Polynomial 변환을 이용한 고속 2 차원 FFT

최환석*, 김원하**, 한승수***

Two dimensional FFT by Polynomial Transform

Hwan-Serk Choi*, Won-Ha Kim**, Seung-Soo Han***
*Department of Electronics Engineering, Myongji University, Yongin Korea
**Electronic & Information Engineering, Kyunghee University, Suwon Korea
***Department of Information Engineering, Myongji University, Seoul Korea
E-mail : *andxor@mju.ac.kr, **wonha@khu.ac.kr, ***shan@mju.ac.kr

Abstract

We suggest 2 dimensional Fast Fourier Transform using Polynomial Transform and integer Fast Fourier Transform. Unlike conventional 2D-FFT using the direct quantization of twiddle factor, the suggested 2D-FFT adopts implemented by the lifting so that the suggested 2D-FFT is power adaptable and reversible. Since the suggested FFT performs integer-to-integer mapping, the transform can be implemented by only bit shifts and additions without multiplications. In addition, polynomial transform severely reduces the multiplications of 2D-FFT. While preserving the reversibility, complexity of this algorithm is shown to be much lower than that of any other algorithms in terms of the numbers of additions and shifts.

I. Introduction

The Fast Fourier Transform (FFT) is one of the most fundamental tools in digital signal processing [2]. Fast algorithms such as the Cooley-Tukey Fast Fourier Transform, the Good-Thomas Fast Fourier Transform, and the Winogard Small Fast Fourier Transform are available to compute 1D-FFT efficiently. For two or higher dimensional FFT, the row-column method, the vector-radix FFT and the polynomial transform algorithms are commonly used fast algorithms [3]. These algorithms 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 [4]. A difficulty of these algorithms is occurred on the implementation. In actual DSP implementation, it is impossible to retain infinite resolution of the signal samples and the transform coefficients [2]. So, the numbers produced during these FFTs are often quantized to a fixed number of bits. The most significant bits (MSBs) of the result after each operation will be kept up to bits, and the tail will be truncated. A disadvantage of this conventional fixed-point arithmetic method destroys the invertibility of the transform because the FFT coefficients are quantized.

To preserves the invertibility, this paper uses the integer FFT[5] to quantize the FFT coefficients. Since the suggested FFT performs integer-to-integer mapping, the transform can be implemented by only bit shifts and additions without multiplications while preserving the invertibility. In addition, Polynomial transform [6] is also used to reduce the number of multiplications while computing 2 dimensional FFT. The polynomial transform exploits the modular permutations induced from the coprime relationship between row and column operations. Empirical and analytical results show that the suggested 2D-FFT does not need the multiplications and severely reduces the additions.

This paper is organized as follows. Section II describes the

본 논문은 2001 년도 한국과학재단의 지원에 의하여 연구되었음 (R05-2003-000-11653-0)

2D-FFT obtained by Polynomial transform, which has no multiplications, and fast algorithm by showing a certain symmetry in the suggested polynomial transform. In Section III, 1D-FFT based on Lifting scheme the fast algorithm is derived with optimization of additions. In this section, we will explain how to preserve the invertibility property. Finally, we reach the conclusion in Section V.

II. 2D-DFT via Polynomial

This section explains that 2D-FFT is represented with polynomial and then its fast algorithm is applied in the polynomial expression of 2D-FFT. The conventional fast algorithms of 2D-FFT process have no relations between its row and column. These independent processes can be related with each row and column by module operation, and then row (or column) index is subordinated to column (or row) index. The dependency of row, based on column, means that row indexes have the overlapped factors of column, which are the twiddle factors for column index and those common factors can be substituted by a representative z. This representation of 2D-DFT makes the series of polynomial on the order of z and transform for row is acquired in the process of transform for column at a time. Then this polynomial representation has the symmetry and periodicity for z and fast polynomial transform algorithm [6]. Namely, the relation between even (or odd) row index k and odd (or even) column index l makes 2D-DFT be applied to polynomial transform by module operation. Polynomial representation and its fast transform are described in next section.

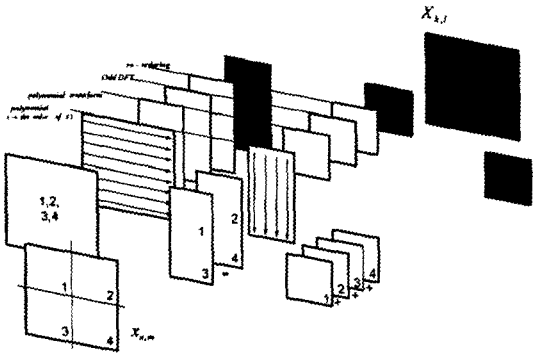

Fig. 1 Abstract diagram of 2D-FFT via Polynomial Transform

The $N \times N$ DFT of input sequence $x_{n,m}$ is defined by

$$X_{k,l} = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} x_{n,m} W_N^{kn} W_N^{lm}, \qquad (1)$$

where $k = 0,1,\cdots N-1$; $l = 0,1,\cdots N-1$; $W_N = e^{-j\frac{2\pi}{N}}$, $N = 2^t$,

2.1 Decomposition of 2D-DFT

In general, the permutation via *module* operation has the periodicity property, therefore it has something in common with DFT. So we start the derivation of polynomial transform from decomposition of 2D-DFT, which is using the relation of row and column index to be capable of permutation by *module*. We represent 2D-DFT by means of polynomial to apply to the Polynomial transform.

Now, 2D-DFT is decomposed as parity of k and l.

We divide $X_{k,l}$ into two parts, depending on the parity of l.

$$X_{k,2l} = \sum_{n=0}^{N-1}\left\{\sum_{m=0}^{N/2-1}\left(x_{n,m} + x_{n,m+N/2}\right)W_{N/2}^{lm}\right\}W_N^{kn} \qquad (2)$$

$$X_{k,2l+1} = \sum_{n=0}^{N-1}\left\{\sum_{m=0}^{N/2-1}\left(\left(x_{n,m} - x_{n,m+N/2}\right)W_N^{m}\right)W_{N/2}^{lm}\right\}W_N^{kn} \qquad (3)$$

And, we divide $X_{k,2l}$ into two parts over again, depending on the parity of k.

$$X_{2k,2l} = \sum_{n=0}^{N/2-1}\left\{\sum_{m=0}^{N/2-1}\left(x_{n,m} + x_{n,m+N/2} + x_{n+N/2,m} + x_{n+N/2,m+N/2}\right)W_{N/2}^{lm}\right\}W_{N/2}^{kn} \qquad (4)$$

$$X_{2k+1,2l} = \sum_{n=0}^{N/2-1}\left\{\left[\sum_{m=0}^{N/2-1}\left(x_{n,m} + x_{n,m+N/2} - x_{n+N/2,m} - x_{n+N/2,m+N/2}\right)W_{N/2}^{lm}\right]W_N^{n}\right\}W_{N/2}^{kn} \qquad (5)$$

This is computed by $N/2$ DFT and polynomial transform of more less rows than (3).

We decomposed into (4) and (5) by the parity of k and l. In (4), the same computation process is applied recursively to the DFTs of $N/2 \times N/2$, $N/4 \times N/4$, ... points until the complete decomposition is achieved. The other processes are applied to polynomial transform. Through these decompositions, k can be permutated by l and polynomial representation via *module* operation is explained in following section.

2.2 The representation of DFT via Polynomial

Previously, we have the decompositions of 2D-DFT. When the permutation of (6) is applied to the decompositions of (3) and (5) as follows, their common twiddle factors are substituted by a representative z and these are processed by polynomial transform.

Here we show how to subordinate row-index k to column-index l. In order that permutation by *module* is possible, k must be dependent on l. Then we can permute k as follows.

we define $k(l)$ as $k(l)=k(2l+1)\bmod N$.

$$A=\{(k,l)\,|\,0\le k\le N-1;\ 0\le l\le M/2-1\}$$
$$B=\{(k(l),l)\,|\,0\le k\le N-1;\ 0\le l\le M/2-1\} \quad (6)$$

Then $A=B$.

From (6), we permute k to $k(2l+1)\bmod N$ and then (3) is

$$\begin{aligned}X_{k(2l+1),2l+1}&=\sum_{n=0}^{N-1}\left\{\sum_{m=0}^{N/2-1}\left(x_{n,m}-x_{n,m+N/2}\right)W_N^{(2l+1)m}\right\}W_N^{\{k(2l+1)\bmod N\}n}\\&=\sum_{n=0}^{N-1}\left\{\sum_{m=0}^{N/2-1}\left(x_{n,m}-x_{n,m+N/2}\right)W_N^{(2l+1)m}\right\}W_N^{k(2l+1)n}\\&=\sum_{n=0}^{N-1}\left\{\sum_{m=0}^{N/2-1}\left(x_{n,m}-x_{n,m+N/2}\right)z^m\right\}z^{kn}\Big|_{z=W_N^{2l+1}}\\&=\left[\sum_{n=0}^{N-1}x_n(z)z^{kn}\bmod\left(z^{N/2}+1\right)\right]\bmod\left(z-W_N^{2l+1}\right)\\&=\overline{X}_k(z)\bmod\left(z-W_N^{2l+1}\right)\end{aligned} \quad (7)$$

, where $x_n(z)\equiv\sum_{m=0}^{N-1}x_{n,m}z^m$ and polynomial transform is

$$\overline{X}_k(z)=\sum_{n=0}^{N-1}x_n(z)z^{nk}\bmod\left(z^{N/2}+1\right)\ \text{ for }\ k=0,1,\cdots,N-1. \quad (8)$$

In case of (5), we permute l to $l(2k+1)\bmod N$ and then its expansion is

$$\begin{aligned}X_{2k+1,2l(2k+1)}&=\sum_{m=0}^{N/2-1}\left\{\sum_{n=0}^{N/2-1}\left(x_{n,m}+x_{n,m+N/2}-x_{n+N/2,m}-x_{n+N/2,m+N/2}\right)W_N^{(2k+1)n}\right\}W_N^{2l(2k+1)\bmod N\}m}\\&=\sum_{m=0}^{N/2-1}\left\{\sum_{n=0}^{N/2-1}\left(x_{n,m}+x_{n,m+N/2}-x_{n+N/2,m}-x_{n+N/2,m+N/2}\right)W_N^{(2k+1)n}\right\}W_N^{2l(2k+1)m}\\&=\sum_{m=0}^{N/2-1}\left\{\sum_{n=0}^{N/2-1}\left(x_{n,m}+x_{n,m+N/2}-x_{n+N/2,m}-x_{n+N/2,m+N/2}\right)z^n\right\}z^{lm}\Big|_{z=W_{N/2}^{2k+1}}\\&=\left[\sum_{m=0}^{N/2-1}x_m(z)z^{lm}\bmod\left(z^{N/2}+1\right)\right]\bmod\left(z-W_{N/2}^{2k+1}\right)\\&=\overline{X}_l(z)\bmod\left(z-W_{N/2}^{2k+1}\right)\end{aligned} \quad (9)$$

,where $x_m(z)\equiv\sum_{n=0}^{N-1}\left(x_{n,m}+x_{n,m+N/2}\right)z^n$ and polynomial transform is

$$\overline{X}_l(z)=\sum_{m=0}^{N/2-1}x_m(z)z^{lm}\bmod\left(z^{N/2}+1\right)\ \text{ for }\ l=0,1,\cdots,N/2-1. \quad (10)$$

2D-DFT was decomposed into 3 parts of (4), (3) and (5). $N/2\times N/2$ DFT of (4) is recursively applied to the same computation processes. In here, each (7) and (9) is evaluated by Polynomial transform of (8) and (10), which simplifies the 2 dimensional transform for row (or column) dimension by additions of z 's coefficients, and the odd outputs of a length N DFT, which is sometimes called an odd DFT, in the last steps of (7) and (9).

Polynomial transforms was defined as (8) and (10). These polynomial transform has some important properties such as the symmetry and periodicity that are enable to be computed by fast algorithm. As we know, DFT has the twiddle factor W_N^n, which is periodic and symmetric, and it can be computed by Fast Fourier Transform. On the other hand, the polynomial transform has *module* operation of $\left(z^{N/2}+1\right)$ as the role of twiddle factor. The periodicity and symmetry is shown by

$$\begin{aligned}z^N&\equiv 1\bmod\left(z^{N/2}+1\right)\\z^{N/2}&\equiv -1\bmod\left(z^{N/2}+1\right)\end{aligned} \quad (11)$$

So, polynomial transform has fast algorithm likewise FFT necessarily.

III. Conclusion

In this paper, we have presented a concept of 2D-DFT using integer processing by polynomial transform. 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 general 2D-DFT using the quantization of twiddle factor, the proposed 2D-FFT algorithm is reversible. Also, because the polynomial transform does not have quantization process, but only integer FFT has, the accuracy of the transforms depends on the lifting coefficients.

This new transform is suitable for mobile computing and any handheld devices that 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.

Table I. Number of non trivial real operations for 2D-FFT on complex input data (shadowed area: proposed 2D-FFT)

N×N	Polynomial transform				Row-Column			
	SRFFT		IntFFT		SRFFT		IntFFT	
	×	+	+	Bit-shifts	×	+	+	Bit-shifts
16×16	528	4604	4916	1764	640	4736	6464	2688
32×32	3408	23868	28212	9348	4352	24832	35776	16704
64×64	18384	116668	147508	52164	25088	123392	181760	88832
128×128	91344	550076	723764	277572	132096	590848	882688	445952
256×256	433872	2531004	3465780	1403460	657408	2754560	4140032	2129920
512×512	2003664	11440828	16098868	6804036	3149824	12587008	19040256	9953280
1024×1024	9075408	51018428	73334324	32086596	14688256	56631296	86009856	45463552

References

[1] Jae S. Lim Two-Dimensional Signal and Image Processing, Englewood Cliffs, New Jersey: Prentice-Hall", YEAR = 1990

[2] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall 1992.

[3] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*, MA: Addison-Wesley, 1984.

[4] P. Duhamel, "Implementation of 'Split-Radix' FFT Algorithms for Complex, Real, and Real-Symmetric Data' IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP-34, pp.285-295, Apr. 1986.

[5] S. Oraintara, Y.J. Chen, and T.Q. Nguyen, "Integer Fast Fourier Transform", *IEEE Trans. Signal Processing*, vol.50, No.3, pp.607–618, March. 2002.

[6] H. J. Nussbaumer and P. Quandalle, "Fast Computation of Discrete Fourier Transforms Using Polynomial Transforms", IEEE Trans. Acoust. Speech, Signal Processing, vol. ASSP-27, NO. 3, pp.169-181, Apr. 1979.

[7] H. J. Nussbaumer, "New algorithms for convolutions and DFT based on polynomial transforms" in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, pp.638-641, 1978.

[8] G. Strang and T. Nguyen. *Wavelets and Filter Banks*, Wellesley, MA: Wellesley-Cambridge, 1996.