Radix-2 Self-Recursive Sparse Factorizations of Delay Vandermonde Matrices for Wideband Multi-Beam Antenna Arrays

SIRANI M. PERERA [1], **ARJUNA MADANAYAKE** [2], **(Member, IEEE),**
AND RENATO J. CINTRA [3,4], **(Senior Member, IEEE)**
[1]Department of Mathematics, Embry–Riddle Aeronautical University, Daytona Beach, FL 32114, USA
[2]Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA
[3]Departamento de Estatística, Universidade Federal de Pernambuco, 50740540 Recife, Brazil
[4]Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada

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

ABSTRACT This paper presents a self-contained factorization for the Vandermonde matrices associated with true-time delay based wideband analog multi-beam beamforming using antenna arrays. The proposed factorization contains sparse and orthogonal matrices. Novel self-recursive radix-2 algorithms for Vandermonde matrices associated with true time delay based delay-sum filterbanks are presented to reduce the circuit complexity of multi-beam analog beamforming systems. The proposed algorithms for Vandermonde matrices by a vector attain $\mathcal{O}(N \log N)$ delay-amplifier circuit counts. Error bounds for the Vandermode matrices associated with true-time delay are established and then analyzed for numerical stability. The potential for real-world circuit implementation of the proposed algorithms will be shown through signal flow graphs that are the starting point for high-frequency analog circuit realizations.

INDEX TERMS Sparse matrices, algorithm design and analysis, computational complexity, accuracy, error analysis, fast fourier transforms, antenna arrays, integrated circuits, wireless communication.

I. INTRODUCTION

The realization of narrowband discrete Fourier transform (DFT) multi-beams is itself a hard engineering problem due to circuit complexity of the aperture transceivers. For example, the phasing network required for forming N beams requires N^2 phasing elements. The DFT is a linear operation that maps an N-point input signal $\mathbf{x} = \begin{bmatrix} x[0] & x[1] & \cdots & x[N-1] \end{bmatrix}^\top$ into an N-point output signal $\mathbf{X} = \begin{bmatrix} X[0] & X[1] & \cdots & X[N-1] \end{bmatrix}^\top$ according to the following relationship: $\mathbf{X} = \mathbf{F}_N \cdot \mathbf{x}$, where $\mathbf{F}_N$ is the DFT matrix, whose elements are given by ω_N^{kl}, $k, l = 0, 1, \ldots, N-1$, where $\omega_N = \exp\left(-j\frac{2\pi}{N}\right)$ is the Nth root of unity and $j = \sqrt{-1}$.

Evaluated by means of direct matrix-vector multiplications, the direct computational complexity of the DFT is in $\mathcal{O}(N^2)$, with N^2 complex multiplications and $N(N-1)$ complex additions. The DFT matrix has been studied for the

last 50 years, and there exist a multitude of fast algorithms (collectively called fast Fourier transforms (FFTs)) that compute the DFT using $\mathcal{O}(N \log N)$ operations, which is significantly lower when compared with the direct implementations. The use of a spatial FFT leads to N independent orthogonal RF beams at $\mathcal{O}(N \log N)$ complexity. In fact, by taking a given FFT algorithm and implementing its "Twiddle Factors" (which are intermediate constant complex multiplications found in FFT algorithms) using microwave or analog IC-based phase-shifter implementations has led to the "Butler Matrix" type multi-beam array beamformers that are well known in the literature. However, such FFT beams suffer from frequency dependent beam directions. Known as "beam squint" because the beam directions are strongly dependent on the temporal frequency of operation, DFT based multi-beam beamformers can only be used for narrowband wireless systems.

The FFT is capable of computing the DFT or its inverse in $\mathcal{O}(N \log N)$ complexity. Therefore, FFT-based multi-beam beamformers are very useful for wireless systems having

---

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

narrow bandwidth. However, for emerging 5G mmW systems that exploit increasingly wide bandwidths, the beam-squint problem can be significant. For emerging 5G mmW systems that fully exploit the available bandwidth for increasing system capacity, one must utilize the true time-delay based multi-beam beamformers described by its own delay Vandermonde matrix (DVM). The DVM, however, is equal to the DFT only at a single temporal frequency. Therefore, FFT-based factorizations are not applicable for the DVM matrix. In this paper, we describe the complexity of an FFT-like factorization algorithm for the Vandermonde matrices, in order to be able to implement truly wideband multi-beam mmW beamformers based on true-time-delay networks albeit at $\mathcal{O}(N \log N)$ complexity.

The paper is organized as follows. Section II contains an introduction to complexity metrics of analog and digital parallel computation systems for matrix-vector products. Section III introduces novel self-contained factorizations for Vandermonde matrices and radix-2 algorithms, while in section IV we will derive arithmetic complexity and elaborate on numerical results based on the proposed algorithms for Vandermonde matrices. Next, section V analyzes error bounds and stability in computing radix-2 algorithms for Vandermonde matrices having true time-delays. In section VI we will present signal flow graphs of the proposed radix-2 algorithm for Vandermonde matrices. Finally, section VII concludes the paper.

II. ANALOG IMPLEMENTATIONS FOR 5G AND BEYOND: QUANTIFYING COMPLEXITY

Fast analog radio frequency (RF) integrated circuit (IC) realizations of the beamforming algorithms become necessary when the bandwidths of interest are greater than a few GHz. For emerging 5G, 6G and beyond, the bandwidths of interest are too high for digital computing solutions to keep up. The solution is to replace digital systems with fast analog implementations of wideband beamforming algorithms, which in turn, requires a revisit to traditional algorithm complexity theory because of differences in analog parallel architectures compared to conventional digital approaches. In analog implementations, the bandwidth effectively sets the rate at which the analog computation can be updated. The DVM building block employs true time delays that can be realized using transmission line segments and/or all-pass networks followed by amplification stages.

Let us define DVM fast algorithms as consisting of gain-delay-block (GDB) and addition/subtraction blocks. Instead of computing the number of multiplications for accessing with arithmetic complexity (as one would do for digital systems), we need to count the number of parallel circuit implementations of GDBs in order to access the circuit complexity of analog parallel algorithms. The larger the number of GDBs, the higher the circuit complexity and hence higher chip area and power consumption. In analog fast algorithms, the objective is to factorize the original matrix into products

of sparse matrices, such that the total number of GDBs is reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$.

We remark here that the gain is not equivalent to the coefficient multiplication. Although a delay of t is simply multiplication by $e^{-j\omega t}$ in the mathematical sense, it requires a separate true time delay circuit in the analog domain. Hence, the multiplication complexity is different from GBD counts.

III. SELF-CONTAINED FACTORIZATION AND ALGORITHM FOR VANDERMONDE MATRICES

Low complexity and stable algorithms for the delay Vandermonde matrix, $\mathbf{A}_N = [\alpha^{kl}]_{k=1,l=0}^{N,N-1}$, where $\alpha = e^{-j\omega_t \tau}$ and accounts for the phase rotation associated with the delay τ at frequency f, and $\omega_t = 2\pi f$, have been derived through our previous work [1], [16], [17]. It is important to realize that the matrix elements are integer powers of $\alpha = e^{-j\omega_t \tau}$ which are functions of the temporal frequency variable ω_t; this is an important distinction from the DFT matrix where the elements are constants defined as the primitive Nth roots of unity. Because integer powers of $\alpha = e^{-j\omega_t \tau}$ are dependent on ω_t the DVM frequency responses are functions of two frequency variables: ω_x, which is typically a spatial variable, and ω_t which is typically the temporal frequency variable. The DVM matrix frequency responses are defined using the spatial frequency variable ω_x via 2-D filterbank responses that contain ω_t as a parameter, and given by the expression for the kth filter for $k = 0, 1, \ldots, N-1$ as $H_k(j\omega_x, j\omega_t) = \sum_i \alpha^{ki} e^{-j\omega_x i}$, $i = 0, 1, \ldots, N-1$. Therefore, considering both ω_x and ω_t the DVM defines N 2-D frequency responses.

Further, the DVM is the super-class of the DFT matrix without having nice properties like unitary, periodicity, symmetry, and circular shift. There is no self-contained radix-2 DVM algorithm in the literature. The manuscript [17] proposes a self-contained sparse factorization of DVM with $\mathcal{O}(N^2)$ arithmetic complexity. The displacement structure of Vandermonde-related matrices is used to derive $\mathcal{O}(N \log^2 N)$ arithmetic complexity algorithms in [7], [8] and an $\mathcal{O}(N)$ arithmetic complexity algorithm in [14]. The manuscripts [12], [13], [23] propose $\mathcal{O}(N^2)$ complexity algorithms to compute Vandermonde matrices (having real nodes) by a vector. The DVM algorithm in [17] extends the results in [12], [13], [23] utilizing complex nodes without using displacement equations as in [7], [8], [14]. Moreover, we have addressed the error bounds and stability of the DVM algorithm in [17] by filling the gaps in [12], [13], [23]. The DVM algorithm in [16] is faster than [17] but does not produce arithmetic complexity of order $\mathcal{O}(N \log N)$. On the other hand, there are no constraints for nodes of DVM in [17] as opposed to what we propose here.

In this section, we derive novel self-contained factorization for the Vandermode-type matrices and propose a radix-2 algorithm for the Vandermonde matrices. We will account for the phase rotation associated with delay and frequency in the factorization of Vandermonde matrices.

A. SELF-CONTAINED FACTORIZATION FOR VANDERMONDE MATRICES

Algorithms operating on analog signals for computing Vandermonde matrix by a vector can be seen as the evaluation of $(N - 1)^{th}$ degree polynomial at N points, albeit using a paralleled analog computing circuit as opposed to a digital realization that must operate on samples and quantized signals. Here we derive self-contained factorization of Vandermonde matrices to obtain efficient continuous-time algorithms for implementation on analog circuits while reducing GDB counts.

One can observe the computation of Vandermonde matrix by a vector with arithmetic complexity $\mathcal{O}(N \log^2 N)$ in [4], [7], [8]. Here, arithmetic complexity refers to the number of GDBs in an analog RF-IC circuit implementation, unlike the traditional approach of the number of multipliers and adders in a digital system. There are several mathematical techniques available to derive radix-2 and split-radix FFT algorithms, as described in [3], [10], [18], [20], [22]. It has been shown in [15] that Vandermonde matrices are badly ill-conditioned with a narrow class of exceptions whereas cyclic sequences of nodes are equally spaced on the unit circle $C(0, 1)$. In here, we propose self-contained and sparse factorization for the well-conditioned Vandermonde matrices and extend the results for $C(0, r)$, where $r > 1$ (i.e. circle of radius r centered at the origin in the complex plane). The proposed factorizations will then be used to derive fast algorithms while reducing GDB counts.

Theorem 1: Let the Vandermonde matrix $\mathbf{V}_N = [v_k^l]_{k,l=0}^{N-1}$ be defined by equally spaced nodes $\{v_0, v_1, \ldots, v_{N-1}\}$ on $C(0, 1)$ (in counterclockwise direction) and $N = 2^t$ $(t \geq 1)$. Then $\mathbf{V}_N(v_0, v_1, \ldots, v_{N-1})$ can be factored into

$$\mathbf{V}_N = P_N^T \begin{bmatrix} \mathbf{V}_{\frac{N}{2}} & \\ & \mathbf{V}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I_{\frac{N}{2}} & \\ & \dot{D}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I_{\frac{N}{2}} & I_{\frac{N}{2}} \\ I_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I_{\frac{N}{2}} & \\ & c \cdot I_{\frac{N}{2}} \end{bmatrix} \quad (1)$$

where P_N is the even-odd permutation matrix, $I_{\frac{N}{2}}$ is the identity matrix, $\dot{D}_{\frac{N}{2}} = \text{diag}[e^{l(\frac{2\pi j}{N})}]_{l=0}^{\frac{N}{2}-1}$, $c = e^{\frac{j\theta N}{2}}$, and $0 \leq \theta < 2\pi$.

Proof: Let us permute rows of $\mathbf{V}_N$ by multiplying with P_N and write the result as the block matrices:

$$P_N \mathbf{V}_N = \left[\begin{array}{c|c} \left[v_{2k}^l\right]_{k,l=0}^{\frac{N}{2}-1} & \left[v_{2k}^{\left(\frac{N}{2}+l\right)}\right]_{k,l=0}^{\frac{N}{2}-1} \\ \hline \left[v_{2k+1}^l\right]_{k,l=0}^{\frac{N}{2}-1} & \left[v_{2k+1}^{\left(\frac{N}{2}+l\right)}\right]_{k,l=0}^{\frac{N}{2}-1} \end{array} \right]. \quad (2)$$

It is clear that the (1,1) block of the product $P_N \mathbf{V}_N$ is $\mathbf{V}_{\frac{N}{2}}$. Now, we consider (1,2), (2,1), and (2,2) blocks of $P_N \mathbf{V}_N$ (2) and represent each of these by $\mathbf{V}_{\frac{N}{2}}$ and the product of diagonal matrices.

By (1,2) block of (2) we get:

$$\left[v_{2k}^{\left(\frac{N}{2}+l\right)} \right]_{k,l=0}^{\frac{N}{2}-1} = \text{diag}\left[v_{2k}^{\frac{N}{2}}\right]_{k=0}^{\frac{N}{2}-1} \cdot \left[v_{2k}^l\right]_{k,l=0}^{\frac{N}{2}-1}. \quad (3)$$

Since nodes are equally spaced on $C(0, 1)$, we have $v_{2k+1} = v_{2k} \cdot e^{\frac{2\pi j}{N}}$, for $k = 0, 1, \ldots, \frac{N}{2} - 1$. Now by (2,1) block of (2) we get:

$$\left[v_{2k+1}^l\right]_{k,l=0}^{\frac{N}{2}-1} = \left[v_{2k}^l\right]_{k,l=0}^{\frac{N}{2}-1} \cdot \text{diag}[e^{l(\frac{2\pi j}{N})}]_{l=0}^{\frac{N}{2}-1}. \quad (4)$$

By (2,2) block of (2) we get:

$$\left[v_{2k+1}^{\left(\frac{N}{2}+l\right)} \right]_{k,l=0}^{\frac{N}{2}-1} = -\text{diag}\left[v_{2k}^{\frac{N}{2}}\right]_{k=0}^{\frac{N}{2}-1} \cdot \left[v_{2k}^l\right]_{k,l=0}^{\frac{N}{2}-1} \cdot$$
$$\text{diag}[e^{l(\frac{2\pi j}{N})}]_{l=0}^{\frac{N}{2}-1}. \quad (5)$$

Thus by (3), (4), and (5), we can state (2) as:

$$P_N \mathbf{V}_N = \left[\begin{array}{c|c} \mathbf{V}_{\frac{N}{2}} & D_{\frac{N}{2}} \cdot \mathbf{V}_{\frac{N}{2}} \\ \hline \mathbf{V}_{\frac{N}{2}} \cdot \dot{D}_{\frac{N}{2}} & -D_{\frac{N}{2}} \cdot \mathbf{V}_{\frac{N}{2}} \cdot \dot{D}_{\frac{N}{2}} \end{array} \right] \quad (6)$$

where $D_{\frac{N}{2}} = \text{diag}\left[v_{2k}^{\frac{N}{2}}\right]_{k=0}^{\frac{N}{2}-1}$. Let us consider the product of mth row of $\mathbf{V}_N$ and lth column of $\mathbf{V}_N^H$, where $\mathbf{V}_N^H$ is the conjugate transpose of $\mathbf{V}_N$. Thus, we have:

$$\mathbf{V}_N(m, :) \cdot \mathbf{V}_N^H(:, l)$$
$$= 1 + v_{m-1}\bar{v}_{l-1} + v_{m-1}^2 \bar{v}_{l-1}^2 + \cdots + v_{m-1}^{(N-1)}\bar{v}_{l-1}^{(N-1)}$$
$$= \begin{cases} N, & \text{when } m = l, \\ 0, & \text{when } m \neq l. \end{cases}$$

In the above, the first equality follows as $v_k, \bar{v}_k \in C(0, 1)$ for $k = 0, 1, \ldots, N - 1$ and the second equality follows as $v_{2k+1} = v_{2k} \cdot e^{\frac{2\pi j}{N}}$. Hence, $\mathbf{V}_N$ is unitary up to scaling by $\frac{1}{\sqrt{N}}$. By using this we can state (6) as:

$$P_N \mathbf{V}_N = \begin{bmatrix} \mathbf{V}_{\frac{N}{2}} & \\ & \mathbf{V}_{\frac{N}{2}} \end{bmatrix}$$
$$\times \left[\begin{array}{c|c} I_{\frac{N}{2}} & \frac{2}{N} \cdot \mathbf{V}_{\frac{N}{2}}^H \cdot D_{\frac{N}{2}} \cdot \mathbf{V}_{\frac{N}{2}} \\ \hline \dot{D}_{\frac{N}{2}} & -\frac{2}{N} \cdot \mathbf{V}_{\frac{N}{2}}^H \cdot D_{\frac{N}{2}} \cdot \mathbf{V}_{\frac{N}{2}} \cdot \dot{D}_{\frac{N}{2}} \end{array} \right]. \quad (7)$$

Now let us consider the product $\mathbf{V}_{\frac{N}{2}}^H \cdot D_{\frac{N}{2}} \cdot \mathbf{V}_{\frac{N}{2}}$ i.e. the product of mth row of $\mathbf{V}_{\frac{N}{2}}^H \cdot D_{\frac{N}{2}}$ -say $\hat{\mathbf{V}}_{\frac{N}{2}}$ and lth column of $\mathbf{V}_{\frac{N}{2}}$. Therefore, we have that

$$\hat{\mathbf{V}}_{\frac{N}{2}}(m, :) \cdot \mathbf{V}_{\frac{N}{2}}(:, l)$$
$$= \bar{v}_0^{m-1} v_0^{\frac{N}{2}} v_0^{l-1} + \bar{v}_2^{m-1} v_2^{\frac{N}{2}} v_2^{l-1} + \bar{v}_4^{m-1} v_4^{\frac{N}{2}} z_4^{l-1}$$
$$+ \cdots + \bar{v}_{N-2}^{m-1} v_{N-2}^{\frac{N}{2}} v_{N-2}^{l-1}$$

$$= \begin{cases} \sum_{k=0}^{\frac{N}{2}-1} v_{2k}^{\frac{N}{2}}, & \text{when } m = l, \\ 0, & \text{when } m \neq l. \end{cases}$$

In the above, the first equality follows as $v_{2k}, \bar{v}_{2k} \in C(0,1)$ and the second equality follows as v_{2k} are nodes of $\mathbf{V}_{\frac{N}{2}}$ and $v_{2k+2} = v_{2k} \cdot e^{\frac{4\pi j}{N}}$. Thus, by following the above one can see the (m, l) entry of $\hat{\mathbf{V}}_{\frac{N}{2}} \cdot \mathbf{V}_{\frac{N}{2}} \cdot \dot{D}_{\frac{N}{2}}$ as

(m, l) entry of $\hat{\mathbf{V}}_{\frac{N}{2}} \cdot \mathbf{V}_{\frac{N}{2}} \cdot \dot{D}_{\frac{N}{2}}$

$$= \begin{cases} \left(\sum_{k=0}^{\frac{N}{2}-1} v_{2k}^{\frac{N}{2}} \right) e^{l\left(\frac{2\pi j}{N}\right)}, & \text{when } m = l, \\ 0, & \text{when } m \neq l. \end{cases}$$

Notice that even nodes on $C(0,1)$ can be expressed as $v_{2k} = e^{j\left(\theta + \frac{4\pi k}{N}\right)}$ for $k = 0, 1, \ldots, \frac{N}{2} - 1$. Thus, by raising each even node to the power of $\frac{N}{2}$ and taking average we get $c = e^{\frac{j\theta N}{2}}$ where $j^2 = -1$. Hence,

$$\mathbf{V}_N = P_N^T \begin{bmatrix} \mathbf{V}_{\frac{N}{2}} & \\ & \mathbf{V}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I_{\frac{N}{2}} & c \cdot I_{\frac{N}{2}} \\ \hline \dot{D}_{\frac{N}{2}} & -c \cdot \dot{D}_{\frac{N}{2}} \end{bmatrix} \quad (8)$$

and the claim of the theorem follows. □

Remark 2: The last matrix in the factorization (8) has been split into three sparse matrices in (1) to reduce the multiplication counts and hence for efficient hardware implementation.

Corollary 3: Let the Vandermonde matrix $\tilde{\mathbf{V}}_N = [\tilde{v}_k^l]_{k,l=0}^{N-1}$ be defined by equally spaced nodes $\{\tilde{v}_0, \tilde{v}_1, \ldots, \tilde{v}_{N-1}\}$ on $C(0, r)$, where $r > 1$ (in counterclockwise direction) and $N = 2^t$ ($t \geq 1$). Then $\tilde{\mathbf{V}}_N(\tilde{v}_0, \tilde{v}_1, \ldots, \tilde{v}_{N-1})$ can be factored into

$$\tilde{\mathbf{V}}_N = \mathbf{V}_N \tilde{\mathbf{D}}_N \quad (9)$$

where $\tilde{\mathbf{D}}_N = \mathrm{diag}[r^l]_{l=0}^{N-1}$ and $\mathbf{V}_N$ is defined via (1).

Proof: This is trivial as $\tilde{v}_k = r \cdot v_k$ for $k = 0, 1, \ldots, N - 1$. □

The following self-contained factorization for the Vandermonde matrices is proposed in connection to the phase rotation associated with delay τ and frequency $\omega_t = 2\pi f$.

Theorem 4: Let the Vandermonde matrix $\mathbf{V}_N = [v_k^l]_{k,l=0}^{N-1}$ be defined by equally spaced nodes $\{v_0, v_1, \ldots, v_{N-1}\}$ on $C(0, 1)$ (in clockwise direction) and $N = 2^t$ ($t \geq 1$). Then $\mathbf{V}_N(v_0, v_1, \ldots, v_{N-1})$ can be factored into

$$\mathbf{V}_N = P_N^T \begin{bmatrix} \mathbf{V}_{\frac{N}{2}} & \\ & \mathbf{V}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I_{\frac{N}{2}} & \\ & \bar{D}_{\frac{N}{2}} \end{bmatrix} \begin{bmatrix} I_{\frac{N}{2}} & I_{\frac{N}{2}} \\ \hline I_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{bmatrix}$$
$$\begin{bmatrix} I_{\frac{N}{2}} & \\ & \bar{c} \cdot I_{\frac{N}{2}} \end{bmatrix} \quad (10)$$

where $I_{\frac{N}{2}}$ is the identity matrix, $\bar{D}_{\frac{N}{2}} = \mathrm{diag}[e^{-l\left(\frac{2\pi j}{N}\right)}]_{l=0}^{\frac{N}{2}-1}$, $\bar{c} = e^{-\frac{j\theta N}{2}}$, and $\theta = 2\pi f \tau = \omega_t \tau$, s.t. $0 \leq \theta < 2\pi$.

Proof: The proof follows similar lines as that of Theorem 1, except $\bar{D}_{\frac{N}{2}} = \mathrm{diag}[e^{-l\left(\frac{2\pi j}{N}\right)}]_{l=0}^{\frac{N}{2}-1}$ instead of $\dot{D}_{\frac{N}{2}} = \mathrm{diag}[e^{l\left(\frac{2\pi j}{N}\right)}]_{l=0}^{\frac{N}{2}-1}$ and $\bar{c}$ instead of c. □

Remark 5: Theorem 4 has proposed a self-contained factorization, as opposed to a scaled DFT matrix. If one chooses to scale DFT matrices to factor $\mathbf{V}_N$, it results in the computation of small complex numbers and leads to zero matrices [9]. The proposed factorization for $\mathbf{V}_N$ in (10) overcomes this barrier.

Corollary 6: Let the Vandermonde matrix $\tilde{\mathbf{V}}_N = [\tilde{v}_k^l]_{k,l=0}^{N-1}$ be defined by equally spaced nodes $\{\tilde{v}_0, \tilde{v}_1, \ldots, \tilde{v}_{N-1}\}$ on $C(0, r)$, where $r > 1$ (in clockwise direction) and $N = 2^t$ ($t \geq 1$). Then $\tilde{\mathbf{V}}_N(\tilde{v}_0, \tilde{v}_1, \ldots, \tilde{v}_{N-1})$ can be factored into

$$\tilde{\mathbf{V}}_N = \mathbf{V}_N \tilde{\mathbf{D}}_N \quad (11)$$

where $\tilde{\mathbf{D}}_N = \mathrm{diag}[r^l]_{l=0}^{N-1}$ and $\mathbf{V}_N$ is defined via (10).

Proof: This is trivial as $\tilde{v}_k = r \cdot v_k$ for $k = 0, 1, \ldots, N - 1$. □

Remark 7: When $\theta = 0$ and $r = 1$, the proposed factorization for the Vandermode matrices given in Theorem 4, reduces to the well known self-contained DFT matrix factorization [3], [19], [22], [24]. Thus, we can use this property to define a delay Vandermonde matrix to solve the beam squint problem as well as allow high-speed analog realizations for future high bandwidth applications where the slowing down of Moore's law prevents the adoption of digital parallel processing architectures.

B. SELF-RECURSIVE ALGORITHMS FOR VANDERMONDE MATRICES

In the following, we will state self-recursive radix-2 algorithms for Vandermonde matrices with the help of the Theorem 1, Theorem 4, Corollary 3 and Corollary 6. Let us call the corresponding algorithms **vanc(N)**, **vancc(N)**, **vancr(N)**, and **vanccr(N)** respectively, e.g., the acronym **vancr(N)** was selected to refer to the factorization for the Vandemode matrices having clockwise nodes on the circle of radius r. We use the following notation for the inputs of the algorithms i.e. N for the size of the matrices, θ, where $0 \leq \theta < 2\pi$, for the angle of rotation from the positive real axis (positive or negative based on counterclockwise or clockwise direction), r for the magnitude, and $\mathbf{z}$ for the input vector.

Before stating algorithms, let us use the following notation to denote sparse matrices which will be used hereafter for $N \geq 4$.

$$\hat{\mathbf{D}}_N = \begin{bmatrix} I_{\frac{N}{2}} & \\ & \dot{D}_{\frac{N}{2}} \end{bmatrix}, \quad \check{\mathbf{D}}_N = \begin{bmatrix} I_{\frac{N}{2}} & \\ & \bar{D}_{\frac{N}{2}} \end{bmatrix}$$

$$\hat{\mathbf{I}}_N = \begin{bmatrix} I_{\frac{N}{2}} & I_{\frac{N}{2}} \\ \hline I_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{bmatrix},$$

$$\mathbf{C}_N = \begin{bmatrix} I_{\frac{N}{2}} & \\ & c \cdot I_{\frac{N}{2}} \end{bmatrix}, \quad \text{and} \quad \bar{\mathbf{C}}_N = \begin{bmatrix} I_{\frac{N}{2}} & \\ & \bar{c} \cdot I_{\frac{N}{2}} \end{bmatrix}. \quad (12)$$

Algorithm 8: **vancc(z, N)**

Input: $N = 2^t$ $(t \geq 1)$, $N_1 = \frac{N}{2}$, θ, and $\mathbf{z} \in \mathbb{R}^n$ or $\mathbb{C}^n$.

1) *If $N = 2$, then*
$$\mathbf{y} = \begin{bmatrix} 1 & e^{j\theta} \\ 1 & -e^{j\theta} \end{bmatrix} \mathbf{z}.$$
2) *If $N \geq 4$, then*
$\mathbf{u} := \mathbf{C}_N \mathbf{z}$,
$\mathbf{v} := \hat{\mathbf{I}}_N \mathbf{u}$,
$\mathbf{w} := \hat{\mathbf{D}}_N \mathbf{v}$,
$\mathbf{s1} := \mathbf{vancc}\left([w_i]_{i=0}^{N_1-1}, N_1 \right)$,
$\mathbf{s2} := \mathbf{vancc}\left([w_i]_{i=N_1}^{N}, N_1 \right)$,
$\mathbf{y} := \mathbf{P}_N^T \left(\mathbf{s1}^T, \mathbf{s2}^T \right)^T$.

Output: $\mathbf{y} = \mathbf{V}_N \mathbf{z}$.

Algorithm 9: **vanc(z, N)**

Input: $N = 2^t$ $(t \geq 1)$, $N_1 = \frac{N}{2}$, θ, and $\mathbf{z} \in \mathbb{R}^n$ or $\mathbb{C}^n$.

1) *If $N = 2$, then*
$$\mathbf{y} = \begin{bmatrix} 1 & e^{-j\theta} \\ 1 & -e^{-j\theta} \end{bmatrix} \mathbf{z}.$$
2) *If $N \geq 4$, then*
$\mathbf{u} := \mathbf{C}_N \mathbf{z}$,
$\mathbf{v} := \hat{\mathbf{I}}_N \mathbf{u}$,
$\mathbf{w} := \hat{\mathbf{D}}_N \mathbf{v}$,
$\mathbf{s1} := \mathbf{vanc}\left([w_i]_{i=0}^{N_1-1}, N_1 \right)$,
$\mathbf{s2} := \mathbf{vanc}\left([w_i]_{i=N_1}^{N}, N_1 \right)$,
$\mathbf{y} := \mathbf{P}_N^T \left(\mathbf{s1}^T, \mathbf{s2}^T \right)^T$.

Output: $\mathbf{y} = \mathbf{V}_N \mathbf{z}$.

Algorithm 10: **vanccr(z, N)**

Input: $N = 2^t$ $(t \geq 1)$, $N_1 = \frac{N}{2}$, r, θ, and $\mathbf{z} \in \mathbb{R}^n$ or $\mathbb{C}^n$.

1) *If $N = 2$, then*
$$\mathbf{y} = \begin{bmatrix} 1 & re^{j\theta} \\ 1 & -re^{j\theta} \end{bmatrix} \mathbf{z}.$$
2) *If $N \geq 4$, then*
$\mathbf{u} := \tilde{\mathbf{D}}_N \mathbf{z}$,
$\mathbf{y} := \mathbf{vancc}\left([u_i]_{i=0}^{N-1}, N \right)$.

Output: $\mathbf{y} = \tilde{\mathbf{V}}_N \mathbf{z}$.

Algorithm 11: **vancr(z, N)**

Input: $N = 2^t$ $(t \geq 1)$, $N_1 = \frac{N}{2}$, r, θ, and $\mathbf{z} \in \mathbb{R}^n$ or $\mathbb{C}^n$.

1) *If $N = 2$, then*
$$\mathbf{y} = \begin{bmatrix} 1 & re^{-j\theta} \\ 1 & -re^{-j\theta} \end{bmatrix} \mathbf{z}.$$
2) *If $N \geq 4$, then*
$\mathbf{u} := \tilde{\mathbf{D}}_N \mathbf{z}$,
$\mathbf{y} := \mathbf{vanc}\left([u_i]_{i=0}^{N-1}, N \right)$.

Output: $\mathbf{y} = \tilde{\mathbf{V}}_N \mathbf{z}$.

IV. ANALOG GDB-COMPLEXITY

The number of additions and multiplications required to carry out a computation is called the arithmetic complexity in a digital computing system. Here, because our intention is to realize these algorithms as high-speed analog computing circuits operating at RF, we use the modified arithmetic complexity metric where we are counting the number of GDBs instead of multipliers. In this section, the GDB counts

of the proposed self-contained factorization for the Vandermonde matrices via algorithms **vanc(z, N)**, **vancc(z, N)**, **vancr(z, N)**, and **vanccr(z, N)** will be addressed. The direct analog computation of the Vandermonde matrix by a vector $\mathbf{z} \in \mathbb{C}$ in the usual way requires $\mathcal{O}(N^2)$ GDB circuits to be realized in parallel in the RF-IC analog computing device.

However, we will show in this section that the proposed self-recursive radix-2 algorithms can be utilized to compute Vandermonde matrices by a vector with $\mathcal{O}(N \log N)$ GDB counts.

This is a dramatic circuit complexity reduction of Vandermonde matrices by a vector in the literature. Although the computation speed is still the same, the new factorization reduces chip area and power consumption due to the smaller amount of GDB circuits that have to be physically realized on the analog computing device.

A. GDB COUNTS OF ANALOG FAST ALGORITHMS FOR VANDERMONDE MATRICES

Here we analyze the analog GDB counts of the radix-2 algorithms for Vandermonde matrices presented in Section III-A. Let us denote the number of complex/real additions (say $\#a\mathbb{C}/\#a\mathbb{R}$ respectively) and complex/real multiplications (say $\#m\mathbb{C}/\#m\mathbb{R}$ respectively) required to compute $\mathbf{y} = \mathbf{V}_N \mathbf{z}$ and $\mathbf{y} = \tilde{\mathbf{V}}_N \mathbf{z}$ having $\mathbf{z} \in \mathbb{C}^N$ or $\mathbb{R}^N$. We do not count multiplication by ± 1 and permutation.

Let us first analyze the complex GDB counts of the radix-2 algorithms for Vandermonde matrices by a complex input vector. We recall that the GDBs implement a complex multiplication defined in the frequency domain ω_t which requires a time-domain delay to implement on the DVM signal flow graphs. We recall that the independent frequency variable of the DVM is ω_x and that ω_t is the temporal frequency parameter associated with the matrix elements α. This is why the complex multiplication operations, which contain $e^{-j\omega_t \tau}$ terms, must in practice be realized in the time domain using time-delays.

Theorem 12: Let $N = 2^t (\geq 2)$ and θ be given. The complex GDB counts of the proposed **vancc(z, N)** algorithm with $\mathbf{z} \in \mathbb{C}^N$ is given by

$$\#a\mathbb{C}(VanCC, N) = Nt,$$
$$\#m\mathbb{C}(VanCC, N) = Nt - N + 1. \quad (13)$$

Proof: Referring to the algorithm **vancc(z, N)**, we get

$$\#a\mathbb{C}(VanCC, N) = 2 \cdot \#a\mathbb{C}\left(VanCC, \frac{N}{2} \right) + \#a\mathbb{C}\left(\hat{\mathbf{D}}_N \right)$$
$$+ \#a\mathbb{C}\left(\hat{\mathbf{I}}_N \right) + \#a\mathbb{C}\left(\mathbf{C}_N \right). \quad (14)$$

By following the structures of $\hat{\mathbf{D}}_N$, $\hat{\mathbf{I}}_N$ and $\mathbf{C}_N$,

$$\#a\mathbb{C}\left(\hat{\mathbf{D}}_N \right) = 0, \quad \#m\mathbb{C}\left(\hat{\mathbf{D}}_N \right) = \frac{N}{2} - 1,$$
$$\#a\mathbb{C}\left(\hat{\mathbf{I}}_N \right) = N, \quad \#m\mathbb{C}\left(\hat{\mathbf{I}}_N \right) = 0,$$
$$\#a\mathbb{C}\left(\mathbf{C}_N \right) = 0, \quad \#m\mathbb{C}\left(\mathbf{C}_N \right) = \frac{N}{2}. \quad (15)$$

Thus by using the above, we could state (14) as the first order difference equation with respect to $t \geq 2$

$$\#a\mathbb{C}(\text{VanCC}, 2^t) - 2 \cdot \#a\mathbb{C}\left(\text{VanCC}, 2^{t-1}\right) = 2^t.$$

Solving the above difference equation using the initial condition $\#a\mathbb{C}(\text{VanCC}, 2) = 2$, we can obtain

$$\#a\mathbb{C}(\text{VanCC}, 2^t) = Nt.$$

Now by using the algorithm **vancc**($\mathbf{z}, \mathbf{N}$) and (15), we could obtain another first order difference equation with respect to $t \geq 2$

$$\#m\mathbb{C}(\text{VanCC}, 2^t) - 2 \cdot \#m\mathbb{C}\left(\text{VanCC}, 2^{t-1}\right) = 2^t - 1.$$

Solving the above difference equation using the initial condition $\#m\mathbb{C}(\text{VanCC}, 2) = 1$, we can obtain

$$\#m\mathbb{C}(\text{VanCC}, 2^t) = Nt - N + 1.$$

$\square$

Corollary 13: Let $N = 2^t(\geq 2)$, r and θ be given. The complex GDB counts of the proposed **vanccr**($\mathbf{z}, \mathbf{N}$) algorithm with $\mathbf{z} \in \mathbb{C}^N$ is given by

$$\#a\mathbb{C}(VanCCR, N) = Nt,$$

$$\#m\mathbb{C}(VanCCR, N) = Nt - \frac{1}{2}N. \tag{16}$$

Proof: The multiplication of the diagonal matrix $\tilde{\mathbf{D}}_N$ with a complex input counts no addition and $\frac{N}{2} - 1$ multiplications. Thus by using **vanccr**($\mathbf{z}, \mathbf{N}$) algorithm and GDB counts in (13), the complex GDB counts can be obtained as in (13). $\square$

Theorem 14: Let $N = 2^t(\geq 2)$ and θ be given. The complex GDB counts of the proposed **vanc**($\mathbf{z}, \mathbf{N}$) algorithm with $\mathbf{z} \in \mathbb{C}^N$ is given by

$$\#a\mathbb{C}(VanC, N) = Nt,$$

$$\#m\mathbb{C}(VanC, N) = Nt - N + 1. \tag{17}$$

Proof: The proof follows similar lines as that of Theorem 12 except $\check{\mathbf{D}}_N$ instead of $\hat{\mathbf{D}}_N$ and $\bar{\mathbf{C}}_N$ instead of $\mathbf{C}_N$. $\square$

Corollary 15: Let $N = 2^t(\geq 2)$, r and θ be given. The complex GDB counts of the proposed **vancr**($\mathbf{z}, \mathbf{N}$) algorithm with $\mathbf{z} \in \mathbb{C}^N$ is given by

$$\#a\mathbb{C}(VanCR, N) = Nt,$$

$$\#m\mathbb{C}(VanCR, N) = Nt - \frac{1}{2}N. \tag{18}$$

Proof: The multiplication of the diagonal matrix $\tilde{\mathbf{D}}_N$ with a complex input counts no addition and $\frac{N}{2} - 1$ multiplications. Thus by using **vancr**($\mathbf{z}, \mathbf{N}$) algorithm and GDB counts in (17), the complex GDB counts can be obtained as in (18). $\square$

Let us analyze the real GDB counts of the radix-2 algorithms for Vandermonde matrices by a real input vector. Here we count the multiplication of two complex numbers with 2 real additions and 4 real multiplications.

Theorem 16: Let $N = 2^t(\geq 2)$ and θ be given. The real GDB counts of the proposed **vancc**($\mathbf{z}, \mathbf{N}$) algorithm with $\mathbf{z} \in \mathbb{R}^N$ is given by

$$\#a\mathbb{R}(VanCC, N) = Nt,$$

$$\#m\mathbb{R}(VanCC, N) = 2Nt - \frac{5}{2}N + 2. \tag{19}$$

Proof: Referring to the algorithm **vancc**($\mathbf{z}, \mathbf{N}$), we get

$$\#m\mathbb{R}(\text{VanCC}, N) = 2 \cdot \#m\mathbb{R}\left(\text{VanCC}, \frac{N}{2}\right) + \#m\mathbb{R}\left(\hat{\mathbf{D}}_N\right)$$
$$+ \#m\mathbb{R}\left(\hat{\mathbf{I}}_N\right) + \#m\mathbb{R}\left(\mathbf{C}_N\right). \tag{20}$$

By following the structures of $\hat{\mathbf{D}}_N$, $\hat{\mathbf{I}}_N$ and $\mathbf{C}_N$,

$$\#a\mathbb{R}\left(\hat{\mathbf{D}}_N\right) = 0, \quad \#m\mathbb{R}\left(\hat{\mathbf{D}}_N\right) = N - 2,$$
$$\#a\mathbb{R}\left(\hat{\mathbf{I}}_N\right) = N, \quad \#m\mathbb{R}\left(\hat{\mathbf{I}}_N\right) = 0,$$
$$\#a\mathbb{R}\left(\mathbf{C}_N\right) = 0, \quad \#m\mathbb{R}\left(\mathbf{C}_N\right) = N. \tag{21}$$

$\square$

Thus by using the above, we could state (20) as the first order difference equation with respect to $t \geq 2$

$$\#m\mathbb{R}(\text{VanCC}, 2^t) - 2 \cdot \#m\mathbb{R}\left(\text{VanCC}, 2^{t-1}\right) = 2 \cdot 2^t - 2.$$

Solving the above difference equation using the initial condition $\#m\mathbb{R}(\text{VanCC}, 2) = 1$, we can obtain

$$\#m\mathbb{R}(\text{VanCC}, 2^t) = 2Nt - \frac{5}{2}N + 2$$

Now by using the algorithm **vancc**($\mathbf{z}, \mathbf{N}$) and (15), we could obtain another first order difference equation with respect to $t \geq 2$

$$\#a\mathbb{R}(\text{VanCC}, 2^t) - 2 \cdot \#a\mathbb{R}\left(\text{VanCC}, 2^{t-1}\right) = 2^t.$$

Solving the above difference equation using the initial condition $\#a\mathbb{R}(\text{VanCC}, 2) = 2$, we can obtain

$$\#a\mathbb{R}(\text{VanCC}, 2^t) = Nt.$$

Corollary 17: Let $N = 2^t(\geq 2)$, r and θ be given. The real GDB counts of the proposed **vanccr**($\mathbf{z}, \mathbf{N}$) algorithm with $\mathbf{z} \in \mathbb{R}^N$ is given by

$$\#a\mathbb{R}(VanCCR, N) = Nt,$$

$$\#m\mathbb{R}(VanCCR, N) = 2Nt - \frac{3}{2}N + 1. \tag{22}$$

Proof: $\tilde{\mathbf{D}}_N$ is a diagonal matrix with real entries so the number of additions will remain the same as in (19) while the number of multiplications will be increased by $N - 1$ in (19). $\square$

Theorem 18: Let $N = 2^t(\geq 2)$ and θ be given. The real GDB counts of the proposed **vanc**($\mathbf{z}, \mathbf{N}$) algorithm with $\mathbf{z} \in \mathbb{R}^N$ is given by

$$\#a\mathbb{R}(VanC, N) = Nt,$$

$$\#m\mathbb{R}(VanC, N) = 2Nt - \frac{5}{2}N + 2. \tag{23}$$

TABLE 1. Complex GDB counts of the proposed radix-2 algorithms (i.e. vanc(z, N), vancr(z, N), vancc(z, N),and vanccr(z, N)) vs direct computation.

N	Direct Add/Multi	$\#a\mathbb{C}(VanC,N)/$ $\#a\mathbb{C}(VanCR,N)/$ $\#a\mathbb{C}(VanCC,N)/$ $\#a\mathbb{C}(VanCCR,N)$	$\#m\mathbb{C}(VanC,N)/$ $\#m\mathbb{C}(VanCC,N)$	$\#m\mathbb{C}(VanCR,N)/$ $\#m\mathbb{C}(VanCCR,N)$
4	12	8	5	6
8	56	24	17	20
16	240	64	49	56
32	992	160	129	144
64	4032	384	321	352
128	16256	896	769	832
256	65280	2048	1793	1920
512	261632	4608	4097	4352
1024	1047552	10240	9217	9728
2048	4192256	22528	20481	21504
4096	16773120	49152	45057	47104

Proof: The proof follows similar lines as that of Theorem 16 except $\check{D}_N$ instead of $\hat{\mathbf{D}}_N$ and $\bar{C}_N$ instead of $\mathbf{C}_N$. □

Corollary 19: Let $N = 2^t (\geq 2)$, r and θ be given. The real GDB counts of the proposed **vancr(z, N)** algorithm with $\mathbf{z} \in \mathbb{R}^N$ is given by

$$\#a\mathbb{R}(VanCR, N) = Nt,$$

$$\#m\mathbb{R}(VanCR, N) = 2Nt - \frac{3}{2}N + 1. \quad (24)$$

Proof: $\tilde{\mathbf{D}}_N$ is a diagonal matrix with real entries so the number of additions will remain the same as in (23) while the number of multiplications will be increased by $N - 1$ in (23). □

B. NUMERICAL RESULTS

Here we provide numerical results for the GDB counts of the proposed radix-2 algorithms **vanc(z, N)**, **vancc(z, N)**, **vancr(z, N)**, and **vanccr(z, N)**. We consider the direct computation of Vandermonde matrices $\mathbf{V}$ and $\tilde{\mathbf{V}}$ by the vector $\mathbf{z} \in \mathbb{C}^N$ with $N(N-1)$ complex additions and multiplications (note that $\mathbf{V}$ and $\tilde{\mathbf{V}}$ have 1's along the first column so we counted the multiplication count as $N(N - 1)$ as opposed to N^2). Also, the direct computation of Vandermonde matrices $\mathbf{V}$ and $\tilde{\mathbf{V}}$ by the vector $\mathbf{z} \in \mathbb{R}^N$ is taken as $N(2N - 1)$ real additions and $2N(N - 1)$ real multiplications (since $v_k = e^{-j\left(\theta + \frac{2\pi k}{N}\right)}$ we have considered on computing the powers of nodes using $v_k^l = e^{-jl\left(\theta + \frac{2\pi k}{N}\right)}$ for $l = 2, 3, \cdots, N - 1$). Note that we have not counted the multiplication by 1 in the Vandermonde matrices. The numerical results for the GDB counts of the proposed algorithms **vanc(z, N)**, **vancr(z, N)**, **vancc(z, N)**,and **vanccr(z, N)** with corresponding matrices $\mathbf{V}_N$ and $\tilde{\mathbf{V}}_N$ varying sizes from 4 × 4 to 4096 × 4096 are shown in Tables 1, 2, and 3.

Following Tables 1, 2, and 3, the proposed radix-2 algorithms for the Vandermonde matrices have shown significant arithmetic complexity reduction as opposed to the DVM algorithms presented in [1], [16], [17]. At the same time, we should recall that the DVM algorithms proposed in [1], [16], [17] have no restriction for nodes or delays as in this

TABLE 2. Real GDB counts of the proposed radix-2 algorithms (i.e. vanc(z, N) and vancc(z, N)) vs direct computation.

N	Direct Add	$\#a\mathbb{R}(VanC,N)/$ $\#a\mathbb{R}(VanCC,N)$	Direct Multi	$\#m\mathbb{R}(VanC,N)/$ $\#m\mathbb{R}(VanCC,N)$
4	28	8	24	8
8	120	24	112	30
16	496	64	480	90
32	2016	160	1984	242
64	8128	384	8064	610
128	32640	896	32512	1474
256	130816	2048	130560	3458
512	523776	4608	523264	7938
1024	2096128	10240	2095104	17922
2048	8386560	22528	8384512	39938
4096	33550336	49152	33546240	88066

TABLE 3. Real GDB counts of the proposed radix-2 algorithms (i.e. vancr(z, N) and vanccr(z, N)) vs direct computation.

N	Direct Add	$\#a\mathbb{R}(VanCR,N)/$ $\#a\mathbb{R}(VanCCR,N)$	Direct Multi	$\#m\mathbb{R}(VanCR,N)/$ $\#m\mathbb{R}(VanCCR,N)$
4	28	8	24	11
8	120	24	112	37
16	496	64	480	105
32	2016	160	1984	273
64	8128	384	8064	673
128	32640	896	32512	1601
256	130816	2048	130560	3713
512	523776	4608	523264	8449
1024	2096128	10240	2095104	18945
2048	8386560	22528	8384512	41985
4096	33550336	49152	33546240	92161

paper. Moreover, the proposed radix-2 algorithms for Vandermonde matrices have reduced GDB counts extensively opposed to the direct computation of Vandermonde matrices by a vector. More importantly, we have achieved the lowest GDB counts of radix-2 algorithms on computing Vandermonde matrices by a vector in the literature while covering radix-2 DFT algorithms as a subclass of the proposed radix-2 algorithms.

V. ERROR BOUND AND NUMERICAL STABILITY OF RADIX-2 VANDERMONDE ALGORITHMS
A. THEORETICAL ANALYSIS
Error bounds and numerical stability when computing the radix-2 Vandermonde algorithms associated with true time delays are the main concern in this section. To derive analytic

results for error bound, we will use the perturbation of the product of matrices (stated in [9]). Following the proposed radix-2 algorithms **vancc**$(\mathbf{z}, \mathbf{N})$ and **vanc**$(\mathbf{z}, \mathbf{N})$, we have to compute weights $e^{\pm k(\frac{2\pi j}{N})} = \omega_{\pm}^{k} (say)$, where $\omega_{\pm} = e^{\pm \frac{2\pi j}{N}}$ for $k = 0, 1, \ldots, \frac{N}{2} - 1$. The way we compute weights affects the accuracy of the algorithms. Thus, we will assume that the computed weights $\widehat{\omega}_{\pm}^{k}$ are used and satisfy for all $k = 0, 1, \ldots, \frac{N}{2} - 1$

$$\widehat{\omega}_{\pm}^{k} = \omega_{\pm}^{k} + \epsilon_{k_{\pm}}, \quad |\epsilon_{k_{+}}| \le \mu_{+}, |\epsilon_{k_{-}}| \le \mu_{-}, \quad (25)$$

where $\mu_{+} := c_1 u$ and $\mu_{-} := c_2 u$ u is the unit roundoff, and c_1 and c_2 are constants that depend on the method [22].

Let's recall the perturbation of the product of matrices stated in [9, Lemma 3.7] i.e. if $\mathbf{A}_k + \Delta \mathbf{A}_k \in \mathbb{R}^{N \times N}$ satisfies $|\Delta \mathbf{A}_k| \le \delta_k |\mathbf{A}_k|$ for all k, then

$$\left| \prod_{k=0}^{m} (\mathbf{A}_k + \Delta \mathbf{A}_k) - \prod_{k=0}^{m} \mathbf{A}_k \right|$$

$$\le \left(\prod_{k=0}^{m} (1 + \delta_k) - 1 \right) \prod_{k=0}^{m} |\mathbf{A}_k|$$

where $|\delta_k| < u$. Moreover, recall $\prod_{k=1}^{N} (1 + \delta_k)^{\pm 1} = 1 + \theta_N$ where $|\theta_N| \le \frac{Nu}{1 - Nu} =: \gamma_N$ and $\gamma_k + u \le \gamma_{k+1}, \gamma_k + \gamma_j + \gamma_k \gamma_j \le \gamma_{k+j}$ from [9, Lemma 3.1 and Lemma 3.3], and for $x, y \in \mathbb{C}$, $fl(x \pm y) = (x + y)(1 + \delta)$ where $|\delta| \le u, fl(xy) = (xy)(1 + \delta)$ where $|\delta| \le \sqrt{2}\gamma_2$ from [9, Lemma 3.5].

To carry out error analysis of the proposed algorithms in complex arithmetic, we implement complex arithmetic using real arithmetic operations computed according to number of additions and multiplications of non-unit numbers. Thus, we multiply $\hat{\mathbf{I}}_N$ (because it has only block identity matrices) and $\hat{\mathbf{D}}_N$, which were defined in (12), and name as $\mathbf{B}_N$ s.t.

$$\mathbf{B}_N = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{I}_{\frac{N}{2}} \\ \dot{\mathbf{D}}_{\frac{N}{2}} & -\dot{\mathbf{D}}_{\frac{N}{2}} \end{bmatrix}.$$ Similarly, we multiply $\check{\mathbf{I}}_N$ (because it

has only block identity matrices) and $\check{\mathbf{D}}_N$, which were defined in (12), and name as $\check{\mathbf{B}}_N$ s.t. $\check{\mathbf{B}}_N = \begin{bmatrix} \mathbf{I}_{\frac{N}{2}} & \mathbf{I}_{\frac{N}{2}} \\ \bar{\dot{\mathbf{D}}}_{\frac{N}{2}} & -\bar{\dot{\mathbf{D}}}_{\frac{N}{2}} \end{bmatrix}.$

Theorem 20: Let $\widehat{\mathbf{y}} = fl(\mathbf{V}_N \mathbf{z})$, where $N = 2^t (t \ge 2)$, be computed using the algorithm **vancc**$(\mathbf{z}, \mathbf{N})$, and assume that (25) holds. Then

$$\frac{\|\mathbf{y} - \widehat{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \le \frac{t v_{+}}{1 - t v_{+}} N^{\frac{1}{2}} \quad (26)$$

where $v_{+} = \eta_{+} \gamma_3 + \eta_{+} + \gamma_3$ and $\eta_{+} = \mu_{+} + \gamma_4 (1 + \mu_{+})$.

Proof: Using the algorithm **vancc**$(\mathbf{z}, \mathbf{N})$ and the computed matrices $\widehat{\mathbf{B}}(k)$ (in terms of computed weights $\widehat{\omega}_{+}^{k}$) for $k = 0, 1, \cdots, t - 2$: we have

$$\widehat{\mathbf{y}} = fl\Big(\mathbf{P}(0)\mathbf{P}(1)\cdots \mathbf{P}(t-2) \, \mathbf{V}(t-1) \, \widehat{\mathbf{B}}(t-2)\mathbf{C}(t-2)\cdots$$

$$\widehat{\mathbf{B}}(1)\mathbf{C}(1)\widehat{\mathbf{B}}(0)\mathbf{C}(0) \, \mathbf{z}\Big)$$

$$= \mathbf{P}(0)\mathbf{P}(1)\cdots \mathbf{P}(t-2) \, (\mathbf{V}(t-1) + \Delta \mathbf{V}(t-1))$$
$$(\widehat{\mathbf{B}}(t-2) + \Delta \widehat{\mathbf{B}}(t-2))(\mathbf{C}(t-2) + \Delta \mathbf{C}(t-2))\cdots$$
$$(\widehat{\mathbf{B}}(1) + \Delta \widehat{\mathbf{B}}(1))(\mathbf{C}(1) + \Delta \mathbf{C}(1))$$
$$(\widehat{\mathbf{B}}(0) + \Delta \widehat{\mathbf{B}}(0))(\mathbf{C}(0) + \Delta \mathbf{C}(0)) \, \mathbf{z}.$$

Each block diagonal matrix $\mathbf{P}(k)$ and $\widehat{\mathbf{B}}(k)$ is formed by 2^k number of $P_{\frac{N}{2^k}}^{T}$'s and $\mathbf{B}_{\frac{N}{2^k}}$'s respectively, in block diagonal positions. Using the fact that each $\mathbf{B}_{\frac{N}{2^k}}$ has only two non-zeros per row and recalling that we are using complex arithmetic, we get:

$$\left| \Delta \widehat{\mathbf{B}}(k) \right| \le \gamma_4 \left| \widehat{\mathbf{B}}(k) \right| \quad for \ k = 0, 1, \cdots, t - 2.$$

Using the fact that $\widehat{\mathbf{B}}(k)$ are computed using the computed weights $\widehat{\omega}_{+}^{k}$, we get:

$$\widehat{\mathbf{B}}(k) = \mathbf{B}(k) + \Delta \mathbf{B}(k), \quad |\Delta \mathbf{B}(k)| \le \mu_{+} |\mathbf{B}(k)|.$$

Each block diagonal matrix $\mathbf{C}(k)$ is formed by 2^k number of $\mathbf{C}_{\frac{N}{2^k}}$'s in block diagonal positions. Using the fact that each $\mathbf{C}_{\frac{N}{2^k}}$ has only one non-zeros per row and recalling that we are using complex arithmetic, we get:

$$|\Delta \mathbf{C}(k)| \le \gamma_3 \, |\mathbf{C}(k)| \quad for \ k = 0, 1, \cdots, t - 2.$$

$\mathbf{V}(t - 1)$ is a block diagonal matrix and formed by 2^{t-1} number of $\mathbf{V}_2$'s in diagonal positions. Hence

$$|\Delta \mathbf{V}(t - 1)| \le \gamma_3 \, |\mathbf{V}(t - 1)|.$$

Thus overall,

$$\widehat{\mathbf{y}} = \mathbf{P}(0)\mathbf{P}(1)\cdots \mathbf{P}(t-2)(\mathbf{V}(t-1) + \Delta \mathbf{V}(t-1))$$
$$(\mathbf{B}(t-2) + \mathbf{E}(t-2))(\mathbf{C}(t-2) + \Delta \mathbf{C}(t-2))\cdots$$
$$(\mathbf{B}(1) + \mathbf{E}(1))(\mathbf{C}(1) + \Delta \mathbf{C}(1))$$
$$(\mathbf{B}(0) + \mathbf{E}(0))(\mathbf{C}(0) + \Delta \mathbf{C}(0)) \, \mathbf{z}$$

where $|\mathbf{E}(k)| \le (\mu_{+} + \gamma_4 (1 + \mu_{+}))|\mathbf{B}(k)| = \eta_{+}|\mathbf{B}(k)|$. Hence

$$|\mathbf{y} - \widehat{\mathbf{y}}| \le [(1 + \eta_{+})^{t-1}(1 + \gamma_3)^{t} - 1]\mathbf{P}(0)\mathbf{P}(1)\cdots \mathbf{P}(t-2)$$
$$|\mathbf{V}(t-1)||\mathbf{B}(t-2)||\mathbf{C}(t-2)|\cdots |\mathbf{B}(1)||\mathbf{C}(1)|$$
$$|\mathbf{B}(0)||\mathbf{C}(0)||\mathbf{z}|.$$

Since each $\mathbf{C}(k)$ is an unitary matrix, and each $\mathbf{B}(k)$ and $\mathbf{V}(t-1)$ are unitary matrices up to scaling, we get $\|\mathbf{C}(k)\|_2 = 1$ and $\|\mathbf{B}(k)\|_2 = \|\mathbf{V}(t-1)\|_2 = \sqrt{2}$. Hence,

$$\|\mathbf{y} - \widehat{\mathbf{y}}\|_2 \le \frac{t v_{+}}{1 - t v_{+}} 2^{t} \|\mathbf{z}\|_2,$$

where $v_{+} = \eta_{+}\gamma_3 + \eta_{+} + \gamma_3$. Now following $\mathbf{V}_N \mathbf{V}_N^{H} = N \cdot \mathbf{I}_N$, we get $\|\mathbf{y}\|_2 = \sqrt{n}\|\mathbf{z}\|_2$, and hence the result. $\quad \square$

Corollary 21: Let $\widehat{\mathbf{y}} = fl(\mathbf{V}_N \mathbf{z})$, where $N = 2^t (t \ge 2)$, be computed using the algorithm **vancc**$(\mathbf{z}, \mathbf{N})$, and assume that (25) holds. Then the proposed radix-2 algorithm for Vandermonde matrices i.e. **vancc**$(\mathbf{z}, \mathbf{N})$ is numerically stable.

Proof: Theorem 20 immediately follows that the proposed radix-2 algorithm for Vandermonde matrices i.e. **vancc**$(\mathbf{z}, \mathbf{N})$ can be computed with tiny forward error

provided that the weights i.e. ω_+^k are computed stably. On the other hand, $\widehat{\mathbf{y}} = \mathbf{y} + \Delta\mathbf{y} = \mathbf{V}_N\mathbf{z} + \Delta\mathbf{y}$. Thus, we get $\widehat{\mathbf{y}} = \mathbf{V}_N(\mathbf{z}+\Delta\mathbf{z})$ and $\frac{\|\Delta\mathbf{z}\|_2}{\|\mathbf{z}\|_2} = \frac{\|\Delta\mathbf{y}\|_2}{\|\mathbf{y}\|_2}$. If we compute $\mathbf{y} = \mathbf{V}_N\mathbf{z}$ using the brute force computation, we get

$$|\mathbf{y} - \widehat{\mathbf{y}}| \leq \gamma_{N+2}|\mathbf{V}_N||\mathbf{z}|.$$

Since $\mathbf{V}_N$ is unitary w. r. t. scaling, this immediately reduces to

$$\frac{\|\mathbf{y} - \widehat{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \leq \gamma_{N+2}N^{\frac{1}{2}}. \tag{27}$$

As $\mu_+ = \mathcal{O}(u)$, the error (26) of the proposed radix-2 algorithm is much more smaller than that in (27). Thus, the proposed algorithm is backward stable. Hence, the proposed algorithm is numerically stable. □

Theorem 22: Let $\widehat{\mathbf{y}} = fl(\mathbf{V}_N\mathbf{z})$, *where* $N = 2^t(t \geq 2)$, *be computed using the algorithm* **vanc(z, N)***, and assume that* (25) *holds. Then*

$$\frac{\|\mathbf{y} - \widehat{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \leq \frac{t\nu_-}{1-t\nu_-}N^{\frac{1}{2}} \tag{28}$$

where $\nu_- = \eta_-\gamma_3 + \eta_- + \gamma_3$ *and* $\eta_- = \mu_- + \gamma_4(1 + \mu_-)$.

Proof: The proof follows similar lines as that of Theorem 20 except $\widetilde{\mathbf{B}}(k)$, $\widetilde{\mathbf{C}}(k)$, $\widehat{\omega}_-^k$, and μ_- instead of $\widehat{\mathbf{B}}(k)$, $\mathbf{C}(k)$, $\widehat{\omega}_+^k$, and μ_+, respectively. □

Corollary 23: Let $\widehat{\mathbf{y}} = fl(\mathbf{V}_N\mathbf{z})$, *where* $N = 2^t(t \geq 2)$, *be computed using the algorithm* **vanc(z, N)***, and assume that* (25) *holds. Then the proposed radix-2 algorithm for Vandermonde matrices i.e.* **vanc(z, N)** *is numerically stable.*

Proof: The proof follows similar lines as in Corollary 21. □

B. NUMERICAL RESULTS
We will now state numerical results in connection to the error bounds of the proposed radix-2 algorithms for Vandermonde matrices and compare the results with the error bound of the radix-2 FFT algorithm analyzed in [9]. With the help of the radix-2 factorization of the DFT matrices in [22], it was proved in [9] that the error bound on computing radix-2 FFT algorithm is given by;

$$\frac{\|\mathbf{y} - \widehat{\mathbf{y}}\|_2}{\|\mathbf{y}\|_2} \leq \frac{t\eta}{1-t\eta}N^{\frac{1}{2}} \tag{29}$$

where $\widehat{\mathbf{y}} = fl(\mathbf{F}_N\mathbf{x})$, $\mathbf{F}_N$ is the DFT matrix, $N = 2^t$, $\eta = \mu + \gamma_4(1+\mu)$, and μ depends on the methods for computing the weights as specified in [22]. We compare the error bounds of the proposed radix-2 algorithms for Vandermonde matrices shown in (26) and (28) with the radix-2 FFT algorithm (29) using MATLAB(R2014a version). In these calculations, we have chosen $\mu = \mu_+ = \mu_- = 10^{-15}$ and $\gamma_N = \frac{Nu}{1-Nu}$ where $N = 2^t$ and u is the machine precision. Since $\mu = \mathcal{O}(u)$, we have chosen $u = 10^{-15}$. Table 4 shows the error bounds of the proposed radix-2 algorithms for Vandermonde matrices and radix-2 FFT algorithm in [9].

Based on the numerical results shown in Table 4, the proposed radix-2 algorithms for Vandermode matrices and

TABLE 4. Error bounds of the proposed radix-2 algorithms (i.e. vancc(z, N) and vanc(z, N)) vs radix-2 FFT algorithm [9].

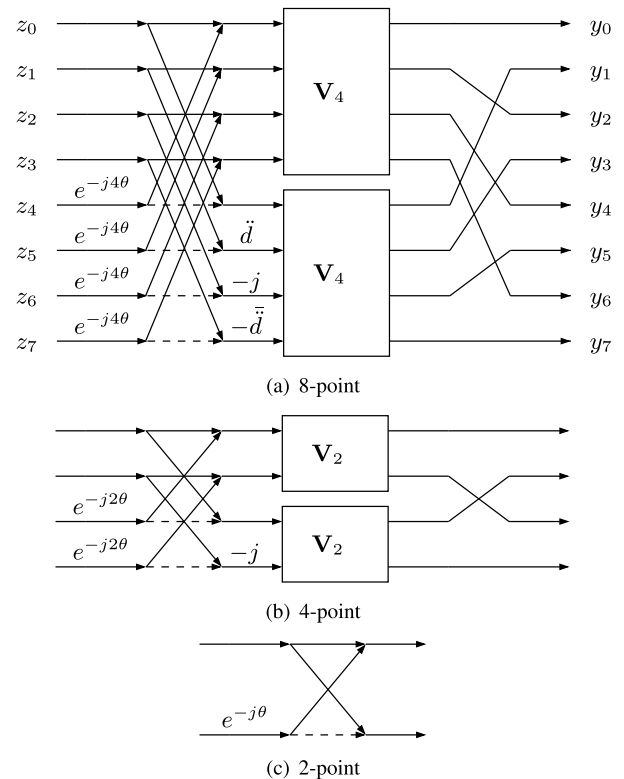| N	Error Bound **vancc(z, N)/vanc(z, N)**	Error Bound FFT
4	3.2×10^{-14}	2×10^{-14}
8	6.8×10^{-14}	4.2×10^{-14}
16	1.3×10^{-13}	8×10^{-14}
32	2.3×10^{-13}	1.4×10^{-13}
64	3.8×10^{-13}	2.4×10^{-13}
128	6.3×10^{-13}	4×10^{-13}
256	1×10^{-12}	6.4×10^{-13}
512	1.6×10^{-12}	1×10^{-12}
1024	2.6×10^{-12}	1.6×10^{-12}
2048	4×10^{-12}	2.5×10^{-12}
4096	6.1×10^{-12}	3.8×10^{-12}

(a) 8-point

(b) 4-point

(c) 2-point

FIGURE 1. Signal flow graph of the 2-, 4-, and 8-point vanc decompositions, where $\ddot{d} = \frac{\sqrt{2}}{2}(1-j)$ and dashed arrows represent multiplication by -1.

radix-2 FFT algorithm have the same error orders except for $N = 16$, and 256. Even with these two N values, error orders of the proposed algorithms and FFT vary only by 10^{-1} and relatively very low. To sum up, Table 4 shows that the proposed radix-2 algorithms for Vandermonde matrices provide tiny forward errors.

VI. SIGNAL FLOW GRAPHS FOR RADIX-2 VANDERMONDE ALGORITHMS
In this section, we use signal flow graphs to illustrate the connection between algebraic operations used in sparse and orthogonal factorization of Vandermonde matrices with

the fundamental signal flow graphs (SFG) building blocks (i.e. adders and multipliers). We provide two signal flow graphs to show the simplicity of the proposed radix-2 algorithms for Vandermonde matrices. Being pivotal for efficient physical implementation in hardware, SFGs should represent a numerical algorithm in its fully factorized form in such a way that more sparse matrices are resulted and, as a consequence, less arithmetic operations demanded. Thus, Fig. 1 displays the SFG for the proposed **vanc**(**z**, **N**) algorithm for the case $N = 8$. The recursive nature is evident as we express the 8-point SFG in terms of the 4- and 2-point SFGs. Notice that, the SFG of the **vancc**(**z**, **N**) algorithm is not presented because the delays have been replaced with time advances that are not realizable in real-time circuits. But for the software implementation purposes, we have proposed **vancc**(**z**, **N**) algorithm in Section III-B to effectively compute Vandermonde matrices.

VII. CONCLUSION

We have proposed novel self-recursive radix-2 algorithms for Vandermonde matrices. These algorithms have sparse and orthogonal factors. We have shown that the well known radix-2 DFT algorithm is a subclass of the proposed algorithms for the Vandermonde matrices. The proposed algorithms attain the lowest gain-delay-block counts for Vandermonde matrices by a vector, in the literature. Theoretical error bounds on computing the radix-2 algorithms and stability of the proposed algorithms are established. Numerical results of the forward error bounds of the proposed radix-2 algorithms are compared with the radix-2 FFT algorithm. The proposed radix-2 algorithms have shown tiny forward and backward errors when weights are computed stably. Signal flow graphs were presented to show the simplicity of the proposed algorithm and to realize high-frequency analog circuits. Using the radix-2 algorithms for Vandermonde matrices associated with true time delay based delay-sum filterbanks, we have reduced the circuit complexity of multi-beam analog beamforming systems significantly.

REFERENCES

[1] 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. Moratuwa Eng. Res. Conf. (MERCon)*, May 2017, pp. 236–241.

[2] J. F. Canny, E. Kaltofen, and L. Yagati, "Solving systems of non-linear equations faster," in *Proc. ACM-SIGSAM Internat. Symp. Symbolic Algebr. Comput.*, New York, NY, USA, 1989, pp. 34–42.

[3] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, vol. 19, no. 90, pp. 297–301, May 1965.

[4] J. R. Driscoll, D. M. Healy, and D. N. Rockmore, "Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs," *SIAM J. Comput.*, vol. 26, no. 4, pp. 1066–1099, Aug. 1997.

[5] W. Gautschi and G. Inglese, "Lower bounds for the condition number of Vandermonde matrices," *Numerische Mathematik*, vol. 52, pp. 241–250, May 1988.

[6] W. Gautschi, "How unstable are Vandermonde systems?" in *Asymptotic and Computational Analysis: Conference in Honor of Frank W. J. Olver's 65th Birthday* (Lecture Notes in Pure and Applied Mathematics), vol. 124, R. Wong, Ed. New York, NY, USA: Marcel Dekker, 1990, pp. 193–210.

[7] I. Gohberg and V. Olshevsky, "Complexity of multiplication with vectors for structured matrices," *Linear Algebra Appl.*, vol. 202, pp. 163–192, Apr. 1994.

[8] I. Gohberg and V. Olshevsky, "Fast algorithms with preprocessing for matrix-vector multiplication problems," *J. Complex.*, vol. 10, no. 4, pp. 411–427, Dec. 1994.

[9] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. Philadelphia, PA, USA: SIAM, 1996.

[10] 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.

[11] T. Kailath and V. Olshevsky, "Displacement-structure approach to polynomial Vandermonde and related matrices," *Linear Algebra Appl.*, vol. 261, nos. 1–3, pp. 49–90, Aug. 1997.

[12] H. Oruç and H. K. Akmaz, "Symmetric functions and the Vandermonde matrix," *J. Comput. Appl. Math.*, vol. 172, no. 1, pp. 49–64, Nov. 2004.

[13] H. Oruç and G. M. Phillips, "Explicit factorization of the Vandermonde matrix," *Linear Algebra Appl.*, vol. 315, nos. 1–3, pp. 113–123, Aug. 2000.

[14] V. Y. Pan, "Fast approximate computations with Cauchy matrices and polynomials," *Math. Comp.*, vol. 86, no. 308, pp. 2799–2826, Apr. 2017.

[15] V. Y. Pan, "How bad are Vandermonde matrices?" *SIAM J. Matrix Anal. Appl.*, vol. 37, no. 2, pp. 676–694, Jan. 2016.

[16] S. M. Perera, A. Madanayake, and R. J. Cintra, "Efficient and self-recursive delay vandermonde algorithm for multi-beam antenna arrays," *Linear Algebra Appl.*, to be published.

[17] 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 using low-complexity algorithms and mixed-signal integrated circuits," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 2, pp. 368–382, May 2018.

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

[19] G. Strang, "Wavelets," *Amer. Scientist*, vol. 82, no. 3, pp. 250–255, 1994.

[20] G. Strang, *Introduction to Applied Mathematics*. Wellesley, MA, USA: Wesley-Cambridge Press, 1986.

[21] E. E. Tyrtyshnikov, "How bad are Hankel matrices?" *Numerische Mathematik*, vol. 67, no. 2, pp. 261–269, 1994.

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

[23] S.-L. Yang, "On the LU factorization of the Vandermonde matrix," *Discrete Appl. Math.*, vol. 146, no. 1, pp. 102–105, Feb. 2005.

[24] R. Yavne, "An economical method for calculating the discrete Fourier transform," in *Proc. AFIPS (Fall, part I)*, vol. 33, 1968, pp. 115–125.

SIRANI M. PERERA received the B.Sc. (Hons.) degree in mathematics from the University of Sri Jayewardenepura, Sri Lanka, in 2004, the Part III Mathematical Tripos (Hons.) degree 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 was the first Sri Lankan to attend the Center for Mathematical Sciences at the University of Cambridge, U.K., through the full Shell Centenary Scholarship (under the Cambridge Commonwealth Trust). Since 2015, she has been an Assistant Professor in mathematics with Embry–Riddle Aeronautical University (ERAU), USA. She is working in the field of numerical linear algebra. Her research interests include scientific computing, structured matrices, including Vandermonde, quasiseparable, semiseparable, Bezoutian, and banded, matrices with displacement structure, DCT, DST, FFT, fast and stable algorithms, complexity and performance of algorithms, signal processing, and image processing. She is a member of the SIAM.

ARJUNA MADANAYAKE (Member, IEEE) received the B.Sc. degree in electronics and telecommunication engineering from the University of Moratuwa, Sri Lanka, and the M.S. and Ph.D. degrees from the University of Calgary, Canada, both in electrical engineering. He is an Associate Professor of electrical and computer engineering with Florida International University (FIU), Miami, FL, USA. His research interests include signal processing, circuits, electronics, and antenna array processing.

RENATO J. CINTRA (Senior Member, IEEE) received the D.Sc. degree in electrical engineering from the Universidade Federal de Pernambuco (UFPE), Recife, Brazil, in 2005. He joined the Centre for Natural and Exact Sciences, UFPE, in 2005. He was a Visiting Professor with the Département Informatique, INSA, Lyon, France, and the University of Calgary, Canada. His long-term topics of research include approximation theory for discrete transforms, theory and methods for digital signal/image processing, and probabilistic methods. He is also a member of the SIAM. He was an Associate Editor of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS and *Circuits, Systems, and Signal Processing* (Springer). He currently serves as Associate Editor for *IET Circuits, Devices and Systems* and *the Journal of Communication and Information Systems*, and a Subject Editor of *Electronics Letters*.

· · ·