INTEGER FAST FOURIER TRANSFORM (INTFFT)

Soontorn Oraintara, Ying-Jui Chen, Truong Nguyen*

*University of Texas at Arlington, EE Dept., Arlington, TX, Email: oraintar@uta.edu
Boston University, ECE Dept., Boston MA, Emails: nguyent, yrchen@bu.edu

ABSTRACT

In this paper, a concept of integer fast Fourier transform (IntFFT) for approximating the discrete Fourier transform is introduced. Unlike the fixed-point fast Fourier transform (FxpFFT), the new transform has properties that it is an integer-to-integer mapping, power adaptable and also reversible. Lifting scheme is used to approximate complex multiplications appearing in the FFT lattice structures. Split-radix FFT is used to illustrate the approach for the case of 2^N-point FFT. The transform can be implemented by using only bit shifts and additions but no multiplication. While preserving the reversibility, the IntFFT is shown experimentally to yield the same accuracy as the FxpFFT when their coefficients are quantized to a certain number of bits. Complexity of the IntFFT is shown to be much lower than that of the FxpFFT in terms of the numbers of additions and shifts.

1. INTRODUCTION

Discrete Fourier transform (DFT) is one of the most fundamental operations in digital signal processing. Because of the efficiency of the convolutional property, the DFT is often used in linear filtering found in many applications such as quantum mechanics [1], noise reduction [2] and image reconstruction [3]. However, the computational requirements for completing the DFT of a finite length signal are relatively intensive. In particular, if the input signal has length N, directly calculating its DFT requires N^2 complex multiplications ($4N^2$ real multiplications) and some additional additions. In 1965, Cooley and Tukey introduced the fast Fourier transform (FFT) which efficiently and significantly reduces the computational cost of calculating N-point DFT from $O(N^2)$ to $O(Nlog_2N)$ [4]. Since then, there have been numerous further developments that extended Cooley and Tukey's original contribution. Many efficient structures for computing DFT have been discovered by taking advantage of the symmetry and periodicity properties of the roots of unity $e^{j2\pi k/N}$ such as the radix-2 FFT, radix-4 FFT and split-radix FFT [6]. In this paper, since we mainly focus on the fast structures of the DFT, the terms DFT and FFT will be used interchangeably.

The order of the multiplicative complexity is commonly used to measure and compare the efficiency of the algorithms since multiplications are intrinsically more complicated among all operations [5]. It is well-known in the field of VLSI that among the digital arithmetic operations (addition, multiplication, shifting and addressing, etc.), multiplication is the operation which consumes most of the time and power required for the entire computation, and therefore causes the resulting devices to be large and expensive. Therefore reducing the number of multiplications in digital chip design is usually a desirable task. In this paper, utilizing the existing efficient structures, a novel structure for approximating the DFT is presented. This proposed structure is shown to be a reversible integer-to-integer mapping called Integer FFT (IntFFT). All coefficients can be represented by finite-length binary numbers. The complexity of the proposed IntFFT will be compared with the conventional fixed-point implementation of the FFT (FxpFFT).

The invertibility of the DFT is guaranteed by orthogonality. The inverse (the IDFT) is just the conjugate transpose. In practice, fixed-point arithmetic is often used to implement the DFT in hardware since it is impossible to retain infinite resolution of the coefficients and operations [6, 7]. The complex coefficients of the transform are normally quantized to a certain number of bits depending on the tradeoff between the cost (or power) and the accuracy of the transform. However, direct quantization of the coefficients used in the conventional structures, including both direct and reduced-complexity (e.g. radix-2, radix-4, etc.) methods, destroys the invertibility of the transform. The novel approach presented in this paper guarantees the invertibility property of the transform while keeping the coefficients of the forward and inverse transforms to be finite-length binary numbers.

In this paper, we refer a real integer to a quantity whose real part has integer value and imaginary part is zero. Similarly, a complex integer is defined as a quantity whose real and imaginary parts have integer values.

1.1. Previous Works

Recently, there has been a reasonable amount of attention in trying to approximate the existing floating-point orthogonal transforms such as the DCT [8, 9, 10] or DFT [11] with invertibility property preserved. In [8], the 8-point DCT which is used in the image and video coding standards is approximated by a brute force technique i.e. the transform coefficients are optimized over the set of real integers by having orthogonality property (a system of nonlinear equations) as a constraint. The same approach is extended to the case of 8-point DFT whose coefficients are selected from the set of complex integers [11]. Although this approach is simple and straight forward, it is very difficult to extend the approach to the case of large N–imagine of solving a big system of nonlinear equations. In addition, once a set of coefficients is obtained, it is not trivial how to adjust them

for different transform accuracy unless one re-optimizes the coefficients.

In [9] and [10], lifting factorization is proposed to replace the 2×2 orthogonal matrices appearing in the 8-point DCT using, respectively, the fast structure and the Hadamard structure [12]. The resulting transforms are shown to be simple and invertible even though the lifting coefficients are quantized, and also are power-adaptable, i.e. different quantization step-sizes can be used to quantize the lifting coefficients without destroying the invertibility. In this paper, lifting factorization is proposed to be used in the fast structures of the DFT where complex multiplications are expressed in terms of liftings. The approach can be used in many existing structures such as radix-2, radix-4 and split-radix with arbitrary sizes. However, the split-radix structure will be used to illustrate the proposed method which can also be extended to other existing FFT structures as well.

2. THE DISCRETE FOURIER TRANSFORM

The DFT of an N-point discrete-time signal $x(n)$ is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad \text{for } k = 0,\ 1,\ \cdots, N-1 \quad (1)$$

where $W_N = e^{-j2\pi/N}$. Similarly, the IDFT can be given by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}, \text{ for } n = 0,\ 1,\ \cdots, N-1, \quad (2)$$

which follows from the orthogonal property of the DFT matrix.

Discarding the factor $1/N$ in (2), it is clear that, in order to calculate one coefficient of the DFT or IDFT, it requires N complex multiplications and $N-1$ complex additions. Therefore the total number of complex multiplications for computing an N-point DFT is N^2. However this direct computation is inefficient and can be significantly simplified by taking the advantages of the symmetry and periodicity properties of the twiddle factor W_N, i.e. $W_N^{k+N/2} = -W_N^k$ and $W_N^{k+N} = W_N^k$.

There are many existing fast structures to compute the DFT depending on the length of the input. In this paper, the split-radix structure which is suitable for input with length of $N = 2^K$ will be used to illustrate the proposed approach of `IntFFT`. The approach can also be applied to other structures such as radix-2 and radix-4 as well. In this paper, the split-radix FFT is chosen to demonstrage the paper where all the twiddle factor W_N^k are implemented by the proposed lifting steps as described following.

3. CONVERTING COMPLEX NUMBERS TO REAL LIFTING STEPS

Recall that all the coefficients appearing in the FFT and IFFT structures are complex numbers with magnitude one, i.e. every coefficient can be expressed as $e^{j\theta}$ where θ is some real number. Since these coefficients are scalar with magnitude one, the inverses are simply their complex conjugates. However, if a coefficient is quantized, the inverse of the new coefficient is no longer guaranteed to be its complex conjugate. Specifically, let a be a complex number with magnitude one, i.e.

$$a = c + js,$$

where c and s are real numbers and $c^2 + s^2 = 1$. Let a^q be the quantized version of a, i.e.

$$a^q = c^q + js^q$$

where c^q and s^q are finite-word length approximations of c and s respectively. Hence the reciprocal of a^q is

$$\frac{1}{a^q} = \frac{c^q}{|a^q|^2} - j\frac{s^q}{|a^q|^2}.$$

In general, $|a^q|$ is not one although $|a| = 1$. Instead, $1/a^q$ may not even be a finite word-length complex number even though a^q is. This is the reason why the conventional fixed-point arithmetic does not preserve the PR property.

The PR property can be preserved via the lifting scheme. Each complex multiplication is equivalent to four real multiplications. Specifically, let $x = x_r + jx_i$ be a complex number and hence $y = ax = (cx_r - sx_i) + j(cx_i + sx_r)$ or in the vector-matrix form:

$$y = [1\ j] \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x_r \\ x_i \end{bmatrix} = [1\ j]\mathbf{R} \begin{bmatrix} x_r \\ x_i \end{bmatrix}. \quad (3)$$

where $\mathbf{R} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$. Figure 1 shows the butterfly structure of a single complex multiplication. Notice that a has

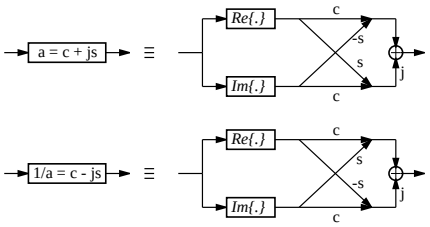

Figure 1: A butterfly structure for implementing a complex multiplication.

magnitude one if and only if $\mathbf{R}$ is an orthonormal matrix. Assuming that $c^2 + s^2 = 1$ with $s \neq 0$ (if $s = 0$ then $a = 1$ or -1 which does not need to be quantized), it is easy to see that $\mathbf{R}$ can be decomposed into three lifting steps as follow [13]:

$$\mathbf{R} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} 1 & \frac{c-1}{s} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{c-1}{s} \\ 0 & 1 \end{bmatrix}. \quad (4)$$

Figure 2 illustrates the conversion from one complex multiplication to three-step lifting scheme. In order to distinguish the coefficients in the lifting structure from the original coefficients c and s, we now call the new coefficients $\frac{c-1}{s}$

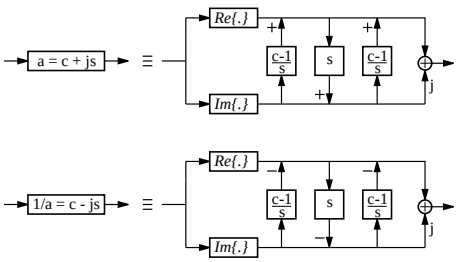

Figure 2: A lifting structure for implementing a complex multiplication and its inverse.

and s as *lifting coefficients* and refer the original coefficients c and s as *butterfly coefficients*.

The advantages of this factorization are two fold. First the number of real multiplications is reduced from four to three although the number of real additions is increased from two to three. Secondly, it allows for quantization of the lifting coefficients and the quantization of the result of each multiplication without destroying the PR property. To be specific, instead of quantizing a directly, the lifting coefficients s and $\frac{c-1}{s}$ are quantized and therefore its inverse $\frac{1}{a}$ also consists of three lifting steps with the same lifting coefficients but with opposite signs. Further than that, nonlinear operators can also be applied to the product at each lifting step.

The dynamic range at the internal nodes is one of important factors. In order for the transform to be perfectly reversible, the number of bits at internal nodes N_n has to be at least this dynamic range. Under an assumption that the resolution of each quantizer (of a lifting coefficient or the product after a lifting coefficient) is sufficiently high, the following conclusion can be made.

Theorem 3.1 *The lifting implementation of each twiddle factor increases the resolution of its input by at most one bit.*

The detailed discussion can be found in [14]. Let us consider the case of split-radix FFT with size $N = 2^k$. Define $f(k)$ to be least upper bound of the number of bits at internal nodes which guarantees the reversibility of the transform. The following theorem gives an upper bound for $f(k)$.

Theorem 3.2 $f(2r) \leq N_i + 3r - 1$ *and* $f(2r+1) \leq N_i + 3r + 1$, *where* $r \geq 1$, *where* N_i *is the number of bits used to represent the input signal.*

4. PERFORMANCES AND COMPLEXITIES

4.1. The Accuracy of the IntFFT

In this section, the performances of the IntFFT are experimentally evaluated and compared with the conventional FxpFFT. The experiment is performed for the case of $N = 256$. The input signal is quantized to 16 bits while the internal nodes of both structures are set to 27 bits. Obtained from Theorem 3.2, 27 bits will ensure that the IntFFT is reversible, and thus zero reconstruction error. Figure 3 compares the errors of the Fourier transform using the FxpFFT and the IntFFT for different values of N_c. In the structure of the FxpFFT, N_c is the number of bits used to quantize the twiddle factors while, in case of the IntFFT case, N_c is the number of bits used to quantize the lifting coefficients. It is evident that the proposed IntFFT is slightly more accurate than the conventional FxpFFT especially when N_c is high. Figure 4 shows the reconstruction error for the case of FxpFFT and FxpIFFT while the reconstruction error of the proposed IntFFT and IntIFFT remains zero.

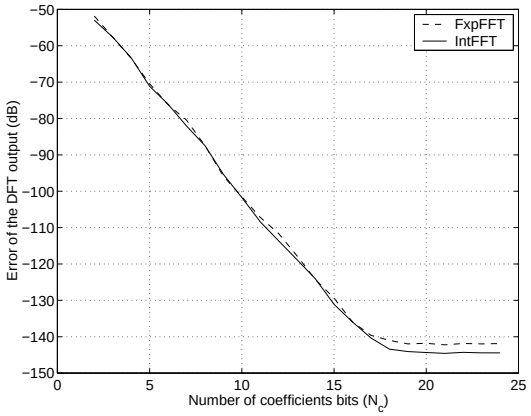

Figure 3: A comparison of the accuracy of the conventional fixed-point arithmetic FFT and the new IntFFT measured in frequency domain.

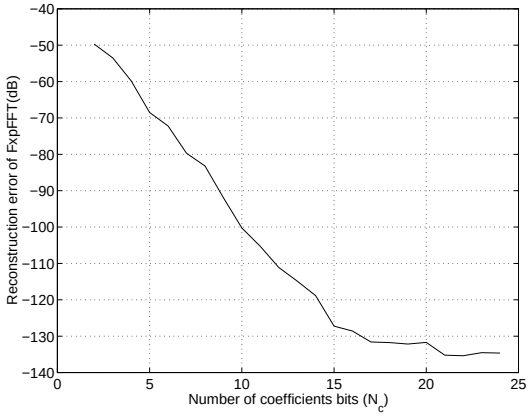

Figure 4: The reconstruction error of the conventional fixed-point arithmetic FFT.

4.2. Complexity of the IntFFT

In this section, the method for minimizing the number of adders discussed in section V of [14] is used to estimate the computational complexities of FxpFFT and IntFFT of different sizes. In particular, the numbers of 1's used in the 10-bit binary representations of the butterfly coefficients in FxpFFT and of the lifting coefficients in IntFFT are counted. Table 1 summarizes the numbers of real multiplications and real additions needed to perform N-point split-radix FFT [15],

Table 1: Computational complexities of the split-radix FFT and the integer versions (`FxpFFT` and `IntFFT`) when the coefficients are quantized to $N_c = 10$ bits.

	FFT		`FxpFFT`		`IntFFT`	
N	Multiplications	Additions	Additions	Shifts	Additions	Shifts
16	20	148	262	144	202	84
32	68	388	746	448	559	261
64	196	964	1910	1184	1420	694
128	516	2308	4674	2968	3448	1742
256	1284	5380	10990	7064	8086	4160
512	3076	12292	25346	16472	18594	9720
1024	7172	27652	57398	37600	41997	22199

and the numbers of real additions and shifts required in N-point `FxpFFT` and `IntFFT`. From Table 1, the numbers of additions of `FxpFFT` and `IntFFT` are approximately 100% and 50% more than that of the exact FFT, however, no real multiplication is needed. Comparing between `FxpFFT` and `IntFFT`, the number of additions of `IntFFT` is 29 - 37% less than `FxpFFT` while the number of shifts is 69 - 72% less.

5. CONCLUSION

In this paper, we have presented a concept of `IntFFT` which can be used to construct FFT with integer coefficients. It provides a new method for approximating the DFT without using any multiplication, and can simply be applied to the case of large-size DFT. Unlike the `FxpFFT` which is the fixed-point arithmetic version of FFT, the `IntFFT` is reversible when the coefficients are quantized. Its inverse `IntIFFT` can be computed with the same computational cost as that of the forward transform. The new transform is suitable for mobile computing and any handheld devices which run on batteries since it is adaptable to available power resources. Specifically, the coefficients appearing in the proposed structures can be quantized directly for different resolutions, i.e. different computational costs, while preserving the reversibility property.

Although a large class of FFT structures such as radix-2, radix-4 and split-radix, can be approximated by this approach, however, the split-radix structure is used to illustrate the technique. According to the simulation, the complexity of `IntFFT` is lower than that of `FxpFFT` by a significant margin.

The accuracy of the transforms is compared experimentally. It is evident from the simulations that both `IntFFT` and `FxpFFT` have approximately the same distortion from ideal FFT when computing forward transform. On the other hand, when the inverse transform is performed after the forward transform, fixed-point arithmetic approach results in reconstruction error while the proposed approach can reconstruct the input perfectly for any fixed resolution of the coefficients.

6. REFERENCES

[1] J.S. Walker, *Fast Fourier Transforms*, CRC Press, 2nd edition, 1996.

[2] L. Krasny and S. Oraintara, "Noise reduction for speech signals using voice activity detector," Technical report EUS/TR/X-98:1662, Ericsson Inc., October 1998.

[3] A.K. Jain, *Fundamental of Digital Image Processing*, Prentice-Hall, 1989.

[4] J.W. Cooley and J.W. Tukey, "An algorithm for the machine caculation of complex fourier series," *Math. Comp.*, vol. 19, pp. 297–301, April 1965.

[5] M.T. Heideman, *Multiplicative Complexity Convolutional, and the DFT*, Springer-Verlag, 1988.

[6] A.V. Oppenheim and R.W. Schafer, *Discrete-time Signal Processing*, Pretice Hall, 1989.

[7] J.G. Proakis and D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*, Prentice-Hall, Englewood Cliffs, NJ, second edition, 1992.

[8] T-C. Pang, C-S. O. Choi, C-F. Chan, and W-K. Cham, "A self-timed ict chip for image coding," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, no. 6, 1999.

[9] T. Tran, "Fast multiplierless approximation of the dct," in *CISS 99*, 1999.

[10] Y-J. Chen, S. Oraintara, and T. Nguyen, "Video compression using integer dct," in *ICIP*, 2000.

[11] S-C. Pei and J-J. Ding, "Integer discrete fourier transform and its extension to integer trigomatric transforms," in *ISCAS*, 2000.

[12] K.R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, 1990.

[13] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," Technical report, Bell Laboratories, Lucent Technologies, 1996.

[14] S. Oraintara, Y-J. Chen, and T.Q. Nguyen, "Integer fast fourier transform (`intfft`)," Submitted to IEEE Trans. on Signal Processing, October 2000.

[15] M.T. Heideman and C.S. Burrus, "On the number of multiplications necessary to compute a length 2^n dft," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-34, no. 1, pp. 91–95, February 1986.