A Fast DVM Algorithm for Wideband Time-Delay Multi-Beam Beamformers

Sirani M. Perera, Levi Lingsch, Arjuna Madanayake, *Member, IEEE,*, Soumyajit Mandal, *Senior Member, IEEE,* and Nicola Mastronardi.

***Abstract*—This paper presents a sparse factorization for the delay Vandermonde matrix (DVM) along with fast, exact, radix-2, and recursive algorithms to compute the DVM-vector product for wideband multi-beam antenna arrays. The proposed algorithms enable low-complexity wideband beamformers in emerging millimeter-wave wireless communication networks by reducing the complexity of N-beam wideband beamforming from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$, where $N = 2^r (r \geq 1)$. As a result, the algorithms are faster than the brute-force computation of the DVM-vector product and more efficient than the direct realization of true-time-delay-based multi-beam beamformers. The proposed low-complexity algorithms' signal flow graph (SFG) is also presented to highlight their suitability for hardware implementations. The 2-D frequency responses of DVM-based beamformers are explained through an array signal processing example. Simulation results suggest that integrated circuit (IC) implementations of the SFG significantly reduce chip area and power consumption.**

***Index Terms*—Delay Vandermonde matrix, Radix-2, fast recursive algorithms, algorithmic complexity, millimeter wave, wireless communications, multi-beam beamforming.**

I. Introduction

Wideband beamforming is important for cognitive radios, legacy wireless networks, emerging mm-wave (mmW) communication systems, radar, and ultrasound imaging. For example, beamforming allows high-capacity wireless connections to be established in multipath environments by overcoming path loss [1], [2]. Many beamforming techniques can be efficiently implemented using FFT-based algorithms, which may utilize either time-domain pre-FFT schemes or frequency-domain post-FFT schemes [3], [4]. However, use of the FFT results in beams with frequency-dependent axes, and thus poor control over beam orientations for broadband input signals [3]. This fundamental issue is known as the beam-squint problem [5]–[7]. Fortunately, true-time-delay (TTD) beamformers can generate wideband squint-free beams in both analog and digital signal domains. An example of such a wideband squint-free multi-beam beamformer is a passive Rotman lens [8], but such lenses are challenging to model and fabricate at mmW frequencies. Recent advances in silicon technology, mmW circuit/antenna/package design, and beamforming techniques have allowed high-bandwidth active beamforming systems (either digital or analog) to emerge as attractive alternatives. Such TTD beamformers use a Vandermonde structure of delay-based steering vectors to overcome the beam squint problem, resulting in a DVM beamforming matrix [5]–[7], [9].

A single TTD beam is formed by delaying and summing N elements such that constructive interference is emulated along a particular direction. For example, consider receiving a signal at a direction of arrival (DOA) θ, measured counter-clockwise from the broadside direction, using an N-element uniform linear array (ULA). The received signals $u_k(t), k = 1, 2, ...N$, must be combined according to $y(t) = \sum_{k=1}^{N} u_k(t - k\tau)$ where $\tau = \Delta x \sin\theta / c$, Δx is the inter-element spacing of the ULA, and c is the wave speed. In the frequency domain, the TTD beamformer can be viewed as a tapped-delay spatio-temporal 2-D filter $Y(\omega_x, \omega) = U(\omega_x, \omega)[1 + \sum_{k=1}^{N} \alpha^k e^{-j\omega_x k}]$ where filter coefficients are defined using $\alpha = e^{-j\omega\tau} \in \mathbb{C}$. Here, ω_x is the independent spatial frequency variable, and temporal frequency of interest ω is a parameter. The two spatial and temporal frequencies are related via $\omega_x = \omega \sin\theta$ for propagating far-field planar waves received by the array. For a particular frequency ω the directional response against angle ϕ reduces to the normalized form $H(\phi, \omega) = (1/N) \sum_{k=0}^{N} e^{-j\omega k(\sin\theta - \sin\phi)} \in \mathbb{C}$ where θ is the beam angle. Typically, beams are plotted in the polar domain as magnitude functions $\|H(\phi, \omega)\|$ (i.e., array patterns).

Extending the formulation above to multiple output beams requires a matrix-vector representation, where the matrix has a DVM structure with each row containing progressive wideband phase-shifts for a particular beam. Thus, wideband multi-beam beamforming requires computing the multiplication of a DVM $\mathbf{A}_N$ by a vector $\mathbf{x}$, s.t. $\mathbf{y} = \mathbf{A}_N \mathbf{x}$ at times $t \in \mathbb{R}$, where $\mathbf{x}$ and $\mathbf{y}$ are input and output vectors containing time-domain signals $u_k(t)$ and $y_k(t)$ respectively. We denote the 1-D temporal Fourier transform of $u_k(t)$ as $x_k \equiv \int_{-\infty}^{+\infty} u_k(t) e^{-j\omega t} dt$, since adopting a frequency domain variable allows the representation of TTDs of duration T as multiplications by $e^{-j\omega T}$ within the elements of the DVM. Readers are reminded that in practice, signals $u_k(t)$ are fed into a linear time invariant (LTI) system to realize the necessary matrix-vector multiplication. For example, direct realization of this operation requires LTI combinations among N^2 TTDs that describe the elements of

This work was partially supported by the National Science Foundation award numbers 1902283 and 2229473.

Sirani M. Perera is with the Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, Florida 32114, USA (e-mail: pereras2@erau.edu).

Levi Lingsch is with the Department of Aerospace Engineering, Embry-Riddle Aeronautical University, Daytona Beach, Florida 32114, USA (e-mail: lingschl@my.erau.edu).

Arjuna Madanayake is with the Department of Electrical and Computer Engineering, Florida International University, Miami, Florida 33174, USA (e-mail: amadanay@fiu.edu).

Soumyajit Mandal is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida 32611, USA (e-mail: soumyajit@ece.ufl.edu).

Nicola Mastronardi is with Istituto per le Applicazioni del Calcolo, "M. Picone," Consiglio Nazionale delle Ricerche (CNR), Sez. Bari, Via Amendola 122/I, Bari, 70126, Italy. (e-mail: nicola.mastronardi@cnr.it).

the DVM in the temporal frequency domain.

A. Relationship to Prior Work

Our earlier work [10] described entries of the DVM $\mathbf{A}_N$ as coefficients of a TTD-based multi-beam beamformer. Note that $\mathbf{A}_N$ equals the discrete Fourier transform (DFT) matrix only at a single temporal frequency. Since the DVM is a superclass of the DFT matrix, we are generally unable to factorize it to obtain a self-contained and radix-2 factorization, unlike for the DFT matrix. Instead, we had obtained a bidiagonal factorization for the DVM when $N = 4$. Next, we extended the bi-diagonal factorization of the 4×4 DVM into an $N \times N$ DVM in [5]. The bidiagonal factorization of the N-beam DVM in [5] leads to a 60% reduction in circuit complexity for IC implementations based on TTD blocks. Furthermore, this N-beam DVM algorithm uses the product of complex 1-band upper and lower matrices, thus extending upon the results in [11]–[13] by utilizing complex nodes instead of real nodes. The DVM factorizations in [5], [10] do not utilize either quasiseparability and displacement equations as in [14]–[16], or a $\mathcal{O}(N^3)$ algorithm to compute the SVD of polynomial Vandermonde matrices as in [17]. On the other hand, we have addressed error bounds and stability of the DVM algorithm for nodes on, inside, and outside the unit circle in [5] by filling the gaps of the bidiagonal factorization of Vandermonde matrices in [11]–[13]. We recall here that an alternative bidiagonal factorization for Vandermonde matrices having nodes inside the unit circle was presented in [18].

In our later work [7], we developed an exact, efficient (i.e., more efficient than brute-force multiplication of the DVM by a vector, but not radix-2), and self-recursive DVM algorithm by using a matrix factorization of the DVM and a polynomial evaluation associated with its nodes to analyze multi-beam antenna arrays. This exact DVM algorithm can be used to reduce the complexity of RF N-beam analog beamforming systems, but not to $\mathcal{O}(N \log N)$. Although the DVM algorithms proposed in [5], [7], [10] are more efficient than brute force matrix-vector calculation, their order of arithmetic complexity is still significantly greater than $\mathcal{O}(N \log N)$. On the other hand, our work in [6] is based on stable and $\mathcal{O}(N \log N)$ algorithms for Vandermonde matrices having nodes on the unit-circle (not necessarily roots of unity) and also nodes on a circle with the center as the origin and a radius greater than unity. The computational efficiency and rapid convergence of these algorithms enable their use in narrowband communication systems. The existing DVM algorithms are neither radix-2 nor can be utilized for wideband communication systems. Thus, it is necessary to have a DVM algorithm to reduce the algorithmic complexity of generating N-parallel wideband beams from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log N)$ for wideband multi-beam antenna arrays. This paper proposes a novel factorization for the DVM followed by new $\mathcal{O}(N\log N)$ algorithms to compute the DVM-vector product for wideband multi-beam antenna arrays and describes its realization using signal flow graphs and analog ICs. As described earlier, the complexity of the DVM algorithms proposed in our previous work [5], [7], [10] exceed $\mathcal{O}(N \log N)$ operations. Also, the stable and $\mathcal{O}(N \log N)$ DVM algorithms in [6] were proposed for narrowband multibeam beamforming. Moreover, the nodes of the Vandermonde matrices in [6] are a special case, i.e., complex nodes are equally spaced on the unit circle (not necessarily the primitive roots of unity) and any other circle having a radius more than the unity. Thus, an $\mathcal{O}(N \log N)$ DVM algorithm enabling wideband multibeam beamforming is still an open question. More importantly, there is no explicit DVM algorithm that can execute recursively with the well-known FFT algorithm to solve the beam-squint problem. It has been mentioned in [15], [19] that the Vandermonde matrix-vector product can be computed using an FFT-like algorithm. However, an explicit and highly sparse factorization for computing the Vandermonde matrix-vector product using the DFT factorization, and the corresponding algorithms, are missing. On the other hand, one could start from [15], [19] and obtain an explicit and highly sparse factorization for the DVM (having distinct nodes $\{\alpha^k\}_{k=1}^N$ s.t. $\alpha = e^{-j\omega\tau} \in \mathbb{C}$) utilizing the well-known DFT matrix factorization. In this paper, we propose a $\mathcal{O}(N \log N)$ DVM algorithm which recursively executes with the well-known FFT algorithm, thus enabling wideband multibeam beamforming.

B. Overview of the Proposed Approach

There are several mathematical techniques available to derive radix-2 and split-radix FFT algorithms, as described in [20]–[23], [26], [27]. Even though the derivation of size-N DFT into two size-$\frac{N}{2}$ DFTs can be done easily, the extension of this idea to the DVM is cumbersome because the useful DFT matrix properties, like periodicity and unitary, are **not** present in the DVM. However, we propose a radix-2 factorization for the DVM. To do so, we first factor the DVM into a product of diagonal matrices and a Toeplitz matrix. Thus, the problem of computing a $\mathcal{O}(N\log N)$ DVM algorithm is transformed into the multiplication of diagonal and Toeplitz matrices by a vector. Next, we use matrix embedding methods from [28], [29] to transform the Toeplitz matrix into a circulant matrix of double size. Next, the similarity transform of the circulant matrix is obtained using the DFT matrices. Finally, we scale the transformation by rectangular sparse matrices, which compose of identity and zero matrices, to calculate the matrix-vector product using $\mathcal{O}(N \log N)$ operations. The proposed fast DVM algorithm thus solves the longstanding beam-squint problem while reducing the complexity of N-beam wideband beamformers from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$.

The rest of the paper is organized as follows. Section II proposes a sparse factorization for the DVM leading to a fast, exact, radix-2, and recursive algorithm, while Section III derives the arithmetic complexity (quantified using the number of necessary adders and gain-delay blocks) and associated numerical results for the proposed DVM algorithm and compares them with the brute-force matrix-vector product. In Section IV, the proposed algorithm is implemented via signal flow graphs. Section V presents an array processing example using the proposed algorithm. Circuit implementations of the algorithm for wideband wireless transceivers are discussed in Section VI, while Section VII concludes the paper.

II. Sparse Factorizations for the DVMs and Radix-2 DVM Algorithms

In this section, we introduce novel factorizations for DVM and scaled DVM into sparse and orthogonal matrices. Let us first introduce all the notations before discussing factorizations for DVM and scaled DVM matrices.

A. Frequently Used Notations

Here we introduce notations for sparse and orthogonal matrices which will frequently be used in this paper. We first define the DVM by

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

where $N = 2^r (r \geq 1)$, $\{\alpha, \alpha^2, \ldots, \alpha^N\}$ are distinct complex nodes s.t. $\alpha = e^{-j\omega\tau}$, $j^2 = -1$, ω is the temporal frequency, and τ is the delay. The scaled DVM (by a diagonal matrix) is defined as

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

For a given vector $\mathbf{x} = [x_0, x_1, \cdots, x_{N-1}]^T \in R^N$, let us introduce an even-odd permutation matrix $\mathbf{P}_N$ ($N \geq 3$) by

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

We also define the DFT matrix by $\mathbf{F}_N = \frac{1}{\sqrt{N}}\,[w_N^{kl}]_{k,l=0}^{N-1}$ where $w_N = e^{-\frac{2\pi j}{N}}$, a scaled DFT matrix by $\tilde{\mathbf{F}}_N = \sqrt{N}\,\mathbf{F}_N$ and its conjugate transpose by $\mathbf{F}_N^*$, a highly sparse matrix by $\mathbf{J}_{M\times N} = \left[\frac{\mathbf{I}_N}{\mathbf{0}_N}\right]$ where $M = 2N$, $\mathbf{I}_N$ is the identity matrix and $\mathbf{0}_N$ is the zero matrix, a scaled orthogonal matrix by $\mathbf{H}_N = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{I}_{\frac{N}{2}} \\ \grave{\mathbf{D}}_{\frac{N}{2}} & -\grave{\mathbf{D}}_{\frac{N}{2}} \end{bmatrix}$ where $\grave{\mathbf{D}}_{\frac{N}{2}} = \operatorname{diag}\left[w_N^l\right]_{l=0}^{\frac{N}{2}-1}$, and its conjugate transpose by $\mathbf{H}_N^*$, diagonal matrices by $\mathbf{D}_N = \operatorname{diag}[\alpha^k]_{k=0}^{N-1}$, $\hat{\mathbf{D}}_N = \operatorname{diag}\left[\alpha^{\frac{k^2}{2}}\right]_{k=0}^{N-1}$ and $\breve{\mathbf{D}}_M = \operatorname{diag}\left[\tilde{\mathbf{F}}_M\mathbf{c}\right]$ where a circulant matrix $\mathbf{C}_M$ defined by the first column $\mathbf{c}$ s.t. $\mathbf{c} = \left[1, \alpha^{-\frac{1}{2}}, \cdots, \alpha^{-\frac{(N-1)^2}{2}}, 1, \alpha^{-\frac{(N-1)^2}{2}}, \alpha^{-\frac{(N-2)^2}{2}}, \cdots, \alpha^{-\frac{1}{2}}\right]^T$.

B. Sparse and Recursive Factorizations of DVMs

Toeplitz and Hankel matrices can be factored by using Vandermonde and diagonal matrices [38], [39], [41]. In fact, the factorization proposed in [39], [41] is a special case of the Carathéodory parametrization of covariance matrices in [40]. However, the factorization proposed in [38]–[41] requires $\mathcal{O}(N^2)$ operations to decompose Toeplitz and Hankel matrices via Vandermonde matrices. As a result, the factorization proposed in these papers can be used to compute the inverse of the Vandermonde matrix or to solve the system of equations having the Vandermonde as the coefficient matrix with $\mathcal{O}(N^2)$ arithmetic complexity [41]. Thus, the earlier factorization algorithms for Vandermonde matrices do not obtain a sparse factorization (as presented here), and their complexity is significantly greater than $\mathcal{O}(N \log N)$ operations. By contrast, here we propose sparse and recursive factorizations for the delay Vandermonde and scaled delay Vandermonde matrices with $\mathcal{O}(N \log N)$ complexity. To obtain a sparse factorization, we transform the DVM into diagonal, sparse, and Toeplitz matrices followed by the matrix embedding of Toeplitz into a double-sized circulant matrix, as in [28], [29]. Finally, we use the similarity transform of the circulant using the DFT matrices followed by sparse factorization to compute the matrix-vector product with $\mathcal{O}(N \log N)$ complexity.

Before proceeding, we emphasize that the focus of our work is to factorize the Vandermonde-structured DVM into sparse matrices, resulting in a $\mathcal{O}(N \log N)$ matrix-vector multiplication algorithm. Related problems, such as computing the inverse of a Vandermonde matrix (resulting in inversion formulas or inversion algorithms) or solving systems of equations with a Vandermonde matrix as the coefficient matrix [42]–[51], are not considered in this paper.

Theorem II.1. *Let the scaled DVM $\widetilde{\mathbf{A}}_N = [\alpha^{kl}]_{k,l=0}^{N-1}$ be defined by nodes $\{1, \alpha, \alpha^2, \ldots, \alpha^{N-1}\} \in \mathbb{C}$, $N = 2^r$ ($r \geq 1$), and $M = 2N$. Then $\widetilde{\mathbf{A}}_N$ can be factored into*

$$\widetilde{\mathbf{A}}_N = \hat{\mathbf{D}}_N[\mathbf{J}_{M\times N}]^T\mathbf{F}_M^*\breve{\mathbf{D}}_M\mathbf{F}_M\mathbf{J}_{M\times N}\hat{\mathbf{D}}_N \tag{1}$$

Proof. The matrix $\widetilde{\mathbf{A}}_N$ can be factored into a product of matrices s.t.

$$\widetilde{\mathbf{A}}_N = \hat{\mathbf{D}}_N\mathbf{T}_N\hat{\mathbf{D}}_N, \tag{2}$$

where $\mathbf{T}_N$ is a symmetric Toeplitz matrix defined by its first column $\left[1, \alpha^{-\frac{1}{2}}, \cdots, \alpha^{-\frac{(N-1)^2}{2}}\right]^T$ and $\hat{\mathbf{D}}_N$ is defined above. We use matrix embedding to construct a circulant matrix using Toeplitz matrices s.t.

$$\mathbf{C}_M = \begin{bmatrix} \mathbf{T}_N & \grave{\mathbf{T}}_N \\ \grave{\mathbf{T}}_N & \mathbf{T}_N \end{bmatrix}, \tag{3}$$

where the symmetric Toeplitz matrix $\grave{\mathbf{T}}_N$ is defined by its first column $\left[1, \alpha^{-\frac{(N-1)^2}{2}}, \alpha^{-\frac{(N-2)^2}{2}}, \cdots, \alpha^{-\frac{1}{2}}\right]^T$. We use the similarity transformation of the circulant matrix defined via the non-singular DFT matrices s.t.

$$\mathbf{C}_M = \mathbf{F}_M^*\breve{\mathbf{D}}_M\mathbf{F}_M, \tag{4}$$

where $\mathbf{F}_M^* = [\bar{\mathbf{F}}_M]^T$. Now, we scale the $\mathbf{C}_M$ matrix by rectangular sparse matrices to extract the $\mathbf{T}_N$ matrix s.t.

$$\mathbf{T}_N = \left[\ \mathbf{I}_N \mid \mathbf{0}_N\ \right]\mathbf{C}_M\left[\frac{\mathbf{I}_N}{\mathbf{0}_N}\right]. \tag{5}$$

Thus by (2), (3), (4), and (5), we get the result. □

Corollary II.2. *Let the DVM $\mathbf{A}_N = [\alpha^{kl}]_{k=1,l=0}^{N,N-1}$ be defined by nodes $\{\alpha, \alpha^2, \cdots, \alpha^N\}$ and $N = 2^r (r \geq 1)$. Then the DVM can be factored into*

$$\mathbf{A}_N = \widetilde{\mathbf{A}}_N \cdot \mathbf{D}_N, \tag{6}$$

where $\widetilde{\mathbf{A}}_N$ is defined via (1).

Proof. This is trivial due to the scaling of (1) by $\mathbf{D}_N$. □

C. Radix-2 and Recursive Algorithms for DVMs

Following the DVM factorizations proposed in Section II-B, here we present radix-2 algorithms for DVM and scaled DVM which execute recursively with scaled FFT algorithms (to reduce the multiplication count). To further reduce the multiplication counts in computing the matrix-vector product, we have moved the factor $\frac{1}{\sqrt{M}}$ in $\mathbf{F}_M$ and $\mathbf{F}_M^*$ to the end of the computation, and hence computed $\mathbf{y} = M\widetilde{\mathbf{A}}_N\mathbf{z}$ and $\mathbf{y} = M\mathbf{A}_N\mathbf{z}$.

Before stating algorithms explicitly, let us define all function notations and how the corresponding algorithms execute recursively. $\mathbf{y} = sdvm(\mathbf{z}, N)$ is the function corresponding to the algorithm $sdvm$. This function takes the input vector $\mathbf{z}$ and scalar N, and produces the output vector $\mathbf{y}$, where $\mathbf{y} = M\widetilde{\mathbf{A}}_N\mathbf{z}$. This algorithm executes recursively with the dft algorithm based on the function $\mathbf{v}_1 = dft(\mathbf{u}_2, M)$ having the input vector $\mathbf{u}_2$ and scalar M, and output vector $\mathbf{v}_1$. The $sdvm$ algorithm also executes recursively with the $idft$ algorithm based on the function $\mathbf{y}_1 = idft(\mathbf{v}_2, M)$ having the input vector $\mathbf{v}_2$ and scalar M, and output vector $\mathbf{y}_1$. The dvm algorithm is based on the function $\hat{\mathbf{y}} = dvm(\mathbf{w}, N)$ having input vector $\mathbf{w}$ and scalar N and computes the output $\hat{\mathbf{y}} = M\mathbf{A}_N\mathbf{w}$. This algorithm executes recursively with the $sdvm$ algorithm followed by dft and $idft$ algorithms. We feed $\alpha \in \mathbb{C}$ s.t. $|\alpha| = 1$ as the input for all these algorithms and compute vectors in steps 1 and 3 of these algorithms. For further clarification of the recursive procedure, we present a block diagram in Fig 1.

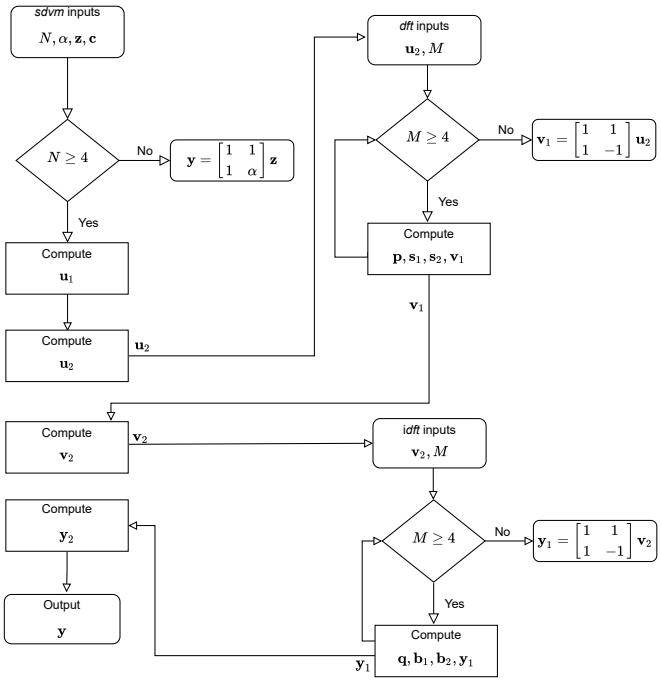

Fig. 1. This block diagram shows the execution of the scaled DVM algorithm corresponding to the function $\mathbf{y} = sdvm(\mathbf{z}, N)$. This function takes vector $\mathbf{z}$ and scalar N as inputs and produces the output vector $\mathbf{y} = M\widetilde{\mathbf{A}}_N\mathbf{z}$. This function executes recursively with the functions $\mathbf{v}_1 = dft(\mathbf{u}_2, M)$ and $\mathbf{y}_1 = idft(\mathbf{v}_2, M)$. Thus, we can calculate the scale DVM by a vector using the well-known sparse factorization of the DFT matrix.

Now, we state the explicit algorithm to compute the product of a scaled DVM by a vector, i.e., $\mathbf{y} = M\widetilde{\mathbf{A}}_N\mathbf{z}$ for a given N, α, $\mathbf{z} \in \mathbb{C}^N$ or $\mathbb{R}^N$ and $\mathbf{c} \in \mathbb{C}^M$.

Algorithm *sdvm*

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

1) **if** $N = 2$, **then**
   $\mathbf{y} \leftarrow \begin{bmatrix} 1 & 1 \\ 1 & \alpha \end{bmatrix} \mathbf{z}$,
2) **end if**
3) **if** $N \geq 4$, **then**
   $\mathbf{u}_1 \leftarrow \hat{\mathbf{D}}_N \cdot \mathbf{z}$
   $\mathbf{u}_2 \leftarrow \mathbf{J} \cdot \mathbf{u}_1$,
   $\mathbf{v}_1 \leftarrow dft(\mathbf{u}_2, M)$,
   $\mathbf{v}_2 \leftarrow \breve{\mathbf{D}}_M \cdot \mathbf{v}_1$,
   $\mathbf{y}_1 \leftarrow idft(\mathbf{v}_2, M)$,
   $\mathbf{y}_2 \leftarrow \mathbf{J}^T \cdot \mathbf{y}_1$,
   $\mathbf{y} \leftarrow \hat{\mathbf{D}}_N \cdot \mathbf{y}_2$,
4) **end if**
5) **return** $\mathbf{y}$

The proposed $sdvm$ algorithm executes recursively with scaled FFTs (we have scaled the DFT and inverse DFT at the end to reduce the multiplication count) in [20], [30]. Let us refer to the scaled FFT and scaled inverse FFT algorithms by *dft* and *idft*, respectively.

Algorithm *dft*

Input: $M = 2^{r_1}$ $(r_1 \geq 1)$, $M_1 = M/2$, and $\mathbf{u}_2 \in \mathbb{C}^M$.
Output: $\mathbf{v}_1 = \hat{\mathbf{F}}_M\mathbf{u}_2$.
Function: $\mathbf{v}_1$=*dft*($\mathbf{u}_2$, M).

1) **if** $M = 2$, **then**
   $\mathbf{v}_1 \leftarrow \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{u}_2$,
2) **end if**
3) **if** $M \geq 4$, **then**
   $\mathbf{p} \leftarrow \mathbf{H}_M \cdot \mathbf{u}_2$
   $\mathbf{s}_1 \leftarrow dft(\mathbf{p}(1 : M_1), M_1)$,
   $\mathbf{s}_2 \leftarrow dft(\mathbf{p}(M_1 + 1 : M), M_1)$,
   $\mathbf{v}_1 \leftarrow \mathbf{P}_M^T \cdot \begin{bmatrix} \mathbf{s}_1^T & \mathbf{s}_2^T \end{bmatrix}^T$,
4) **end if**
5) **return** $\mathbf{v}_1$

Now, we can compute the DVM-vector product through the following dvm algorithm. This algorithm executes recursively with the $sdvm$, dft, and $idft$ algorithms.

Example II.3. *Based on the proposed algorithms, we show building blocks in the signal flow graph drawn for the 16-point scaled DVM algorithm in Section IV. The $sdvm$ algorithm is stated based on the sparse factorization in Theorem II.1, and hence the factorization for the scaled DVM can be stated as*

Algorithm *idft*

Input: $M = 2^{r_1}$ $(r_1 \geq 1)$, $M_1 = M/2$, and $\mathbf{v}_2 \in \mathbb{C}^M$.
Output: $\mathbf{y}_1 = \tilde{\mathbf{F}}_M^* \mathbf{v}_2$.
Function: $\mathbf{y}_1$=*idft*($\mathbf{v}_2$, M).

1) **if** $M = 2$, **then**
$\mathbf{y}_1 \leftarrow \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{v}_2$,
2) **end if**
3) **if** $M \geq 4$, **then**
$\mathbf{q} \leftarrow \mathbf{P}_M \cdot \mathbf{v}_2$
$\mathbf{b}_1 \leftarrow idft(\mathbf{q}(1 : M_1), M_1)$,
$\mathbf{b}_2 \leftarrow idft(\mathbf{q}(M_1 + 1 : M), M_1)$,
$\mathbf{y}_1 \leftarrow \mathbf{H}_M^* \cdot \begin{bmatrix} \mathbf{b}_1^T & \mathbf{b}_2^T \end{bmatrix}^T$,
4) **end if**
5) **return** $\mathbf{y}_1$

Algorithm *dvm*

Input: $N = 2^r$ $(r \geq 1)$, $\alpha \in \mathbb{C}$ s.t. $|\alpha| = 1$, and $\mathbf{w} \in \mathbb{C}^N$ or $\mathbb{R}^N$.
Output: $\hat{\mathbf{y}} = M\mathbf{A}_N\mathbf{w}$.
Function: $\hat{\mathbf{y}}$=*dvm*($\mathbf{w}, N$).

1) **if** $N = 2$, **then**
$\hat{\mathbf{y}} \leftarrow \begin{bmatrix} 1 & \alpha \\ 1 & \alpha^2 \end{bmatrix} \mathbf{w}$
2) **end if**
3) **if** $N \geq 4$, **then**
$\mathbf{z} \leftarrow \mathbf{D}_N \cdot \mathbf{w}$,
$\hat{\mathbf{y}} \leftarrow sdvm(\mathbf{z}, N)$,
4) **end if**
5) **return** $\hat{\mathbf{y}}$

follows.

$$32\widetilde{\mathbf{A}}_{16} = \hat{\mathbf{D}}_{16} \left[\mathbf{I}_{16} | \mathbf{0}_{16}\right] \tilde{\mathbf{F}}_{32}^* \breve{\mathbf{D}}_{32} \tilde{\mathbf{F}}_{32} \begin{bmatrix} \mathbf{I}_{16} \\ \mathbf{0}_{16} \end{bmatrix} \hat{\mathbf{D}}_{16},$$

where $\hat{\mathbf{D}}_{16} = \begin{bmatrix} 1 & & & \\ & \hat{d}_1 & & \\ & & \ddots & \\ & & & \hat{d}_{15} \end{bmatrix}$, $\breve{\mathbf{D}}_{32} = \begin{bmatrix} \breve{d}_0 & & & \\ & \breve{d}_1 & & \\ & & \ddots & \\ & & & \breve{d}_{31} \end{bmatrix}$, $\tilde{\mathbf{F}}_{32}^*$:= *the conjugate transpose of the scaled* $\mathbf{F}_{32}$*, and the sparse factorization for the* $\mathbf{F}_{32}$ *can be obtained from [20].*

Remark II.4. *The proposed scaled DVM and DVM algorithms can execute recursively the FFTs and scaled FFTs in [30] and [23], and will be explored in future work.*

III. Complexity of DVM Algorithms

Here we establish the arithmetic complexities of the proposed scaled DVM and DVM algorithms. The number of additions and multiplications counts are presented in correspondence with the adders and gain-delay blocks, respectively. The DVM describes TTDs of the input-vector of signals, which are given by the Fourier transform property $x(t-\tau) \rightarrow X(\omega)e^{-j\omega\tau}$. The TTD operations can in turn be represented by linear convolutions $x(t-\tau) = x(t) \star \delta(t-\tau)$, which in turn establishes that the complex multiplicative elements of the DVM are purely due to the representation of the inputs in the temporal frequency domain. In fact, a time-domain representation of N inputs $u_k(t), k = 1, 2, ..., N$, would result in a time domain DVM $\mathbf{A}_t$ representation with (n, m) matrix elements defined as $\delta(t - \tau nm)$ and the output vector $v_k(t), k = 1, 2, ..., N$, found through matrix-vector convolution $\mathbf{v}(t) = \mathbf{A}_t \star \mathbf{u}(t)$.

The presented multiplication counts (in frequency domain) is a combination of gains and delays in the time domain. TTDs require a separate TTD circuit in the analog domain. In analog domain, delays are realized in continuous time analog circuits. In the temporal frequency domain, true time delays $\tau \in \mathbb{R}$ are represented by multiplicative terms $e^{-j\omega\tau} \in \mathbb{C}$. Ideally, such filters are represented by complex exponential functions, which are *infinite order* when mapped to their Taylor series polynomials. The use of analog all-pass filters for approximating such delays requires setting a finite order for the ration polynomials that present passive circuits made from inductors, resistors, and capacitors. A typical approximation using cascade of 1-st order all-pass filters is $e^{-s\tau} \approx \phi_{APF}(s) = \left[\frac{1-\frac{s\tau}{2m}}{1+\frac{s\tau}{2m}}\right]^m$ where $m \in \mathbb{Z}^+$ and it can be shown that typically $m \geq 3$ is sufficient for acceptable beamforming performance in real-world systems [24], [25]. Note, $s \in \mathbb{C}$ is the Laplace variable, and the frequency response of the system can be found by setting $s = j\omega$. In the digital domain, the delays can be realized using fractional delay filters, typically of type finite impulse response (FIR) as this offers linea. Here, the delay τ becomes a fraction of the temporal sample period F_S such that $\mu = \tau F_S$. The digital FIR filter, for example, that implement a fractional delay, takes the form $H_{APF}(z) = z^{-\mu}$. For both analog and digital realizations, the objective is to factor the original DVM matrix into a product of sparse matrices, s.t., the total number of adders and gain-delay blocks (all pass filters in analog; fractional delay FIR filters in digital) reduce from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log N)$. Thus, we will show in the following that the arithmetic complexities of the proposed algorithms require $\mathcal{O}(N\log N)$ adders and gain-delay blocks.

In one realization, the analog TTDs are electronic all-pass filters (APFs), and digital TTDs are fractional order filters of order L. Therefore, in digital realizations, each TTD consists of L real multipliers in a filter structure. Since these L multipliers are necessary for both direct and sparse factorized DVM algorithms, we can remove them from the complexity calculation of the fast algorithm, keeping in mind that digital realizations require $L\times$ more multipliers than the TTD count of the DVM algorithm.

A. Arithmetic Complexity of DVM Algorithms

Let us obtain the number of additions (say $\#a$) and number of multiplications (say $\#m$), i.e., adders and gain-delay blocks, respectively, required to compute the algorithms pro-

posed in Section II. Counts are obtained to compute the scaled DVM and DVM algorithms which execute recursively with the scaled FFTs (to reduce the multiplication counts) in [20]. Thus, we take the number of additions and multiplications required to compute $\mathbf{y} = \tilde{\mathbf{F}}_N \mathbf{z}$, where $\mathbf{z} \in \mathbb{C}^N$, as Nr and $\frac{1}{2}Nr - \frac{3}{2}N + 2$, respectively, when the multiplications by ± 1 and $\pm j$ are not counted.

Lemma III.1. *Let $N = 2^r (r \geq 1)$ be given. The scaled DVM algorithm, i.e., Algorithm sdvm, can recursively be computed using sdvm, dft, and idft algorithms with the following arithmetic complexities (respectively, adders and gain-delay blocks):*

$$\begin{aligned} \#a(sdvm, N) &= 4Nr + N, \\ \#m(sdvm, N) &= 2Nr + 2. \end{aligned} \tag{7}$$

Proof. Referring to the scaled DVM algorithm, we get

$$\begin{aligned} \#a(sdvm, N) = 2\left(\#a(\tilde{\mathbf{F}}_M)\right) + 2\left(\#a(\hat{\mathbf{D}}_N)\right) \\ + 2\left(\#a(\mathbf{J})\right) + \#a(\breve{\mathbf{D}}_M). \end{aligned} \tag{8}$$

The similar equation holds for the number of multiplications as well. Following the structures of $\hat{\mathbf{D}}_N, \mathbf{J}$, and $\breve{\mathbf{D}}_M$, and the multiplication of each matrix by a complex input, we have

$$\begin{aligned} \#a\left(\mathbf{J}\right) = 0, \;\; \#m\left(\mathbf{J}\right) = 0, \\ \#a\left(\hat{\mathbf{D}}_N\right) = 0, \;\; \#m\left(\hat{\mathbf{D}}_N\right) = N - 1, \\ \#a\left(\breve{\mathbf{D}}_M\right) = 0, \;\; \#m\left(\breve{\mathbf{D}}_M\right) = M. \end{aligned}$$

Multiplications of $\mathbf{J}$ from the right and left of the DFT do not affect the multiplication counts, but do affect the addition counts (due to $\mathbf{0}_N$ matrix). Hence, we have to subtract $M+N$ addition counts (due to no M and N additions after and before $\hat{\mathbf{D}}_N$, respectively) from the total addition counts. By following Nr additions and $\frac{1}{2}Nr - \frac{3}{2}N + 2$ multiplications to compute $\mathbf{y} = \tilde{\mathbf{F}}_N \mathbf{z}$ with $\mathbf{z} \in \mathbb{C}^N$, we get addition and multiplication counts as in (7) based on the computation of the proposed scaled DVM algorithm. □

The DVM algorithm is obtained by scaling the $sdvm(\mathbf{z}, N)$ algorithm, and hence adders and gain-delay blocks needed to compute $\mathbf{y} = M\mathbf{A}_N\mathbf{z}$ can be obtained as follows.

Lemma III.2. *Let $N = 2^r (r \geq 1)$ be given. The DVM algorithm, i.e., Algorithm dvm , can recursively be computed using dvm, sdvm, dft, and idft algorithms with the following arithmetic complexities (respectively, adders and gain-delay blocks):*

$$\begin{aligned} \#a(dvm, N) &= 4Nr + N, \\ \#m(dvm, N) &= 2Nr + N + 1. \end{aligned} \tag{9}$$

Proof. This is trivial due to scaling. □

B. Numerical Results for the Complexity of DVM Algorithms

Numerical results for the arithmetic complexity, respectively adders and gain-delay blocks, of the scaled DVM and DVM algorithms derived via Lemma III.1 and III.2 versus the direct matrix-vector computations for matrix sizes varying from 4×4 to 4096×4096 are shown in Tables I and II. Here, we consider the direct computation of the scaled DVM by a vector cost of $N(N-1)$ additions (respectively adders) and $(N-1)^2$ multiplications (respectively gain-delay blocks), and the DVM by a vector cost of $N(N-1)$ additions and N^2 multiplications. These counts are calculated without considering the multiplications by ± 1. The last column in Tables I and II shows the percentage reduction (say Pr) of addition and multiplication counts (a total of adders and gain-delay block counts) of the proposed algorithm opposed to the brute-force matrix-vector calculation. The values in these columns are obtained using $Pr = \left(\frac{Td - Ta}{Td}\right) \times 100\%$, where Td is the sum of adders and gain-delay block counts in computing the direct matrix-vector product and Ta is the sum of adders and gain-delay block counts in computing the proposed algorithms.

TABLE I
ADDERS AND GAIN-DELAY BLOCK COUNTS OF THE SCALED DVM ALGORITHM VERSUS THE DIRECT COMPUTATION OF THE SCALED DVM BY A VECTOR

N	Direct add	$\#a(sdvm)$	Direct multiply	$\#m(sdvm)$	Pr
4	12	36	9	18	-157%
8	56	104	49	50	-47%
16	240	272	225	130	14%
32	992	672	961	322	49%
64	4032	1600	3969	770	70%
128	16256	3712	16129	1794	83%
256	65280	8448	65025	4098	90%
512	261632	18944	261121	9218	95%
1024	1047552	41984	1046529	20482	97%
2048	4192256	92160	4190209	45058	98%
4096	16773120	200704	16769025	98306	99%

TABLE II
ADDERS AND GAIN-DELAY BLOCK COUNTS OF THE DVM ALGORITHM VERSUS THE DIRECT COMPUTATION OF THE DVM BY A VECTOR

N	Direct add	$\#a(dvm)$	Direct multiply	$\#m(dvm)$	Pr
4	12	36	16	21	-104%
8	56	104	64	57	-35%
16	240	272	256	145	16%
32	992	672	1024	353	49%
64	4032	1600	4096	833	70%
128	16256	3712	16384	1921	83%
256	65280	8448	65536	4353	90%
512	261632	18944	262144	9729	95%
1024	1047552	41984	1048576	21505	97%
2048	4192256	92160	4194304	47105	98%
4096	16773120	200704	16777216	102401	99%

Tables I and II show that the scaled DVM and DVM algorithms require a very low number of adders and gain-delay blocks compared to the brute-force calculation. When the size of the matrices increases, we observe a significant reduction in arithmetic complexity (equivalently, the total number of adders and delay-gain blocks) for computing the proposed algorithms. Tables I and II indicate that the proposed algorithms are not efficient for matrices of sizes $N = 4, 8$, but significantly fast, i.e. $> 90\%$ for matrices of sizes $N \geq 256$.

IV. Signal Flow Graph of the Fast DVM Algorithm

Signal flow graphs (SFGs) can be used as a tool to analyze and visualize fast algorithms, and also to design their VLSI architectures. The sparse factorization and fast recursive algorithm established in Section II results in relatively simple SFG, as presented in this section. Implementing such simplified SFG requires dramatically reduced chip area, cost, and power consumption compared to conventional algorithms. To illustrate the point, Fig. 2 presents 16-point SFG of the scaled DVM algorithm (i.e., Algorithm *sdvm*). Ignoring the multiplications by ± 1 and $\pm j$ (exactly as followed in Section III), the addition and multiplication counts (respectively adders and delay-gain blocks) in the SFG matches exactly with the complexity results proposed in Lemma III.1, Section III for $N = 16$.

Following the gain-delay blocks in Lemma III.1 and the building blocks of the SFG shown in Fig. 2, the explicit gains, delays, and total multiplication counts in computing the scaled DVM algorithm are shown in the second, third, and the last columns, respectively, of Table III. The fourth column of the table shows the non-trivial anti-causal counts, which are equivalent to the number of entries in the pre-computed matrix $\tilde{\mathbf{D}}_M$. Trivial anti-causal counts arise from the product of the scaled DFT matrix $\tilde{\mathbf{F}}_M$ by a vector $\mathbf{c}$ with entries of the form $e^{jk\omega\tau}$, where τ is a delay, $k = \frac{p}{2}$, and p is 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 III. To realize trivial anti-causal counts, i.e., entries of $\mathbf{c}$, we multiply every entry in $\mathbf{c}$ by the largest magnitude of the anti-causal term, i.e., $\left|\alpha^{\frac{-(N-1)^2}{2}}\right|$, in the pre-computation stage so that anti-causal terms are not a problem for practical realizations. 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 III
Gains, delays, and anti-causal counts of the scaled DVM algorithm

N	Gains	Delays	Anti-causal	$\#m(sdvm)$
4	4	6	8	18
8	20	14	16	50
16	68	30	32	130
32	196	62	64	322
64	516	126	128	770
128	1284	254	256	1794
256	3076	510	512	4098
512	7172	1022	1024	9218
1024	16388	2046	2048	20482
2048	36868	4094	4096	45058
4096	81924	8190	8192	98306

V. DVM Array Signal Processing Systems

A. Array Transceiver Architectures

Wideband transceivers for emerging 5G/6G wireless networks use large-scale arrays to provide high antenna gain, thus compensating for high path loss [32]. The proposed DVM-based multi-beam beamformer can reduce the size, weight, power consumption, and cost (SWaP-C) of such array signal processors. Here we focus on architectures for array receivers; the discussion can be readily extended to transmitters.

The block diagram of a conventional N-element receiver using digital N-beam multi-beamforming is shown in Fig. 3(a). The input plane-wave is received by an N-element ULA, amplified by low-noise amplifiers (LNAs), and then digitized using high-speed analog-to-digital converters (ADCs). The digitized RF outputs are fed into a digital back-end processor that implements the N-beam DVM beamformer algorithm to generate N RF beams. In practice most of the N parallel output beams do not contain useful information and are discarded before further processing. This step is performed by a $N : K$ beam selection multiplexer (where $K \ll N$) whose switches are adaptively set by digital signal processing (DSP) algorithms via a select (SEL) signal.

The all-digital array receiver design shown in Fig. 3(a) suffers from two problems. The first is the need for ultra-high-speed ADCs to digitize the wideband LNA outputs. These ADCs are also susceptible to saturation due to strong unwanted interferers (known as blockers), which are unavoidable in wireless channels. The second is high digital throughput and power consumption because of the need to beamform all N high-bandwidth ADC output streams in parallel. This step is obviously inefficient when $K \ll N$ (as is often the case), since most of the N digital beams are discarded prior to further processing. Fig. 3(b) shows a receiver architecture that is potentially more energy-efficient. This design performs N-beam multi-beamforming and beam selection in the analog domain immediately after the LNAs, followed by frequency down-conversion and low-speed, low-power ADCs. Our prior work has demonstrated suitable analog-domain IC implementations of both FFT- and DVM-based multi-beamformers [2], [5], [10], [33]–[35]. Another advantage of such hybrid analog-digital array receiver architectures is that beam selection is performed before the ADCs (using analog switches), allowing blockers to be removed from the ADC inputs and also significantly reducing the overall ADC count (to $K \ll N$).

B. Multidimensional Region of Support of Propagating Waves

Next, we analyze the spatiotemporal properties of plane-waves incident on the array receivers described in the previous subsection. Let a 3-D spatiotemporal plane-wave signal $w_a(x, y, ct) \in \mathbb{R}^3$ propagating in the Cartesian plane $(x, y) \in \mathbb{R}^2$ be measured on the $y = 0$ line to yield a 2-D spacetime signal $w(x, ct) \in \mathbb{R}^2$. The DOA, measured counterclockwise from the $x = 0$ axis, is given by θ through the planar equation $p = -x \sin\theta + ct$ such that $w(x, ct) = f(p)$ where $f(p), p \in \mathbb{R}$ is a wideband plane-wave bandlimited to B Hz and $\Delta x = c/(2B)$ is the inter-element distance of a ULA located along $y = 0$. Let the 2-D discrete-space Fourier transform be defined within the 2-D Nyquist $\square$ via $W(e^{j\omega_x}, \omega_t) = \int_{t=-\infty}^{\infty} \sum_{n=0}^{N-1} w(n\Delta x, ct) e^{-jn\omega_x \Delta x} d(ct)$. The signals are temporally bandlimited; therefore $-2\pi B \leq \omega_t \leq 2\pi B$. Without loss of generality, normalize $c = 1, \Delta x =$

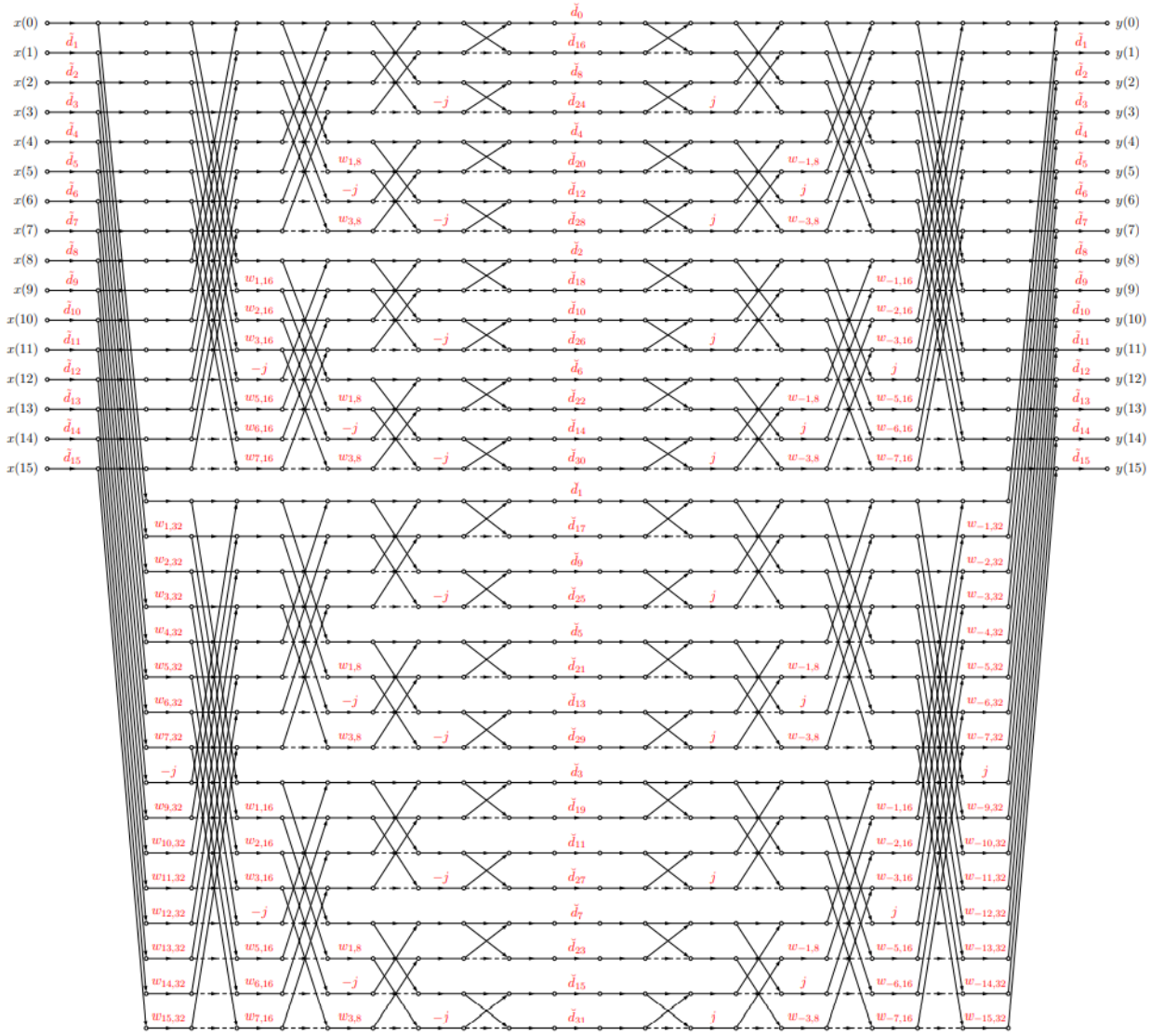

Fig. 2. Signal flow graph for the 16-point scaled DVM algorithm, where $w_{k,n} = e^{-i\frac{2\pi k}{n}}$, $j^2 = -1$, $\hat{d}_k$ is the k^{th} component of $\hat{\mathbf{D}}$, and $\check{d}_k$ is the k^{th} component of $\check{\mathbf{D}}$. Dotted lines represent a multiplication by -1.

1, and $B = 1$. The region of support (ROS) of $W(e^{j\omega_x}, \omega_t)$ is the part of the domain where the spectrum is not defined to be zero. The ROS of $W(e^{j\omega_x}, \omega_t)$ is a line through the origin of $(\omega_x, \omega_t) \in \mathbb{R}^2$ oriented at angle $\beta = \tan^{-1}(\sin\theta)$.

To demonstrate the ROS, we show in Fig. 4 an example propagation scenario where 10 sinusoidal components at $f_i = \{0.4, 0.7, 0.9, 1.0, 2.4, 3.0, 3.5, 4.0, 5.8, 6.0\}$ GHz propagate simultaneously in a single quadrant at angles $\theta = \{0°, 20°, 50°, 90°\}$ to form four wideband plane-waves. Clearly, the temporal spectra are identical among the four waves; however, the 2-D ROS lie along the four directions $\beta = \{\tan^{-1}(\sin 0), \tan^{-1}(\sin 20\pi/180), \tan^{-1}(\sin 50\pi/180), \tan^{-1}(\sin 90\pi/180)\}$, radians, respectively. The received spatially-discrete 2-D array signal is $w(n,t) = \sum_i \sum_k \cos(2\pi f_i(-n\sin\theta_k + t))$ where $0 \leq n < N$. The finite array size imposes a $(\sin\pi\omega_x)/\pi\omega_x$ type window on the 2-D spectrum causing convolutional widening and shaping of the spatial spectrum - readily observable in Fig. 4.

C. 2-D Frequency Responses of DVM Multi-Beamformers

The $N \times N$ DVM matrix represents a 2-D filter bank. Let the k^{th} filter be given by the impulse response $h_k(l, e^{j\omega}) = \alpha^{kl}$. Therefore, the 2-D frequency response can be computed by first finding the spatial $z-$transform $H_k(z) = \sum_{l=0}^{N-1} \alpha^{kl} z^{-l}$ and evaluating the spatial frequency response by setting $z = e^{j\omega_x}$. The 2-D frequency response function is $H_k(e^{j\omega_x}, e^{j\omega_t}) = \sum_{l=0}^{N-1} e^{-j\omega_t \tau k/Nl} e^{-j\omega_x l} = \sum_{l=0}^{N-1} e^{-jl(\omega_t \tau k/N + \omega_x)}$. Defining a plane $\lambda = \omega_t \tau k/N + \omega_x$ we obtain the simplified form $H_k(e^{j\omega_x}, e^{j\omega_t}) = \sum_{l=0}^{N-1} e^{-j\lambda l}$ which reduces to $H_k(e^{j\omega_x}, e^{j\omega_t}) = \frac{1-e^{-jN\lambda}}{1-e^{-j\lambda}}$. Normalizing $\tau = 1$ as per convention and assuming $-\pi \leq \omega_x < \pi$ and $-\pi \leq \omega_t < \pi$ gives the frequency response function as $H_k(e^{j\omega_x}, e^{j\omega_t}) = \frac{\sin N\lambda'/2}{\sin \lambda'/2} e^{-j(N-1)\lambda'/2}$ where $\lambda' = k\omega_x/N + \omega_t$. Let the beam axis of the k^{th} filter be oriented at angle β_k from the ω_t axis. Then, it follows that $\beta_k = \tan^{-1}(k/N), k = 0, 1, ..., N-1$. Let the direction

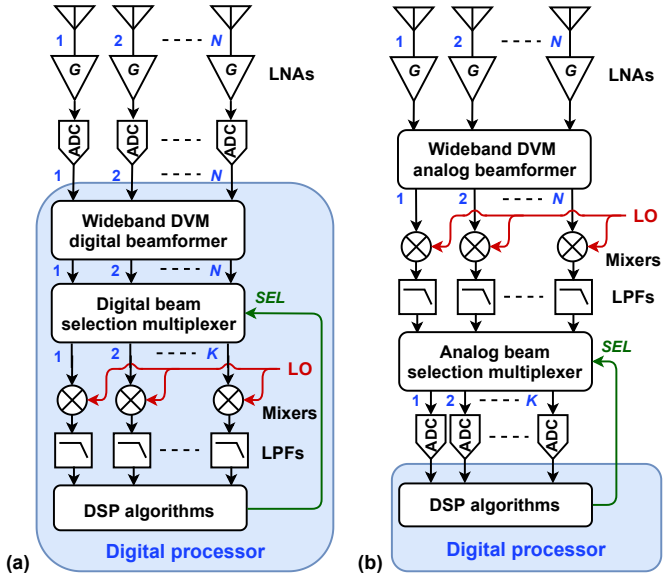

Fig. 3. Architectures for wideband array receivers: (a) using all-digital multi-beamforming. Here, it is assumed that fully-wideband signals are directly sampled at the full RF bandwidth. E.g., as in direct-RF to digital converters for extremely wide bandwidth systems. Figure describes how the full-bandwidth RF signal can be sampled, beamformed, and thereafter selected bandpass signals can be extracted in real-time using digital down-conversion and filtering stages. And in (b) using hybrid analog-digital multi-beamforming where LNA = low-noise amplifier, LO = local oscillator, LPF = low-pass filter.

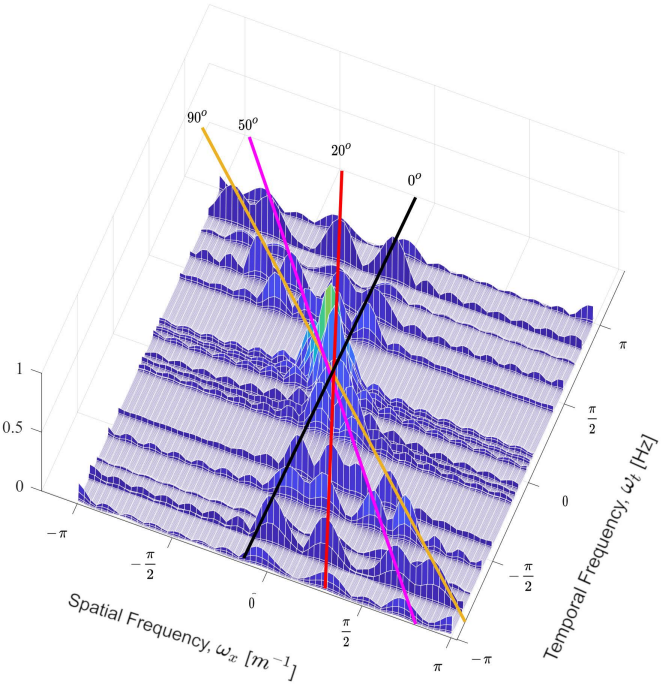

Fig. 4. The 2-D magnitude spectra of four wideband plane-waves having identical temporal frequency spectra arriving at $\theta = \{0°, 20°, 50°, 90°\}$ where each ROS orientation is non-linearly related to DOA via $\tan\beta = \sin\theta$.

of propagation of a far-field electromagnetic plane-wave be given by $-\pi/2 \leq \theta \leq \pi/2$. Due to finite speed of light ($c \approx 3 \times 10^8 \text{ms}^{-1}$) the apparent direction of arrival in spacetime $(x, ct) \in \mathbb{R}^2$ is given by $\beta = \tan^{-1}(\sin\theta)$. The angle for the k^{th} beam of the DVM filter bank is thus $\theta_k = \sin^{-1}(\tan\beta_k)$, $k = 0, 1, ..., N-1$. As an example, Fig. 5 shows the filter bank magnitude responses (i.e., 2-D beam spectra) for a $N = 16$ beam DVM beamformer. Each beam follows a straight line in the (ω_x, ω_t) plane, as expected.

VI. Circuit Implementations of DVMs for Wideband Transceivers

In this section, we discuss circuit-level implementations of the DVM multi-beamformers described in Section V. The designs are more challenging than for conventional narrow-band DFT-based beamformers due to the need to use TTDs (unlike DFT-based schemes, which can use phase shifters) [5]. Additionally, frequency down-conversion (i.e., mixing to a lower center frequency) cannot be performed prior to DVM-based beamforming, unlike for narrowband methods. As a result, the circuit must have sufficient bandwidth to handle the highest-frequency RF or mmW signals of interest.

A. Realizing True Time Delays (TTDs)

Generating the k^{th} squint-free DVM beam using an ULA requires $\sin(\theta_k) = \left(\frac{c\tau}{\Delta x}\right) k$. For beam angles $\theta_k \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and an N-element array, $\sin(\theta_k) = \tan(\beta_k) = 2k/N$ where $k = -N/2, ..., N/2-1$. Thus, the necessary TTDs should be integer multiples of

$$\tau = \frac{2\Delta x}{cN} \approx \frac{1}{f_{max} N}, \tag{10}$$

where f_{max} is the maximum signal frequency (corresponding to the minimum wavelength λ_{min}) and we have used the fact that $\Delta x \approx \lambda_{min}/2$ to avoid spatial aliasing. Thus, the required delay resolution increases both with operating frequency and array size. For example, a $N = 16$ element array at 28 GHz requires $\tau = 2.2$ ps. In digital realizations of DVM beamformers, such high-resolution TTDs can be implemented using linear-phase fractional-delay FIR filters. Such digital filters are part of the larger class of digital interpolation filters and are straightforward to design. Digital FIR fractional delay interpolation filters can be realized using low-complexity polyphase forms for processing RF bandwidths B that are greater than half the clock rate of a typical DSP system.

Analog-domain implementations of wideband DVM beamformers are more energy-efficient, as described in Section V-A. For such systems, the TTDs should ideally be linear-phase delay lines having transfer functions $x(t-\tau) \Leftrightarrow X(j\omega_t)e^{-j\omega_t\tau}$. These can either be passive transmission lines (which are linear and noiseless, but consume significant chip area) or active approximations (which are nonlinear and noisy, but area-efficient). For example, first-order APFs can be cascaded to approximate TTDs on CMOS ICs [31].

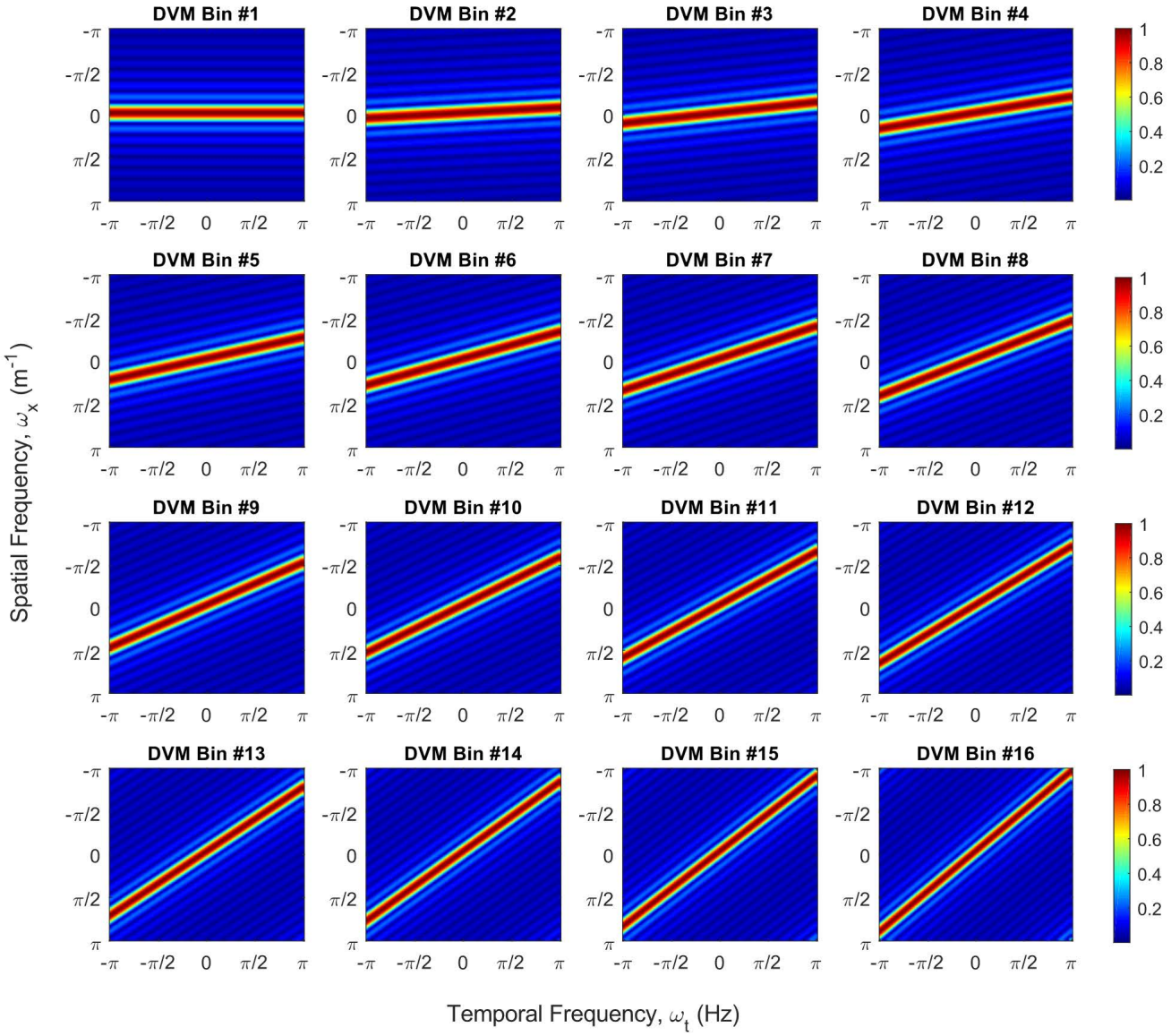

Fig. 5. The 2-D magnitude frequency response of the N-beam DVM beamformer (here $N = 16$). Each beam extracts wideband plane-wave signals based on its direction of arrival (DOA) θ where $\sin\theta = \tan\beta$. Broadside waves fall on the broadside beam ($DVM\ bin\#0$), while waves from the end-fire direction fall on the $\beta = 45^\circ$ beam ($DVM\ bin\#15$).

B. Realizing DVM Multi-Beam Beamformers

Analog-domain implementations of DVM beamformers can operate in either continuous time (CT) or discrete time (DT) [36]. CT realizations have higher bandwidths, eliminate issues with high-frequency clock distribution, and avoid temporal aliasing. However, they are less accurate since the transfer functions (TFs) of CT TTD elements are sensitive to unavoidable device mismatch and process-voltage-temperature (PVT) variations in IC processes. On the other hand, DT realizations have lower bandwidths but are more accurate because i) TTD delays can be set by an accurate external clock, and ii) TF errors due to device mismatch can be removed using various offset-cancellation and dynamic element matching (DEM) methods [37]. In either case, the analog signals can be represented either using voltages (known as voltage-mode) or currents (known as current-mode). Current-mode operation is attractive for wideband implementations because it avoids the need for power-hungry closed-loop op-amps; instead, one can use simple open-loop current mirrors (for gain) and Kirchoff's current law (KCL) (for additions) [34]. The necessary TTDs can be approximated using op-amp RC filters (CT voltage-mode), op-amp switched-capacitor filters (DT voltage-mode), translinear filters (CT current-mode), or switched-current filters (DT current-mode). Here we consider a CT current-mode implementation using passive delay lines since it minimizes power consumption for a given bandwidth.

Fig. 6 shows a programmable current-mode gain-delay block for implementing the proposed analog DVM multi-beam beamformer. The circuit allows rational gain values of the form $N_1/(2N_2)$ to be implemented, where N_1 and N_2 are integers chosen to closely approximate a particular DVM beamformer coefficient. First, the input current is attenuated by N_2 using a

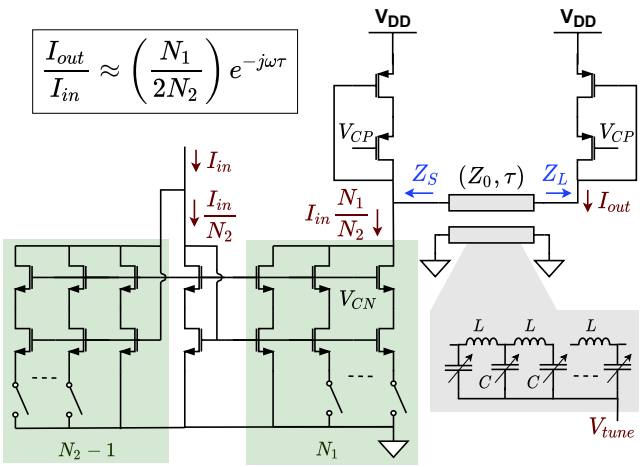

Fig. 6. Schematic of a programmable current-mode gain-delay block using an on-chip LC transmission line (characteristic impedance Z_0, time delay τ). Here V_{CP} and V_{CN} are DC bias voltages for the cascode transistors.

programmable current splitter. The latter is realized by using CMOS switches that set the relative width of the shunt NMOS transistor to a factor of $N_2 - 1$. Next, the attenuated current is amplified by N_1 using another set of switches that set the relative width of the output NMOS transistor. Finally, the amplified current is delayed by τ using a passive LC transmission line with characteristic impedance Z_0. In practice the line is implemented using $M \gg 1$ discrete LC stages, such that $\tau \approx M\sqrt{LC}$ for frequencies $\omega \ll 1/(2\sqrt{LC})$. Device mismatch and PVT variations can be compensated by programming the value of τ (typically over a 1:2 range) through the tuning voltage V_{tune}, which in turn sets the capacitance of the variable capacitors (varactors). Also, the PMOS loads are sized to ensure impedance matching at either end of the line, thus avoiding reflections. That is, we set the PMOS transconductance g_{mp} such that $Z_s = Z_L = 1/g_{mp} = Z_0 = \sqrt{L/C}$. While the input capacitance $C_{in} \approx (C_{gs}+C_{gd})$ of the loads degrades matching at high frequencies, its effects can be removed by absorbing C_{in} into the terminal capacitors of the LC line (which have a nominal value of $C/2$ each). Finally, note that simplified fixed-gain versions of this circuit (with the switches removed) can be used if the beamformer coefficients remain fixed for a given chip.

Detailed simulations and test results of the analog DVM beamformer will be presented in future papers. Here we describe preliminary simulation results of the proposed design using a 45 nm bulk CMOS process. Fig. 7 shows the simulated frequency response and group delay of the circuit for various values of τ (i.e., line length). In this example, the desired gain was set to $5/8 = 0.625$, while τ was set in 50 ps steps. The results show that the simulated group delay is slightly larger than the line delay τ due to delay within the current mirrors; line lengths can be reduced to compensate for this error. Also, the realized gain is within 1.5% of its desired value up to 1 GHz (assuming a maximum delay of $\tau_{max} = 0.8$ ns). Larger bandwidths are available by reducing the value of τ_{max} generated by a single block and instead using multiple cascaded blocks to realize long delays. For example, the useful

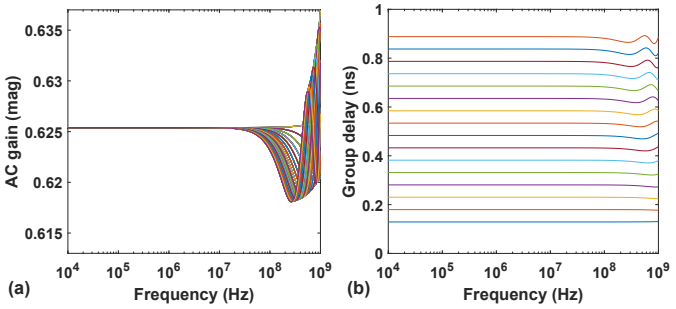

Fig. 7. Simulated (a) frequency response, and (b) group delay of the gain-delay block for a bias current of $I_B = 500$ μA, $V_{DD} = 1.2$ V, $Z_0 = 300$ Ω, and line delay τ varying from 50 ps to 800 ps in 50 ps steps.

bandwidth increases to 2 GHz when $\tau_{max} = 0.4$ ns. However, power consumption increases as M_s where M_s is the number of cascaded stages. Additionally, the rms noise level increases as $\sqrt{M_s}$, thus degrading the signal-to-noise ratio (SNR), i.e., output precision. Thus, there is a design trade-off between bandwidth, precision, and power consumption [52].

The proposed analog gain-delay blocks were used to realize a complete wideband DVM multi-beamformer with $N = 16$ beams. The value of f_{max} 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 DVM matrix $\mathbf{A}_N$ 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. 8(a) and (c) compares the simulated beam shapes obtained at input frequencies of 0.6 GHz and 1 GHz, respectively, with ideal ones obtained from a MATLAB implementation of the algorithm. Also, Fig. 8(b) and (d) show the magnitude of the errors between simulated and ideal beam shapes at the two frequencies. These errors increase with frequency due to the limited frequency response of the gain-delay blocks (see Fig. 7), but are acceptable up to 1.2 GHz.

Further design improvements (e.g., increasing the bandwidth of the current mirrors by implementing them using high-speed heterojunction bipolar transistors [HBTs]) should allow f_{max} to be increased to several GHz, thus enabling the proposed hybrid array receiver architecture (see Fig. 3(b)) to be implemented in various sub-6 GHz wireless bands. Extension to mmW bands requires a different circuit architecture (e.g., based on active APFs) and will be explored in future work.

VII. Conclusion

We have proposed a sparse factorization for the DVM followed by a fast, exact, radix-2, and recursive DVM algorithms to realize microwave/millimeter wave N-beam wideband beamformers while reducing the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$. The proposed DVM algorithms are at least 90% faster than the brute-force DVM by a vector product for the matrices of sizes $N \geq 256$. Since the proposed algorithm is based on the pre-computation of anti-casual, we have suggested a technique to realize anti-casual parts of the signal flow graph using only delay elements which results in the original filter bank with additional latency. The signal flow graph shows the simplicity and beauty of the proposed

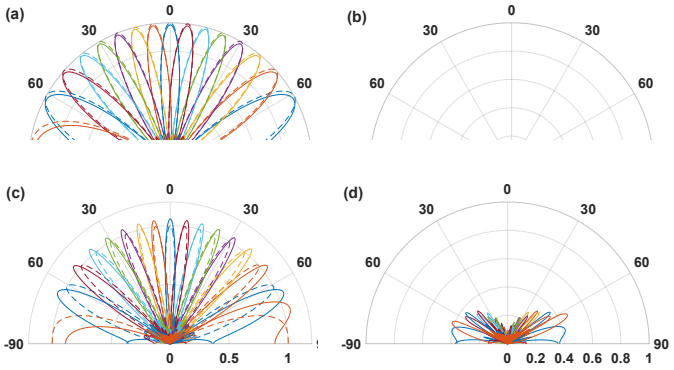

Fig. 8. Left column: Simulated DVM beam shapes for a 16-element ULA with $f_{max} = 1$ GHz (solid lines) compared to ideal shapes obtained from MATLAB (dashed lines). The input frequency was set to (a) 0.6 GHz or (c) 1 GHz, respectively. Right column: Magnitudes of the errors between simulated and ideal beam shapes at (b) 0.6 GHz and (d) 1 GHz, respectively.

sparse factorization which can be used in very large-scale integrated circuit architecture while reducing the chip area, cost, and power consumption significantly. Analog realizations may use passive and active delay lines. Digital realizations may use fractional delay FIR interpolation filters for achieving the DVM array processor in future realizations.

ACKNOWLEDGMENTS

The authors thank Udara De Silva for help with generating publication quality figures.

REFERENCES

[1] H. L. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory*, Wiley Publications, New York, USA, 2002.

[2] 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 TCAS-I: Regular Papers 59(5): 1061-1073, March 2012.

[3] K. W. Masui, J. R. Shaw, C. Ng, K. M. Smith, K. Vanderlinde, and A. Paradise, *Algorithms for FFT beamforming radio interferometers*, ApJ 879(1): 16, July 2019.

[4] S. R. Seydnejad and S. Akhzari,*Performance evaluation of pre- and post-FFT beamforming methods in pilot-assisted SIMO-OFDM systems*, Telecommun. Syst. 61(3): 471-487, March 2016.

[5] S. M. Perera, V. Ariyarathna, N. Udayanga, A. Madanayake, G. Wu, L. Belostotski, Y. Wang, S. Mandal, R. J. Cintra, and T. S. Rappaport, *Wideband N-beam Arrays with Low-Complexity Algorithms and Mixed-Signal Integrated Circuits*, IEEE J Sel Top Signal Process 12(2):368-382, April 2018.

[6] S. M. Perera, A. Madanayake and R. Cintra, *Radix-2 Self-recursive Algorithms for Vandermonde-type Matrices and True-Time-Delay Multi-Beam Antenna Arrays*, IEEE Access 8: 25498-25508, February 2020.

[7] S. M. Perera, A. Madanayake and R. Cintra, *Efficient and Self-Recursive Delay Vandermonde Algorithm for Multi-beam Antenna Arrays*, IEEE OJSP 1:64-76, April 2020.

[8] W. Rotman and R. F. Turner, *Wide-angle microwave lens for line source applications*, IEEE Trans. Antennas Propag. AP-11: 623-632, November 1963.

[9] A. E. A. Blomberg, A. Austeng, R. E. Hansen, *Adaptive beamforming applied to a cylindrical sonary array using and interpolated array transformation*, IEEE J. Ocean. Eng. 37(1): 25-34, January 2012.

[10] V. Ariyarathna, N. Udayanga, A. Madanayake, S. M. Perera, L. Belostotski, and R. J. Cintra, *Design methodology of an analog 9-beam squint-free wideband IF multi-beamformer for mmW applications*, In Proc. of IEEE 2017 MERCon: 236-241, IEEE, July 2017.

[11] H. Oruc and G. M. Phillips, *Explicit factorization of the Vandermonde matrix*, Linear Algebra Appl. 315:113-123, 2000.

[12] H. Oruc and H. K. Akmaz, *Symmetric functions and the Vandermonde matrix*, J. Comput. Appl. Math. 172:49-64, 2004.

[13] S. L. Yang, *On the LU factorization of the Vandermonde matrix*, Discret. Appl. Math. 146:102-105, 2005.

[14] I. Gohberg and V. Olshevsky *Complexity of multiplication with vectors for structured matrices*, Linear Algebra Appl. 202:163-192, 1994.

[15] I. Gohberg and V. Olshevsky. *Fast algorithms with preprocessing for matrix-vector multiplication problems*, J. Complex. 10:411-427, 1994.

[16] V. Y. Pan, *Fast approximate computations with Cauchy matrices and polynomials*, Math. Comput. 86: 2799-2826, 2017.

[17] J. Demmel and P. Koev, *Accurate SVDs of polynomial Vandermonde matrices involving orthonormal polynomials*, Linear Algebra Appl. 417: 382-396, 2006.

[18] A. Marco and J.-J. Martínez, *Accurate computations with Said-Ball-Vandermonde matrices*, Linear Algebra Appl. 432(11): 2894-2908, 2010.

[19] J. F. Canny, E. Kaltofen, and L. Yagati, *Solving systems of non-linear equations faster*, In Proc. ACM-SIGSAM ISSAC: 34-42, 1989.

[20] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp. 19:297-301, 1965.

[21] G. Strang, *Introduction to Applied Mathematics*, Wesley-Cambridge Press, USA, 1986.

[22] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM Publications, Philadelphia, USA, 1992.

[23] S. G. Johnson and M. Frigo, *A modified split-radix FFT with fewer arithmetic operations*, IEEE Trans. Signal Process. 55 (1):111-119, January 2007.

[24] C. Wijenayake, A. Madanayake, Y. Xu, L. Belostotski, and L.T. Bruton, *A Steerable DC-1 GHz all-pass filter-Sum RF space-time 2-D beam filter in 65 nm CMOS,* IEEE Int. Symp. Circuits Syst., pp. 1276-1279, August 2013.

[25] C. Wijenayake, A. Madanayake, L. Belostotski, Y. Xu and L.T. Bruton, *All-Pass Filter-Based 2-D IIR Filter-Enhanced Beamformers for AESA Receivers,* IEEE TCAS-I 61(5): 1331-1342, March 2014.

[26] K. R. Rao, D.N. Kim, J. J. Hwang, *Fast Fourier Transform: Algorithm and Applications*, Springer, New York, USA, 2010.

[27] R. E. Blahut, *Fast Algorithms for Signal Processing*, Cambridge University Press, 2010.

[28] D. A. Bini, *Matrix Structures in Queuing Models* In Benzi M., Simoncini V. (eds), Exploiting Hidden Structure in Matrix Computations: Algorithms and Applications, Lecture Notes in Mathematics 2173, pp. 65-160, Springer, Cham, 2016.

[29] T. Kailath and A. H. Sayed, *Fast Reliable Algorithms for Matrices with Structure*, SIAM, Philadelphia, 1999.

[30] R. Yavne, *An economical method for calculating the discrete Fourier transform*, In Proc. AFIPS FJCC 33, pp. 115-125, San Francisco, California, December 1968.

[31] P. Ahmadi, B. Maundy, A. S. Elwakil, L. Belostotski and A. Madanayake, *A New Second-Order All-Pass Filter in 130-nm CMOS,* IEEE TCAS-II: Express Briefs, 63(3): 249-253, March 2016.

[32] 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 7: 78729-78757, June 2019.

[33] 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. Top. Circuits. 8(3): 466-479, May 2018.

[34] 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 2015 IEEE MTT-S International Microwave Symposium, pp. 1-4. IEEE, 2015.

[35] H. Zhao, S. Mandal, V. Ariyarathna, A. Madanayake, and R. J. Cintra, *An offset-canceling approximate-DFT beamforming architecture for wireless transceivers.* In 2018 IEEE ISCAS, pp. 1-5. IEEE, 2018.

[36] K. Spoof, V. Unnikrishnan, M. Zahra, K. Stadius, M. Kosunen, and J. Ryynänen, *True-time-delay beamforming receiver with RF re-sampling,* IEEE TCAS-I 67(12): 4457-4469, December 2020.

[37] C. C. Enz and G. C. Temes, *Circuit techniques for reducing the effects of op-amp imperfections: autozeroing, correlated double sampling, and chopper stabilization,* In Proc of the IEEE 84(11): 1584-1614, November 1996.

[38] D. L. Boleya, F. T. Luk, and D. Vandevoorde, *A fast method to diagonalize a Hankel matrix*, Linear Algebra Appl: 284(1-3), 41-52, 1998.

[39] T. Bäckström, *Vandermonde factorization of Toeplitz matrices and applications in filtering and warping*, IEEE Trans. Signal Process. 61(24): 6257-6263, 2013.
[40] P. Stoica and R. L. Moses, *Spectral analysis of signals*, Pearson/Prentice Hall Upper Saddle River, NJ, 2005.
[41] T. Bäckström, J. Fischer, and D. Boley, *Implementation and evaluation of the Vandermonde transform*, In 2014 22nd EUSIPCO, Lisbon, Portugal, IEEE, 2014.
[42] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhauser/Springer, Boston, New York, 2001.
[43] F. Parker, *Inverses of Vandermonde matrices*, Amer. Math. Monthly 71: 410 - 411, 1964.
[44] A. Björck and V. Pereyra, *Solution of Vandermonde systems of equations*, Math. Comp. 24: 893-903, 1970.
[45] L. Reichel and G. Opfer, *Chebyshev-Vandermonde systems*, Math. Comp. 57:703-721, 1991.
[46] T. Bella, Y. Eidelman, I. Gohberg, I. Koltracht, and V. Olshevsky, *A Björck-Pereyra-type algorithm for Szegö-Vandermonde matrices based on properties of unitary Hessenberg matrices*, Linear Algebra Appl. 420(2-3): 634-647, 2007.
[47] T. Bella, Y. Eidelman, I. Gohberg, and I., V. Olshevsky, *A fast Björck-Pereyra-type algorithm for solving Hessenberg-quasiseparable-Vandermonde systems*, SIMAX 31(2): 790-815, 2009.
[48] S. M. Perera, *Quasiseparable Approach to Matrices of Vandermonde Type*, Thesis, University of Connecticut, ProQuest Publication, Ann Arbor, USA, (2012), 207 pages, UMI Number: 3533908.
[49] S. M. Perera, A. Madanayake, A. Ogle, D. Silverio, and J. Qi, *A Fast Algorithm to Solve Delay Vandermonde Systems in Phased-Array Digital Receivers*, IEEE Trans. Aerosp. Electron. Syst. 57(4): 2288-2297, 2021.
[50] J. Demmel and P. Koev, *The Accurate and Efficient Solution of a Totally Positive Generalized Vandermonde Linear System*, SIMAX 27(1):142-152, 2005.
[51] T. Kailath, and V. Olshevsky, *Displacement structure approach to polynomial Vandermonde and related matrices*, Linear Algebra Appl., 261, 49-90, 1997.
[52] R. Sarpeshkar, *Analog versus digital: extrapolating from electronics to neurobiology,* Neural Computation 10(7): 1601-1638, 1998.