced.

Figure 5 is a detailed timing diagram for the proposed centering and covariance units. There are \( N \) received signals, and the sample length is \( M \). After taking \( (N \times M)/2 \) cycles to write received signals into the received memory, row vectors \( Y_1 \) and \( Y_2 \) are read by the four CCs in the centering

\[
Y_c = \begin{bmatrix}
Y_1^c \\
Y_2^c \\
\end{bmatrix}
\]
unit in sequence to calculate the average values. \( M \) cycles are consumed in the calculation of the average values. The vectors \( \mathbf{Y}^1 \) and \( \mathbf{Y}^2 \) are read again and the average values are subtracted to generate the centered signals of \( \tilde{\mathbf{Y}}^1 \) and \( \tilde{\mathbf{Y}}^2 \). This also takes \( M \) cycles. The centered signals are saved into the memory and sent to the DSC of the covariance unit at the same time. Therefore, while the centering unit is being processed into the memory and sent to the DSC of the covariance matrix. It can be observed from Fig. 5 that the centering operations are being performed on all \( \tilde{\mathbf{Y}}_{11} \) and \( \tilde{\mathbf{Y}}_{22} \) for the covariance matrix. Subsequently, the centering unit reads vectors \( \mathbf{Y}^3 \) and \( \mathbf{Y}^4 \) sequentially and performs the accumulation. Since the centering unit does not write any data to the memory at this time, the DSC reads \( \mathbf{Y}^1 \), \( \mathbf{Y}^2 \), \( \mathbf{Y}^1 \), and \( \mathbf{Y}^2 \) from the memory and calculates \( \tilde{\mathbf{Y}}_{12} \) for the covariance matrix. It can be observed from Fig. 5 that while the centering operations are being performed on all \( N \) complex signals, \( (N/2) \times 3 \) out of \((1+N) \times N)/2\) elements in the covariance matrix (i.e., those in the L-shaped region marked by a solid line in Fig. 3) are computed at the same time. The remaining covariance elements are decomposed into \((N^2 - 2 \times N)/8\) \( 2 \times 2 \) submatrices and are processed by the ECMMA unit. Figure 5 shows that \( 2 \times M \) cycles are consumed by the ECMMA unit to calculate four elements concurrently.

To maximize the throughput, the centering unit and the DSC can start to process the next \( N \times M \) complex matrix while the ECMMA unit is processing. Figure 5 shows that the processing of a \( 2 \times 2 \) ECMMA for the first matrix and the processing of the centering and covariance units overlap for the second matrix. Therefore, the covariance matrix is generated in every \((N^2 \times M - 2 \times N \times M)/4\) cycles assuming \( N \geq 8 \). For example, the covariance matrix is computed in every 6144 cycles for an \( 8 \times 512 \) complex-valued matrix.

IV. PROPOSED EVD UNIT

A. Proposed EVD Design

The most commonly used approach for EVD is based on the Jacobi algorithm [11], [15]. A series of rotation transformations is exploited to find the eigenvalues and eigenvectors of a matrix. Specifically, in the Jacobi algorithm, an \( N \times N \) matrix is decomposed into multiple \( 2 \times 2 \) submatrices and the submatrices that are on the diagonal of the matrix (known as diagonal submatrices) are diagonalized. The other submatrices (known as non-diagonal submatrices) are rotated according to the angles of the diagonalization process. This diagonalization-rotation process is performed iteratively.

In the ICA preprocessing, the input to the EVD unit is the covariance matrix, which must be a Hermitian matrix. A Hermitian matrix has unique characteristics that can be exploited by the EVD unit to achieve an optimized balance between throughput and complexity. The proposed EVD unit fully utilizes the Hermitian properties in the design of the processing sequence and the architecture for the diagonalization and rotation of a \( 2 \times 2 \) complex-valued matrix. The proposed processing sequence is presented in Fig. 6. It requires five stages for the EVD diagonalization of a \( 2 \times 2 \) submatrix. Note that this \( 2 \times 2 \) submatrix is the diagonal submatrix of the covariance matrix, so it must be Hermitian. In other words, the non-diagonal elements \( p_{12} \) and \( p_{21} \) in Fig. 6 are complex conjugates of each other. The processing sequence is based on calculating the angle and the matrix rotation. Stages 1 and 2 eliminate the imaginary parts of \( p_{12} \) and \( p_{21} \) in the submatrix. The rotation angle \( \theta \) is computed accordingly. In the following stage 3, the corresponding rotation angle \( \phi \) is computed. Finally, stages 4 and 5 eliminate the non-diagonal elements of the submatrix, resulting in a diagonal \( 2 \times 2 \) submatrix.

Note that, since the \( 2 \times 2 \) submatrix is a Hermitian
matrix, the diagonal elements have real values. As a result, all the elements in the submatrix will be real after stage 2. Furthermore, since the non-diagonal elements are complex conjugates of each other, the rotation angle θ needs to be computed only once.

Figure 7 presents the architecture of the EVD unit proposed to process the flow shown in Fig. 6. There are CORDIC elements in each stage, and it is architected as a 10-cycle pipeline. In this figure, DR and DI are the real and imaginary parts of the eigenvalues and ER and EI are the real and imaginary parts of the eigenvectors, respectively. A superscript represents the corresponding position in the submatrices. D is initialized to the value of the submatrix, and E is initialized to the identity matrix. The diagonal 2 × 2 submatrices are sent into the EVD unit one by one. In stage 1, the first CORDIC operates in vectoring mode and obtains the right-hand side (RHS) rotation direction. The corresponding RHS rotations are performed by the other three CORDICs based on the direction provided by the first CORDIC. Thus, the imaginary part of D_{12} is eliminated. Since D_{12} and D_{21} are complex conjugates of each other, in stage 2, two CORDICs perform the left-hand side (LHS) rotation on the submatrix using the same rotation direction as stage 1. The imaginary part of D_{21} is eliminated in stage 2 and the diagonal 2 × 2 submatrix is transformed into a real symmetric matrix. In stage 3, the CORDIC calculates the rotation angle to eliminate the non-diagonal elements of the submatrix for the subsequent stages. In stage 4, the first CORDIC performs a LHS rotation with the rotation angle calculated in stage 3 and provides the rotation direction for the other three CORDICs. In stage 5, eight CORDICs perform the RHS rotation to eliminate the non-diagonal elements based on the rotation direction generated by stage 4. Therefore, each diagonal 2 × 2 submatrix can be converted into a real-valued diagonal matrix. After all diagonal 2 × 2 submatrices have been so converted, the non-diagonal 2 × 2 submatrices are input one by one and the corresponding rotation is performed by the EVD unit operating in rotation mode. The rotation of a non-diagonal 2 × 2 submatrix is through the corresponding rotation direction and angle.

**B. EVD Processing Sequence**

The EVD of an N × N matrix is an iterative process involving the diagonalization of 2 × 2 diagonal submatrices and the corresponding rotations of 2 × 2 non-diagonal submatrices. How those 2 × 2 diagonal and non-diagonal submatrices are constructed and the processing order affect the efficiency of the EVD [13]. To accelerate the iterative process, the ordering is usually parallel [14].

The parallel ordering is illustrated in Fig. 8(a) for an 8 × 8 matrix. This matrix is decomposed into seven orderings, each of which has four 2 × 2 diagonal submatrices. The notation (i, j) denotes the 2 × 2 diagonal submatrix with elements at (i, i), (i, j), (j, i), and (j, j) in the 8 × 8 matrix. For example, given the 8 × 8 matrix with elements h_{ij}, where i is the row and j the column, the notation (1, 2) represents the 2 × 2 submatrix of elements [h_{11}, h_{12}; h_{21}, h_{22}]. In each ordering, the diagonalization processes for the 2 × 2 diagonal submatrices are independent of each other, and thus, in this design, each ordering is processed by the proposed EVD engine in a pipelined fashion. However, the sequence for rotating the non-diagonal submatrices also affects the efficiency of the EVD and needs to be optimized.

To reduce the latency and to improve the throughput, a novel processing flow for the EVD is also proposed in this work. Figure 7 shows that the proposed EVD unit has five stages so that five 2 × 2 submatrices are processed concurrently in a pipeline. To achieve this, the ordering for processing different submatrices needs to be specially designed. In particular, each ordering is processed based on the result from the prior ordering. Thus, for example, the calculations for ordering 2 cannot start until all the submatrices of ordering 1 have been output. This results in processing delays between the orderings. To reduce the latency, this design proposes a processing sequence such that the data dependency between two orderings can be eliminated and the efficiency can be significantly enhanced. Consider an 8 × 8 matrix. The matrix that is reconstructed based on the ordering is illustrated in Fig. 8(b). Orderings 1 and 2 are shown as examples. Each 2 × 2 submatrix in the reconstructed matrix of Fig. 8(b) is denoted as S'. The proposed processing sequence for each ordering is shown in Fig. 8(c).

Figure 9 is a timing analysis diagram for EVD for 8 × 8 matrix based on the proposed EVD architecture. The main purpose of the proposed processing sequence is to make the beginning of the processing for each new ordering and the end of the processing for each previous ordering independent. Figure 9 shows that the processing of the last four submatrices for ordering 1 and the processing of the first four submatrices for ordering 2 can be performed in parallel. This greatly enhances the efficiency of the proposed pipelined EVD engine.

Before the calculation of ordering 2, there must be a 40-cycle idle time to avoid a data hazard. However, by sorting the input submatrices, ordering 2 can be calculated before the submatrices of ordering 1 are completely output and no data hazard will occur.

**V. EXPERIMENTAL RESULTS AND COMPARISONS**

**A. Implementation Results**

Based on the proposed operational flow and architecture, an ICA preprocessor was designed and implemented. Note that while the implemented preprocessor was designed for eight received signals with a sample length of 512, the proposed architecture is configurable for different numbers.
of signals and sample lengths. The preprocessor was implemented, synthesized, and placed-and-routed using the TSMC 90 nm process with a nominal power supply voltage of 1.1 V. Estimates for the operating frequency, latency, throughput, and complexity were based on the post-layout simulations. These are summarized in Table II. Figure 10 is the complete timing diagram for the implemented preprocessor. The figure shows that 6144 and 4480 cycles are consumed for the centering and covariance units and for the EVD unit, respectively. Therefore, a throughput of 40.7 kMatrices per second with a latency of 60.95 µs is achieved given the operating frequency of 250 MHz. Furthermore, the area complexity of this preprocessor is 73.3 kGE in terms of two-input NAND gates with 30 kB of on-chip single-port SRAM. The layout is shown in Fig. 11. The distribution of the centering, covariance, and EVD units can be seen.

B. Comparison with Prior Designs

Table III compares the proposed centering and covariance units with prior work [10], [11], [13]. Since the implementation results reported in the literature are for the entire ICA system, the latency and complexity for the centering and
The latency for the prior designs was scaled for a complex-valued \(8 \times 8\) matrix, as shown in Table IV. We again assumed that the architectures were unchanged. The processing time was scaled according to the size and the type of the matrix. For example, the latency of \([10]\) was increased by two times for the EVD of the complex-valued \(8 \times 8\) matrix since the Jacobi algorithm operates on the real and imaginary parts separately. Furthermore, to evaluate the performance of EVD in ICA preprocessing, the throughput was also estimated by assuming that the centering and covariance units proposed in this work are the front stage of the EVD unit, which is referred to as ICA throughput in Table IV. The throughput for \([10]\) was calculated by assuming that the EVD operates by itself, which is referred to as SA throughput. Since the proposed centering and covariance units output a result every 6144 cycles, the throughput is still dominated by the latency of EVD, so ICA throughput was the same as SA throughput for that design. The latency for \([11]\) was doubled for the complex-valued matrix, and SA throughput was calculated accordingly. Since the architecture of \([14]\) was designed for a complex-valued matrix, the scaled latency is the same as the original latency. The scaled latency was doubled and the scaled SA throughput was halved for the designs in \([15]\) and \([20]\) because of the complex-valued EVD. However, the scaled ICA throughput for \([11], [15], [20]\) were limited by the input of the matrix from the centering and covariance units.

The latency for the prior designs was scaled for a complex-valued \(8 \times 512\) matrix. For example, the latency of \([10]\) was increased by two times for the EVD of the complex-valued \(8 \times 512\) matrix since the Jacobi algorithm operates on the real and imaginary parts separately. Moreover, to evaluate the performance of EVD in ICA preprocessing, the throughput was also estimated by assuming that the centering and covariance units proposed in this work are the front stage of the EVD unit, which is referred to as ICA throughput in Table IV. The throughput for \([10]\) was calculated by assuming that the EVD operates by itself, which is referred to as SA throughput. Since the proposed centering and covariance units output a result every 6144 cycles, the throughput is still dominated by the latency of EVD, so ICA throughput was the same as SA throughput for that design. The latency for \([11]\) was doubled for the complex-valued matrix, and SA throughput was calculated accordingly. Since the architecture of \([14]\) was designed for a complex-valued matrix, the scaled latency is the same as the original latency. The scaled latency was doubled and the scaled SA throughput was halved for the designs in \([15]\) and \([20]\) because of the complex-valued EVD. However, the scaled ICA throughput for \([11], [15], [20]\) were limited by the input of the matrix from the centering and covariance units.

It can be seen from Table IV that compared to \([10]\), the proposed EVD unit achieves a higher throughput with reduced complexity due to the highly optimized pipelined processing flow. Furthermore, the architecture in \([11]\) is based on the systolic array structure, so its latency is greatly reduced. While the latency of \([11]\) in terms of clock cycles is lower than the proposed design, the throughput is limited by the very low operating frequency. Moreover, the fully pipelined design in \([14]\) achieves high SA throughput because multiple matrices can be processed at the same time. However, the throughput is degraded for ICA preprocessing. On the other hand, the proposed EVD unit is specifically optimized for ICA, and compared with \([14]\), it strikes the best balance between throughput and complexity. In addition, the architecture of \([15]\) maximizes the throughput by using multiple fully pipelined CORDIC elements and matrix multipliers. Compared to \([15]\), the throughput of the proposed EVD unit is lower while the complexity is also much reduced. The design in \([20]\) leads to a heavily degraded throughput due to the low operating...
TABLE III: Comparisons of the centering and covariance units for different designs of preprocessor

| Parameter                | Original | [10] | Scaled | Original | [11] | Scaled | Original | [13] | Scaled | This work |
|--------------------------|----------|------|--------|----------|------|--------|----------|------|--------|------------|
| Technology (nm)          | 90       | 90   | 90     | 90       | 90   | 90     | 90       | 90   | 90     | 90         |
| Architecture type        | Real     | Real | Real   | Real     | Real | Real   | Real     | Real | Real   | Complex    |
| Arithmetic type          | Fixed    | Fixed| Fixed  | Floating | Floating | Floating | Floating | Floating | Floating | Fixed      |
| Frequency (MHz)          | 100      | 100  | 11     | 11       | 11   | 11     | 100      | 100  | 100    | 250        |
| Matrix type              | Real     | Complex | Real | Complex | Complex | Real | Complex | Complex | Complex | Complex    |
| Matrix size              | 8 × 256  | 8 × 512 | 8 × 256 | 8 × 512 | 8 × 1024 | 8 × 512 | 8 × 512 | 8 × 512 | 8 × 512 | 8 × 512 |
| Latency (cycles)         | 24576a   | 196608b | 24576  | 196608b | 24576a | 24576b | 10757   | 10757 | 10757  |           |
| Throughput (µMatrices/cycle) | 40.7a | 5.1b  | 40.7a  | 5.1b    | 40.7a | 40.7b | 162.76  | 162.76 | 162.76  |           |
| Complexity (kGE)         | 2.76a   | 2.76a | 1.16a  | 1.16a   | 486a  | 486a  | 11.8    |        |        |           |
| Power (mW)               | 16.35a  | n/a  | 0.0816c | n/a      | 65.0c | n/a   | 50.2c   |        |        |           |
| Efficiencyc              | 14.75   | 1.85 | 35.086 | 4.397    | 0.08  | 0.08  | 13.793  |        |        |           |

aLatency and complexity are analyzed proportionally according to the architecture diagram reported in the literature. The latency for the design [10] is analyzed based on Fig. 2 in [10] and the complexity for [10] is analyzed based on Figs. 2 and 18 in [10]. Furthermore, the throughput and complexity for [10] are analyzed based on Table I in [10]. Moreover, the latency for the design [13] is analyzed based on Fig. 3.2 in [13] and the complexity for [13] is analyzed based on Fig. 4.5 in [13].
bLatency and throughput are scaled based on the processing of the complex-valued 8 × 512 matrix. It is assumed that the architecture remains the same and the processing time is scaled for processing larger complex-valued matrices. The memory is scaled for storing a complex-valued 8 × 512 matrix.
cPower consumption is for the entire chip, not just the centering and covariance units.

cEfficiency = Throughput (µMatrices/cycle) / Complexity (kGE).

TABLE IV: Comparison of EVD units for different designs of preprocessor

| Parameter                | Original | [10] | Scaled | Original | [11] | Scaled | Original | [13] | Scaled | This work |
|--------------------------|----------|------|--------|----------|------|--------|----------|------|--------|------------|
| Technology (nm)          | 90       | 90   | 90     | 90       | 90   | 90     | 90       | 90   | 90     | 90         |
| Architecture type        | Real     | Real | Real   | Real     | Real | Real   | Real     | Real | Real   | Complex    |
| Arithmetic type          | Floating | Floating | Fixed | Fixed    | Fixed | Fixed | Floating | Floating | Floating | Fixed      |
| Frequency (MHz)          | 100      | 100  | 11     | 11       | 11   | 11     | 100      | 100  | 100    | 250        |
| Matrix type              | Real     | Complex | Real | Complex | Complex | Real | Complex | Complex | Complex | Complex    |
| Matrix size              | 8 × 8    | 8 × 8  | 8 × 8  | 8 × 8     | 8 × 8 | 8 × 8  | 8 × 8    | 8 × 8 | 8 × 8  | 8 × 8       |
| Latency (cycles)         | 112896c  | 225792b | 1680  | 3360b    | 1336a | 1336a  | 700a     | 1400a | 840    | 4480       |
| SA throughput (µMatrices/cycle) | 8.9a | 4.43b | 595.24a | 297.62b | 2200 | 2200 | 8929.89 | 4464.95b | 1190a | 595b       |
| SA throughput (kMatrices/s) | 0.89b | 0.443 | 6.55a  | 3.23b    | 1100 | 1100 | 7620     | 3630b  | 1190b | 5.95b      |
| ICA throughput (µMatrices/cycle) | n/a | 4.43b | n/a  | 162.76c | n/a  | 162.76c | n/a      | 162.76c | n/a    | 162.76c    |
| ICA throughput (kMatrices/s) | n/a | 0.443 | n/a  | 1.79b    | n/a  | 1.79b  | n/a      | 81.38b  | n/a    | 132.32b    |
| Complexity (kGE)         | 75.6c   | 75.6c | 61.2a  | 61.2a    | 481  | 481   | 156a     | 156a   | 61.5   |            |
| Power (mW)c              | 16.55c  | n/a  | 0.0816c | n/a      | 152  | n/a   | 50.2c    |        |        |            |
| SA efficiencyc           | 0.12    | 0.06 | 9.736  | 4.864    | 11.458 | 11.458 | 18.565   | 9.283  | 7.6    | 3.629      |
| ICA efficiencyd          | n/a     | 0.06 | n/a   | 2.659    | n/a  | 0.848  | n/a      | 0.338  | n/a    | 1.04       |

aLatency and complexity are analyzed based on the timing and architecture diagram reported in the literature. The latency for the design [10] and [13] are analyzed based on Fig. 4 in [10] and the complexity is analyzed based on Fig. 4.5 in [13]. Furthermore, the latency and complexity of [11] and [14] are analyzed based on Table I in [11] and Table I and Figs. 5 and 6 in [14] respectively. The latency for [15] is analyzed based on the calculations mentioned in Section IV-B in [15] and the throughput of [20] is computed based on Table 3 of [20].
bThe latency and throughput are scaled based on the EVD of a complex-valued 8 × 8 matrix. It is assumed that the architecture remains the same and the processing time is scaled for complex-valued matrices.
cStandalone (SA) is where the throughput is estimated by assuming only the EVD is operating. ICA is where the throughput is calculated by assuming the centering and covariance units reported in this work are the front stages of the EVD.
dThe power is for the entire chip.

cEfficiency = Throughput (µMatrices/cycle) / Complexity (kGE).

frequency. The efficiency FOM is also used to compare the proposed EVD in ICA preprocessing by the different designs. Table V shows that the proposed design outperforms the other designs in terms of efficiency. This is attributed to the highly optimized processing flow and the highly efficient utilization of hardware resources.

Table VI compares the proposed preprocessor with the prior ICA preprocessor designs. The latency and complexity of the ICA are due to the centering and covariance units as well as the EVD unit, as shown in Tables III and IV. Furthermore, the total complexity in Table VI includes the logic as well as the memory. Specifically, to store an 8 × 512 complex-valued matrix, the size of the memory in [10] and [11] has to be increased by four times. Since the design in [13] has multipliers operating in parallel, the size of the memory is the same for a reduced-size complex matrix. Table VI shows that the preprocessor of [10] results in low throughput due to the serialized processing flow. Furthermore, the EVD in [10] has a floating-point structure, which results in high complexity. Moreover, the throughput of [11] is degraded due to the lower efficiency of the processing flow and the low operating frequency. In addition, due to its fully floating-point design, the throughput of [13] is higher at the cost of greatly increased area complexity. Table VI shows that the proposed design outperforms the other designs in terms of the efficiency FOM. This is attributed to the highly optimized processing flow and the highly efficient utilization of hardware resources.

VI. CONCLUSION

This paper presents a high-throughput and highly efficient configurable preprocessor for the ICA algorithm. Moreover, a high-performance MMA is also described. The proposed
prior work, the proposed preprocessor achieves the highest second with a complexity of 73.3 kGE. Compared with
preprocessor achieves a throughput of 40.7 kMatrices per
estimates are based on post-layout evaluations. The proposed
and implemented with a circuit design flow. The performance
can be minimized. The proposed ICA preprocessor is designed
maximized while the required number of hardware elements
thermore, the novel processing flow of the proposed preproces-
sion.

MMA architecture uses time-multiplexed processing, which
greatly increases the efficiency of hardware utilization.
Furthermore, the novel processing flow of the proposed preproces-
sor is highly optimized, so that the centering, the calculation of
the covariance matrix, and the EVD are conducted in parallel and are pipelined. Thus, the processing throughput is
maximized while the required number of hardware elements can be minimized. The proposed ICA preprocessor is designed
and implemented with a circuit design flow. The performance estimates are based on post-layout evaluations. The proposed
preprocessor achieves a throughput of 40.7 kMatrices per
second with a complexity of 73.3 kGE. Compared with prior work, the proposed preprocessor achieves the highest processing throughput and best efficiency.

**REFERENCES**

[1] J. V. Stone, *Independent Component Analysis and Blind Source Separation*, 2004, pp. 1–3.

[2] R. K. Yadav, R. Mehta, and N. Dubey, “Blind audio source separation using weight initialized independent component analysis,” in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, 2015, pp. 563–566.

[3] A. Hyvarinen, “Fast and robust fixed-point algorithms for independent component analysis,” *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 626–634, 1999.

[4] M. E. Fouda, S. Shaboyan, A. Elezabi, and A. Eltawil, “Application of ica on self-ferference cancellation of in-hand full duplex systems,” *IEEE Wireless Communications Letters*, vol. 9, no. 7, pp. 924–927, 2020.

[5] H.-H. Lu, M. E. Fouda, C.-A. Shen, and A. Eltawil, “Full-duplex self cancellation techniques using independent component analysis,” in *2020 54th Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 900–904.

[6] M. Bkassiny and S. K. Jayaweera, “Independent component analysis-based source separation with noise cancelling for cyclostationary detection in cognitive radios,” *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, 2018.

[7] H. Wu, Z. Xiang, G. T. Nguyen, Y. Shen, and F. H. Fitzek, “Component-dependent independent component analysis for time-sensitive applications,” *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020.

[8] H. Wu, Z. Xiang, G. T. Nguyen, Y. Shen, and F. H. Fitzek, “Computing meets network: Coin-aware offloading for data-intensive blind source separation,” *IEEE Network*, vol. 35, no. 5, pp. 21–27, 2021.

[9] X.-L. Li and T. Adali, “Independent component analysis by entropy bound minimization,” *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5151–5164, 2010.

**TABLE V: Comparison with state-of-the-art ICA preprocessors**

| Parameter                  | Original | Scaled | Original | Scaled | Original | Scaled |
|----------------------------|----------|--------|----------|--------|----------|--------|
| Technology (nm)            | 90       | 90     | 90       | 90     | 90       | 90     |
| Architecture type          | Real     | Real   | Real     | Real   | Real     | Real   |
| Arithmetic type            | Hybrid   | Hybrid | Fixed    | Fixed  | Floating | Floating|
| Frequency (MHz)            | 100      | 100    | 11       | 11     | 100      | 100    |
| Matrix type                | Real     | Complex| Complex  | Complex| Real     | Complex|
| Matrix size                | 8 × 256  | 8 × 512| 8 × 256  | 8 × 512| 8 × 1024 | 8 × 512|
| Latency (cycles)           | 137472   | 422400 | 26256    | 199968 | 137472   | 250368 |
| Throughput (µMatrices/cycle)| 7.27µa  | 2.37µa | 38.09µa  | 5.001µ | 7.27µa   | 3.99µa |
| Throughput (kMatrices/s)   | 0.727µs  | 0.237µs| 0.419µs  | 0.055µs | 0.727µs  | 0.399µs
| Memory (kB)                | 8        | 32     | 3.5      | 14     | 32       | 32     |
| Logic complexity (kGE)     | 79.26    | 79.26  | 62.36    | 62.36  | 562.5    | 562.5  |
| Memory complexity (kGE)    | 50.81    | 129.91 | 24.7     | 63.75  | 129.91   | 129.91 |
| Total complexity (kGE)     | 130.07   | 209.17 | 87.06    | 126.11 | 692.41   | 692.41 |
| Power (mW)                 | 16.35    | n/a    | 0.0816   | n/a    | 65.0     | n/a    |
| Efficiency                 | 0.06     | 0.011  | 0.437    | 0.04   | 0.01     | 0.006  |

*Latency, complexity, and memory are analyzed based on Tables III and IV. Throughput is calculated based on the processing latency. Power is for the entire chip, not just the preprocessor. The memory block is generated, and the complexity is estimated based on the area of two-input NAND gates. The total complexity is the sum of the logic complexity and the memory complexity. Efficiency = Throughput (µMatrices/cycle) / Complexity (kGE).*
[24] H. Yang, H. Zhang, J. Zhang, and L. Yang, “Digital self-interference cancellation based on blind source separation and spectral efficiency analysis for the full-duplex communication systems,” *IEEE Access*, vol. 6, pp. 43946–43955, 2018.

[25] A. Sheikhi, S. M. Razavizadeh, and I. Lee, “A comparison of tdd and fdd massive mimo systems against smart jamming,” *IEEE Access*, vol. 8, pp. 72068–72077, 2020.

[26] T. Riihonen and R. Wichman, “Analog and digital self-interference cancellation in full-duplex mimo-ofdm transceivers with limited resolution in a/d conversion,” in *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2012, pp. 45–49.

[27] D. Kim, H. Lee, and D. Hong, “A survey of in-band full-duplex transmission: From the perspective of phy and mac layers,” *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2017–2046, 2015.

[28] M. He and C. Huang, “Self-interference cancellation for full-duplex mimo ofdm with single rf chain,” *IEEE Wireless Communications Letters*, vol. 9, no. 1, pp. 26–29, 2020.

Mohammed E. Fouda received a B.Sc. degree (Hons.) in Electronics and Communications Engineering and an M.Sc. degree in Engineering Mathematics from the Faculty of Engineering, Cairo University, Cairo, Egypt, in 2011 and 2014, respectively. Fouda received a Ph.D. degree from the University of California, Irvine, USA, in 2020. Currently, he works as an assistant researcher at the University of California, Irvine. Fouda has published more than 100 peer-reviewed journal and conference papers, one Springer book, and three book chapters. His H-index is 22, and he has been cited more than 1500 times. His research interests include analog AI hardware, neuromorphic circuits and systems, brain-inspired computing, memristive circuit theory, fractional circuits and systems, and analog circuits. He serves as a peer reviewer for many prestigious journals and conferences. He also serves as a review editor in *Frontiers in Electronics* and the *International Journal of Circuit Theory and Applications*, in addition to serving as a technical program committee member in many conferences. He was the recipient of the best paper award in ICM for 2013 and 2020 and the Broadcom foundation fellowship for 2016/2017.

Ahmed E. Eltawil (S’97–M’03–SM’14) received a doctorate from the University of California, Los Angeles, in 2003 and M.Sc. and B.Sc. degrees (with honors) from Cairo University, Giza, Egypt, in 1999 and 1997, respectively. Since 2019, he has been a professor at the Computer, Electrical, and Mathematical Science and Engineering Division of King Abdullah University of Science and Technology, Thuwal, KSA. Since 2005, he has been with the Electrical Engineering and Computer Science Department at the University of California, Irvine. His research interests are in the general area of low-power digital circuit and signal processing architectures, with an emphasis on mobile systems. Dr. Eltawil has been on the technical program committees and steering committees for numerous workshops, symposia, and conferences. He has received several awards, as well as distinguished grants, including an NSF career grant that supports his research into low-power systems.

Hsi-Hung Lu received the B.Sc. and M.Sc. degree from National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan in 2017 and 2021.

Chung-An Shen received a B.Sc. degree from the National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, in 2000 and an M.Sc. degree from Ohio State University, Columbus, OH, USA, in 2003. He earned a Ph.D. degree from the University of California, Irvine, in 2012. Dr. Shen joined the Department of Electronic and Computer Engineering at NTUST in 2012, where he is currently an associate professor. Dr. Shen’s recent research activities are in low-power and highly efficient digital circuits and signal processing architectures for wireless communication systems. He collaborates with industry in the design and implementation of FPGA-based accelerators and offloading engines for 5G cellular networks. Before joining NTUST, Dr. Shen held several positions in industry and worked on digital signal processing and wireless system design.