Asian Journal of Information Technology 5(5) : 476-479, 2006

A Fast Algorithm for the Computation of Radix-4 Two-Dimensional Fourier Transform and its Parallel Impementation

Djamel Chikouche, Nourredine Amardjia, Nabil Khenfer, Rais El'hadi Bekka and Fairouz Belilita
Department of Electronics, Faculty of Engineering, University of Sétif, 19000 Sétif, Algeria

Abstract: In this study, we propose a fast algorithm for computing radix-4 two-dimensional Fourier transform that is suitable for implementation on a parallel architecture. Our algorithm is derived in this paper from a Cooley decimation-in-time algorithm by using an appropriate indexing process. It is proved that the number of multiplications necessary to compute our proposed algorithm is significantly reduced while the number of additions remains almost identical to that of Cooley 2D FFT's. Comparison results show the good performance of the proposed 2D FFT algorithm against the row-column FFT transform.

Key words: Two-dimensional FFT, decimation-in-time algorithm, indexing process, radix-4, computational complexity, parallel architecture

INTRODUCTION

In recent years, there has been a growing interest regarding the development of efficient computational algorithms for the discrete Fourier transform[1-11]. These algorithms must be capable of matching the advantages offered by the high speed digital computer and the rapid advances in VLSI technology[12-13]. The evaluation of the 2D DFT is based on three widely used classes of FFTs. There are the row-column, the vector radix and the polynomial transform FFT[4,5].

In this study, we propose an algorithm for the computation of the radix-4 two-dimensional Fourier transform presented in a simple matrix form which allows straight forward VLSI implementation. We will present first the conventional Cooley 2D decimation-in-time algorithm and the ideas proposed in this paper to improve its computational complexity. Second, we derive our proposed algorithm on the basis of an improvement in the indexing process involved. Third we analyze the computational complexity and relations of our algorithm against the 2D FFT conventional algorithms and propose an implementation on a parallel architecture.

THE 2D DISCRETE FOURIER TRANSFORM

The two-dimensional DFT transform (2D DFT) of a sequence $x(k_1,,k_2)$ is defined as[1]:

$$X(n_1,n_2)=\sum_{k_1=0}^{N-1}\cdot\sum_{k_2=0}^{N-1}x(k_1,k_2)W_N^{\sum_{j=1}^{2}n_jk_j} \quad (1)$$

where $n_j\in[0,N-1]$ and $W_N=\exp(-j2\pi/N)$
or in a matrix form as[4]:

$$X=W_N^2\,x \quad (2)$$

The basic matrix W_N^2 ($N^2\times N^2$) is generated by a Kronecker product of the matrix W_N[6].

$$W_N^2=W_N\otimes W_N \quad (3)$$

The direct computation of N^2 points 2D DFT of equation *(2)* requires: N^4 complex multiplications, $N^2(N^2-1)$ complex additions and $2N^2$ loads and stores.

TRADITIONAL TECHNIQUE FOR THE COMPUTATION OF THE 2D DFT

The usual way to compute this 2D DFT N points is by performing the computation of 2N distinct 1D DFT N points[4,8].

$$\begin{aligned}X_1(n_1,k_2)&=\sum_{k_1=0}^{N-1}x(k_1,k_2)W_N^{n_1k_1}\\ X_2(n_1,n_2)&=\sum_{k_2=0}^{N-1}X_1(n_1,k_2)W_N^{n_2k_2}\end{aligned} \quad (4)$$

If a radix-4 FFT is used, then the number of complex multiplications necessary for the entire 2D DFT is

$$\frac{3}{4}N^2\left(\log_2 N-2\right)$$

and the number of complex additions is.

$$2N^2\log_2 N$$

Corresponding Author: Djamel Chikouche, Electronics Department, Faculty of Engineering, University of Sétif, 19000 Sétif, Algeria

We will see that our constructed algorithm reduces notably this considerable amount of computations.

THE PROPOSED 2D DFT ALGORITHM

Let $N=4^r$, where 4 is the radix of the 2D DFT, r 4-ary digits are necessary to represent the indices n_j and k_j in Eq (1).

Using decimation-in-time Cooley-Tukey mapping to equ. (1), we get the intermediate steps in the two-dimensional case from the recursive relation Eq (5):

$$X_i(m_{i1},m_{i2})=\sum_{(k_1)_{r-i}=0}^{3}\sum_{(k_2)_{r-i}=0}^{3}X_{i-1}(m_{(i-1)1},m_{(i-1)2})\times W_N^{\sum_{j=1}^{2}(k_j)_{r-i}C_{i-1}(m_{ij})} \quad I=1,2,\ldots,r \tag{5}$$

Then the i^{th} intermediate step is obtained when the i^{th} sums over the $(k_j)_{r-i}$ are performed.
The index $m_{(i-1)j}$ is defined as:
$m_{(i-1)j}=(n_j)_0\ldots..(n_j)_{i-2}(k_j)_{r-i}(k_j)_{r-(i+1)}\ldots..(k_j)_0$

$$C_{i-1}(m_{ij})=\left[\sum_{h=0}^{i-1}4^h(m_{ij})_{r-1-h}\right]4_{r-i} \tag{6}$$

and the index m_{ij} can be expressed as:
$m_{ij}=(n_j)_0\ldots..(n_j)_{i-2}(n_j)_{i-1}(k_j)_{r-(i+1)}\ldots..(k_j)_0$
$=(m_{ij})_{r-1}\ldots..(m_{ij})_{r-(i-1)}(m_{ij})_{r-i}(m_{ij})_{r-(i+1)}\ldots..(m_{ij})_0$

The recursive relations (5) use $\log_4 N$ intermediate steps $X_1, X_2,\ldots..,X_r$ to obtain an entire spectrum. We can easily see that the j^{th} indices of two successive intermediate steps are different in one digit only. We use this observation in order to simplify the indexing process.

After some mathematical manipulations, the recursive equation that computes the intermediate steps can be expressed as follows:

$$V_i(m_1,m_2)=\sum_{k_1=0}^{3}\sum_{k_2=0}^{3}V_{i-1}(k_1,k_2)W_N^{\sum_{j=1}^{2}k_jC_{i-2}(p_j)}W_4^{\sum_{j=1}^{2}k_jm_j}$$

or in a matrix form:

$$V_i=W_4^{\,2}\,D_4^{\,2}V_{i-1} \tag{7}$$

where:

$$V_i(m)=V_i(m_1,m_2)=X_i(p_1+m_1 4^{r-i},\ p_2+m_2 4^{r-i}) \tag{8}$$
$$V_{i-1}(k)=V_{i-1}(k_1,k_2)=X_{i-1}(p_1+k_1 4^{r-i},p_2+k_2 4^{r-i}) \tag{9}$$
$$W_4^{\,2}=W_4\otimes W_4 \tag{10}$$

- with

$$C_{i-2}(p_j)=\left[\sum_{h=0}^{i-2}4^h(p_j)_{r-1-h}\right]4_{r-i} \tag{11}$$

and $C_{-1}(P_j)=0$ for:

- $p_j=0$ to N-1 except$\{\ p_j+m_j4^{r-i},/\ m_j=0,1,2,3\ \}$, $j=1,\ 2$.

The 2D DFT is derived from the last intermediate step (i=r) after a digit-reversal of its indices[10].
The matrix $W_4^{\,2}$ can be written as:

$$W_{4^2}=\prod_{j=1}^{2}\left(I_{4^{j-1}}\otimes W_4\otimes I_{4^{2-j}}\right) \tag{12}$$

By using the optimal factorization of W_4 and the tensor product properties, we get:

$$W_{4^2}=\prod_{j=1}^{2}\left(I_{4^{j-1}}\otimes Z_4\otimes I_{4^{2-j}}\right)\left(I_{4^{j-1}}\otimes S_4\otimes I_{4^{2-j}}\right)\times\left(I_{4^{j-j}}\otimes R_4\otimes I_{4^{2-j}}\right) \tag{13}$$

Where:

$$W_4=\begin{bmatrix}1&1&1&1\\1&-j&-1&j\\1&-1&1&-1\\1&j&-1&-j\end{bmatrix}=\begin{bmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-j\end{bmatrix}\begin{bmatrix}1&0&1&0\\1&0&-1&0\\0&1&0&1\\0&1&0&-1\end{bmatrix}=Z_4S_4R_4 \tag{14}$$

This last factorization of the matrix $W_4^{\,2}$ contains only "0","±1" and "±j" elements. Thus it involves only 4×4^2 complex additions in the computation of equ. (7). Therefore, the total number of operations required to perform radix-4 2D DFT is:

$$4\times4^2\frac{N^2}{4^2}\log_4 N=2N^2\log_2 N$$

complex additions and

$$(4^2-1)\frac{N^2}{4^2}(\log_4 N-1)=\frac{(4^2-1)}{2\times4^2}N^2(\log_2 N-2)$$

$$=\frac{15}{32}N_2(\log_2 N-2)$$

complex multiplications

Table 1: Comparison between the proposed radix-4 2D FFT algorithm and traditional algorithms

		The traditional method based on	Proposed algorithm
	2D DFT	1D radix-4 FFT	radix-4 2D FFT
Number of complex additions	$N^2 (N^N - 1)$	$2N^2 \log_2 N$	$2N^2 \log_2 N$
Number of complex multipli-cations	N4	$\frac{3}{4} N^2 \log_2 \frac{N}{4}$	$\frac{15}{32} N^2 \log_2 \frac{N}{4}$

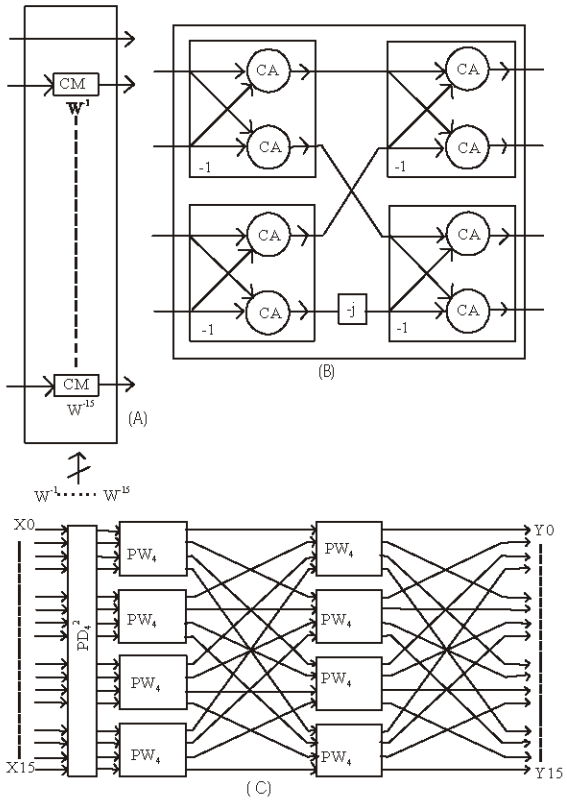

Fig. 1 (a): PD_4^2 : aD_4^2 processor with CM : Complex multiplication
(b): PW^4 : a W^4 processor with CA : a complex addition
(c): P_{FFTB4} : a $W_4^2 D_4^2$ processor

COMPARISON BETWEEN THE PROPOSED RADIX-4 FFT ALGORITHM AND TRADITIONAL ALGORITHMS

Table 1 shows that the number of multiplications necessary to compute the 2D DFT using our algorithm is significantly reduced while the number of additions, for most cases, remains at the same level as traditional Cooley-Tukey methods.

HARDWARE IMPLEMENTATION

From Equ. (7) we see that the radix-4 2D DFT is computed in a recursive manner and that the vector V_i at stage i is fully defined from and only from its predecessor V_{i-1} using a butterfy structure. The radix-4 2D DFT can then be computed in place, using a parallelized and pipelined implementation. The basic butterfly processor, in Fig. 1.(c), denoted PE_{FFTB4}, performs the $W_4^2 D_4^2$ operation where D_4^2 and W_4^2 functions are given in Figs. (Fig la,b) respectively, with CA and CM performing complex addition and multiplication and Wi representing the twiddle factors.

CONCLUSION

The proposed algorithm for computing the radix-4 2D DFT combines the advantages of the Cooley-Tukey method, the Kronecker product and an efficient indexing process to give an optimal radix-4 2D fast Fourier transform algorithm expressed in a simple matrix form. This has resulted in a substantial computational savings compared to standard row-column FFT algorithms. Furthermore, this matrix form of the algorithm can lead to a parallel implementation in a forward manner and provides the needed details to implement it in a computer program.

REFERENCES

1. Cooley, W. and J.W. Tukey, 1965. An algorithm for the machine calculation of complex fourier series, Math. Comput., pp: 297-301.
2. Sorensen, H.V. and C.S. Burrus, 1993. Efficient computation of the DFT with only a subset of input/output points, IEEE Trans. on Sig. Proc., pp: 1184-1200.
3. Hinshaw, W.S., *et al.*, 1983. An introduction to NMR imaging: From the Block Equation to the Imaging Equation, Proc. IEEE.
4. Gertner, I. and Shamash, 1987. VLSI Architectures for multidimensional fourier transform processing, IEEE Trans. on Computers, pp: 1265-1274.
5. Angelopoulos, G. and I. Pitas, 1991. Two-dimensional FFT algorithms on hypercube machines, Proc. Transputer Applications 91, Glasgow.
6. Granata, J., M. Conner and R. Tolimieri, 1992. The Tensor Product: a mathematical Programming language for FFTs and other fast DSP operations, IEEE SP Magazine, pp: 40-48.
7. Zapata, E.L., F.F. Rivera, I. Benavides, J.M. Carazo and R. Peskin, 1990. Multidimensional fast Fourier Transform into SIMD Hypercubes, IEEE proc., pp: 257-260.
8. Niemel, L.P.W. and R. Prasad, 1994. On the reduction of multidimensional DFT to separable DFT by smith normal Form theorem, European Trans. on Telecomm. and Related Techn., pp: 377.

9. Falkawski, B.J., 1994. Properties and ways of calculation of multi-polarity generalized walsh Transforms, IEEE Trans. on Circ. and Sys.-II: Analog and Digital Signal Processing, pp: 380-391.
10. Rius, J.M. and R. De Porrata-Doria, 1995. T bit-reversal rlgorithm, IEEE Trans. on Sig. Proc., pp: 991-994.
11. Chikouche, D., S. Bouguezel and R.E. Bekka, 2004. An efficient algorithm for the computation of the two-dimensional Fourier transform, Proc. of the 7th African Conference on Research in Computer Sci. CARI'04, Hammamet, Tunisia, pp: 43-50.
12. Hyesook, Lim and E. Earl and J.R. Swartzlander, 1999. Multidimensional Systolic Arrays for the implementation of Discrete Fourier Transforms, IEEE Trans. on Sig. Proc., N°. 5, pp: 1359-1370.
13. Yun-Nan Chang and K.K. Parhi, 2003. An efficient pipelined FFT architecture, IEEE Trans. on Circuits and systems II:Analog and Digital Signal Processing, Issue 6, pp: 322- 325.