Multiplierless Approximation of Transforms With Adder Constraint

Ying-Jui Chen, *Student Member, IEEE*, Soontorn Oraintara, *Member, IEEE*, Trac D. Tran, *Member, IEEE*, Kevin Amaratunga, and Truong Q. Nguyen, *Senior Member, IEEE*

***Abstract*—This letter describes an algorithm for systematically finding a multiplierless approximation of transforms by replacing floating-point multipliers with VLSI-friendly binary coefficients of the form $k/2^n$. Assuming the cost of hardware binary shifters is negligible, the total number of binary adders employed to approximate the transform can be regarded as an index of complexity. Because the new algorithm is more systematic and faster than trial-and-error binary approximations with adder constraint, it is a much more efficient design tool. Furthermore, the algorithm is not limited to a specific transform; various approximations of the discrete cosine transform are presented as examples of its versatility.**

***Index Terms*—Binary coefficients, BinDCT, IntDCT, integer DCT, lifting scheme, matrix approximation, multiplierless approximation.**

I. Introduction

RECENTLY, there has been increasing interest in approximating a given floating-point transform using only very large scale integration-friendly binary, multiplierless coefficients of the form $k/2^n$ [1]–[8]. Because only binary coefficients are needed, the resulting transform approximation is multiplierless, and the overall complexity of hardware implementation can be measured in terms of the total number of adders and/or shifters required in the implementation.

Usually, a higher complexity can achieve a higher accuracy. Since the cost of a hardware bit shifter is negligible as compared with that of an adder, the overall complexity can safely be measured by the total number of adders only. Thus, given a total number of adders as the design constraint, it is desirable to come up with a good adder allocation among the various multipliers so that the highest possible accuracy can be achieved. However, little attention has been paid to this issue.

We propose a new quasi-coordinate-descent-based algorithm for systematically finding the multiplierless approximation of a given transform. Specifically, we will use the discrete cosine transform (DCT) [9] as an example to illustrate how the algorithm works. Extending this to other transforms is straightforward. Furthermore, the proposed algorithm will be applied on an efficient, sparse representation of the given transform, so as to minimize the number of floating-point multipliers. In particular, the lifting factorization [10] will be the efficient representation of choice. This permits perfect reconstruction or reversibility. Strang [11] details how to get such a lifting-like factorization for an $N \times N$ nonsingular constant matrix.

The letter is organized as follows. Section II discusses the minimum-adder representation of an integer and the corresponding reducibility issue in terms of adders. These properties are used in Section III to derive the proposed algorithm for finding the multiplierless approximation of a transform with adder constraint. Two DCT approximation examples are presented in Section IV. Section V concludes the letter.

Manuscript received August 15, 2001; revised July 8, 2002. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Elias S. Manolakos.

Y.-J. Chen and K. Amaratunga are with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

S. Oraintara is with the Department of Electrical and Computer Engineering, University of Texas, Arlington, TX 76019-0016 USA (e-mail: oraintar@uta.edu).

T. D. Tran is with the Electrical and Computer Engineering Department, The Johns Hopkins University, Baltimore, MD 21218-2608 USA.

T. Q. Nguyen is with the Electrical and Computer Engineering Department, University of California, San Diego (UCSD), La Jolla, CA 92093 USA.

Digital Object Identifier 10.1109/LSP.2002.804419

II. Minimum-Adder Multiplications

A. Integers

An integer multiplication is equivalent to bit-shifting the multiplicand to the left by different numbers of bits and summing up these bit-shifted versions. The total number of shifts and adds required can be counted from the binary representation of the integer multiplier. For example, multiplication by $5 = (101)_2$ can be implemented by one adder and one shift. Similarly, multiplication by $7 = (111)_2$ can be done using two adders and two shifts. However, this is not the minimum number of adders needed to multiply a number by 7, because if we express

$$7 = 8 - 1 = (1000)_2 - (1)_2$$

it is immediately clear that only one adder and one shift are required. In essence, this involves the following signed digit representation of numbers [12]–[14].

To begin with, let us first introduce the concept of multiplicative irreducibility in terms of adders.

Definition 1—Multiplicative Irreducibility: A positive integer multiplier X is said to be multiplicatively irreducible in terms of adders if the minimum number of adders required to implement its multiplication is equal to $N_X - 1$ where N_X is the number of 1s in the binary representation of X.

As a consequence, the following condition on the binary patterns results.

Fact 1: A positive integer X is multiplicatively irreducible if and only if its binary representation contains not more than two consecutive 1s and if any pairs of two consecutive 1s are separated by at least two 0s.

The multiplicative irreducibility is important in determining the minimum-adder representation of an integer multiplier, as follows.

Fact 2—Minimum-Adder Representation: An integer X can be decomposed into the form $X = A - B$ where $A, B \in \mathbb{N}$ are multiplicatively irreducible containing N_A and N_B binary 1s, respectively. Furthermore, the minimum number of adders required to implement the multiplication by X is $N_A + N_B - 1$.

Definition 2—Irreducible Form: Given an integer X, the above minimum-adder representation is said to be the irreducible form of X.

Now, to find out the minimum-adder representation of a given integer X, the idea is to look for both

1) n consecutive 1s ($n \geq 3$) in the binary representation;
2) two groups of n consecutive 1s ($n \geq 2$) separated by only one 0.

For instance, the following number has a group of three consecutive 1s and two groups of four consecutive 1s:

$$\begin{aligned} X &= \overbrace{1111}^{4}00\overbrace{111}^{3}00\overbrace{1111}^{4} \quad \Longrightarrow 10 \text{ adders, } 10 \text{ shifts} \\ &= \left.\begin{array}{r} 10\,000\,01\,000\,01\,0000 \\ -1\,00\,001\,00\,0001 \end{array}\right\} \Longrightarrow 5 \text{ adders, } 5 \text{ shifts} \end{aligned}.$$

Here is another example:

$$\begin{aligned} X &= \overbrace{1111}^{4}0\overbrace{111}^{3} \quad \Longrightarrow 6 \text{ adders, } 6 \text{ shifts} \\ &= \left.\begin{array}{r} 1111\,1\,000 \\ -1 \end{array}\right\} \Longrightarrow 5 \text{ adders, } 5 \text{ shifts} \\ &= \left.\begin{array}{r} 10\,000\,0\,000 \\ -1\,001 \end{array}\right\} \Longrightarrow 2 \text{ adders, } 2 \text{ shifts} \end{aligned}.$$

B. Binary Fractions

A binary fractional multiplier of the form $\pm k/2^b$, $k, b \in \mathbb{N}$, k odd, can also be implemented using only integer arithmetic. The multiplicand is first multiplied by $\pm k$, and the result is right-shifted by b bits. Therefore, in our setup, the minimum number of adders required for implementing a given binary fraction is equal to that for implementing its numerator.

III. Adder-Constrained Multiplierless Approximation Algorithm

Let T denote the given transform of interest whose sparse matrix factorization consists of M (usually floating-point) multipliers λ_i, $i = 1, 2, \ldots, M$. Equivalently, T can be thought of as being parameterized by these λ_i

$$T = T(\lambda_1, \lambda_2, \ldots, \lambda_M). \tag{1}$$

Usually, such sparse factorizations of T are not unique, and one is preferred whenever all the λ_i satisfy $|\lambda_i| \leq 1$. These multipliers are usually floating-point numbers. Let $[\lambda]_p$ denote the best achievable binary (fractional) approximation of λ with only p adders. Specifically, if

$$[\lambda_i]_{n_i} = \pm\frac{k_i}{2^{b_i}}, \qquad i = 1, 2, \ldots, M$$

where $b_i \in \mathbb{N}$ and k_i is odd, then the irreducible form of k_i contains a total of $(n_i + 1)$ binary 1s. Call $[\lambda]_n$ the n-adder binary approximation (or n-ABA) of λ.

A. Finding the n-ABA

To compute the n-ABA, $[\lambda]_n$, of some floating-point number λ, its binary representation $k_b/2^b, k_b \in \mathbb{Z},\ b \in \mathbb{N}$, is first calculated with a sufficient precision, namely, with b large enough, depending on the dynamic range of n. Then based on the irreducible form of $|k_b|$, $|k_b| = A_{k_b} - B_{k_b}$, a positive integer m is determined such that the total number of 1s in the binary representations of integers $\lfloor A_{k_b}/2^m \rfloor$ and $\lfloor B_{k_b}/2^m \rfloor$ is equal to $n + 1$. Then the n-ABA of λ is given by

$$[\lambda]_n = \text{sgn}(\lambda)\left(\left\lfloor \frac{A_{k_b}}{2^m} \right\rfloor - \left\lfloor \frac{B_{k_b}}{2^m} \right\rfloor\right) 2^{m-b}. \tag{2}$$

B. Quasi-Coordinate Descent

In (1), if all the λ_i are replaced by the respective n_i-ABAs, $[\lambda_i]_{n_i}$, the resulting transform

$$\hat{T} = T\left([\lambda_1]_{n_1}, [\lambda_2]_{n_2}, \ldots, [\lambda_M]_{n_M}\right)$$

becomes a multiplierless approximation of the original T. In this case, the minimum number of adders required to implement $\hat{T}$, $N_{\hat{T}}$, is given by

$$N_{\hat{T}} = N_0 + \sum_{i=1}^{M} n_i$$

where N_0 is the number of "basic" adders associated with the particular sparse matrix factorization structure used to parameterize T. In other words, N_0 is the number of the adders that remain when all the λ_i in T are set to zero, which corresponds to configurations C9 in Tables I and II. The significance of N_0 should become clear in Section IV where sparse factorizations of the DCT are presented.

Now, the goal is to find a good adder allocation, subject to the given signal statistics and a given value of $N_{\hat{T}}$. Two common performance measures defining a good adder allocation are 1) the transform coding gain of $\hat{T}$ and 2) the mean-square error (MSE) between the outputs of T and $\hat{T}$.

Let Φ denote the performance measure, defined over the same parameter space as $\hat{T}$. Then, based on the chosen Φ, the proposed algorithm to find the optimal adder allocation (assuming $N_{\hat{T}} \geq N_0$) is given as follows:

```
Initialize all n_i = 0
for j = 1, 2, ..., (N_T̂ − N_0)
  for i = 1, 2, ..., M
    ρ_i = Φ([λ_1]_{n_1}, [λ_2]_{n_2}, ..., [λ_i]_{n_i+1}, ..., [λ_M]_{n_M})
  end
  i* = arg min_{i=1}^{M} {ρ_i}  or  i* = arg max_{i=1}^{M} {ρ_i}
  n_{i*} := n_{i*} + 1
end
```

The algorithm begins with a given value of $N_{\hat{T}}$, which serves as the adder constraint, with all the multipliers initialized to zero ($n_i = 0$). Then, in each iteration (indexed by j), the most "effective" multiplier (λ_{i^*}) is identified and is assigned one more adder ($n_{i^*} := n_{i^*} + 1$) to increase its accuracy. This is repeated until all the $N_{\hat{T}}$ adders are exhausted.

Upon termination of the algorithm, the final n_i represent the desired adder allocation. Note that the proposed algorithm com-

TABLE I
PROPOSED MULTIPLIERLESS APPROXIMATIONS OF THE DCT BASED ON THE IntDCT STRUCTURE, WITH VARIOUS ADDER CONSTRAINTS. C1 HAS BETTER MSE PERFORMANCE THAN WAS PREVIOUSLY PUBLISHED AT THE SAME ADDER CONSTRAINT. C2–C8 ARE OUR NEW DESIGNS

	Floating-point	IntDCT-C1	C2	C3	C4	C5	C6	C7	C8	C9
p_1	0.1989123674	1/8	1/8	1/8	0	0	0	0	0	0
u_1	−0.3826834324	−3/8	−1/4	−1/4	−1/4	−1/4	−1/4	0	0	0
p_2	0.1989123674	1/8	0	0	0	0	0	0	0	0
p_3	−0.6681786379	−5/8	−5/8	−1/2	−1/2	−1/2	−1/2	−1/2	−1/2	0
u_2	0.9238795325	1	1	1	1	1	1	1	0	0
p_4	−0.6681786379	−5/8	−5/8	−5/8	−5/8	−1/2	−1/2	−1/2	−1/2	0
p_5	−0.6681786379	−1/2	−1/2	−1/2	−1/2	−1/2	−1/2	−1/2	0	0
u_3	0.9238795325	1	1	1	1	1	1	0	0	0
p_6	−0.6681786379	−5/8	−1/2	−1/2	−1/2	−1/2	−1/2	0	0	0
p_7	−0.8206787908	−13/16	−13/16	−3/4	−3/4	−3/4	−3/4	−3/4	−1/2	0
u_4	0.9807852804	1	1	1	1	1	1	1	0	0
p_8	−0.8206787908	−1/2	−1/2	−1/2	−1/2	−1/2	−1/2	−1/2	−1/2	0
p_9	−0.3033466836	−1/4	−1/4	−1/4	−1/4	0	0	0	0	0
u_5	0.5555702330	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0	0
p_{10}	−0.3033466836	−1/4	−1/4	−1/4	−1/4	−1/4	0	0	0	0
Shifts	-	9	6	4	3	2	2	2	0	0
Adds	-	45	42	40	39	37	36	33	28	24
MSE	-	$8.6e{-4}$	$1.7e{-3}$	$2.5e{-3}$	$3.0e{-3}$	$6.0e{-3}$	$8.5e{-3}$	$1.9e{-2}$	$8.4e{-2}$	$1.6e{-1}$
C_g (dB)	-	8.7352	8.6800	8.5863	8.5708	8.3338	8.0205	7.8286	6.9356	7.9461

TABLE II
PROPOSED MULTIPLIERLESS APPROXIMATIONS OF THE DCT BASED ON THE BinDCT STRUCTURE, WITH VARIOUS ADDER CONSTRAINTS. THE MSE PERFORMANCE (C2–C8) IS BETTER THAN WAS PREVIOUSLY PUBLISHED AT THE SAME ADDER CONSTRAINTS, EXCEPT FOR C1

	Floating-point	BinDCT-C1	C2	C3	C4	C5	C6	C7	C8	C9
p_1	0.4142135623	13/32	13/32	13/32	3/8	3/8	3/8	1/4	1/4	0
u_1	0.3535533905	11/32	5/16	5/16	5/16	5/16	1/4	1/4	0	0
p_2	0.6681786379	21/32	21/32	21/32	5/8	5/8	5/8	1/2	1/2	0
u_2	0.4619397662	15/32	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0
p_3	0.1989123673	3/16	3/16	3/16	3/16	3/16	3/16	1/8	0	0
u_3	0.1913417161	3/16	3/16	1/8	1/8	1/8	1/8	0	0	0
p_4	0.4142135623	13/32	13/32	13/32	13/32	3/8	3/8	3/8	1/4	0
u_4	0.7071067811	11/16	11/16	11/16	11/16	11/16	5/8	1/2	1/2	0
p_5	0.4142135623	13/32	13/32	13/32	13/32	13/32	3/8	3/8	1/4	0
Shifts	-	25	23	22	20	19	16	11	7	1
Adds	-	42	40	39	37	36	33	28	24	18
MSE	-	$1.3e{-5}$	$3.0e{-5}$	$4.0e{-5}$	$8.4e{-5}$	$1.2e{-4}$	$3.3e{-4}$	$1.4e{-3}$	$4.2e{-3}$	$2.9e{-2}$
C_g (dB)	-	8.8244	8.8201	8.8189	8.8148	8.8127	8.7947	8.7009	8.5564	7.9204

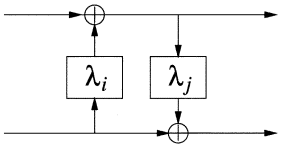

Fig. 1. The lifting structure, used exclusively in the design examples of the proposed algorithm, consists of several one-wing butterflies with multipliers λ. The adder $\oplus$ will effectively vanish if the corresponding λ_i or λ_j is zero.

pletes in a finite number of steps equal to the number of the excess adders $(N_{\hat{T}} - N_0)$. Also, the choice in Step 6 depends on the performance measure Φ. For example, we wish to minimize the MSE, while maximizing the coding gain. In essence, given one more adder to the intermediate system at stage j, the deepest coordinate-descent direction is found, and the added adder is allocated to the corresponding multiplier.

C. Adaptation to Lifting Structures

The lifting structure [10] (as shown in Fig. 1) will be used exclusively in this letter for the factorization of the DCT kernel. Now, all the λ_i in (1) are the lifting multipliers, and it is immediately clear that if some λ_i is zero, the corresponding lifting step and, hence, the associated adder will vanish.

To take this into account, the proposed algorithm is modified by initializing all the n_i to -1 and defining $[\lambda_i]_{-1} \equiv 0$. With these modifications, the lifting-adapted algorithm is as follows:

1: Initialize all $n_i = -1$
2: for $j = -(M-1), \ldots, -1, 0, 1, 2, \ldots, (N_{\hat{T}} - N_0)$
3: for $i = 1, 2, \ldots, M$
4: $\rho_i = \Phi([\lambda_1]_{n_1}, [\lambda_2]_{n_2}, \ldots, [\lambda_i]_{n_i+1}, \ldots, [\lambda_M]_{n_M})$
5: end
6: $i^* = \arg\min_{i=1}^{M}\{\rho_i\}$ or $i^* = \arg\max_{i=1}^{M}\{\rho_i\}$
7: $n_{i^*} := n_{i^*} + 1$
8: end

IV. DESIGN EXAMPLES

In this section, DCT is selected to demonstrate how the proposed algorithm works. A lifting-like factorization of the DCT is considered. Throughout this section, it is assumed that the input signal to the DCT is an AR(1) process with $\rho = 0.95$ and unit variance.

A. IntDCT

In [4], a Walsh–Hadamard-based factorization of the DCT was used, and each of the resulting rotation angles was lifted. Fig. 2 illustrates how the DCT kernel is factored in this case: there are $M = 15$ multipliers, p_i and u_j, for $1 \le i \le 10$ and $1 \le j \le 5$. The minimum number of adders is $N_0 = 24$, which is the number of the remaining adders when all the p_i and u_j are set to zero. Table I shows the results of the IntDCT-based multiplierless approximations of the DCT with various adder con-

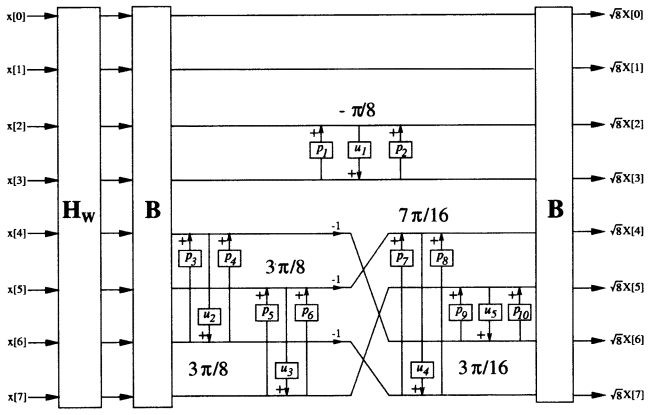

Fig. 2. The IntDCT parameterized by p_i and u_j. $\mathbf{H_w}$ is the Walsh–Hadamard transform, and $\mathbf{B}$ is bit-reversal. $N_0 = 24$ is the number of the adders that remain when all the p_i and u_j are set to zero. Actually, these are the adders internal to $\mathbf{H_w}$.

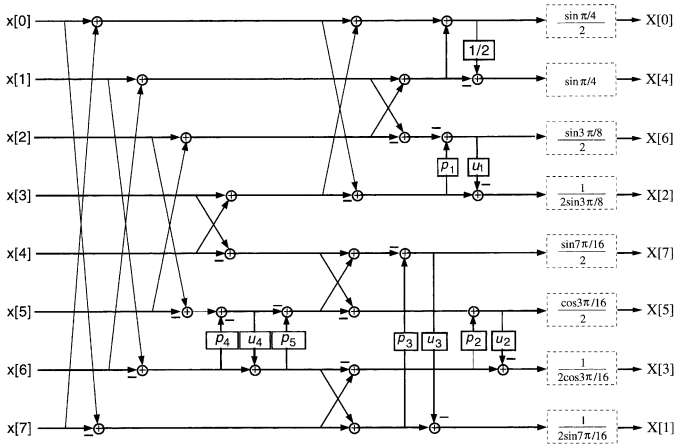

Fig. 3. The BinDCT parameterized by p_i and u_j. $N_0 = 18$ is the number of the adders that remain when all the p_i and u_j are set to zero.

straints. The MSE was chosen as the performance measure. As a comparison, the configuration reported in [4] requires 45 adds and 18 shifts with MSE $= 1.2e-3$, while the MSE of configuration C1 in Table I is only $8.6e-4$. C2–C8 are our new designs. Note that configuration C9 is nothing but the Walsh–Hadamard transform. The IntDCT has uniform scaling of each subband, which is helpful in applications such as embedded coding, because no coefficient realignment is required.

B. BinDCT

Fig. 3 shows the structure of the BinDCT proposed in [5] and [6], with $N_0 = 18$. The proposed algorithm is also applied to this factorization structure of the DCT, and these BinDCT-based multiplierless approximations are shown in Table II with various adder constraints. The MSE was used as the performance measure. Our algorithm results in a better MSE perormance (C2–C8) than was reported in [6, Table II] at the same adder constraints (with the exception of configuration C1), which confirms the effectiveness of our algorithm. To further examine its quality, an exhaustive search over all possible n-ABAs of the lifting multipliers of the BinDCT structure has been conducted, and the results coincide with the proposed multiplierless approximations C1–C9, except for C6. Examing the difference $\mathbf{e}$ between the floating-point lifting multipliers and their n-ABAs for C6 reveals that the proposed algorithm minimizes $||\mathbf{e}||$. The same is true for BinDCT-C1.

In both examples, the MSE decreases monotonically with the increased number of adders. Our new designs in Tables I and II yield lower MSE as compared with previous solutions presented in [4] and [6].

V. Conclusion

A quasi-coordinate-descent algorithm has been presented for systematically finding, with adder constraint, a multiplierless approximation of transforms. Based on a particular sparse matrix factorization used, the given transform is parameterized by a few (floating-point) multipliers in terms of which a performance measure is formed, and the proposed algorithm finds the binary approximations of the (floating-point) multipliers using only a finite number of evaluations and comparisons of the performance measure, and therefore good approximations are readily available even in the case where exhaustive search becomes intractible. When necessary, the resulting binary approximations may serve as the initial conditions for other more sophisticated approximation algorithms. Because the new algorithm is more systematic and faster than trial-and-error adder-constrained binary approximations, it manifests itself as a more efficient design tool. Furthermore, the algorithm is not limited to a specific transform; various multiplierless approximations of the DCT have been presented to demonstrate its versatility.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive suggestions.

References

[1] F. A. M. L. Bruekers and A. W. M. van den Enden, "New networks for perfect inversion and perfect reconstruction," *IEEE J. Sel. Areas Commun.*, vol. 10, Jan. 1992.
[2] S. C. Chan, W. Liu, and K. L. Ho, "Multiplierless perfect reconstruction modulated filter banks with sum-of-powers-of-two coefficients," *IEEE Signal Processing Lett.*, vol. 8, pp. 163–166, June 2001.
[3] M. D. Adam and F. Kossentni, "Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and analysis," *IEEE Trans. Image Processing*, vol. 9, pp. 1010–1024, June 2000.
[4] Y.-J. Chen, S. Oraintara, and T. Q. Nguyen, "Video compression using integer DCT," in *Proc. ICIP*, Sept. 2000.
[5] T. D. Tran, "The BinDCT: Fast multiplierless approximation of the DCT," *IEEE Signal Processing Lett.*, vol. 7, pp. 141–145, June 2000.
[6] J. Liang and T. D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme," *IEEE Trans. Signal Processing*, Dec. 2001.
[7] S. Oraintara, Y.-J. Chen, and T. Q. Nguyen, "Integer fast Fourier transform," *IEEE Trans. Signal Processing*, pp. 607–618, Mar. 2002.
[8] Y. Zheng, G. Bi, and Z. Lin, "Integer sinusoidal transforms based on lifting factorization," in *Proc. ICASSP*, Salt Lake City, UT, May 2001.
[9] K. R. Rao and P. Yip, *Discrete Cosine Transform, Algorithms, Advantages, Applications*. New York: Academic, 1990.
[10] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 247–269, 1998.
[11] G. Strang, "Every unit matrix is a *LULU*," *Linear Algebra Appl.*, vol. 265, pp. 165–172, 1997.
[12] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. 10, pp. 389–400, 1961.
[13] G. W. Reitwiesner, "Binary arithmetic," *Adv. Comput.*, vol. 1, pp. 232–308, 1960.
[14] H. L. Garner, "Number systems and arithmetic," *Adv. Comput.*, vol. 6, pp. 131–194, 1965.