IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 May 2025, accepted 19 May 2025, date of publication 27 May 2025, date of current version 4 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3573930

RESEARCH ARTICLE

A Fast True Time-Delay Wideband Multi-Beam Beamforming Algorithm Based on a 16-Beam Approximate-DVM

SIRANI M. PERERA[1], LEVI LINGSCH[2], ALP TUZTAS[3], AND ARJUNA MADANAYAKE[4], (Senior Member, IEEE)

[1]Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114, USA
[2]Department of Mathematics, ETH AI Center, ETH Zurich, 8092 Zürich, Switzerland
[3]Department of Aerospace Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114, USA
[4]Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA

Corresponding author: Sirani M. Perera (pereras2@erau.edu)

This work was supported in part by the National Science Foundation under Award 2229473 and Award 2229471.

ABSTRACT True-time-delay (TTD) beamformers can generate wideband squint-free beams in analog and digital signal domains. The delay Vandermonde matrix (DVM) was introduced as a mathematical model that represents TTD-based multi-beam beamformers while reducing the delays from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$, where $N = 2^r (r \geq 1)$ is the number of beams. In this paper, we propose to reduce the complexity of delays from $\mathcal{O}(N \log N)$ to nearly $\mathcal{O}(N)$ for a small number of beams. More precisely, we present a recursive algorithm to compute the DVM-vector product with a complexity reduction of at least 21% to at most 52% compared to our most recent work, and at least 39% to at most 98% compared to the brute-force DVM-vector calculation. This enhancement was achieved by using 16-beam approximate-DVM (ADVM) building blocks that recursively execute with the DVM algorithm. The reduced complexity DVM algorithm achieves nearly linear complexity for smaller input sizes, specifically when $N \leq 1024$. This modification results in a complexity reduction when compared to the $\mathcal{O}(N \log N)$ complexity of the DVM algorithm, spanning from 8 to 1024 beams. For example, by computing the DVM-vector product for $N = 8$ to 1024 elements antenna arrays, we can obtain wideband RF beams while reducing the required chip area and power consumption by at least 21% at 1024 beams to at most 52% at 16 beams compared to radix-2 DVM algorithm, and also at least 39% at 8 beams to at most 98% at 1024 beams compared to the brute-force DVM-vector product computation. With this reduction, we show that the proposed DVM algorithm is better suited for end-to-end RF-IC design that includes multiple wideband channels. At the end, a signal flow graph, simulated beam patterns at 150 MHz, 300 MHz, 600 MHz, and 1 GHz frequencies based on the proposed ADVM algorithm, and a digital overview are provided to demonstrate the simplicity, efficiency, and accuracy of the proposed TTD multibeam beamformers for RF-IC design.

INDEX TERMS Wideband multi-beam beamforming, true-time delays (TTDs), low-complexity algorithm, antenna arrays, numerical approximation, discrete Fourier transform, delay Vandermonde matrix, sparse matrices, performance of algorithms, matrix norms, signal flow graphs, wireless communication systems.

I. INTRODUCTION

Wideband multi-beam beamforming is the key to meet the ever-expanding range of applications for both frequency range three (FR3) 6-24 GHz as well as frequency range 2 (FR2) - the mm-waves (mmW) in the 24-300 GHz bands - in next-generation communication systems [1], [2], wireless networks [3], [4], [5], [6], [7], [8], and radar systems [9], [10], [11]. For instance, the fifth-generation (5G) ideally requires a frequency range of 30 to 300GHz [12] while

The associate editor coordinating the review of this manuscript and approving it for publication was Olutayo O. Oyerinde.

6G systems will likely be over the FR3 band [13]. Therefore, implementing the traditional beamforming process in the digital domain becomes practically infeasible due to its high power consumption and costs [14]. With it, the FR2 and FR3 frequency bands have garnered significant interest as a solution to meet 5G/6G networks' demands for network capacity [12], [13], [15].

Beamforming is a signal processing technique that improves the strength of signals received by antennas by adjusting their amplitude and phase in a manner such that the signal of interest is subject to constructive interference, while radio frequency interference (RFI) is diminished via destructive interference. This allows for the creation of a stronger beam of signal in the desired direction, having improved signal-to-noise and interference ratio (SNR) [16]. Beamforming overcomes path loss. Therefore, beamforming enables high-capacity wireless connections to be established even in environments with multiple signal paths [17]. Along with the technological advancement, there has been increasing interest in estimating temporal and spatial (directional) parameters [18].

TTD beamformers with simultaneous multi-beams have applications in defense for tactical situations. Multi-beam TTD beamformers will be useful for the development of multifunctional and adaptable RF sensors and systems that can meet the need for national defense for improved operations in spectrally congested and contested environments. Such truly wideband TTD multi-beams can help support a force which can operate while under electromagnetic attack due to the ability to operate across the entire band of interest. The TTD multi-beam capability will advance electronic warfare EW and signals intelligence (SIGINT) capabilities to meet emerging needs in national defense.

Recent advances in beamforming techniques and silicon integrated circuit (IC) technology have led to the development of high-bandwidth active beamforming systems. These systems are becoming increasingly attractive alternatives, especially in 5G/6G systems that aim to maximize the available bandwidth of up to 300 GHz to enhance system capacity [12]. To achieve this, wideband multibeam beamformers based on TTD are essential, and these can be mathematically modeled using the delay Vandermonde matrix (DVM) [4], [5], [6], [19]. Furthermore, the TTD-based DVM beamformers are squint-free and wideband in temporal and spatial frequency responses. This makes the TTD elements more appropriate for wideband applications, as signals can be processed with instantaneous wide bandwidth without introducing significant distortions [20]. However, the discrete Fourier transform (DFT) matrix-based beamformers are narrowband and suffer from the beam squint problem, i.e., TTD systems providing multiple beams with frequency-dependent beam steering [4], [21], [22], [23], [24], [25]. To wit, DVM beamformers are needed for emerging wideband systems, while legacy systems that are relatively narrowband in temporal frequency response can use DFT beamformers via the fast Fourier transform (FFT) algorithm. As we move to wideband systems that operate over extreme frequency ranges, we have to move to DVM beamformers, as DFT/FFT beamformers suffer from the beam squint problem.

The DFT beamformers are realized via FFTs with $\mathcal{O}(NlogN)$ complexity algorithms. The Cooley-Tukey FFT algorithm is well-regarded for effectively processing band-limited and sampled discrete-domain signals by efficiently computing DFTs [26], [27], [28], [29], [30]. The FFT algorithm drastically reduces multiplicative complexity, resulting in significantly improved computation speed [26], [31], [32], [33], [34]. The use of FFT algorithms has overreached conventional applications, extending into ML computations [26], [35], [36], [37], [38], [39], followed by our work on the development of low-complexity neural networks to realize TTD-based multi-beam beamformers [40], [41].

On the other hand, low-complexity beamforming algorithms using an approximate DFT (ADFT) for uniformly linear or rectangular arrays were proposed in [42] and [43]. Here, a sparse factorization was obtained for the DFT matrix to improve the efficiency of beamformers, reducing circuit complexities and power consumption in comparison to FFT for narrowband beamformers [42]. Thus, with ADFT, one can realize an N-number of non-overlapping beams for an N-element array, enabling area-power efficient FFT beamformers [44]. Hence, the ADFT leads to reduce the multiplicative complexity of the FFT from $\mathcal{O}(NlogN)$ to $\mathcal{O}(N)$ while preserving its additive complexity at $\mathcal{O}(N \log N)$ [10], [45]. We note here that the ADFT block contains a certain amount of computational error that prevents it from producing an exact DFT. Despite this, the applications of most ADFT algorithms in wireless communication can accommodate these minor errors without substantially impacting performance [26]. For instance, ADFT beamformers have worst-case sidelobe levels degraded by about 1.5 dB compared to DFT beamformers. Thus, by employing ADFTs instead of FFTs, these networks can possibly reduce the required computational operations, which in turn helps reduce overall circuit complexity, chip area, and power consumption [10], [46], [47], [48], [49], [50], [51], [52].

A. OUR PRIOR WORK

In [4], we showed a reduction of nearly 60% in addition and multiplication counts when computing the DVM-vector product, compared to the brute-force calculation using $\mathcal{O}(N^2)$ beamforming techniques. The DVM, as a superclass of DFT matrices, does not possess the periodic and unitary properties that are typical of DFT matrices [4], [5]. Our work presented in [19] introduces numerically stable DVM algorithms with a complexity of $\mathcal{O}(N \log N)$, specifically designed for TTD narrowband multi-beam beamformers. Furthermore, the nodes of the Vandermonde matrices discussed in [19] are a specific case: they consist of complex nodes that are equally distributed on the unit circle or any circle with a radius larger than unity, rather than being limited to the primitive roots of unity. In contrast, our latest work shows an impressive

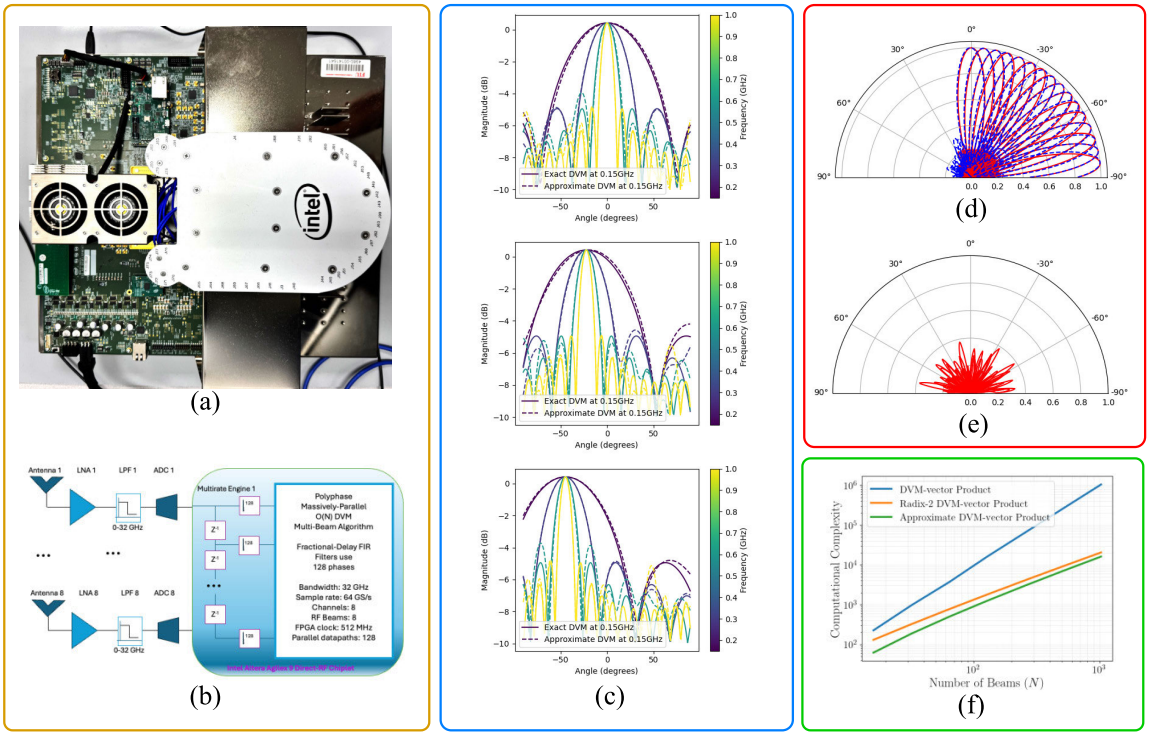


FIGURE 1. Proposed work: (a) A 16-element DVM beamformer with 16 beams can be implemented with eight parallel copies of the Stratix AX-10 FPGA development system supporting two channels.(b) The 8-channel 0-32 GHz $\mathcal{O}(N)$ wideband direct-digital DVM beamformer can be seen through the RF Chiplet. (c) Simulated 16-beam beamformed signals are shown for beams 0, 4, and 8, using both approximate-DVM and the exact DVM across various frequencies.(d) The full set of simulated 16-beam DVM beamformers at 1.0 GHz is shown here. (e) The error between the true and approximate 16-beam DVM beamformers, with each beam normalized to 0 dB. (f) The scaling of the approximate DVM algorithm is improved compared to exact DVM-vector and radix-2 DVM-vector calculations.

complexity reduction compared to $\mathcal{O}(N^2)$, leading to an $\mathcal{O}(N \log N)$ DVM algorithm to realize TTD wideband multibeam beamformers [6]. Thus, we have identified the critical need for a DVM algorithm to reduce complexity from $\mathcal{O}(N\ log\ N)$ to almost linear, enabling the realization of wideband multibeam beamformers with $\mathcal{O}(N)$ delays, rather than $\mathcal{O}(N^2)$ or $\mathcal{O}(N\ log\ N)$. We have also recognized the urgent need for a fast DVM algorithm that leverages the system's structure, setting a new standard for reducing both chip area and power consumption in the implementation of TTD wideband multi-beam beamformers. On the other hand, we also emphasize that no TTD beamformers have reduced complexity than $\mathcal{O}(N\ log\ N)$ in Butler matrix/FFT type beamformers. In fact, TTD N-beam networks are of complexity $\mathcal{O}(N^2)$. Therefore, such beamformers are not feasible even for moderately large N. Thus, we propose an algorithmic interpretation to obtain a DVM algorithm while reducing the complexity to almost linear complexity for a small number of beams.

B. PROPOSED APPROACH

The realization of wideband multi-beam beamformers presents a significant challenge due to the inherent complexity of the aperture transceivers. An N-element receiver array requires N^2 time delays or phasing elements to form N beams in the beamforming network. While the concept may be straightforward, realizing wideband multi-beam beamformers with a reduced complexity–from $O(N^2)$ to $\mathcal{O}(N \log N)$–presents significant challenges due to the complexity of signal flow graphs. Building on our previous work [6], which reduced delay complexity to $\mathcal{O}(N \log N)$, we present a new algorithm that uses sparse matrix factorization of the DVM aiming to reduce complexity of beamformers from $\mathcal{O}(N \log N)$ to nearly $\mathcal{O}(N)$, for a small number of antenna array elements. This reduction will be achieved by utilizing the radix-2 exact DVM algorithm in [6] followed by the highly sparse and multiplierless ADFT matrices in [45], [53]. Simply, we use matrix embedding in the development of an almost linear delay DVM algorithm ranging from 8-point to 1024-point. Additionally, we will obtain an approximate DVM (ADVM) that requires minimal multiplication (or analog delay [54]) operations. Although we have proposed multiple different DVM algorithms [4], [5], [19], we have specifically chosen the DVM factorization presented in [6] because it is the exact radix-2 DVM algorithm that executes recursively with the DFT to realize wideband multi-beam beamformers to present linear ADVM algorithm for small number of antenna array elements. The primary motivation of the reduced complexity DVM algorithm is to minimize chip area and power consumption in integrated circuit design for wideband multi-beam beamformers. This is vital in mm-wave wireless

communication, where minimizing the number of delays is of utmost importance. To pave the way for that, we will demonstrate a simple Signal Flow Graph (SFG) for the approximated 16-point DVM algorithm (16-point ADVM) to answer the complexity demand of SFGs. Finally, we present simulated beam patterns for a wideband DVM multi-beam beamformers with $N = 16$ beams, along with the state-of-the-art in linear wideband direct-digital DVM beamformers using the Intel Altera Agilex 9 Direct-RF Chiplet. Thus, we show Figure 1 to illustrate the overall architecture, incorporating the simulated approximate-DVM beamformed signals, errors, and complexity reduction.

C. ORGANIZATION OF THE PAPER

In Section II, we propose highly sparse factors to compute an ADVM from 8-point to N-point using the exact DVM and ADFT factorizations. Section III provides algorithms for computing the ADVM algorithm and discusses the arithmetic (or delay) complexity of the proposed ADVM algorithm. At the end of the section, we show that the proposed N-point ADVM has almost linear-order complexity for a small number of antenna array elements. Section IV presents numerical results to show both the error bound and the proximity of the proposed ADVM to the DVM, evaluated through the spectral norm. In section V, the proposed ADVM algorithm will be shown using an SFG, illustrating the bijection between the system of linear equations and the fundamental building blocks of the flow graphs, showing the linear complexity for the 16-point ADVM algorithm. The proposed ADVM algorithm will be utilized to show the beamforming patterns for 16 beams, as shown in section VI. Within the same section, we will provide a brief overview of the state-of-the-art (SOTA) chiplet using the Intel Altera Agilex 9. Finally, Section VII concludes the paper.

II. SPARSE FACTORIZATION TO COMPUTE AN APPROXIMATE-DVM

This section presents sparse factors to obtain an approximate DVM using scaled DVM and ADFT, aiming to reduce multiplication (delay) complexity. The goal of the sparse factorization is to develop a linear order DVM algorithm for a small number of antenna array elements, as depicted in section III. After presenting the ADVM algorithm, we will clarify the meaning of the terminology "delay complexity." Thus, before obtaining factorization formulas for the DVM, we state the frequently used notations as follows.

A. FREQUENTLY USED NOTATIONS

In this section, we will state the notations for sparse and orthogonal matrices that will be utilized throughout this paper.

DVM definition [4], [5], [6], [19]: The DVM is defined by

$$A_N := [A_{kl}]_N = [\alpha^{kl}]_{k=1,l=0}^{N,N-1},$$

where $N = 2^r (r \geq 1)$ is the number of beams, $\{\alpha, \alpha^2, \ldots, \alpha^N\}$ are distinct complex nodes, $\alpha = e^{-j\omega\tau}$, $j^2 = -1$, ω is the temporal frequency, and τ is the delay. We can utilize the definition of the DVM to establish a scaled DVM, which is the DVM scaled by a diagonal matrix, as given below

$$\widetilde{A}_N := [\widetilde{A}_{kl}]_N = [\alpha^{kl}]_{k,l=0}^{N-1}.$$

We note that the coefficient α^{kl} is a temporal Fourier transform of a pure time-delay of duration τ. The signal $x(t)$ with Fourier transform $X(\omega)$ is related to the delayed version of the signal $x(t - p\tau)$ via the relationship $X(\omega)e^{-jp\omega\tau}$, where p is a positive integer. So the corresponding phase rotation of $X(\omega)$ is simply $\alpha^p = e^{-j\omega p\tau}$. Crucially, the DVM contains closed-form complex functions of ω given as complex phase rotations in integer powers p of α. That is, α^p are not necessarily numerical values, but they could represent complex functions of frequency ω raised to power p, i.e., seen $e^{-j\omega p\tau}$ as a function of ω. So an $N \times N$ DVM by an $N \times 1$ vector containing powers of α *resembles a matrix-vector product*, but we are in the temporal Fourier domain; so this is defined $\mathcal{O}(N^2)$ delay-and-sum product, in contrast to the conventional $\mathcal{O}(N^2)$ multiply-and-sum product.

DVM factorization [6]: This paper introduces a factorization for an approximate-DVM, aligning with the execution of the scaled DVM algorithm in [6]. Thus, let us recall the factorization for the scaled DVM in [6] given via

$$\tilde{A}_N = \hat{D}_N [J_{M\times N}]^T F_M^* \check{D}_M F_M J_{M\times N} \hat{D}_N,$$

where $M = 2N$, $J_{M\times N} = \begin{bmatrix} I_N \\ 0_N \end{bmatrix}$ is a sparse matrix consisting of an identity matrix I_N and a zero matrix 0_N, $\hat{D}_N = diag\left[\alpha^{\frac{k^2}{2}}\right]_{k=0}^{N-1}$ and $\check{D}_M = diag\left[\tilde{F}_M c\right]$ are diagonal matrices, C_M is a circulant matrix defined by the first column c s.t.

$$c = \left[1, \alpha^{-\frac{1}{2}}, \cdots, \alpha^{-\frac{(N-1)^2}{2}}, 1, \alpha^{-\frac{(N-1)^2}{2}}, \alpha^{-\frac{(N-2)^2}{2}}, \cdots, \alpha^{-\frac{1}{2}}\right]^T,$$

where ^{T} represents the transpose operation, $F_N = \frac{1}{\sqrt{N}} [w_N^{kl}]_{k,l=0}^{N-1}$ is the DFT matrix, $\tilde{F}_N = \sqrt{N} F_N$ is the scaled DFT matrix, and $w_N = e^{-\frac{2\pi j}{N}}$.

DFT factorization [26], [27], [31]: The DVM algorithm in [6] executes recursively via the FFT algorithm. Thus, to grasp this, we outline the factorization of the DFT that leads to the FFT algorithm, following [27] and [31]. The scaled DFT factorization is given via $F_N = P_N^T \begin{bmatrix} F_{\frac{N}{2}} & 0_{\frac{N}{2}} \\ 0_{\frac{N}{2}} & F_{\frac{N}{2}} \end{bmatrix} H_N$, where for a given vector $x = [x_0, x_1, \cdots, x_{N-1}]^T \in R^N$, P_N ($N \geq 3$) is an even-odd permutation matrix given via

$$P_N \mathbf{x} = \begin{cases} [x_0, x_2, \cdots, x_{N-2}, x_1, x_3, \cdots, x_{N-1}]^T & \text{even } N \\ [x_0, x_2, \cdots, x_{N-1}, x_1, x_3, \cdots, x_{N-2}]^T & \text{odd } N, \end{cases}$$

$H_N = \begin{bmatrix} I_{\frac{N}{2}} & I_{\frac{N}{2}} \\ \grave{D}_{\frac{N}{2}} & -\grave{D}_{\frac{N}{2}} \end{bmatrix}$ is a scaled orthogonal matrix, $\grave{D}_{\frac{N}{2}} = diag\left[w_N^l\right]_{l=0}^{\frac{N}{2}-1}$ is a diagonal matrix, and H_N^* is the conjugate transpose of H_N.

ADFT definition: To obtain a linear order 16-beam ADVM algorithm, we will incorporate ADFT factorization in place of the DFT [26], [55], [56] within the factorization of the scaled DVM. Thus, the 16-beam ADVM algorithm executes using the ADFT factors, leading us to revisit the sparse and condensed ADFT factorization formula in [26] with the fewest factors. Hence, we recall the factorization for the 16-point ADFT [45], [57] and also the 32-point ADFT [53], [56], [58], [59], s.t., $\widehat{F}_{32} = W_8 \cdot W_7 \cdot W_6 \cdot W_5 \cdot W_4 \cdot W_3 \cdot W_2 \cdot W_1$, which we utilize as an initialization of the *Adft* algorithm in section III-A, where $\widehat{F}_N$ represent the N-point ADFT. For an in-depth identification of the elements within the matrices and the factorization of 16-point and 32-point ADFT matrices, we refer the readers to [45], [53], [56], and [59].

B. A FACTORIZATION FOR APPROXIMATE-DVM

This section presents a factorization for approximate-DVM to reduce the number of delays from $\mathcal{O}(N^2)$ to nearly linear, enabling the realization of N-wideband multibeam beamformers having highly sparse factors. Along with the factorization of the N beams DVM-based beamformers, we also obtain 16-point DVM factorization, which will be utilized as an initialization of the *Asdvm* algorithm in section III-A. Furthermore, we utilize the matrix embedding to replace DFT matrices via a 32-point ADFT matrix to present DVM factorization for 16 beams. Thus, we start this section by presenting a modified 32-point ADFT factorization based on [53], [56], and [58].

Proposition 2.1: The factorization of the 32-point approximate DFT can be expressed as:

$$\widehat{F}_{32} = S_4 \cdot S_3 \cdot S_2 \cdot S_1 \cdot S_0, \tag{1}$$

where $S_0 = W_1$, $S_1 = W_3 \cdot W_2$, $S_2 = W_5 \cdot W_4$, $S_3 = W_7 \cdot W_6$, *and* $S_4 = W_8$.

Proof: This is trivial by the matrix multiplication while setting $S_1 = W_3 \cdot W_2$, $S_2 = W_5 \cdot W_4$, and $S_3 = W_7 \cdot W_6$ in [53], [56], and [58]. These S_i matrices for $i = 0, 1, \cdots, 4$, are explicitly given in Appendix . ■

Remark 2.2: Following the above result, the conjugate transpose of the ADFT can be easily observed as $\widehat{F}^*_{32} = S_0^T \cdot S_1^T \cdot S_2^T \cdot S_3^T \cdot S_4^*$. *Thus, we can leverage both* $\widehat{F}^*_{32}$ *and* $\widehat{F}_{32}$ *to obtain a factorization for the 16-beam ADVM.*

Let's present a factorization to obtain an approximation for the scaled DVM from 8-point to N-point to reduce the arithmetic complexity. We show that our proposed approximate-DVM algorithm has linear order multiplication complexity for a small number of antenna array elements.

Proposition 2.3: Let the scaled DVM, i.e., $\widetilde{A}_N = [\alpha^{kl}]^{N-1}_{k,l=0}$ *be defined by nodes* $\{1, \alpha, \alpha^2, \ldots, \alpha^{N-1}\} \in \mathbb{C}$, $N = 2^r$ $(r \geq 4)$, *and* $M = 2N$. *Then, an approximation for the N-point scaled DVM denoted as* $\widehat{A}_N$, *can be obtained through the following*:

$$\widehat{A}_N = \hat{D}_N [J_{M\times N}]^T \widehat{F}^*_M \check{D}_M \widehat{F}_M J_{M\times N} \hat{D}_N, \tag{2}$$

where $\widehat{F}_M = P_M^T \begin{bmatrix} \widehat{F}_N & \\ & \widehat{F}_N \end{bmatrix} H_M$.

Proof: We can obtain the above factorization for ADVM by utilizing the matrix embeddings of both $\widehat{F}_M$ and $\widehat{F}^*_M$ in places of F_M and F^*_M, respectively, within the factorization of DVM in [6] followed by the initialization with Proposition 2.1. ■

The factorization mentioned above will be utilized to develop an ADVM algorithm for sizes $N \geq 16$. Thus, in the following, we present an 8-point ADVM factorization based on the ADFT factorization in [45].

Corollary 2.4: Let 8-point scaled DVM, i.e., $\widetilde{A}_8 = [\alpha^{kl}]^7_{k,l=0}$ *be defined by nodes* $\{1, \alpha, \alpha^2, \ldots, \alpha^7\} \in \mathbb{C}$. *Then, an approximation for the 8-point scaled DVM denoted as* $\widehat{A}_8$, *can be obtained through the following*:

$$\widehat{A}_8 = \hat{D}_8 [J_{16\times 8}]^T \widehat{F}^*_{16} \check{D}_{16} \widehat{F}_{16} J_{16\times 8} \hat{D}_8. \tag{3}$$

Proof: This follows directly from Proposition 2.3 with the substitution for $N = 16$, along with the replacement of $\widehat{F}^*_{16}$ using the approximated 16-point DFT matrix in [45]. ■

III. APPROXIMATE-DVM ALGORITHM TO REDUCE COMPLEXITY

Following the ADVM factorization for N beams presented in Section II, we introduce an approximated scaled DVM algorithm that executes recursively with an approximate 16-point scaled DVM algorithm, followed by the matrix embedding of the 32-point ADFT. More specifically, we present an approximate scaled DVM algorithm for $N \geq 32$ points, referred to as *Asdvm*, alongside the 16-point approximate scaled DVM algorithm, called *Asdvm16*. We note that *Asdvm16* algorithm executes based on the embeddings of the sparse factors of the ADFT. To optimize the computation of the matrix-vector product and minimize multiplication counts, we have relocated the factor $\frac{1}{\sqrt{M}}$ in F_M and F^*_M to the end of the computation, and hence computed scaled ADVM algorithm s.t. $y = M\widehat{A}_N z$. We will show at the end of the section that the proposed ADVM algorithm has nearly linear complexity as opposed to $\mathcal{O}(N \log N)$ complexity for a small number of antenna array elements.

A. LOW-COMPLEXITY APPROXIMATED SCALED DVM ALGORITHM

Let's present *Asdvm* and *Asdvm16* algorithms for obtaining approximated N-point DVM algorithms for $N \geq 16$. As described in Section II, we find that the factorization for the ADVM is self-explanatory for $N = 4, 8$, and obtaining these algorithms may not be necessary. We now explicitly state an approximate scaled DVM algorithm to compute the product of a scaled DVM by a vector, i.e., $y = M\widehat{A}_N z$ for a given N, α, $z \in \mathbb{C}^N$ or $\mathbb{R}^N$ and $c \in \mathbb{C}^M$.

The proposed approximate scaled DVM algorithm *Asdvm* executes recursively with the approximate scaled FFTs, initialized with the 16-point approximate DVM algorithm. We shall refer to the scaled approximate FFT and approximate inverse FFT algorithms as *Adft* and *Aidft*, respectively. We will begin with the 16-point ADVM algorithm, *Asdvm16*, as follows.

Algorithm 1 *Asdvm*

Input: $N = 2^r$ $(r \geq 4)$, $M = 2N$, $\alpha \in \mathbb{C}$ s.t. $|\alpha| = 1$, $z \in \mathbb{C}^N$ or $\mathbb{R}^N$, and $c \in \mathbb{C}^M$.
Output: $y = M\widehat{A}_N z$.
Function: y=*Asdvm*(z, N).

1) **if** $N = 16$, **then**
   $y \leftarrow Asdvm16(z, N)$,
2) **end if**
3) **if** $N \geq 32$, **then**
   $u_1 \leftarrow \hat{D}_N \cdot z$
   $u_2 \leftarrow J \cdot u_1$,
   $v_1 \leftarrow Adft(u_2, M)$,
   $v_2 \leftarrow \check{D}_M \cdot v_1$,
   $y_1 \leftarrow Aidft(v_2, M)$,
   $y_2 \leftarrow J^T \cdot y_1$,
   $y \leftarrow \hat{D}_N \cdot y_2$,
4) **end if**
5) **return** y

Algorithm 2 *Asdvm16*

Input: $N = 16$, $M = 2N$ and $z \in \mathbb{C}^N$ or $\mathbb{R}^N$.
Output: $y = M \cdot \widehat{A}_{16} z$
Function: y=*Asdvm16*(z, N).

1) **if** $N = 16$, **then**
   $w_1 \leftarrow \hat{D}_N \cdot z$
   $w_2 \leftarrow J \cdot w_1$,
   $w_3 \leftarrow \widehat{F}_M \cdot w_2$,
   $w_4 \leftarrow \check{D}_M \cdot w_3$,
   $w_5 \leftarrow \widehat{F}^*_M \cdot w_4$,
   $w_6 \leftarrow J^T \cdot w_5$,
   $y \leftarrow \hat{D}_N \cdot w_6$,
2) **end if**
3) **return** y

Algorithm 3 *Adft*

Input: $M = 2^{r_1}$ $(r_1 \geq 5)$, $M_1 = M/2$, and $u_2 \in \mathbb{C}^M$.
Output: $v_1 = \widehat{F}_M u_2$.
Function: v_1=*Adft*(u_2, M).

1) **if** $M = 32$, **then**
   $v_1 \leftarrow \widehat{F}_{32} u_2$
2) **end if**
3) **if** $M \geq 64$, **then**
   $p \leftarrow H_M \cdot u_2$
   $a_1 \leftarrow Adft(p(1 : M_1), M_1)$,
   $a_2 \leftarrow Adft(p(M_1 + 1 : M), M_1)$,
   $v_1 \leftarrow P^T_M \cdot \left[a_1^T \ a_2^T\right]^T$,
4) **end if**
5) **return** v_1

Algorithm 4 *Aidft*

Input: $M = 2^{r_1}$ $(r_1 \geq 5)$, $M_1 = M/2$, and $v_2 \in \mathbb{C}^M$.
Output: $y_1 = \widehat{F}^*_M v_2$.
Function: y_1=*Aidft*(v_2, M).

1) **if** $M = 32$, **then**
   $y_1 \leftarrow \widehat{F}^*_{32} v_2$
2) **end if**
3) **if** $M \geq 64$, **then**
   $q \leftarrow P_M \cdot v_2$
   $b_1 \leftarrow Aidft(q(1 : M_1), M_1)$,
   $b_2 \leftarrow Aidft(q(M_1 + 1 : M), M_1)$,
   $y_1 \leftarrow H^*_M \cdot \left[b_1^T \ b_2^T\right]^T$,
4) **end if**
5) **return** y_1

Example 3.1: Based on the proposed algorithms, we show building blocks in the SFG drawn for the 16-point scaled ADVM algorithm in Section V. The Asdvm algorithm is stated based on the sparse factorization in Proposition 2.3, and hence, the factorization for the scaled ADVM can be stated as follows.

$$32\widehat{A}_{16} = \hat{D}_{16}\,[I_{16}|0_{16}]\,\widehat{F}^*_{32}\check{D}_{32}\widehat{F}_{32}\begin{bmatrix} I_{16} \\ 0_{16} \end{bmatrix}\hat{D}_{16},$$

where $\hat{D}_{16} = [\hat{d}_k]_{k=0}^{15}$, $\check{D}_{32} = [\check{d}_k]_{k=0}^{31}$, $\widehat{F}^*_{32} :=$ *the conjugate transpose of the ADFT, i.e.,* $\widehat{F}_{32}$*, and the sparse factorization for the* $\widehat{F}_{32}$ *can be obtained from Proposition 2.1.*

B. MULTIPLICATION COMPLEXITY, i.e., GAIN-DELAY BLOCK COUNTS, OF THE APPROXIMATE-DVM ALGORITHM

This section focuses on the gain-delay block counts [6], which are based on the computation of the approximated scaled DVM algorithms in section II. We note here that the gain-delay block counts can be interpreted as multiplication counts that relate to the brute-force computation of the scaled DVM by a vector. The multiplication counts presented in the frequency domain represent a combination of gains and delays in the time domain. TTDs require a dedicated TTD circuit within the analog domain, where delays are realized through continuous-time analog circuits. In the temporal frequency domain, true time delays $\tau \in \mathbb{R}$ are expressed as multiplicative terms $e^{-j\omega\tau} \in \mathbb{C}$. Therefore, we refer to the multiplication counts involved in DVM-vector computations as gain-delay counts, which are subsequently followed by the delay complexity associated with the ADVM algorithm.

Let us obtain the number of multiplications (say #m), i.e., gain-delay blocks [6], required to compute the ADVM algorithm. We take the number of multiplications required to compute $y = \widehat{F}_M z$, where $z \in \mathbb{C}^M$, as 0 for $M = 16$ *and* 32, when the multiplications by ± 1 and $\pm j$ are not counted, and use *Adft, and Aidft* algorithms to obtain counts to compute $y = \widehat{F}_M z$ for $M > 32$.

Lemma 3.2: Let $N = 2^r (r \geq 4)$ *be given. The approximated scaled DVM algorithm, i.e., Asdvm algorithm,*

can be computed recursively utilizing the Adft, and Aidft algorithms with the following gain-delay block counts (number of multiplications) :

$$\begin{aligned}
&\#m(Asdvm, N) = 4N - 2, \ \ for \ \ N = 16\\
&\#m(Asdvm, N) = 6N - 6, \ \ for \ \ N = 32\\
&\#m(Asdvm, N) = 8N - 14, \ \ for \ \ N = 64\\
&\#m(Asdvm, N) = 10N - 30, \ \ for \ \ N = 128\\
&\#m(Asdvm, N) = 12N - 62, \ \ for \ \ N = 256\\
&\#m(Asdvm, N) = 14N - 126, \ \ for \ \ N = 512\\
&\#m(Asdvm, N) = 16N - 254, \ \ for \ \ N = 1024\\
&\#m(Asdvm, N) = 2Nr - \frac{17}{4}N + 2, \ \ for \ \ N > 1024. \quad (4)
\end{aligned}$$

Proof: Following the approximated scaled DVM algorithm in Proposition 2.3, we could obtain the number of multiplications, i.e., gain-delay block counts, using the equation:

$$\begin{aligned}
\#m(Asdvm, N) = 2\left(\#m(\widehat{F}_M)\right) + 2\left(\#m(\hat{D}_N)\right)\\
+ 2\left(\#m(J)\right) + \#m(\check{D}_M). \quad (5)
\end{aligned}$$

Moreover, we could utilize the structures of $\hat{D}_N$, J, and $\check{D}_M$ followed by the multiplication of each matrix by a vector to obtain number of gain-delay block counts via:

$$\#m\left(J\right) = 0, \#m\left(\hat{D}_N\right) = N - 1, \#m\left(\check{D}_M\right) = M. \quad (6)$$

Now, we utilize the 32-point ADFT factorization in Proposition 2.1 followed by the *Adft and Aidft* algorithms to obtain the multiplication counts corresponding to the computation of the ADFT by a vector via: $\#m\left(\widehat{F}_M\right) = 0, N - 2, 2N - 6, \cdots, 6N - 126$, respectively, for $N = 16, 32, 64, \cdots, 1024$. Moreover, to compute the approximated DFT by a vector, i.e., $y = \widehat{F}_M u$ with $u \in \mathbb{C}^M$, for $N > 1024$, we get $Nr - \frac{33}{8}N + 2$ multipliers because we executed the 16-point ADFT via an fft-type algorithm. Thus, using these along with the equations (5) and (6), we get the multiplication counts, i.e., gain-delay block counts as in (4) based on the computation of the proposed approximated scaled DVM algorithm. ■

Remark 3.3:
1) *The paper [45] along with the equations (2) and (4), we conclude that the gain-delay complexity of the 8-point scaled ADVM algorithm is* $4N - 2$.
2) *We had shown that the proposed approximated scaled DVM algorithm has almost linear order, i.e., gain-delay complexity of* $\mathcal{O}(N)$ *as opposed to the FFT-like* $\mathcal{O}(N \log N)$ *algorithms, for a small number of antenna array elements. While the gain-delay blocks show linear behavior, the adder counts of the ADVM algorithm reach* $\mathcal{O}(N^2)$ *for smaller values of* N*, specifically when* N *is 8 or 16, due to the embedding of 16-point or 32-point ADFT, respectively, in Corollary 2.4 or Proposition 2.3. However, as* N *increases, i.e.,* $N \geq 32$*, the adder complexity improves from* $\mathcal{O}(N^2)$ *to* $\mathcal{O}(N \log N)$ *as we embed only a 32-point ADFT in Proposition 2.3 to execute the 16-point ADVM, but not for* $N \geq 32$.
3) *We mention that employing the linear order ADFT algorithm [53] can significantly reduce the complexity of the DVM-vector product for any value of* N*. We assert that this reduction in complexity is relevant only for gain-delay block counts and not for adder counts. The decrease in adder counts transitions from* $\mathcal{O}(N^2)$ *to* $\mathcal{O}(N \log N)$*, remains as it is in [6] for large* N*. But, in integrated circuit (IC) design, gain-delay blocks carry a significantly higher cost compared to adders. Therefore, prioritizing cost in IC design makes ADVM worthwhile compared to the exact and radix-2 DVM algorithms for a small number of antenna array elements.*
4) *Although the gain-delay complexity of the proposed ADVM algorithm is nearly linear* $\mathcal{O}(N)$*, the ADVM beamfomers may not be practical in cases where sidelobe selectivity and gain are crucial. This is because the ADVM algorithm utilizes a matrix embedding based on 32-point ADFT, which has the worst-case sidelobe intensity levels degraded by approximately 1.5 dB when compared to DFT beamformers [10], [46], [47], [48], [49]. Nevertheless, we have shown a trade-off between the ADVM beamformers and the exact DVM beamformers while comparing the simulated beam patterns obtained at input frequencies of 150 MHz, 300 MHz, 600 MHz, and 1 GHz, with ideal beams in section VI. We have observed discrepancies at each frequency that were marginally greater than those produced by the 16-point exact DVM algorithm [6], but this difference is expected since we used an ADVM rather than the exact DVM algorithm.*

TABLE 1. Gain-delay block counts of the scaled approximate-DVM algorithm, the direct computation of the scaled DVM by a vector, $\mathcal{O}(N \log N)$ DVM algorithm, and percentage reduction of complexities, i.e., P_A w.r.t. $\mathcal{O}(N \log N)$ and P_L w.r.t. $\mathcal{O}(N^2)$.

N	Direct Multiply	$\#m(sdvm)$ [6]	$\#m(Asdvm)$	P_A	P_L
8	49	50	30	40%	39%
16	225	130	62	52%	72%
32	961	322	186	42%	81%
64	3696	770	498	35%	87%
128	16129	1794	1250	30%	92%
256	65025	4098	3010	27%	95%
512	261121	9218	7042	24%	97%
1024	1046529	20482	16130	21%	98%

The proposed DVM is an approximated algorithm as opposed to the exact DVM algorithms in [4], [5], [6], and [19]. Thus, we provide the detailed counts in Table 1 to enhance readers' comprehension of the explicit gain-delay or multiplication counts along with the percentage reduction of complexities as opposed to the best-known DVM algorithm having $\mathcal{O}(N \log N)$ complexity [6] and the brute-force DVM-vector calculation.

Table 1 shows the numerical results for the multiplication complexity (the gain-delay blocks) of the approximated scaled DVM algorithm in Lemma 3.2 compared to the brute-force DVM-vector computations and the radix-2 exact DVM algorithm. The matrix sizes in the comparison range vary from 16×16 to 1024×1024. Let us consider the direct computation of the scaled DVM by a vector for gain-delay blocks as $(N-1)^2$, where the number of gain-delay blocks being denoted by $\#m(sdvm)$ in order to compute the exact scaled DVM algorithm presented in [6]. The percentage reductions, denoted as P_A and P_L, represent the decreases in gain-delay block counts based on the proposed approximated scaled DVM algorithm when compared to the $\mathcal{O}(N \log N)$ complexity DVM algorithm in [6] and the $\mathcal{O}(N^2)$ brute-force DVM-vector computation. The values in the column before the last are obtained using

$$P_A(w.r.t.\ \mathcal{O}(N \log N)) = \left(\frac{Te - Ta}{Te}\right) \times 100\%, \quad (7)$$

where Te is the gain-delay block counts of the DVM algorithm in [6], i.e., $\mathcal{O}(N \log N)$ complexity, and Ta is the gain-delay block counts in computing the proposed ADVM algorithm.

The values in the last column are obtained using

$$P_L(w.r.t.\ \mathcal{O}(N^2)) = \left(\frac{Td - Ta}{Td}\right) \times 100\%, \quad (8)$$

where Td is the gain-delay block counts in computing the direct matrix-vector product, i.e., $\mathcal{O}(N^2)$ complexity, and Ta is the gain-delay block counts in computing the proposed ADVM algorithm.

Table 1 shows that the scaled ADVM algorithm requires a significantly low number of gain-delay blocks compared to the brute-force calculation. When the size of the matrices increases, we observe a significant reduction in multiplication complexity (equivalently, delay-gain blocks) for computing the proposed algorithm. More preciously, numerical results in Table 1 shows that the proposed ADVM algorithm could be utilized to compute the DVM-vector product with a complexity reduction of at least 21% at 1024 beams to at most 52% at 16 beams compared to our most recent work based on the $\mathcal{O}(N \log N)$ exact DVM algorithm in [6]. We also emphasize that the proposed ADVM algorithm has a percentage reduction in complexity, having at least 39% at 8 beams to at most 98% at 1024 beams compared to the brute-force DVM-vector calculation to realize TTD wideband multibeam beamformers. These complexity reductions will pave the way for implementing the ADVM algorithm in RF chiplet design, addressing the demand for chip area and power consumption while reducing complexity by at least 39% at 8 beams to at most 98% at 1024 beams, as opposed to brute-force techniques-based chiplet architectures.

We also observe that the proposed scaled ADVM algorithm has a low number of gain-delay blocks compared to the exact DVM algorithms with $\mathcal{O}(N \log N)$ complexity in [6] and $\mathcal{O}(N^2)$ complexity algorithms in [4], [5], and [60]. Since the scaled DVM algorithms presented in [4], [5], and [60] exhibit $\mathcal{O}(N^2)$ complexity, equivalent to that of brute-force calculations, we have chosen not to include these results in our comparison Table 1. Instead, we compared the proposed approximated scaled DVM algorithm with our TTD wideband multi-beam beamformer-based exact scaled DVM algorithm having $\mathcal{O}(N \log N)$ complexity [6]. Although the radix-2 DVM and scaled DVM algorithms are stated in [19], we haven't compared the proposed ADVM algorithm with the complexity results in [19] because the latter paper is primarily focused on narrowband scenarios, rather than wideband multi-beam beamformers. On the other hand, we haven't compared the complexity result of the proposed ADVM algorithm with [20] as the TTD-based multi-beam beamformers in the later paper are realized with the $\mathcal{O}(N^2)$ complexity for N-beam. To sum up, utilizing the ADFT instead of the FFT for the ADVM algorithm shows that the complexity reduction percentage decreases with larger matrix sizes. However, the ADVM maintains linearity for a small number of beams.

IV. NUMERICAL RESULTS SHOWING THE PROXIMITY OF THE APPROXIMATION WITH THE EXACT DVM

The proposed ADVM algorithm is somewhat closer to the exact DVM algorithm. Thus, knowing the numerical values showing how close the ADVM algorithm is to the exact DVM algorithm, yielding to the lower computational complexity, is an approximation problem. Mathematically, the approximation problem can be stated as a constrained optimization problem with respect to the constrained imposed to reduce the error between the exact DVM with the ADVM. The required constraint is that the computational cost of ADVM should be less than that of the exact DVM, in both theory and implementation. Thus, in the context of approximations for scaled delay Vandermonde matrices $\widetilde{A}_N$, ADVM is a transformation matrix such that $\widehat{y} = \widehat{A}_N z \approx y = \widetilde{A}_N z$. In essence, the exact and approximate transform domain signals are closely related in a quantifiable manner. Thus, an approximated matrix must fulfill the conditions s.t. (i) preserving key properties of the exact matrix; (ii) maintaining mathematical proximity with the exact matrix; and (iii) significantly reducing computational costs compared to the exact DVM-vector product computation.

To ensure the accurate interpretation of the approximate spectrum, it is common practice to seek approximate scaled DVMs that closely resemble the associated exact scaled DVMs. A way to achieve this is by minimizing the difference between the exact and the approximated scaled DVMs, through a matrix norm. This minimization process should also take into account the constraint of maintaining low complexity. We know that the matrix norms satisfy the equivalence property, so we utilize the spectral norm, i.e., induced 2-norm, which is compatible with the Frobenius, 1-norm, or ∞-norm. Since norms are equivalent, one could also show the existence of positive constants c_1 and c_2 satisfying $c_1||\widetilde{A}_N - \widehat{A}_N||_\nu \leq ||\widetilde{A}_N - \widehat{A}_N||_2 \leq ||\widetilde{A}_N - \widehat{A}_N||_\nu$, where ν can be Frobenius, 1-norm, or ∞-norm, showing the

convergence of the ADVM algorithm with respect to the other norms. Hence, we show a possible minimization problem of deriving an approximated scaled DVM for the best possible $\alpha \in \mathbb{C}$ through a constrained optimization problem s.t.

$$error(\widetilde{A}_N, \widehat{A}_N) := min_{\alpha \in \mathbb{C}} ||\widetilde{A}_N - \widehat{A}_N||_2, \tag{9}$$

to minimize error between the exact DVM and the ADVM, where $||\cdot||_2$ is the spectral norm. As a result of this, we will present numerical results based on the equation (9) to minimize the error as stated in Table 2.

TABLE 2. Minimizing the difference between the exact and the approximated scaled DVMs through the spectral norm with the best possible $\alpha_1, \alpha \in \mathbb{C}$, where $\alpha_1 = e^{-j(2q+1)\pi/2}$ for $q \in \mathbb{Z}_0^+$.

N	error $(\widetilde{A}_N, \widehat{A}_N)$ for α_1 values	α values	error $(\widetilde{A}_N, \widehat{A}_N)$ for α values
16	4.31e-14	$e^{-j\pi/(4096)}$	0.1633
32	1.2926e-13	$e^{-j\pi/(8192)}$	0.6343
64	7.9307e-12	$e^{-j\pi/(65536)}$	0.6466
128	3.6255e-11	$e^{-j\pi/(524288)}$	0.6525
256	1.2945e-09	$1.4142 \times 10^{-6} - j$	0.7460
512	4.1470e-09	$2 \times 10^{-7} - j$	0.8467
1024	2.0806e-09	$0.25 \times 10^{-7} - j$	0.8480

Table 2 values are obtained in the IEEE floating point format with the significant decimal digits precision of 10^{-16}. The above-tabulated data shows that the approximated scaled DVM algorithm convergences w.r.t. the spectral norm for a set of α values defined as $e^{-j(2q+1)\pi/2}$, where $q \in \mathbb{Z}_0^+$, having the error bound from 10^{-14} to 10^{-9}. For these α values, we had chosen integer delays s.t. $\tau = 2q + 1$ followed by $\omega = \pi/2$. Furthermore, we have shown numerical results for the other specific α values as listed in the third column of Table 2 to indicate the convergence of the ADVM algorithm, albeit with an error bound of 10^{-1}. For these α values, we had chosen fractional delay values s.t. $0 < \tau < 1$ followed by an appropriate ω satisfying $\alpha = e^{-j\omega\tau}$, e.g., at $N = 16$ when we had chosen $\alpha = 1/16, 1/32, 1/64$ we selected $\omega = \pi/256, \pi/128, \pi/64$, respectively. Furthermore, for a given N, we executed the proposed DVM algorithm 20 times while varying α values, in order to minimize the spectral norm. For instance, if we had fixed α values at $N = 256, 512, 1024$ to be $1.4142 \times 10^{-6} - j$, the corresponding spectral norms are 0.7460, 5.9829, and 47.4408, respectively. However, our goal is to identify the optimal α that minimizes the spectral norm. Therefore, we selected the α values presented in the table to achieve the minimum spectral norm. Hence, for the given $\alpha, \alpha_1 \in \mathbb{C}$ values, the ADVM algorithm shows the proximity to the exact scaled DVM with the minimum spectral norm while preserving the DVM structure along with the execution of low-complexity *Asdvm*16 and *Asdvm* algorithms.

We note here that the derivation of a brute-force approximation for 8×8 DVM using only $\{\pm 1, 0, \pm j\}$ would need an enormous combinatorial search of 5^{64}. Therefore, it becomes essential to explore optimization techniques to achieve a viable solution within a reasonable time frame. On the other hand, this power increase in the matrix search space renders the current methods unfeasible for finding much larger approximation matrices, e.g., $N > 64$. But, using the matrix embedding with the 16-point ADFT along with the *Asdvm*16 and *Asdvm* algorithms allows us to reduce the error between the approximate and exact DVM while greatly decreasing the computational effort needed. With this said, we have numerically shown that, even for some large $N = 64$, we have attained spectral norm ranging from at least 10^{-12} to at most 10^{-1}, as shown in Table 2 with the gain-delay block counts of $8N - 14$, shown in Lemma 3.2. We remind that the gain-delay block count is the key factor in determining the computational complexity.

On the other hand, conventional analog wideband multi-beamformers are based on applying the spatiotemporal DVM to signals captured by antenna arrays. Thus, analog implementations of exact DVM may be difficult due to the need to convert $\mathcal{O}(N^2)$ fractional complex coefficients into RF circuits. Hence, achieving a transformation similar to the N-point DVM while reducing the complexity of delay-gain blocks from $\mathcal{O}(N \log N)$ to $\mathcal{O}(N)$ and keeping minimum errors could enhance bit-shifting operations in digital hardware and software.

V. SIGNAL FLOW GRAPH SHOWING THE APPROXIMATE-DVM ALGORITHM

The algorithms of many fast transforms often arise from a recursive structure. Famously, the FFT has been dubbed the *butterfly algorithm*, based on the symmetric, butterfly-like patterns that appear when the algorithm is represented visually in a graph [27], [31], [32], [61]. Such graphs illustrate how signals or data are processed at each stage of an algorithm, and so-called signal flow graphs. These graphs are used as a tool to design and improve algorithms and can be utilized as a building block for IC design. Thus, we utilize an SFG to denote the 16-point ADVM algorithm in Figure 2. The dashed lines in the figure represent a multiplication by -1, red lines represent a multiplication by $j = \sqrt{-1}$, and dashed red lines represent a multiplication by $-j$. With the SFG, it is straightforward to verify the multiplication and addition complexity of the algorithm for a given value of N, by simply counting the number of convergences as additions, and the number of elements above the arrows as multiplications, or gain-delay block counts. Therefore, we can utilize the SFG for implementing the 16-point approximated scaled DVM algorithm directly in the IC design.

Based on the gain-delay blocks in Lemma 3.2 and the foundational framework of the SFG shown in Fig. 2, the explicit gains (excluding the multiplication by ± 1 and $\pm j$), as well as the delays and the total multiplication counts, i.e., gain-delay block counts in computing the scaled ADVM algorithm, are depicted in the second, third, and final columns of Table 3. The fourth column of the table presents the non-trivial anti-causal counts [6], which correspond to the number of entries in the pre-computed matrix $\check{D}_M$. The trivial anti-causal counts emerge from the multiplication of the

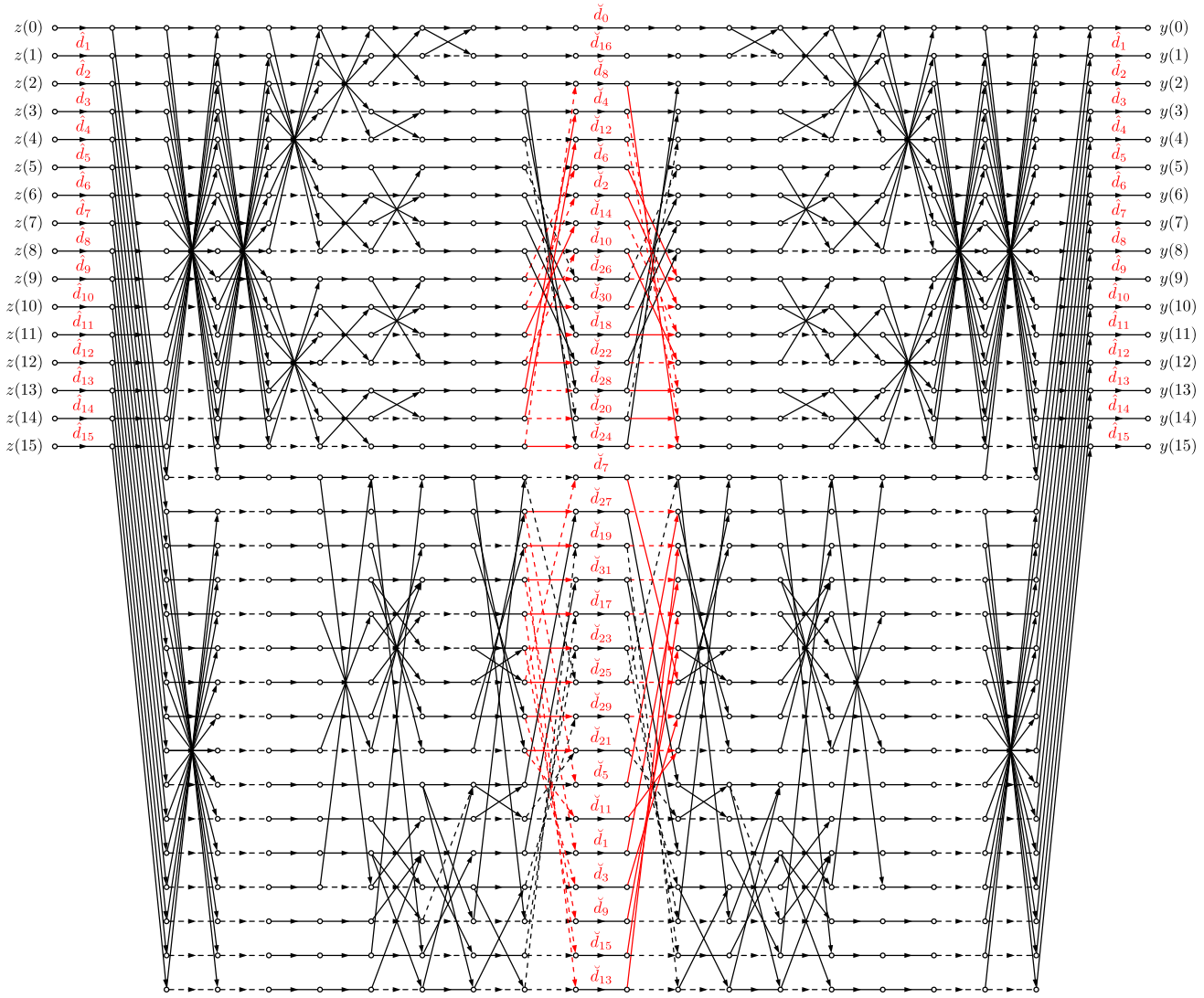


FIGURE 2. The 16-point approximated scaled DVM algorithm *Asdvm16* representing dashed lines for the multiplication by -1, red lines for the multiplication by $j = \sqrt{-1}$, dashed red lines for the multiplication by $-j$, $\check{d}_k$ for the k^{th} component of $\hat{D}_{16}$, and $\check{d}_k$ for the k^{th} component of $\check{D}_{32}$. $\hat{D}_{16}$ and $\check{D}_{32}$ are also shown in Example 3.1.

scaled DFT matrix $\tilde{F}_M$ by a vector c, whose entries are expressed as $e^{jk\omega\tau}$, where τ represents a delay, $k = \frac{p}{2}$, and p denotes a non-negative integer. Since these counts are calculated in the pre-computation stage of the algorithm, we have included only the non-trivial anti-causal counts in Table 3. To efficiently realize trivial anti-causal counts, i.e., the entries of c, we multiply every entry in c by the largest magnitude of the anti-causal term, which is $\left|\alpha^{\frac{-(N-1)^2}{2}}\right|$. This approach is implemented during the pre-computation stage to ensure that anti-causal terms do not pose any challenges for practical implementations. To exemplify, consider the simplest transfer function $P(\omega) = e^{j\omega\tau} + e^{-j\omega\tau}$. We can obtain the same magnitude function by modifying $P(\omega)$ to $P'(\omega) = 1 + e^{-2\omega\tau}$, which is simply the original filter with extra latency τ.

TABLE 3. Gains, delays, and anti-causal counts of the scaled approximate-DVM algorithm.

N	Gains	Delays	Anti-causal	$\#m(Asdvm)$
16	0	30	32	62
32	60	62	64	186
64	244	126	128	498
128	740	254	256	1250
256	1988	510	512	3010
512	4996	1022	1024	7042
1024	12036	2046	2048	16130

VI. REALIZING MULTI-BEAM BEAMFORMERS USING APPROXIMATE-DVM

Analog-domain implementations of DVM beamformers can function in either continuous time (CT) or discrete time (DT). CT realizations offer superior bandwidths,

mitigate challenges associated with high-frequency clock distribution, and prevent temporal aliasing [6], [62]. Nonetheless, their accuracy is compromised because the transfer functions (TFs) of CT TTD elements are highly sensitive to inevitable device mismatches and variations in process, voltage, and temperature (PVT) within IC manufacturing. On the other hand, DT implementations boast reduced bandwidths while achieving greater accuracy. This is due to two key factors: first, TTDs can be calibrated with a precise external clock; second, TF errors arising from device mismatches can be effectively mitigated through various offset-cancellation techniques and dynamic element matching (DEM) methods [6], [63].

We note here that it's too early to make definitive statements regarding power consumption. However, by following sections II to V, we can assert that power consumption will scale almost linearly with N, rather than logarithmically with $N \log N$, for the small number of antenna array elements. Nevertheless, we show the simulated beam patterns for a wideband DVM multi-beam beamformer with $N = 16$ beams followed by the state-of-the-art in DVM beamformers on RF chiplet.

The proposed analog gain-delay blocks were used to realize a complete wideband DVM multi-beam beamformer with $N = 16$ beams. The f_{max} value was set to 1 GHz, resulting in $\tau = 62.5$ ps. A fixed time-delay of $\tau \times (N^2/4) = 4.0$. ns was added to all elements of the ADVM $\widehat{A}_N$ to ensure that the time-delays are causal (i.e., positive); the resulting maximum time-delay is $\tau \times N(N-1)/2 = 7.5$ ns. Fig. 3 compares the simulated beam shapes obtained at input frequencies of 150 MHz, 300 MHz, 600 MHz, and 1 GHz, with ideal beams obtained from a Python implementation of the algorithm. Moreover, the intensity of these 16-beam beamformed signals is visualized in dB scale on a Cartesian grid in Fig. 4.

To enhance the clarity of these beamformed signals and visualize the intensity of particular beams at each frequency in Figure 4, we also arbitrarily selected ADVM beam numbers 0, 4, 8 and 12 and their comparisons to the ideal DVM-form as a function of frequency in Figure 5. Finally, we highlight that due to the errors observed at each frequency, the sidelobe levels are slightly worse (by about 1-2 dB) when compared to the simulated exact DVM beam patterns produced by the 16-point exact scaled DVM algorithm presented in [6]. Therefore, there is about 2 dB worst case tradeoff in sidelobe level when we reduce the arithmetic complexity from $\mathcal{O}(N \log N)$ down to $\mathcal{O}(N)$ by adoption of the ADFT in place of the DFT in the DVM factorization. This discrepancy is anticipated, as the proposed algorithm utilizes an approximated scaled DVM rather than the exact scaled DVM algorithm in [6].

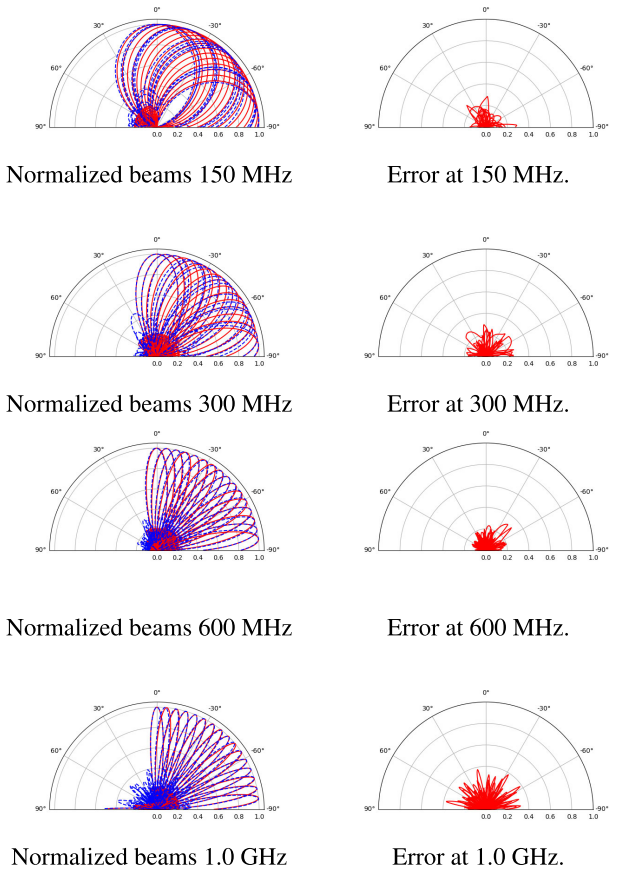
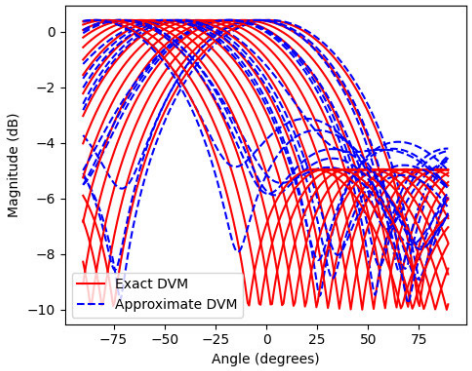


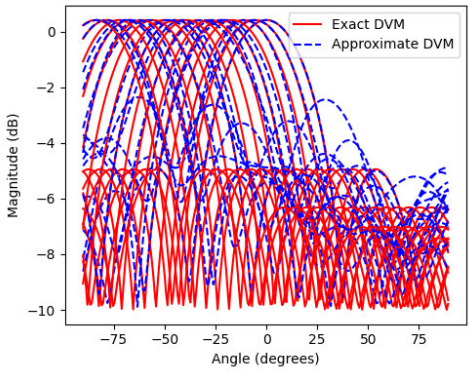
FIGURE 3. Left column: Simulated approximate-DVM beam shapes for a 16-element ULA with $f_{max} = 1$ GHz (blue dashed lines) compared to ideal shapes obtained from simulations (red solid lines). Four input frequencies from 150 MHz to 1 GHz were analyzed. The beam shapes are shown in the left column, while the error between the ideal and approximate shapes is shown in the right column. Each beam is normalized to 0 dB maximum gain.

A. DIGITAL BACKEND

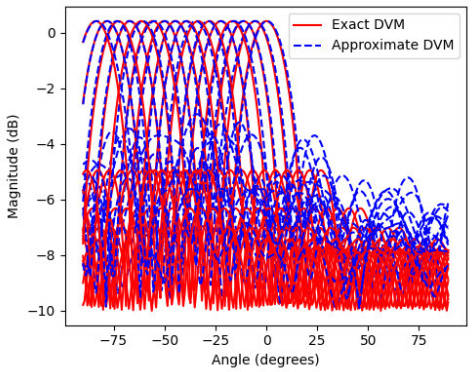
Fig. 7 shows the FPGA platform based on the gain-delay blocks for realizations of wideband multi-beam beamforming networks. For example, consider Intel Stratix-10 AX FPGA Development Kit which contains the 10 AX SoC FPGA with integrated wideband data converters [64]. The on-package data converters support sampling rates of up to 64 GSps via 14nm process technology [65]. The FPGAs contain with two A-Tile Direct RF transceivers (which run at 64 Gsps), 24 H-Tile and 24 E-Tile multi gigabit transceiver channels. The Intel Stratix-10 AX development board provides a state-of-art architecture, offering flexible and programmable compute, robust connectivity and frequency agility across multiple bands going up to 32 GHz. These advanced RF signal processing features make the Intel chiplets [64], [65] well suited for applications in wideband beamforming for radar, signals intelligence, and wireless communication systems. The Intel FPGA system can sample at 64 GSps across 256 phases in a polyphase signal processing framework, where each digital phase supports a clock rate of 256 MHz on the digital programmable FPGA fabric. The chiplets allow 64 GSps sample rates in real-time by adopting this massively polyphase processing approach where each subchannel supports 256 MHz of bandwidth.



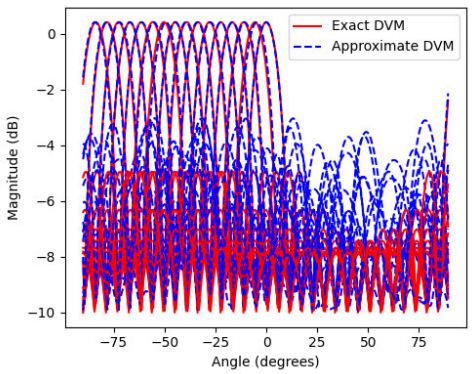
Beams at 150 MHz



Beams at 300 GHz

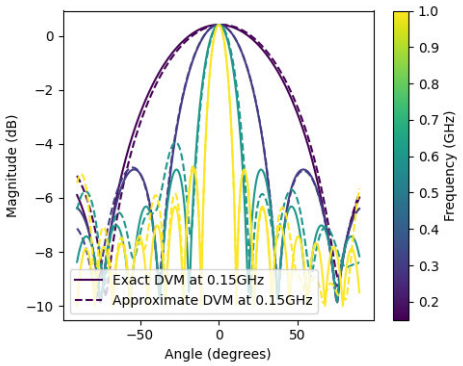


Beams at 600 MHz

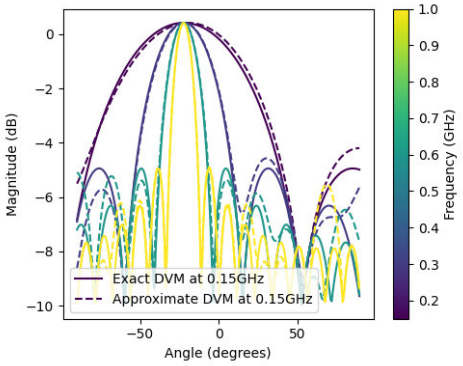


Beams at 1.0 GHz

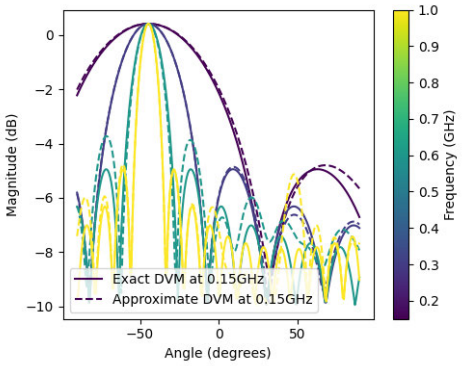
FIGURE 4. Along with the polar plots, beam patterns for 16 beams were simulated from 150 MHz to 1 GHz, in dB scale with the Cartesian grid.



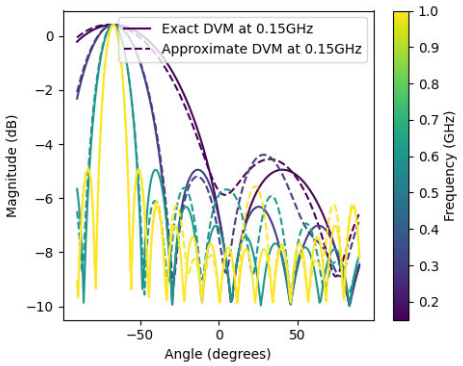
Beam 0



Beam 4



Beam 8



Beam 12

FIGURE 5. A specific beam numbers, i.e., Beam 0, 4, 8, and 12, are selected and plotted across the frequency spectrum at 0.15, 0.3, 0.6, and 1 GHz.

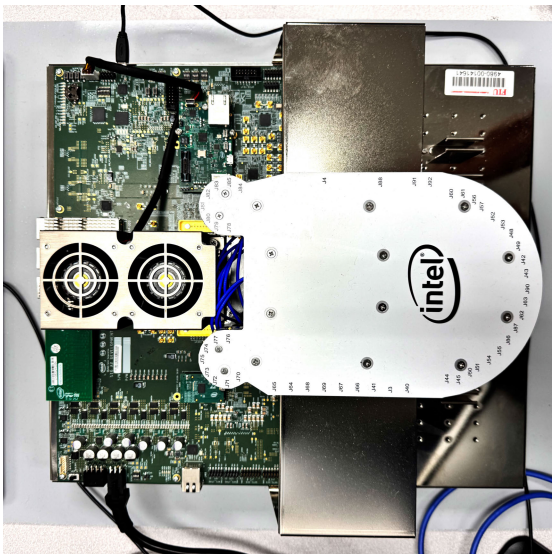


FIGURE 6. Stratix AX-10 FPGA development system supporting two channels at 32 GHz/channel with 64 GS/s sampling rate. A 16-element DVM beamformer with 16 beams will require eight parallel copies of the AX10 board.

Fig. 6 shows an example of a Stratix FPGA development board from Intel Altera having 8 channels, with an instanteneous bandwidth of 4 GHz per channel. The board can support 8 channels at 4 GHz of bandwidth, allowing wideband beamforming in the 0-4 GHz band for an 8-element array, and up to 2 channels at 32 GHz of instantaneous bandwidth. An 8-element array with 32 GHz of instantaneous bandwidth would thus require 4 of the above Intel Stratix boards. We have, at the moment, configured a single channel with the full 32 GHz of instantaneous bandwidth necessary for the full demonstration of the proposed algorithm [66], [67]. However, due to the excessive hardware requirements of 4 boards, we will aim at an experimental demonstration in future work.

B. RF FRONT-END

For receive mode beamforming, the RF front-end per channel has to support the full bandwidth of the signals of interest, without using a down-conversion stage as the DSP digital backend samples at twice the highest frequency component available at RF. Therefore, the RF front-ends are mixedless and directly process the full RF bandwidth within the passband of the particular wideband antenna used. A Vivaldi antenna is a typical sensor for high-sensitivity receive mode array processors with several octaves of RF bandwidth and high gain. Low noise amplifiers (LNAs) must cover the full band. For the DC-32 GHz range, a suitable wideband amplifier such as Minicircuits AVA-054-DG+ with a gain of 17 dB and a maximum output level of 19 dBm may be useful. However, these amplifiers require specialized packaging technology, such as glass-based packages are needed for integration in a printed circuit board (PCB). Transmit mode beamforming is even more challenging with

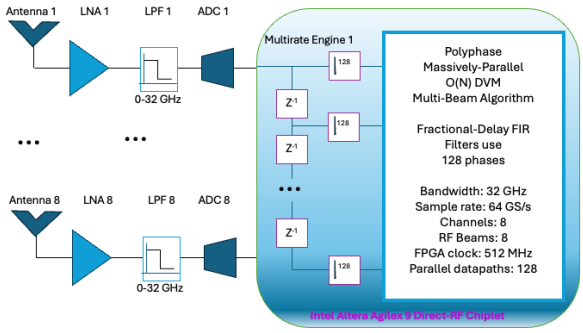


FIGURE 7. The envisaged 8-channel 0-32 GHz $\mathcal{O}(N)$ wideband direct-digital DVM beamformer using Intel Altera Agilex 9 Direct-RF Chiplet technology.

components having wideband behavior, which are rare and also difficult to design. For example, Minicircuits PMA5-83-2W-DG+ has a frequency response of up to 10 GHz with a maximum output power of 31 dBm. Generally speaking, higher power levels in transmit beamforming make system design much more expensive than that of receivers. To wit, the proposed algorithms are better suited for end-to-end RF-IC design that includes multiple wideband channels, data conversion steps, and digital processors on the same chip/package with a multi-chiplet approach to system integration.

C. POTENTIAL ERRORS, UNCERTAINTIES, OR LIMITATIONS

The proposed ADVM factorization has been achieved by adoption of a ADFT in place of the conventional DFT matrix only at 16 beams and execute recursively with the exact DVM algorithm; this was shown to lead to lower complexity of $\mathcal{O}(N)$ compared to the use of FFTs which leads to complexity of $\mathcal{O}(N \log N)$, a small number of antenna array elements. The deviation of the beam shapes (gain, sidelobes) was studied extensively in the mathematical domain, and the level of deviation was deemed to be acceptable for many real-world wireless and RF applications. Nevertheless, a practical measurement in a real-time system with extremely high bandwidth would be desirable from an algorithm evaluation standpoint. Particular designs of antennas having high bandwidth have a pivotal role in beam fidelity and array gain. For example, tightly coupled dipole array (TCDA) antenna technology shows more than 46:1 bandwidth [68]; however, the antenna gains are relatively low compared to conventional antenna technologies such as Vivaldi elements, which show much higher gain but are also much larger in physical size. All in all, the particular antenna type to be used depends on the use case. Other sources of error include the effect of clock jitter in the high-speed ADC/DAC systems, typically based on time-interleaved sampling, necessary for sampling extremely wideband signals. However, the use of a phase-stable reference oscillator operating at 10/100 MHz for synchronizing the internal ADC/DAC clocks via phase

locked loops (PLLs) would be an approach for mitigating jitter issues.

VII. CONCLUSION

Multi-beam true-time delay beamformers have circuit complexity that is proportional to the number of amplifiers and delay lines utilized within the multi-beam beamforming network. A direct realization of an N-beam beamformer has amplifier/gain and delay-line complexity proportional to $\mathcal{O}(N^2)$ for an N-element N-beam system. In recent work, we proposed a low-complexity N-beam true time delay beamformer having amplifier/gain and delay complexity corresponding to the delay Vandermonde matrix (DVM)-vector product with $\mathcal{O}(N \log N)$ complexity algorithm. In this new work, we have proposed an N-point approximate-DVM (ADVM) algorithm that utilizes highly sparse factors in an approximate computing framework, thus even further reducing the gain-delay complexity of the DVM-vector product from our recent work, i.e., $\mathcal{O}(N \log N)$ to a new reduced complexity. Thus, leading to a significant reduction of complexity by a further factor of $\log N$ for the small number of antenna array elements; for example, for 1024 beams, that's a $10\times$ reduction. Hence, we can provide N true time delay beams on an N-element array at amplifier/gain and delay circuit complexity reduced from the brute-force calculation of $\mathcal{O}(N^2)$ down to $\mathcal{O}(N^s)$, where $s \approx 1$. The proposed 16-point ADVM algorithm executes based on a 32-point approximated DFT algorithm with linear complexity. The N-point ADVM algorithm executes recursively with the 16-point ADVM algorithm. The resulting recursion leads to reducing the gain-delay complexity of the DVM-vector product to almost linear order. The ability to form large numbers of true time delay multi-beams over an antenna array aperture having N-elements, with circuit complexity scaling as almost linear $\mathcal{O}(N^s)$ - instead of the usual quadratic growth, can significantly enhance the practical realizability of wideband multi-antenna beamforming systems in the future, including mm-wave and FutureG wireless systems, massive-MIMO wireless systems, radar and electronic warfare, and RF imaging applications. The ADVM algorithm leverages the pre-computation of anti-causal components, and these anti-causal segments of the signal flow graph were implemented using solely delay elements. This approach yields the original filter bank but introduces some additional latency. Moreover, we have simulated beam shapes based on the ADVM algorithm at frequencies of 150 MHz, 300 MHz, 600 MHz, and 1 GHz. These simulated beams were generated through a Python implementation of the ADVM algorithm and subsequently compared with the beam patterns produced by the 16-point exact-scaled DVM algorithm. The simulated beams of the ADVM algorithm showed a slight deviation from the exact DVM algorithm while ensuring the precision of the proposed algorithm. Thus, the signal flow graph, simulated beam patterns, and FPGA overview elegantly demonstrate the simplicity, efficiency, and realizability of the proposed highly sparse factorization of the ADVM, leading to massively parallel $\mathcal{O}(N^s)$ DVM multi-beam algorithm with a significant reduction of cost, chip area, and power consumption in IC design.

APPENDIX

We provide an explicit factorization for the ADFT for 32-points $\widehat{F}_{32}$ [53], [56], [58], where $\widehat{F}_{32} = S_4 \cdot S_3 \cdot S_2 \cdot S_1 \cdot S_0$. Let us define the matrices in the factorization of $\widehat{F}_{32}$ [53], [56], [58] s.t.

$$S_0 = \begin{bmatrix} S_{0,11} & 0_{17} \\ 0_{15} & S_{0,22} \end{bmatrix}, \tag{10}$$

where the block submatrices $S_{0,11}$ and $S_{0,22}$ are given via [53], [58]

$$S_{0,11} = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&-1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&-1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&-1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&-1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&-1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&-1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&-1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

and

$$S_{0,22} = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&-1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&-1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&-1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&-1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&-1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&-1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&-1\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}.$$

The S_1 matrix is defined as

$$S_1 = \begin{bmatrix} S_{1,11} & S_{1,12} \\ S_{1,21} & S_{1,22} \end{bmatrix}, \tag{11}$$

where the block submatrices $S_{1,11}$, $S_{1,12}$ $S_{1,21}$, and $S_{1,22}$ are given via

$$S_{1,11} = \begin{bmatrix}
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&-1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&-1&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&-1&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&-1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&-1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&-1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&-1
\end{bmatrix},$$

$$S_{1,12} = \begin{bmatrix}
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&-1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&-1&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&-1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&-1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&-1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&-1
\end{bmatrix},$$

$$S_{1,21} = \begin{bmatrix} 0&0\\ 0&I_{15} \end{bmatrix},$$

and $S_{1,22}$, as shown at the bottom of the page.

$$S_{1,22} = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&-1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&-1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&-1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&-1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&-1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&-1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&-1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&-1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&-1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&-1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-1
\end{bmatrix}$$

$$S_{2,11} = \begin{bmatrix}
1&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&-1&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&-1&-1&0&0&0&0&0&0&0&0&0&0&0\\
-1&1&0&-1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&1&0&-1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&-1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&1&-1&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&0&-1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&-1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&1&-1&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&-1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&-1
\end{bmatrix}$$

The S_2 is defined as

$$S_2 = \begin{bmatrix} S_{2,11} & 0_{16} \\ 0_{16} & S_{2,22} \end{bmatrix}, \tag{12}$$

where the block submatrices $S_{2,11}$, as shown at the bottom of previous page, and $S_{2,22}$, as shown at the bottom of the page, are given via.

$$S_{2,22} = \begin{bmatrix}
-1&0&0&0&0&0&0&0&0&0&0&0&-1&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&-1&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&1&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&-1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&-1&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&-1&0&0&0&0&0&0&0&0\\
0&0&1&0&-1&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&1&0&-1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&1\\
-1&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&-1&0&0\\
1&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&-1
\end{bmatrix}$$

$$S_4 = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-j&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-j&0&0&0&0&-1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-j&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&-1&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&-1&0&0&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-j&0&0&0&0&0&0&-1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-j&0&-1&0&0&0&0&0&0&0\\
0&0&0&0&-1&0&0&0&0&0&0&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0&-1\\
0&0&0&0&0&0&0&1&0&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-j&0&0&0&0&0&0&0&0&-1&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&j&0&0&0&0&0&0&0&0&0&0&-1&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0&-1\\
0&0&0&0&-1&0&0&0&0&0&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&j&0&-1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&j&0&0&0&0&-1&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&-1&0&0&0&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-1&0&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&j&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&-j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&j&0&0&0&0&-1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&j&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&j&0&0&0&0&0&0&0&1&0&0&0&0&0&0
\end{bmatrix}$$

The S_3 matrix is defined as

$$S_3 = \begin{bmatrix} S_{3,11} & 0_{16} \\ 0_{16} & S_{3,22} \end{bmatrix}, \tag{13}$$

where the block submatrices $S_{3,11}$ and $S_{3,22}$, as shown at the bottom of the page, are given via

$$S_{3,11} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & I_{14} \end{bmatrix}$$

$$S_{3,22} = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&1&0&0&0&-1&0&0\\
0&1&0&0&0&0&1&0&-1&1&0&0&0&0&0&0\\
0&1&-1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&-1&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&-1&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&-1&-1&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&0&1&0&-1&-1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&0&0&1&0&0&-1\\
0&0&0&0&0&0&0&0&1&-1&0&0&1&0&0&-1\\
0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0\\
1&0&0&0&0&0&0&0&-1&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&-1&0&0\\
0&0&0&0&0&0&0&0&-1&0&0&1&0&0&-1&0
\end{bmatrix}$$

$$F_{32,11} = \begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1-j&1-j&1-j&-j&-j&-j&-j&-j&-1-j&-1-j&-1-j&-1&-1\\
1&1&1-j&-j&-j&-j&-1-j&-1&-1&-1&-1+j&j&j&j&1+j&1\\
1&1-j&-j&-j&-1-j&-1&-1&-1+j&j&1+j&1&1&1-j&-j&-j&-1-j\\
1&1-j&-j&-1-j&-1&-1+j&j&1+j&1&1-j&-j&-1-j&-1&-1+j&j&1+j\\
1&1-j&-j&-1&-1+j&j&1&1-j&-j&-1-j&-1&j&1+j&1&-j&-1-j\\
1&-j&-1-j&-1&j&1&1-j&-j&-1&j&1+j&1&-j&-1&-1+j&j\\
1&-j&-1&-1+j&1+j&1-j&-j&-1&j&1&-j&-1-j&-1+j&1+j&1&-j\\
1&-j&-1&j&1&-j&-1&j&1&-j&-1&j&1&-j&-1&j\\
1&-j&-1&1+j&1-j&-1-j&j&1&-j&-1&j&1-j&-1-j&-1+j&1&-j\\
1&-j&-1+j&1&-j&-1&1+j&-j&-1&j&1-j&-1&j&1&-1-j&j\\
1&-1-j&j&1&-1-j&j&1&-1-j&j&1-j&-1&j&1-j&-1&j&1-j\\
1&-1-j&j&1-j&-1&1+j&-j&-1+j&1&-1-j&j&1-j&-1&1+j&-j&-1+j\\
1&-1-j&j&-j&-1+j&1&-1&1+j&-j&-1+j&1&-1&1+j&-j&j&1-j\\
1&-1&1+j&-j&j&-j&-1+j&1&-1&1&-1-j&j&-j&j&1-j&-1\\
1&-1&1&-1-j&1+j&-1-j&j&-j&j&-j&j&1-j&-1+j&1-j&-1&1
\end{bmatrix}$$

$$F_{32,12} = \begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
-1&-1&-1&-1+j&-1+j&-1+j&j&j&j&j&j&1+j&1+j&1+j&1&1\\
1&1&1-j&-j&-j&-j&-1-j&-1&-1&-1&-1+j&j&j&j&1+j&1\\
-1&-1+j&j&j&1+j&1&1&1-j&-j&-1-j&-1&-1&-1+j&j&j&1+j\\
1&1-j&-j&-1-j&-1&-1+j&j&1+j&1&1-j&-j&-1-j&-1&-1+j&j&1+j\\
-1&-1+j&j&1&1-j&-j&-1&-1+j&j&1+j&1&-j&-1-j&-1&j&1+j\\
1&-j&-1-j&-1&j&1&1-j&-j&-1&j&1+j&1&-j&-1&-1+j&j\\
-1&j&1&1-j&-1-j&-1+j&j&1&-j&-1&j&1+j&1-j&-1-j&-1&j\\
1&-j&-1&j&1&-j&-1&j&1&-j&-1&j&1&-j&-1&j\\
-1&j&1&-1-j&-1+j&1+j&-j&-1&j&1&-j&-1+j&1+j&1-j&-1&j\\
1&-j&-1+j&1&-j&-1&1+j&-j&-1&j&1-j&-1&j&1&-1-j&j\\
-1&1+j&-j&-1&1+j&-j&-1&1+j&-j&-1+j&1&-j&-1+j&1&-j&-1+j\\
1&-1-j&j&1-j&-1&1+j&-j&-1+j&1&-1-j&j&1-j&-1&1+j&-j&-1+j\\
-1&1+j&-j&j&1-j&-1&1&-1-j&j&1-j&-1&1&-1-j&j&-j&-1+j\\
1&-1&1+j&-j&j&-j&-1+j&1&-1&1&-1-j&j&-j&j&1-j&-1\\
-1&1&-1&1+j&-1-j&1+j&-j&j&-j&j&-j&-1+j&1-j&-1+j&1&-1
\end{bmatrix}$$

$$F_{32,21} = \begin{bmatrix}
1&-1&1&-1&1&-1&1&-1&1&-1&1&-1&1&-1&1&-1\\
1&-1&1&-1+j&1-j&-1+j&-j&j&-j&j&-j&1+j&-1-j&1+j&-1&1\\
1&-1&1-j&j&-j&j&-1-j&1&-1&1&-1+j&-j&j&-j&1+j&-1\\
1&-1+j&-j&j&-1-j&1&-1&1-j&j&-1-j&1&-1&1-j&j&-j&1+j\\
1&-1+j&-j&1+j&-1&1-j&j&-1-j&1&-1+j&-j&1+j&-1&1-j&j&-1-j\\
1&-1+j&-j&1&-1+j&-j&1&-1+j&-j&1+j&-1&-j&1+j&-1&-j&1+j\\
1&j&-1-j&1&j&-1&1-j&j&-1&-j&1+j&-1&-j&1&-1+j&-j\\
1&j&-1&1-j&1+j&-1+j&-j&1&j&-1&-j&1+j&-1+j&-1-j&1&j\\
1&j&-1&-j&1&j&-1&-j&1&j&-1&-j&1&j&-1&-j\\
1&j&-1&-1-j&1-j&1+j&j&-1&-j&1&j&-1+j&-1-j&1-j&1&j\\
1&j&-1+j&-1&-j&1&1+j&j&-1&-j&1-j&1&j&-1&-1-j&-j\\
1&1+j&j&-1&-1-j&-j&1&1+j&j&-1+j&-1&-j&1-j&1&j&-1+j\\
1&1+j&j&-1+j&-1&-1-j&-j&1-j&1&1+j&j&-1+j&-1&-1-j&-j&1-j\\
1&1+j&j&j&-1+j&-1&-1&-1-j&-j&1-j&1&1&1+j&j&j&-1+j\\
1&1&1+j&j&j&j&-1+j&-1&-1&-1&-1-j&-j&-j&-j&1-j&1\\
1&1&1&1+j&1+j&1+j&j&j&j&j&j&-1+j&-1+j&-1+j&-1&-1
\end{bmatrix}$$

$$F_{32,22} = \begin{bmatrix}
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
-1 & 1 & -1 & 1-j & -1+j & 1-j & j & -j & j & -j & j & -1-j & 1+j & -1-j & 1 & -1 \\
1 & -1 & 1-j & j & -j & j & -1-j & 1 & -1 & 1 & -1+j & -j & j & -j & 1+j & -1 \\
-1 & 1-j & j & -j & 1+j & -1 & 1 & -1+j & -j & 1+j & -1 & 1 & -1+j & -j & j & -1-j \\
1 & -1+j & -j & 1+j & -1 & 1-j & j & -1-j & 1 & -1+j & -j & 1+j & -1 & 1-j & j & -1-j \\
-1 & 1-j & j & -1 & 1-j & j & -1 & 1-j & j & -1-j & 1 & j & -1-j & 1 & j & -1-j \\
1 & j & -1-j & 1 & j & -1 & 1-j & j & -1 & -j & 1+j & -1 & -j & 1 & -1+j & -j \\
-1 & -j & 1 & -1+j & -1-j & 1-j & j & -1 & -j & 1 & j & -1-j & 1-j & 1+j & -1 & -j \\
1 & j & -1 & -j & 1 & j & -1 & -j & 1 & j & -1 & -j & 1 & j & -1 & -j \\
-1 & -j & 1 & 1+j & -1+j & -1-j & -j & 1 & j & -1 & -j & 1-j & 1+j & -1+j & -1 & -j \\
1 & j & -1+j & -1 & -j & 1 & 1+j & j & -1 & -j & 1-j & 1 & j & -1 & -1-j & -j \\
-1 & -1-j & -j & 1 & 1+j & j & -1 & -1-j & -j & 1-j & 1 & j & -1+j & -1 & -j & 1-j \\
1 & 1+j & j & -1+j & -1 & -1-j & -j & 1-j & 1 & 1+j & j & -1+j & -1 & -1-j & -j & 1-j \\
-1 & -1-j & -j & -j & 1-j & 1 & 1 & 1+j & j & -1+j & -1 & -1 & -1-j & -j & -j & 1-j \\
1 & 1 & 1+j & j & j & j & -1+j & -1 & -1 & -1 & -1-j & -j & -j & -j & 1-j & 1 \\
-1 & -1 & -1 & -1-j & -1-j & -1-j & -j & -j & -j & -j & -j & 1-j & 1-j & 1-j & 1 & 1
\end{bmatrix}$$

The S_4 matrix is defined as shown at the bottom of the 16 page.

We also denote the explicit approximated 32-point DFT matrix as follows [53] and [58]

$$\widehat{F}_{32} = \begin{bmatrix} F_{32,11} & F_{32,12} \\ F_{32,21} & F_{32,22} \end{bmatrix}, \tag{14}$$

where [53], [58]. $F_{32,11}$, $F_{32,12}$, and $F_{32,21}$, as shown at the bottom of the previous page, and $F_{32,22}$, as shown at the top of the page.

REFERENCES

[1] T. S. Rappaport, Y. Xing, G. R. MacCartney, A. F. Molisch, E. Mellios, and J. Zhang, "Overview of millimeter wave communications for fifth-generation (5G) wireless Networks—With a focus on propagation models," *IEEE Trans. Antennas Propag.*, vol. 65, no. 12, pp. 6213–6230, Dec. 2017.

[2] T. S. Rappaport, Y. Xing, O. Kanhere, S. Ju, A. Madanayake, S. Mandal, A. Alkhateeb, and G. C. Trichopoulos, "Wireless communications and applications above 100 GHz: Opportunities and challenges for 6G and beyond," *IEEE Access*, vol. 7, pp. 78729–78757, 2019.

[3] T. Long, Y. Li, W. Zhang, Q. Liu, X. Chen, W. Tian, and X. Yang, *Wideband Radar*. Singapore: Springer, 2022, ch. 6.

[4] S. Perera, V. Ariyarathna, N. Udayanga, A. Madanayake, G. Wu, L. Belostotski, R. Cintra, and T. Rappaport, "Wideband N-beam arrays with low-complexity algorithms and mixed-signal integrated circuits," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 2, pp. 368–382, Feb. 2018.

[5] S. M. Perera, A. Madanayake, and R. J. Cintra, "Efficient and self-recursive delay Vandermonde algorithm for multi-beam antenna arrays," *IEEE Open J. Signal Process.*, vol. 1, pp. 64–76, 2020.

[6] S. M. Perera, L. Lingsch, A. Madanayake, S. Mandal, and N. Mastronardi, "A fast DVM algorithm for wideband time-delay multi-beam beamformers," *IEEE Trans. Signal Process.*, vol. 70, pp. 5913–5925, 2022.

[7] W. Hong, Z. H. Jiang, C. Yu, D. Hou, H. Wang, C. Guo, Y. Hu, L. Kuai, Y. Yu, Z. Jiang, Z. Chen, J. Chen, Z. Yu, J. Zhai, N. Zhang, L. Tian, F. Wu, G. Yang, Z.-C. Hao, and J. Y. Zhou, "The role of millimeter-wave technologies in 5G/6G wireless communications," *IEEE J. Microw.*, vol. 1, no. 1, pp. 101–122, Jan. 2021.

[8] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[9] I. I. Immoreev and P. G. S. D. V. Fedotov, "Ultra wideband radar systems: Advantages and disadvantages," in *Proc. IEEE Conf. Ultra Wideband Syst. Technol.*, Mar. 2002, pp. 201–205. [Online]. Available: https://api.semanticscholar.org/CorpusID

[10] S. Kulasekera, A. Madanayake, C. Wijenayake, F. M. Bayer, D. Suarez, and R. J. Cintra, "Multi-beam 8×8 RF aperture digital beamformers using multiplierless 2-D FFT approximations," in *Proc. Moratuwa Eng. Res. Conf. (MERCon)*, Apr. 2015, pp. 260–264.

[11] A. M. Elbir, K. V. Mishra, S. A. Vorobyov, and R. W. Heath, "Twenty-five years of advances in beamforming: From convex and nonconvex optimization to learning techniques," *IEEE Signal Process. Mag.*, vol. 40, no. 4, pp. 118–131, Jun. 2023.

[12] T. E. Bogale, X. Wang, and L. B. Le. (2017). *Mmwave Communication Enabling Techniques for 5G Wireless Systems*. [Online]. Available: https://api.semanticscholar.org/CorpusID

[13] S. Kang, M. Mezzavilla, S. Rangan, A. Madanayake, S. B. Venkatakrishnan, G. Hellbourg, M. Ghosh, H. Rahmani, and A. Dhananjay, "Cellular wireless networks in the upper mid-band," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 2058–2075, 2024.

[14] J. Zhang, W. Liu, C. Gu, S. S. Gao, and Q. Luo, "Multi-beam multiplexing design for arbitrary directions based on the interleaved subarray architecture," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 11220–11232, Oct. 2020.

[15] V. Ariyarathna, A. Madanayake, X. Tang, D. Coelho, R. J. Cintra, L. Belostotski, S. Mandal, and T. S. Rappaport, "Analog approximate-FFT 8/16-beam algorithms, architectures and CMOS circuits for 5G beamforming MIMO transceivers," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 466–479, Sep. 2018.

[16] S. Shahbazpanahi and Y. Jing. (2018). *Recent Advances in Network Beamforming*. [Online]. Available: https://api.semanticscholar.org/CorpusID

[17] C. Wijenayake, Y. Xu, A. Madanayake, L. Belostotski, and L. T. Bruton, "Rf analog beamforming fan filters using CMOS all-pass time delay approximations," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 5, pp. 1061–1073, May 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID

[18] M. Viberg and H. Krim, "Two decades of array signal processing," *IEEE Signal Process. Mag.*, vol. 13, no. 4, pp. 67–94, Jul. 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID

[19] S. M. Perera, A. Madanayake, and R. J. Cintra, "Radix-2 self-recursive algorithms for vandermonde-type matrices and true-time-delay multi-beam antenna arrays," *IEEE Access*, vol. 8, pp. 25498–25508, 2020.

[20] T.-S. Chu and H. Hashemi, "True-Time-Delay-Based multi-beam arrays," *IEEE Trans. Microw. Theory Techn.*, vol. 61, no. 8, pp. 3072–3082, Aug. 2013.

[21] W. Rotman and R. Turner, "Wide-angle microwave lens for line source applications," *IEEE Trans. Antennas Propag.*, vol. AP-11, no. 6, pp. 623–632, Nov. 1963. [Online]. Available: https://api.semanticscholar.org/CorpusID

[22] N. Tiwari and T. R. Rao, "A switched beam antenna array with Butler matrix network using substrate integrated waveguide technology for 60 GHz communications," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Aug. 2015, pp. 2152–2157. [Online]. Available: https://api.semanticscholar.org/CorpusID

[23] D. H. Archer and M. J. Maybell, "Rotman lens development history at raytheon electronic warfare systems 1967–1995," in *Proc. IEEE Antennas Propag. Soc. Int. Symp.*, vol. 2B, Jul. 2005, pp. 31–34. [Online]. Available: https://api.semanticscholar.org/CorpusID

[24] X. Mo, W. Ma, L. Gui, L. Zhang, and X. Sang, "Beamspace channel estimation with beam squint effect for the millimeter-wave MIMO-OFDM systems," *IEEE Access*, vol. 9, pp. 153037–153049, 2021.

[25] B. Wang, M. Jian, F. Gao, G. Y. Li, and H. Lin, "Beam squint and channel estimation for wideband mmWave massive MIMO-OFDM systems," *IEEE Trans. Signal Process.*, vol. 67, no. 23, pp. 5893–5908, Dec. 2019, doi: 10.1109/TSP.2019.2949502.

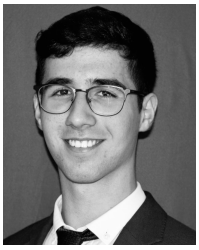
[26] L. Portella, D. F. G. Coelho, F. M. Bayer, A. Madanayake, and R. J. Cintra, "Radix-N algorithm for computing N^{2^n}-point DFT approximations," *IEEE Signal Process. Lett.*, vol. 29, pp. 1838–1842, 2022.
[27] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, p. 297, Apr. 1965.
[28] J. W. Cooley, "The re-discovery of the fast Fourier transform algorithm," *Mikrochimica Acta*, vol. 93, nos. 1–6, pp. 33–45, Jan. 1987.
[29] R. Shirbhate, T. Panse, and C. Ralekar, "Design of parallel FFT architecture using cooley Tukey algorithm," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2015, pp. 0574–0578.
[30] R. E. Blahut, *Fast Algorithms for the Discrete Fourier Transform*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[31] R. Yavne, "An economical method for calculating the discrete Fourier transform," in *Proc. AFIPS*, 1968, p. 115.
[32] P. Duhamel and M. Vetterli, "Fast Fourier transforms: A tutorial review and a state of the art," *Signal Process.*, vol. 19, no. 4, pp. 259–299, 1990.
[33] J. Y. Stein, *Digital Signal Processing: A Computer Science Perspective*. NY, USA: Wiley, 2000.
[34] S. G. Johnson and M. Frigo, "A modified split-radix FFT with fewer arithmetic operations," *IEEE Trans. Signal Process.*, vol. 55, no. 1, pp. 111–119, Jan. 2007.
[35] R. J. Cintra, S. Duffner, C. Garcia, and A. Leite, "Low-complexity approximate convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 5981–5992, Dec. 2018.
[36] X. Huang, Q. Wang, S. Lu, R. Hao, S. Mei, and J. Liu, "NUMA-aware FFT-based convolution on ARMv8 many-core CPUs," in *Proc. IEEE Intl Conf Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Sep. 2021, pp. 1019–1026.
[37] T. Lou, M. S. Park, M. Ramezanali, and V. Tang, "Fnetar: Mixing tokens with autoregressive Fourier transforms," *ArXiv*, vol. abs/2107.10932, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236318364
[38] Y. Rao, W. Zhao, Z. Zhu, J. Lu, and J. Zhou, "Global filter networks for image classification," 2021, *arXiv:2107.00645*.
[39] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "FNet: Mixing tokens with fourier transforms," in *Proc. Conf. North American Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.* Seattle, CA, USA: Association for Computational Linguistics, 2022, pp. 4296–4313.
[40] H. Aluvihare, C. Shanahan, S. M. Perera, S. Sivasankar, U. Kumarasiri, A. Madanayake, and X. Li, "A low-complexity LSTM network to realize multibeam beamforming," in *Proc. IEEE Int. Conf. Wireless Space Extreme Environments (WiSEE)*, Dec. 2024, pp. 11–16.
[41] H. Aluvihare, S. Sivasankar, X. Li, A. Madanayake, and S. M. Perera, "A low-complexity structured neural network approach to intelligently realize wideband multi-beam beamformers," *IEEE J. Radio Freq. Identificat.*, 2025.
[42] V. A. Coutinho, V. Ariyarathna, D. F. G. Coelho, R. J. Cintral, and A. Madanayake, "An 8-Beam 2.4 GHz digital array receiver based on a fast multiplierless spatial DFT approximation," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Jun. 2018, pp. 1538–1541.
[43] V. Ariyarathna, S. Kulasekera, A. Madanayake, K.-S. Lee, D. Suarez, R. J. Cintra, F. M. Bayer, and L. Belostotski, "Multi-beam 4 GHz microwave apertures using current-mode DFT approximation on 65 nm CMOS," in *IEEE MTT-S Int. Microw. Symp. Dig.*, May 2015, pp. 1–4.
[44] R. Palisetty, G. Eappen, J. L. G. Rios, J. C. M. Duncan, S. Domouchtsidis, S. Chatzinotas, B. Ottersten, B. Cortazar, S. D'Addio, and P. Angeletti, "Area-power analysis of FFT based digital beamforming for GEO, MEO, and LEO scenarios," in *Proc. IEEE 95th Veh. Technol. Conference: (VTC-Spring)*, Jun. 2022, pp. 1–5. [Online]. Available: https://api.semanticscholar.org/CorpusID
[45] V. Ariyarathna, D. F. G. Coelho, S. Pulipati, R. J. Cintra, F. M. Bayer, V. S. Dimitrov, and A. Madanayake, "Multibeam digital array receiver using a 16-point multiplierless DFT approximation," *IEEE Trans. Antennas Propag.*, vol. 67, no. 2, pp. 925–933, Feb. 2019.
[46] S. Sun, T. S. Rappaport, R. W. Heath, A. Nix, and S. Rangan, "MIMO for millimeter-wave wireless communications: Beamforming, spatial multiplexing, or both?" *IEEE Commun. Mag.*, vol. 52, no. 12, pp. 110–121, Dec. 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID
[47] H. Zhao, S. Mandal, V. Ariyarathna, A. Madanayake, and R. J. Cintra, "An offset-canceling approximate-DFT beamforming architecture for wireless transceivers," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5. [Online]. Available: https://api.semanticscholar.org/CorpusID
[48] R. W. Heath, N. González-Prelcic, S. Rangan, W. Roh, and A. M. Sayeed, "An overview of signal processing techniques for millimeter wave MIMO systems," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 3, pp. 436–453, Apr. 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID
[49] R. Méndez-Rial, C. Rusu, N. González-Prelcic, A. Alkhateeb, and R. W. Heath, "Hybrid MIMO architectures for millimeter wave communications: Phase shifters or switches?" *IEEE Access*, vol. 4, pp. 247–267, 2016.
[50] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing Point of View," *Proc. IEEE*, vol. 108, no. 3, pp. 394–399, Mar. 2020.
[51] J. Du, K. Chen, P. Yin, C. Yan, and W. Liu, "Design of an approximate FFT processor based on approximate complex multipliers," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2021, pp. 308–313.
[52] S. Queiroz, J. P. Vilela, and E. Monteiro, "Is FFT fast enough for beyond 5G communications? A throughput-complexity analysis for OFDM signals," *IEEE Access*, vol. 10, pp. 104436–104448, 2022.
[53] A. Madanayake, V. Ariyarathna, S. Madishetty, S. Pulipati, R. J. Cintra, D. Coelho, R. Oliviera, F. M. Bayer, L. Belostotski, S. Mandal, and T. S. Rappaport, "Towards a low-SWaP 1024-beam digital array: A 32-beam subsystem at 5.8 GHz," *IEEE Trans. Antennas Propag.*, vol. 68, no. 2, pp. 900–912, Feb. 2020.
[54] N. H. Silva, A. Madanayake, S. Mandal, and J. Delva, "Inductorless analog time-delays for nonlinear RF signal processing circuits in 180nm CMOS," in *Proc. IEEE Microw., Antennas, Propag. Conf. (MAPCON)*, Dec. 2023, pp. 1–5.
[55] S. Kulasekera, A. Madanayake, D. Suarez, R. J. Cintra, and F. M. Bayer, "Multi-beam receiver apertures using multiplierless 8-point approximate DFT," in *Proc. IEEE Radar Conf. (RadarCon)*, May 2015, pp. 1244–1249.
[56] R. J. Cintra, "An approximation for the 32-point discrete Fourier transform," 2024, *arXiv:2407.12708*.
[57] V. A. Coutinho, V. Ariyarathna, D. F. G. Coelho, S. K. Pulipati, R. J. Cintra, A. Madanayake, F. M. Bayer, and V. S. Dimitrov, "A low-SWaP 16-beam 2.4 GHz digital phased array receiver using DFT approximation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 5, pp. 3645–3654, Oct. 2020.
[58] A. Madanayake, K. Lawrance, B. U. Kumarasiri, S. Sivasankar, T. K. Gunaratne, C. U. S. Edussooriya, and R. J. Cintra, "Design of multichannel spectrum intelligence systems using approximate DFT algorithm for antenna array based spectrum perception applications," *Algorithms*, vol. 17, p. 338, 2024.
[59] A. Madanayake, R. J. Cintra, N. Akram, V. Ariyarathna, S. Mandal, V. A. Coutinho, F. M. Bayer, D. Coelho, and T. S. Rappaport, "Fast radix-32 approximate DFTs for 1024-beam digital RF beamforming," *IEEE Access*, vol. 8, pp. 96613–96627, 2020.
[60] S. M. Perera, G. Rathnasekara, and A. Madanayake, "Thiran filters for wideband DSP-based multi-beam true time delay RF sensing applications," *Sensors*, vol. 24, no. 2, p. 576, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID
[61] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.
[62] K. Spoof, V. Unnikrishnan, M. Zahra, K. Stadius, M. Kosunen, and J. Ryynänen, "True-Time-Delay beamforming receiver with RF re-sampling," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 12, pp. 4457–4469, Dec. 2020.
[63] C. C. Enz and G. C. Temes, "Circuit techniques for reducing the effects of op-amp imperfections: Autozeroing, correlated double sampling, and chopper stabilization," *Proc. IEEE*, vol. 84, no. 11, pp. 1584–1614, 1996.
[64] *Intel Stratix AX-10 FPGA Description*. Accessed: Jan. 1, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/fpga/stratix/10/ax.html
[65] A. Raghu and B. Jenkins, "Addressing the need of all-digital beamforming systems," Tech. Rep., 2024.
[66] B. Gayanath, H. Weerasooriya, M. Nilan, K. Lawrance, R. Cintra, and A. Madanayake, "Direct-RF 64 GS/s Wideband Approximate DFT Analysis Filterbanks for Spectrum AI-Perception," in *Proc. Int. Appl. Comput. Electromagn. Soc. Symp. (ACES)*, 2025, pp. 1–12.
[67] B. Gayanath, K. Karunanayke, H. Abdellatif, S. B. Venkatakrishnan, E. A. Alwan, J. Jornet, A. Singh, and A. Madanayake, "D-band SDR with 64 GHz B/W on COTS chiplets," *IEEE Access*, vol. 13, pp. 94488-94507, 2025.
[68] S. B. Venkatakrishnan and J. Volakis, "Techniques to improve TCDA bandwidth beyond 46:1," in *Proc. Int. Appl. Comput. Electromagn. Soc. Symp. (ACES)*, Mar. 2023, pp. 1–2.

SIRANI M. PERERA received the B.S. degree (Hons.) in mathematics from the University of Sri Jayewardenepura, Sri Lanka, in 2004, the Master of Advanced Studies (MASt) degree (Hons.) in mathematics from the University of Cambridge, U.K., in 2006, and the Ph.D. degree in mathematics from the University of Connecticut, USA, in 2012.

She became the first Sri Lankan mathematics undergraduate to be selected by the Center for Mathematical Sciences, University of Cambridge, U.K., with a full Cambridge Commonwealth Trust Scholarship. From 2007 to 2008, she was a Lecturer of mathematics with the University of Colombo, Sri Lanka. Later, she joined Embry-Riddle Aeronautical University (ERAU) and has been working as an Associate Professor of mathematics. In 2023, she was selected as A Convergence Research (CORE) Fellow at the CORE Institute: The University of California San Diego, under the NSF Convergence Accelerator Program. She is the inventor of the patent titled "Reduced Multiplicative Complexity Discrete Cosine Transform (DCT) Circuitry" by the USPTO in January 2021. Her research interests include applied linear algebra, computational mathematics, and scientific computing, along with the development of low-complexity classical and machine learning algorithms.

Dr. Perera serves as the Principal Investigator (PI) and the Co-PI on multiple National Science Foundation (NSF) awards, including the Division of Mathematical Sciences (DMS), the Division of Electrical, Communications, and Cyber Systems(ECCS), and the Division of Undergraduate Education. In 2024, she was recognized with the Research and Scholarly Excellence Award by the College of Arts and Sciences, ERAU, and also named as a Rising Star in Science by the Academy of Science, Engineering, and Medicine in Florida (ASEMFL).

LEVI LINGSCH is currently pursuing the Ph.D. degree with the ETH AI Center, ETH Zurich. He is a member of seminar for applied mathematics with ETH Zürich. His research interests include machine learning for scientific computing, specifically with approaches in neural operators and physics informed machine learning. More recently, he has focused on applications of these technologies for foundation models in weather and climate, in collaboration with the IBM Research Laboratory, Zürich, Switzerland.



ALP TUZTAS is currently pursuing the B.Sc. degree in aerospace engineering with Embry-Riddle Aeronautical University, USA. His research interests include signal processing, radar systems, and fast algorithms.

ARJUNA MADANAYAKE (Senior Member, IEEE) received the B.Sc. degree in electronic and telecommunication engineering (Hons.) from the University of Moratuwa, Sri Lanka, in 2001, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Calgary, Alberta, Canada, in 2004 and 2008, respectively. He joined the University of Akron, as a tenure track Assistant Professor, in 2009, and he moved to Florida International University (FIU), Miami, FL, USA, in 2018, as a tenured Associate Professor. He is currently a tenured Associate Professor of electrical and computer engineering with FIU. He was selected as the Most Outstanding Candidate in Electrical Engineering and Computing Sciences at the NSERC 2009 Postdoctoral Fellowship Competition. His research has been recently supported by multiple awards from NSF, ONR, DARPA, NIH, and other agencies.