ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31, 2000, Geneva, Switzerland

INTEGER DISCRETE FOURIER TRANSFORM AND ITS EXTENSION TO INTEGER TRIGOMATRIC TRANSFORMS

Soo-Chang Pei *Jian-Jiun Ding*

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C

Email address: pei@cc.ee.ntu.edu.tw

ABSTRACT

DFT has the good quality of performance and fast algorithm. But when we implement the DFT, we will require the floating-points multiplication. In this paper, we will introduce the *integer Fourier transform* (ITFT). ITFT is approximated to the DFT, but all the entries in the transform matrix are integer numbers. So it only requires the fixed-points multiplication, and the implementation can be much simplified, especially for VLSI. This new transform will work very similar as the original DFT, for example, the transform results are similar and the shifting-invariant property is also preserved for ITFT. Besides, we will also introduce the general method to derive the integer transform. By this approach, we can derive many types of integer transforms (such as integer cosine, sine, and Hartley transforms).

I. INTRODUCTION

Because of the direct relations with frequency spectrum and the fast algorithm, DFT is a very popular tool for signal processing. But when we implement the DFT, some floating-points multiplication operations are required. The implementation of floating-points multiplication is usually trouble and time-consuming. Besides, for the computer, it requires the floating-points processor, and will be more complex and expensive.

In this paper, we will derive the *integer Fourier transform* (ITFT). It approximates to the DFT, but all the entries in the transform matrix are integer numbers. We can implement the ITFT without any the floating-points multiplication operations because there are no non-integer entries. In section 2, we will introduce the general method to derive the integer transform from some real discrete transform. This method is the generalization and the modification of the method introduced by Cham [1][3] (He used his method to derive the integer cosine transform). In section 3, we will derive the 8-points ITFT. Then, in section 4, we will discuss the performance and the property of ITFT. We will show ITFT remains almost all the performance quality of DFT. In section 5, we make a conclusion.

II. THE GENERAL METHOD TO DERIVE THE INTEGER TRANSFORM

Integer transform is the discrete transform that all the entries in the transform matrix are integer numbers. When we try to find the integer transform analogous to some real transform, we can use following the steps described as below. Here we use **A** to denote the transform matrix of the original real transform, use **B** to denote the forward transform matrix of the integer transform, and use **D** to denote the inverse transform matrix of the integer transform.

(1) Find the ***symmetry relation and the equality relation*** for each row of the original real transform, and find the *sign* of the entries of the transform matrix.

(2) ***Forming the prototype matrix of the forward integer transform*** from the relations obtained in the step 1. The prototype must satisfy the following rules:

(a) If for the original transform matrix,

$A(m, a_1) = C \cdot A(m, a_2)$ where $C = 1, -1, j, -j$

then for the forward and inverse integer transform, the equality relations as below must be satisfied:

$B(m, a_1) = C \cdot B(m, a_2)$ $D(m, a_1) = C \cdot D(m, a_2)$

(b) If for the original transform matrix,

$A(m_1, n_1) = 0$ $A(m_2, n_2) > 0$ $A(m_3, n_3) < 0$

then for the forward and inverse integer transform,

$B(m_1, n_1) = 0$ $B(m_2, n_2) > 0$ $B(m_3, n_3) < 0$

$D(m_1, n_1) = 0$ $D(m_2, n_2) > 0$ $D(m_3, n_3) < 0$

From these rules, we assign the unknowns in each entry of the prototype matrix. To be convenient, all the unknowns are constrained to be positive and real integers. If the prototype matrix has too many unknowns, we can try to make some unknowns be the same, or make some unknowns to be 1.

(3) ***Forming the prototype matrix of the inverse integer transform***. The prototype of transform matrix of the inverse transform is of the same form as the forward transform, but we use another set of unknowns.

(4) ***Constraints for orthogonality***. In this step, we search for the requirements to make the transform matrices of the forward and inverse transform to be orthogonal, and make the inner product of the same rows to be the values of 2^k (k is integer):

$$\sum_{k=0}^{N-1} B(m,k)\overline{D(n,k)} = R_m \delta_{mn} \quad \text{where } R_m = 2^k \qquad (1)$$

From this, we obtain the *equality constraints* for the unknowns of the prototypes matrices.

(5) ***Constraints for inequality***. In this step, we find the inequality relations among the unknowns of the prototypes matrices from the inequality relations for each row of the original non-integer transform matrix. That is, if for the original transform,

$A(m, n_1) > A(m, n_2)$

then for the forward and inverse integer transforms,

$B(m, n_1) > B(m, n_2)$ $D(m, n_1) > D(m, n_2)$

These will be the *inequality constraints* for the unknowns.

0-7803-5482-6/99/$10.00 ©2000 IEEE

(6) ***Assign the values for all the unknowns.*** At last, we assign the values of unknowns. They must be real, positive integer numbers, and satisfy the constraints obtained in steps (4), (5).

We note, the transform matrix **B** of the forward transform and the transform matrix of the inverse transform **D** would be different. Thus, if $X(m)$ is the forward integer transform of $x(n)$,

$$X(m)=\sum_{n=0}^{N-1} B(m,n)\cdot x(n) \qquad (2)$$

Then we can recover $x(n)$ from $X(m)$ by

$$x(n)=\sum_{m=0}^{N-1} D^{*}(m,n)\cdot R_m^{-1}\cdot X(m) \qquad (3)$$

where * represents the conjugation, and R_m is defined in Eq. (1). Eqs. (2), (3) can also be written as

$$\mathbf{X}=\mathbf{B}\cdot\mathbf{x} \qquad \mathbf{x}=\mathbf{D}^{H}\mathbf{C}^{-1}\mathbf{X} \qquad (4)$$

where ^{H} is the Hermitian operation. The method introduced in [1] makes the transform matrix for the forward and the inverse integer transform to be the same, and can't avoid the floating-points multiplication for the inverse transform (since the values of R_m in Eq. (1) would not be the form of 2^k). Here we allow the forward and the inverse integer transform to be different. This enables us to fully avoid the floating-points multiplication operations, no matter for the forward or inverse transform. But since **B, D** are of the same forms, so the structures of the implementation of the forward and inverse transform are basically the same, except for the direction would be reversed and the parameters are different.

From the process introduced above, we can assure the integer transform we derived are very similar to the original non-integer transform, because (1) the symmetry, equality relations for each row, (2) the sign of each entry, (3) the orthogonality property, (4) the inequality relations for each row have been preserved. Thus, many properties of the original transform (such as the shift-invariant property of DFT) will be almost preserved.

III. THE 8-POINTS INTEGER FOURIER TRANSFORM

The original 8-points DFT is:

$$F(m,n)=\exp(-j\,m\,n\,\pi/4) \quad m,n\in[0,1,\ldots 7] \qquad (5)$$

To derive the 8-points integer Fourier transform (ITFT), we first form its prototype from the equality relation and the sign of each entry of the original 8-points DFT (steps 1, 2 in section 2). We list the equality relations of the original 8-points DFT as below:

row	row 0	row 1	row 2	row 3	row 4	row 5	row 6	row 7
G=1	C=1		C=-j		C=-1		C=j	
G=2	C=1	C=-j	C=-1	C=j	C=1	C=-j	C=-1	C=j
G=4	C=1	C=-1	C=1	C=-1	C=1	C=-1	C=1	C=-1

Table 1 the equality relation of the original 8-points DFT

The values of G and C mean the m^{th} row of the original 8-points DFT will have the following relations

$$F(m,n\oplus G)=C\cdot F(m,n) \quad \text{if } n\oplus G>n \qquad (6)$$

$$F(m,n\oplus G)=C^{-1}\cdot F(m,n) \quad \text{if } n\oplus G<n \qquad (7)$$

where $\oplus$ is the exclusive-OR addition:

$$A\oplus B=\sum_{i=0}^{2} a_i\cdot 2^i\oplus\sum_{i=0}^{2} b_i\cdot 2^i=\sum_{i=0}^{2}(a_i\,XOR\,b_i)\cdot 2^i$$

where a_i, b_i = 0, 1. We note, from the step 2 described in section 2, the 8-points ITFT must also have the same equality and symmetry relations for each row as the original 8-points DFT, so Eqs. (6), (7) must also hold for the 8-points ITFT. Then together with the sign of the entries in the 8-points DFT matrix, we can construct the prototype matrix of the 8-points forward ITFT as:

$$\mathbf{FI}_P=\begin{bmatrix} g_1 & g_1 & g_1 & g_1 & g_1 & g_1 & g_1 & g_1 \\ a_1 & a_2-ja_2 & -ja_1 & -a_2-ja_2 & -a_1 & -a_2+ja_2 & ja_1 & a_2+ja_2 \\ g_2 & -jg_2 & -g_2 & jg_2 & g_2 & -jg_2 & -g_2 & jg_2 \\ b_1 & -b_2-jb_2 & jb_1 & b_2-jb_2 & -b_1 & b_2+jb_2 & -jb_1 & -b_2+jb_2 \\ g_3 & -g_3 & g_3 & -g_3 & g_3 & -g_3 & g_3 & -g_3 \\ c_1 & -c_2+jc_2 & -jc_1 & c_2+jc_2 & -c_1 & c_2-jc_2 & jc_1 & -c_2-jc_2 \\ g_4 & jg_4 & -g_4 & -jg_4 & g_4 & jg_4 & -g_4 & -jg_4 \\ d_1 & d_2+jd_2 & jd_1 & -d_2+jd_2 & -d_1 & -d_2-jd_2 & -jd_1 & d_2-jd_2 \end{bmatrix} \qquad (8)$$

We assign the unknowns for the real part and image part of the entries separately to assure all the unknowns will be real numbers. In this prototype matrix, there are totally 12 unknowns. The amount of unknowns seems to be too much, so we will set some unknowns to be 1 and some unknowns to be equal. We set

$$g_1=g_2=g_3=g_4=1, \qquad (9)$$

$$b_1=c_1,\quad b_2=c_2,\qquad d_1=a_1,\quad d_2=a_2 \qquad (10)$$

Then the prototype is simplified as:

$$\mathbf{FI}_P=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a_1 & a_2-ja_2 & -ja_1 & -a_2-ja_2 & -a_1 & -a_2+ja_2 & ja_1 & a_2+ja_2 \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ c_1 & -c_2-jc_2 & jc_1 & c_2-jc_2 & -c_1 & c_2+jc_2 & -jc_1 & -c_2+jc_2 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ c_1 & -c_2+jc_2 & -jc_1 & c_2+jc_2 & -c_1 & c_2-jc_2 & jc_1 & -c_2-jc_2 \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ a_1 & a_2+ja_2 & ja_1 & -a_2+ja_2 & -a_1 & -a_2-ja_2 & -ja_1 & a_2-ja_2 \end{bmatrix} \qquad (11)$$

and there are only 4 unknowns. But we must assure the ITFT can still be derived after the simplification, otherwise we must remove some of the equality relations Eqs. (9), (10). We note, in Eq. (11), the m^{th} row of the prototype matrix will be the conjugation of the $(7-m)^{th}$ row, as the original DFT.

Then we form the prototype for the inverse ITFT. The prototype is of the same form as Eq. (11), but we change the unknowns $\{a_1, a_2, c_1, c_2\}$ as $\{a_3, a_4, c_3, c_4\}$:

$$\mathbf{IFI}_P=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ a_3 & a_4-ja_4 & -ja_3 & -a_4-ja_4 & -a_3 & -a_4+ja_4 & ja_3 & a_4+ja_4 \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ c_3 & -c_4-jc_4 & jc_3 & c_4-jc_4 & -c_3 & c_4+jc_4 & -jc_3 & -c_4+jc_4 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ c_3 & -c_4+jc_4 & -jc_3 & c_4+jc_4 & -c_3 & c_4-jc_4 & jc_3 & -c_4-jc_4 \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ a_3 & a_4+ja_4 & ja_3 & -a_4+ja_4 & -a_3 & -a_4-ja_4 & -ja_3 & a_4-ja_4 \end{bmatrix} \qquad (12)$$

There are 4 unknowns for the inverse ITFT, so there are totally 8 unknowns for ITFT.

Then we search the constraints for the unknowns. First, we try to find the orthogonality constraints to make each row of the forward, inverse ITFT to satisfy the Eq. (1).

From the prototype of the forward, inverse ITFT, we find except for {row 1, 5}, {row 5, 1}, {row 3, 7}, and {row 7, 3}, all pairs of different rows will be orthogonal to each other. And calculating the inner product of {row 1, 5}, {row 5, 1}, {row 3, 7}, and {row 7, 3} directly, we find if we want these pairs to be orthogonal, then the following constraints must be satisfied:

(i) $a_1 c_3 = 2 a_2 c_4$, (ii) $a_3 c_1 = 2 a_4 c_2$

Then, from the requirement that the inner product of the same row must be the power of 2, we obtain the constraint as:

(iii) $a_1 a_3 + 2 a_2 a_4 = 2^k$, (iv) $c_1 c_3 + 2 c_2 c_4 = 2^h$

where k, h are integer numbers. The constraints of (i), (ii), (iii), (iv) will be the equality constraints of the unknowns.

Then, from the inequality relations for each row of original DFT matrix, we obtain the inequality constraints for the 8-points integer DFT as:

(v) $a_1 \geq a_2$ (vi) $c_1 \geq c_2$ (vii) $a_3 \geq a_4$ (viii) $c_3 \geq c_4$

These are the inequality constraints. Thus we have obtained all the constraints for unknowns.

Then we assign the values of unknowns. Since there are 8 unknowns and only 4 equality constraints, so there are infinite choices for the values of unknowns. But to search the values of unknowns to satisfy all the constraints, especially to satisfy the constraints (iii), (iv), is a difficult task. We introduce a process as below. This process will make the work of searching for the values of unknowns to be more efficient.

(a) Choose the values of a_1, a_2 such that a_1, a_2 are integer numbers and

$$2 \cdot a_2 \geq a_1 \geq a_2 \qquad (13)$$

(b) Find the integer values of c_1, c_2 such that $c_1 \geq c_2$, and

$$a_1 \cdot c_2 + a_2 \cdot c_1 = 2^n \qquad n \text{ is integer} \qquad (14)$$

(c) Set the value of a_3, a_4, c_3, c_4 as

$$c_3 = 2a_2, \quad c_4 = a_1, \quad a_3 = 2c_2, \quad a_4 = c_1 \qquad (15)$$

Then the values of unknowns are all obtained. We list some possible choices of the unknowns as below:

(1) a_1=2, a_2=1, c_1=2, c_2=1, a_3=2, a_4=2, c_3=2, c_4=2

This is the smallest, simplest integer solution for the unknowns. And in this case, the value of R_m in Eq. (1) is:

$$R_0 = R_2 = R_4 = R_6 = 2^3, \quad R_1 = R_3 = R_5 = R_7 = 2^4.$$

(2) a_1=7, a_2=5, c_1=13, c_2=9, a_3=18, a_4=13, c_3=10, c_4=7

We note, when we choose the parameters as the values listed above, the ratios of $a_1 : a_2$, $c_1 : c_2$, $a_3 : a_4$, $c_3 : c_4$, are all near to 1.414:1, which is ratio of the original discrete Fourier transform. If R_m is defined as Eq. (1), then in this case

$$R_0 = R_2 = R_4 = R_6 = 2^3, \quad R_1 = R_3 = R_5 = R_7 = 2^{10}.$$

We can implement the forward/inverse 8–points integer Fourier transform (8-points ITFT) in the following ways:

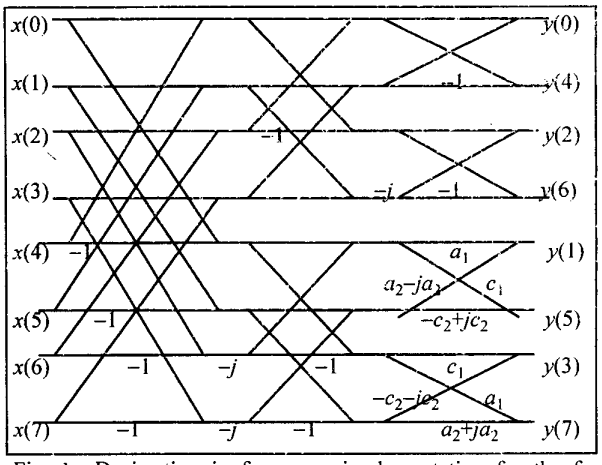

Fig. 1 Decimation–in–frequency implementation for the forward 8–point integer Fourier transform (ITFT)

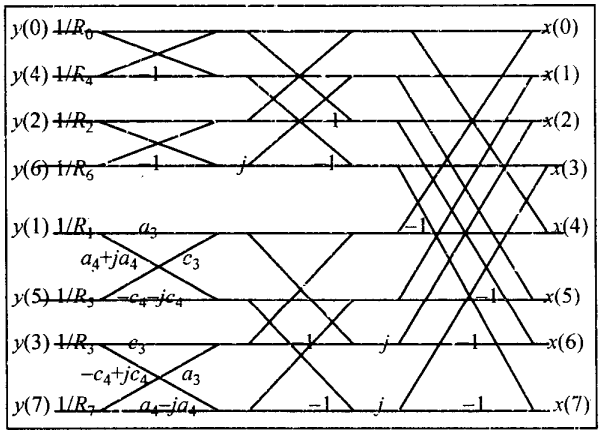

Fig. 2 Decimation–in–frequency implementation for the inverse 8–point integer Fourier transform

We find, for the 8-points ITFT, the number of the multiplication operations required is:

- forward/inverse transform: 8 fixed-points multi. operations
- normalization: 8 fixed-points multi. operations
- total: 24 fixed-points multiplication operations

And for the original 8-points DFT,

- forward/inverse transform: 2 floating-points multi. ops.
- normalization: 8 fixed-points multi. operations
- total: 4 floating-points, 8 fixed-points multi. operations

That is, we replace 4 floating-points multiplication operations with 16 fixed-points multiplication operations. The cost for doing 16 fixed-points multiplication operations is much less than the cost for doing 4 floating-points multiplication operation. Thus, 8-points ITFT will be much faster than the original 8-points DFT. Besides, we can further reduce the number of multiplication operations by setting a_1=c_1, or a_2=c_2, or a_3=c_3, or a_4=c_4, and each of them will save 2 fixed-points multiplication operations.

The advantages of the ITFT are the simpler implementation and saving the computation time, and its performance is much like the original DFT (see section 4). But some properties, such

as the convolution theorem, can just be approximated, not be preserved. We can choose the ratios of $a_1 : a_2$, $c_1 : c_2$, $a_3 : a_4$, $c_3 : c_4$ near to 1.414:1, then the performance of the ITFT will be more approximated to the original DFT.

IV. THE PERFORMANCE OF THE INTEGER FOURIER TRANSFORM

We will see the performance of the 8-points integer Fourier transform (8-points ITFT). Here the parameters we choose are a_1=7, a_2=5, c_1=13, c_2=9, a_3=18, a_4=13, c_3=10, c_4=7.

We first see the transform result. We choose 2 inputs:

$\mathbf{x}_1$ = [2, 3, 4, 5, 4, 5, 2, 3]. (16)

$\mathbf{x}_2$ = [2.8, 4.3-0.6i, 3.7+0.9i, 3.1-0.6i, 4.6, 3.1+0.6i, 3.7-0.9i, 4.3+0.6i] (17)

Then we do the original 8- points DFT, and plot the results in Fig. 3(c), 3(d). And then, we do the 8-points ITFT, and plot the results in Fig. 3(e), 3(f). We will normalize the transform results for comparison. That is, for the transform result of the ITFT

$$X(m) = \sum_{n=0}^{7} FI(m,n) \cdot x(n) \quad (18)$$

We will normalize $X(m)$ as below in Fig. 3(e), 3(f):

$$\tilde{X}(m) = X(m) / FI(m,0). \quad (19)$$

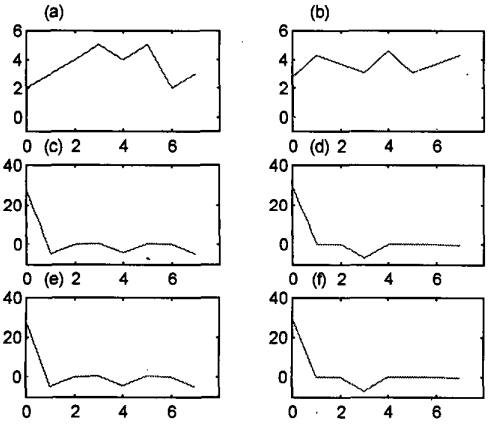

Fig. 3 The transform results of the original DFT and ITFT. (a) $x_1[n]$, (b) $x_2[n]$, (c) original DFT for $x_1[n]$, (d) original DFT for $x_2[n]$, (e) ITFT for $x_1[n]$, (f) ITFT for $x_2[n]$. (heavy lines): real part, (light lines): imaginary part

We find, the normalized transform results for the 8-points ITFT are very similar as the transform results for the original 8-points DFT. Also, we note that input $\mathbf{x}_2$ is the combination of a low frequency component and a high frequency component. As these 2 components can be separated by both the original DFT and the IDFT. So the IDFT can also be used for the filter design.

Then, we see the displacement property of the IDFT. Here we use the displacement of input $x_2[n]$ (see Eq. (19)) as the input:

$x_3[n] = x2[((n+1))_8]$ $x_4[n] = x_2[((n+2))_8]$ (20)

And we plot $x_2[n]$, $x_3[n]$, $x_4[n]$ in Fig. 4(a), 4(c), 4(e). Then we do the 8-points ITFT for $x_2[n]$, $x_3[n]$, $x_4[n]$, and plot the amplitudes of the normalized results in Fig. 4(b), 4(d), 4(f).

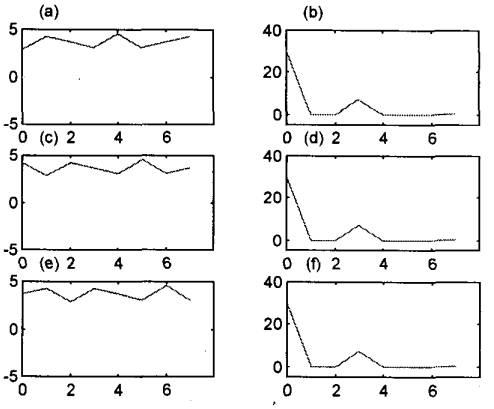

Fig. 4 The space-invariant property for the ITFT. (a) $x_2[n]$, (b) $|X_2[m]|$, (c) $x_3[n]$, (d) $|X_3[m]|$, (e) $x_4[n]$, (f) $|X_4[m]|$, where $X_2(m)$, $X_3(m)$, $X_4(m)$ are the ITFT of $x_2(n)$, $x_3(n)$, $x4(n)$.

We find, for the ITFT, the displacement property also keeps. The signal after displacement will have the same transform amplitude as the original signal. When we compare the phase, there is an interesting phenomenon. We find, if $X_2(m)$, $X_3(m)$, $X_4(m)$ are the transform results of $x_2(n)$, $x_3(n)$, $x4(n)$, then

angle($\mathbf{X}_3$)–angle($\mathbf{X}_2$) = [0, -45, -90, -135, 0, 135, -270, -315]

angle($\mathbf{X}_4$)–angle($\mathbf{X}_2$) = [0, -90, -180, -270, 0, -90, -180, -270]

This is exactly the same as the original DFT.

V. CONCLUSION

In Sec. 3, we have used the method introduced in Sec. 2 to derive the 8 points integer Fourier transform (8 points ITFT). In fact, we can also derive the ITFT for other number of points (such as the 6-points ITFT). We can also derive other types of integer transforms, such as the integer cosine, Hartley transforms.

For the integer transform, we can replace all the floating-points multiplication operations with the fixed-points multiplication operations, so the integer transform is very efficient. If the datum we want to process are all integer number, using the integer transforms is especially efficient. The concept of the integer transform is very new, and there are only a few researches about it. Because they are convenient for implementation, and almost remain the quality of original transform, so they can compete with the original discrete transform in many applications. We believe the integer transform will be very popular in the future.

REFERENCE

[1] W. K. Cham, *IEE Proceeding*, v. 136, n. 4, p 276-282, 1989

[2] A. V. Oppenheim, and R. W. Schafer, "*Discrete-Time Signal Processing*", Prentice-Hall, Inc. , 1989

[3] T. C. J. Pang, C. S. O. Choy, C. F. Chan, and W. K. Cham, *IEEE Trans. Circuits Syst. for Viedo Tech.*, v. 9, n. 6, p. 856-860, 1999