ONE DIMENSIONAL CYCLIC CONVOLUTION ALGORITHMS WITH MINIMAL MULTIPLICATIVE COMPLEXITY

Abraham H. Diaz-Perez
Electrical and Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico,
San Juan PR. 00919-2017
E-mail: adiaz@pupr.edu

Domingo Rodriguez
Electrical and Computer Engineering Department
University of Puerto Rico,
Mayagüez PR. 00681-5000
E-mail: domingo@ece.uprm.edu

ABSTRACT

This document present an enhancement algorithm for one dimensional cyclic convolution based on the Minimal Multiplicative Complexity Theorem proposed by Winograd. Particularly, this work focuses on the arithmetic complexity of the matrix-vector product when this represents polynomial multiplication module the polynomial $u^N - 1$, where N the polynomial length, is a power of 2. The proposed algorithms are compared with the algorithms make use of the Chinese remainder Theorem and it is shown why the formers are more efficient than the latter in terms of calculation steps. The algorithms are also compared with those that use the Fast Fourier Transform to carry out cyclic convolution operation, showing the advantages of the suggested approach and expressing possible improvements in order to perform the cyclic convolution computation in the least amount of time.

1. INTRODUCTION

In digital signal processing, the design of fast and computationally efficient algorithms has been a major focus of research activity. The objective, in most cases, is the design of algorithms and their respective implementation in a manner so as to perform the required computations in the least amount of time. In order to achieve this goal, parallel processing has also received a lot attention in the research community [1].

In the present work the Chinese Remainder Theorem is used for the formulation of new and computationally efficient algorithms for the matrix-vector products when they represent cyclic convolution operations. We will also show the mapping of the resulting operations onto a signal flow diagram that imply parallel computation.

2. THEORETICAL FRAMEWORK

2.1 Periodic or Cyclic Convolution

The periodic or cyclic convolution of two signals denoted by $y[n] = x[n] \otimes_N h[n]$, also of fundamental period N is:

$$y[n] = \sum_{m=0}^{N-1} h[m].x[\langle n-m \rangle_N] \;;\; n = 0,1,2...N-1$$

Note: The operator $\langle P \rangle_N$, denotes P modulo N, and it is the residue of P when is divided by N, or $\langle P \rangle_N = residue(P/N)$.

Associated with a N-point sequence $y[n]$ there exist a $(N-1)$-th degree polynomial in the indeterminate u:

$$y(u) = y_0 + y_1 u + ... + y_{N-1} u^{N-1}$$

Then, the cyclic convolution can be represented as a multiplication of polynomials modulo a monic polynomial:

$$y(u) = \langle x(u)h(u) \rangle_{(u^N - 1)}$$

The limitation of our algorithms to cyclic convolutions is not a serious problem since it is possible to express the aperiodic convolution (normal polynomial multiplication) as a cyclic convolution by a well known process described in [2]:

Consider the expression for an N-point cyclic convolution

$$y[n] = \sum_{m=0}^{N-1} h[m].x[\langle n-m \rangle_N]; \;\; n = 0,1,2...N-1$$

In matrix form, column major representation it can be written as: $\mathbf{y} = \mathbf{X} \cdot \mathbf{h}$

$$\begin{bmatrix} y[0] \\ y[1] \\ . \\ . \\ . \\ y[N-1] \end{bmatrix} = \begin{bmatrix} x[0] & x[N-1] & . & . & . & x[1] \\ x[1] & x[0] & . & . & . & x[2] \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x[N-1] & x[N-2] & . & . & . & x[0] \end{bmatrix} \cdot \begin{bmatrix} h[0] \\ h[1] \\ . \\ . \\ . \\ h[N-1] \end{bmatrix}$$

The matrix in the cyclic convolution is named *circulant* [3].

The *cyclic convolution property* relates the convolution $y[n]$ of two periodic discrete signals $x[n]$ and $h[n]$ by means of their DFT in the following manner:

$$Y[k] = X[k] \cdot H[k] \quad k = 0,1,..., N-1$$

Where $Y[k], H[k]$ and $X[k]$ are the DFT's of $y[n]$, $h[n]$ and $x[n]$ respectively, and $X[k] \cdot H[k]$ represents their Hadamard product.

A different way to determine the cyclic convolution is:

$$y[n] = -\frac{1}{N} \sum_{k=0}^{N-1} H[k] \cdot X[k] \cdot W^{-nk} \quad n = 0,1,2...N-1$$

$H[k]$ and $X[k]$ can be computed in parallel and then the N products $H[k] \cdot X[k]$ should be computed.

2.2 Winograd's Minimal Complexity Theorem.

This theorem will serve as basis to obtain the lower bound in the number of multiplications necessary to obtain the polynomial product.

Let F be some polynomial field. Let $x(u)$ and $h(u)$ be two polynomials of degree N defined on that field; then:

$$y(u) = \langle x(u) \cdot h(u) \rangle_{p(u)}$$

Requires at least $2N-k$ multiplications where k is the number of irreducible factors of $p(u)$ over the field F [2]. If $p(u)$ is prime, then k is 1. If $p(u) = (u^N - 1)$ (as cyclic convolution), then $k = N$ and the minimal number of multiplications is $2N - k = 2N - N = N$.

2.3 The Chinese remainder Theorem.

Optimal algorithms for one dimensional convolution have been constructed by Winograd [4] by means of the Chinese Remainder Theorem. Taken into account the expression:

$$y(u) = \langle x(u) \cdot h(u) \rangle_{p(u)}$$

with

$$p(u) = \prod_{i=1}^{k} p_i(u)$$

The polynomial product can be reduced to the following product of polynomial of smaller degree:

$y_i(u) = \langle x_i(u) \cdot h_i(u) \rangle_{p_i(u)}$, and the total product can be reconstructed by using of the Chinese Remainder Theorem (CRT) [7] by:

$$y(u) = \sum_{i=1}^{k} \langle y_i(u) \cdot R_i(u) \rangle_{p(u)}$$

Where the polynomials $R_i(u)$ are defined as:

$$R_i(u) = 1 \bmod p_i(u),$$
$$0 \bmod p_j(u),\ i \neq j$$

In [4] it is shown that the products $y_i(u)$ are disjoint. It is also proven that if optimal algorithms for the products $y_i(u)$ exist, then the algorithm together with the polynomial reductions module $p_i(u)$ and the Chinese Remainder Theorem reconstruction form an overall optimal algorithm for the computation of the convolution [7].

3. ALGORITHM DEVELOPMENT

This section show the process to obtain recursive algorithms in order to perform the polynomial product of length N modulo the polynomial $p(u) = (u^N - 1)$ when N is a power of 2. We will use the matrix representation of this product to take advantage of the spatial structure of the circulant matrix and the roots of unity as a tool to obtain the lower bound in the number of multiplications established by the Winograd theorem.

Let $\mathbf{y} = \mathbf{X} \cdot \mathbf{h}$, where $N = 2$, and $N - 1 = 1$ is the degree of the polynomials, the polynomial product module $(u^2 - 1)$ can be represented in matrix form as:

$$\mathbf{y} = \begin{bmatrix} x_0 & x_1 \\ x_1 & x_0 \end{bmatrix} \cdot \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} x_0 h_0 + x_1 h_1 \\ x_0 h_1 + x_1 h_0 \end{bmatrix} \qquad (1)$$

The straightforward computation is done by means of 4 multiplications and 2 sums. Winograd [4], [5] show that for computing this matrix-vector product, the following algorithm can be used with only 2 multiplications and 6 sums:

$$\mathbf{y} = \begin{bmatrix} x_0 & x_1 \\ x_1 & x_0 \end{bmatrix} \cdot \begin{bmatrix} h_0 \\ h_1 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 \\ m_1 - m_2 \end{bmatrix}, \quad \text{where:}$$

$$m_1 = \frac{1}{2}(x_0 + x_1)(h_0 + h_1);\ \ m_2 = \frac{1}{2}(x_0 - x_1)(h_0 - h_1) \qquad (2)$$

We are interested in reducing the algorithm's complexity in terms of multiplications, therefore we cannot take into consideration multiplications by $\frac{1}{2}$, 1 and -1. They are elements of our field of constants or ground set G, that will be in the remain the field of the complex number [4], [5].

We can observe the followings relations; the constant employed in this algorithm are the roots of unit of polynomial $u^N - 1$ in our case $u^2 - 1$ (1 and -1), and the value $\frac{1}{2}$ is simply $\frac{1}{N}$. The relation between this algorithm for performing cyclic convolution and the approach that uses the DFT is now evident, the constant used in both algorithms are the same.

The following figure shows, through a numerical example, how the first algorithm can be mapped into a parallel hardware structure:

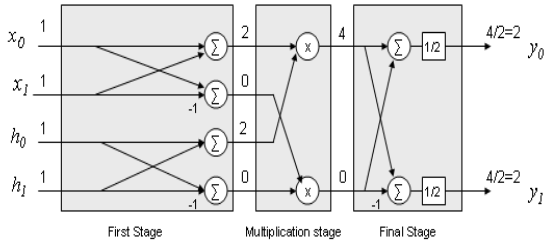

Figure 1. Signal flow diagram for cyclic convolution N=2.

We can appreciate in figure 1 three stages in our algorithm for $N = 2$. The first stage can be associated with the discrete Fourier transform. Actually, the values of the coefficients at the output of this stage are the same that those obtained by applying the discrete Fourier transform. The multiplication stage can be associated with the Hadamard multiplication of the two transformed sequences, and the last stage can be related to the inverse transform. The number of stage of our algorithm is given by the order of the sequences, in this case, for $N = 2^1$, the algorithm shows $S_1 = \log_2(N) = 1$ multiplication stages by our field of constants or ground set G (the roots of the polynomial $u^2 - 1$), and $S_2 = \log_2(N) = 1$ multiplications stages by the roots of the polynomial $u^2 - 1$. We only need a stage to multiply the sequences and it is of size N, which is the limit established by the Winograd theorem.

Now, we want to increase the order of the polynomials to the next power of two say, $N = 2^2 = 4$:

$$\mathbf{X} = \left[\begin{array}{cc|cc} x_0 & x_3 & x_2 & x_1 \\ x_1 & x_0 & x_3 & x_2 \\ \hline x_2 & x_1 & x_0 & x_3 \\ x_3 & x_2 & x_1 & x_0 \end{array}\right], \text{ and } \mathbf{h} = \left[\begin{array}{c} h_0 \\ h_1 \\ \hline h_2 \\ h_3 \end{array}\right], \qquad (3)$$

We can represent the matrix $\mathbf{X}$ and the vector $\mathbf{h}$ as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X_0} & \mathbf{X_1} \\ \mathbf{X_1} & \mathbf{X_0} \end{bmatrix}, \text{ and } \mathbf{h} = \begin{bmatrix} \mathbf{h_0} \\ \mathbf{h_1} \end{bmatrix}, \text{ then}$$
$$\mathbf{y} = \mathbf{Xh} = \begin{bmatrix} \mathbf{X_0 h_0} + \mathbf{X_1 h_1} \\ \mathbf{X_0 h_1} + \mathbf{X_1 h_0} \end{bmatrix} \qquad (4)$$

We can use the same algorithm employed in (2) to calculate vector $\mathbf{y}$.

$$\mathbf{y} = \begin{bmatrix} \mathbf{y_0} \\ \mathbf{y_1} \end{bmatrix} = \begin{bmatrix} \mathbf{X_0} & \mathbf{X_1} \\ \mathbf{X_1} & \mathbf{X_0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{h_0} \\ \mathbf{h_1} \end{bmatrix} = \begin{bmatrix} \mathbf{m_1} + \mathbf{m_2} \\ \mathbf{m_1} - \mathbf{m_2} \end{bmatrix}, \text{where:}$$
$$\mathbf{m_1} = \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{X_0} + \mathbf{X_1})(\mathbf{h_0} + \mathbf{h_1});\ \mathbf{m_2} = \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{X_0} - \mathbf{X_1})(\mathbf{h_0} - \mathbf{h_1}) \qquad (5)$$

The vectors $\mathbf{m_1}$ and $\mathbf{m_2}$ are found by:

For $\mathbf{m_1}$ we will call

$$\begin{bmatrix} x_0' & x_1' \\ x_1' & x_0' \end{bmatrix} = \begin{bmatrix} x_0 + x_1 & x_3 + x_1 \\ x_1 + x_3 & x_0 + x_2 \end{bmatrix}, \text{then}$$

$$\begin{bmatrix} h_0' \\ h_1' \end{bmatrix} = \begin{bmatrix} h_0 + h_2 \\ h_1 + h_3 \end{bmatrix}, \text{where we have 4 sums.} \quad (6)$$

Now, we multiply t o obtain $\mathbf{m_1}$

$$\mathbf{m_1} = \frac{1}{2}\begin{bmatrix} x_0' & x_2' \\ x_1' & x_0' \end{bmatrix}.\begin{bmatrix} h_0' \\ h_1' \end{bmatrix}$$

It has the same properties than (2), thus, it can be computed using 2 multiplications and 6 sums. For $\mathbf{m_2}$, we proceed similarly:

$$\begin{bmatrix} x_2' & -x_3' \\ x_3' & x_2' \end{bmatrix} = \begin{bmatrix} x_0 - x_2 & x_3 - x_1 \\ x_1 - x_3 & x_0 - x_2 \end{bmatrix}, \text{then}$$

$$\begin{bmatrix} h_2' \\ h_3' \end{bmatrix} = \begin{bmatrix} h_0 - h_2 \\ h_1 - h_3 \end{bmatrix}, \text{where we have 4 additional sums.} \quad (7)$$

Now, we multiply to obtain $\mathbf{m_2}$

$$\mathbf{m_2} = \frac{1}{2}\begin{bmatrix} x_2' & -x_3' \\ x_3' & x_2' \end{bmatrix}.\begin{bmatrix} h_2' \\ h_3' \end{bmatrix}$$

Here a different structure appears, however, it is closely related to block circulants. In order to solve this multiplication we will use a new algorithm, given to us by Winograd:

$$\mathbf{m_2} = \frac{1}{2}\begin{bmatrix} x_2' & -x_3' \\ x_3' & x_2' \end{bmatrix}.\begin{bmatrix} h_2' \\ h_3' \end{bmatrix}; \mathbf{m_2} = \frac{1}{2}\begin{bmatrix} (r_1 + r_2) \\ (r_1 - r_2)/i \end{bmatrix}, \text{where}$$

$$r_1 = \frac{1}{2}(x_2' + ix_3')(h_2' + ih_3'); \; r_2 = \frac{1}{2}(x_2' - ix_3')(h_2' - ih_3') \quad (8)$$

The computation of the matrix sums $(\mathbf{X_0} + \mathbf{X_1})$, and $(\mathbf{X_0} - \mathbf{X_1})$ are done using only two sums. The same occur with the computation of vectors sums $(\mathbf{h_0} + \mathbf{h_1})$ and $(\mathbf{h_0} - \mathbf{h_1})$. We can see the multiplication by the roots $1, -1, i$ and $-i$.

The vectors sums to calculate the general results are:

$$\mathbf{v} = \begin{bmatrix} \mathbf{v_1} \\ \mathbf{v_2} \end{bmatrix} = \begin{bmatrix} \mathbf{m_1} + \mathbf{m_2} \\ \mathbf{m_1} - \mathbf{m_2} \end{bmatrix}, \text{calling:}$$

$$\mathbf{m_1} = \begin{bmatrix} x_0'' \\ x_1'' \end{bmatrix} = \frac{1}{2}\begin{bmatrix} x_0' & x_1' \\ x_1' & x_0' \end{bmatrix}.\begin{bmatrix} h_0' \\ h_1' \end{bmatrix};$$

$$\mathbf{m_2} = \begin{bmatrix} x_2'' \\ x_3'' \end{bmatrix} = \frac{1}{2}\begin{bmatrix} x_2' & -x_3' \\ x_3' & x_2' \end{bmatrix}.\begin{bmatrix} h_2' \\ h_3' \end{bmatrix}, \text{Then} \quad (9)$$

$$\mathbf{y} = \begin{bmatrix} \mathbf{y_0} \\ \mathbf{y_1} \end{bmatrix} = \begin{bmatrix} \mathbf{m_1} + \mathbf{m_2} \\ \mathbf{m_1} - \mathbf{m_2} \end{bmatrix} = \begin{bmatrix} x_0'' + x_2'' \\ x_1'' + x_3'' \\ x_0'' - x_2'' \\ x_1'' - x_3'' \end{bmatrix}$$

is computed by means of 4 additional sums.

The constant multiplication by the value $\frac{1}{2}$ was realized two times, so we can use only the constant $\frac{1}{4}$ that is none other than $\frac{1}{N}$, the other constants employed are the roots of $u^4 - 1$. These outcomes can be observed in the figure 2.

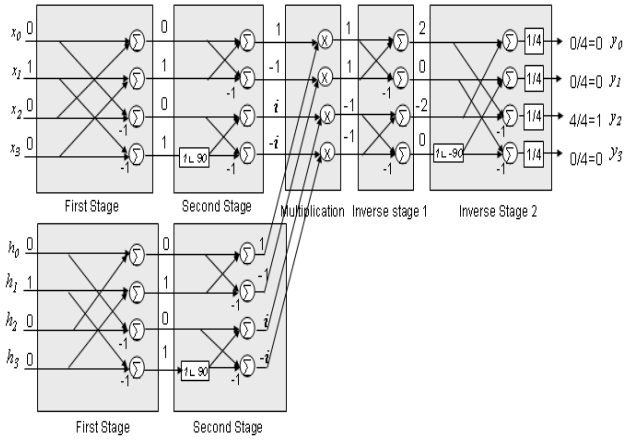

Figure 2. Signal flow diagram for cyclic convolution N=4.

We can note in figure 2 the stages of our algorithm. For $N = 2^2$, the algorithm show $S_1 = \log_2 N = 2$ stages of multiplication by the roots of the polynomial $u^4 - 1$. We can observe also $S_2 = \log_2 N = 2$ stages that are similar to the process of inverse Fourier transform, and one multiplication stage of the "transformed" sequences of size $N = 4$. The total numbers of operation are 4 multiplications and 24 sums. A process similar to the one described above is done with the roots of the polynomial $u^N - 1$, for convolutions of order $N = 2^s$. Since the generalization is straightforward there is no need to follow with a detailed description. The general algorithm uses a field of constants that are organized according to its use in the mapping of the convolution process. The following figure shows an easy way to find the constants for different sequences orders.

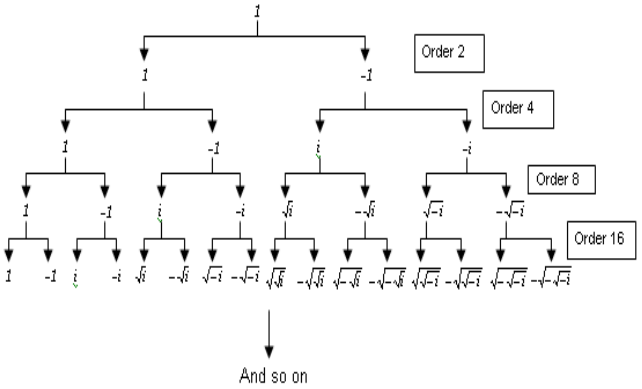

Figure 3. Roots of unit for different sequences orders.

Now we want to show the mathematical foundation of our algorithm by means of an example; consider $y(u) = \langle x(u)h(u) \rangle_{(u^4-1)}$ for the polynomials:

$$x(u) = x_0 + x_1u + x_2u^2 + x_3u^3 \;\; h(u) = h_0 + h_1u + h_2u^2 + h_3u^3$$

$$y(u) = x_0h_0 + x_3h_1 + x_2h_2 + x_1h_3 +$$

$$(x_1h_0 + x_0h_1 + x_3h_2 + x_2h_3)u +$$

$$(x_2h_0 + x_1h_1 + x_0h_2 + x_3h_3)u^2 +$$
$$(x_3h_0 + x_2h_1 + x_1h_2 + x_0h_3)u^3$$

The above product can be computed by definition using 16 multiplications and 12 sums.

By means of the Chinese remainder theorem this polynomial product can be realized following three steps:

1) We will divide the two polynomials by the roots of the monic polynomial $(u^4 - 1)$ and obtain the remainders:

$(x_0 + x_1 + x_2 + x_3)$ and $(h_0 + h_1 + h_2 + h_3)$ for $(u-1)$

$(x_0 - x_1 + x_2 - x_3)$ and $(h_0 - h_1 + h_2 - h_3)$ for $(u+1)$

$(x_0 + ix_1 - x_2 - ix_3)$ and $(h_0 + ih_1 - h_2 - ih_3)$ for $(u-i)$

$(x_0 - ix_1 - x_2 + ix_3)$ and $(h_0 - ih_1 - h_2 + ih_3)$ for $(u+i)$

2) Now, by multiplying those remainders we can obtain:

$$m_1 = (x_0 + x_1 + x_2 + x_3)\ (h_0 + h_1 + h_2 + h_3)/2$$
$$m_2 = (x_0 - x_1 + x_2 - x_3)\ (h_0 - h_1 + h_2 - h_3)/2$$
$$m_3 = (x_0 + ix_1 - x_2 - ix_3)\ (h_0 + ih_1 - h_2 - ih_3)/2$$
$$m_4 = (x_0 - ix_1 - x_2 + ix_3)\ (h_0 - ih_1 - h_2 + ih_3)/2$$

3) By means of a linear combination we obtain:

$$y_0 = (m_1 + m_2 + m_3 + m_4)/2$$
$$y_1 = ((m_1 - m_2) + (m_3 - m_4))/2$$
$$y_2 = ((m_1 + m_2) - (m_3 + m_4))/2$$
$$y_3 = ((m_1 - m_2) - (m_3 - m_4))/2$$

The big advantage of the matrix representation proposed in this paper is that it is not necessary to calculate the polynomials remainders. Other minimal multiplicative complexity algorithms based on CRT such as Winograd's algorithms need to realize this decomposition, for this reason, in our case the whole algorithm complexity is reduced.

Is very interesting to compare the new algorithm with those that use the DFT for convolution; we will compare the pre-Hadamard stages in the figures 3 and 4 for $N = 8$. A comparison between theses figures shows a high correlation in the mapping of the two algorithms to a signal flow diagram. Even though their signal flow diagrams are slightly different, the two figures show the same number of computations ($O(N \cdot \log(N))$). The new advance of our algorithm however, is that it does not need the previous stage of bit reversal operation, for this reason this will improve the time necessary to realize the convolution operation with respect to the FFT-2 and FFT-4 approaches.

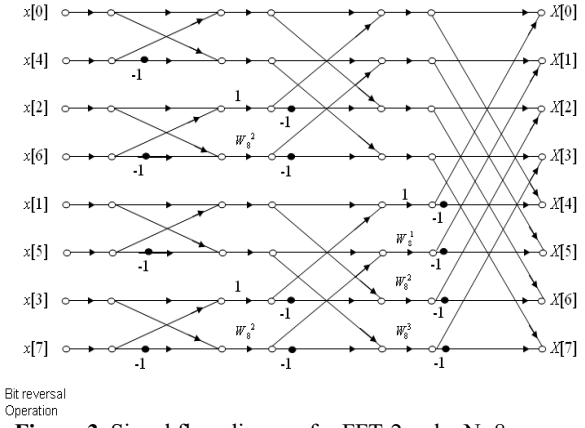

Figure 3. Signal flow diagram for FFT-2 order N=8.

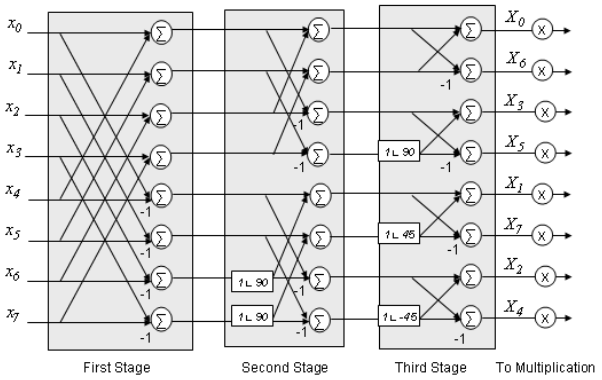

Figure 4. Signal flow diagram for the new algorithm order N=8.

4. CONCLUSION

This work shows a novel algorithm for the fast computation of the product of a circulant matrix with a vector, based in the Chinese Remainder Theorem. The principal goal was to obtain a recursive algorithm, easy to implement, with the advantage of not needing to realize the polynomial divisions by the roots of unit in order to obtain less number of multiplications. The algorithm obtained is limited to polynomials length a power of 2, and its flow diagram suggests the possible use of Kronecker or Tensor products as a tool to improve their performance by means of parallel processing.

5. AKNOWLEDGMENT

The authors would like to thank to the Dr Shmuel Winograd by their helpful suggestions and comments to the development of the algorithms.

6. REFERENCES

[1] H. Krishna, B. Krishna, K. Y. Lin, J. D. Sun, *Computational number theory and digital signal processing* (CRC Boca Raton, Florida 1994).

[2] D.G. Myers, *Efficient convolution and Fourier transform techniques* (Prentice Hall of Australia 1990).

[3] J. Davis, *Circulants matrices* (John Wiley, New York, 1979).

[4] S. Winograd, *Arithmetic complexity of computations* (Society for Industrial and Applied Mathematics, 1980).

[5] M. Heideman, *Multiplicative complexity, convolution, and the DFT* (Springer Verlag, New York 1988)

[6] J. McClellan and C. Rader, Number Theory in Digital Signal Processing. Englewood Cliffs, NJ: Prentice Hall 1979.

[7] J. Cooley, Some applications of computational complexity Theory to Digital Signal Processing. *1981 Joint Automatic Contr. Conf.* University of Virginia, June 17-19 1981.