Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Microprocessors and Microsystems 30 (2006) 184–188
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

Transform decomposition method of pruning the FHT algorithms

K.M.M. Prabhu[a,*], G. Ghurumuruhan[b]

[a] Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai 600 036, India
[b] Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30318, USA

Available online 4 January 2006

Abstract

In this paper, we have applied the transform decomposition (TD) technique of pruning the fast Fourier transform (FFT) flow graph to the fast Hartley transform (FHT) flow graph. We have shown that efficient pruning is possible when the number of output points is limited. Any arbitrary band of spectra can also be computed using the method proposed.

Keywords: Fast Fourier transform (FFT); Fast Hartley transform (FHT); Radix-2 FFT and FHT algorithms; Transform decomposition (TD); Index mapping

1. Introduction

The concept of pruning the fast algorithm was first introduced by Markel [1], and applied to the input of radix-2 decimation-in-frequency (DIF) FFT flow graph. Later, Skinner [2] showed that the application of Markel's pruning algorithm to the input of radix-2 decimation-in-time (DIT) FFT flow graph is more efficient. Subsequently, Sreenivas and Rao [3] combined Skinner's and Markel's pruning strategies to prune both the input and the output of the FFT flow graph. In 1993, Sorensen and Burrus [4] proposed a new method, called the transform decomposition (TD) method, for pruning the input as well as output of the FFT flow graph, which was proved to be much more efficient than the existing methods of pruning.

Narayanan and Prabhu [5] explored the possibility of pruning the input of the radix-2 DIF FHT flow graph, similar to the method followed by Markel. We shall call the algorithm they have proposed as NAP algorithm. Although the NAP algorithm is computationally efficient, we find that its programming complexity is quite high, since the structure of the pruned FHT flow graph is very different and much more difficult to handle than its FFT counterpart.

In this paper, we apply the TD method to prune the output of the FHT flow graph and calculate the reduction in the number of computations. The paper is organized as follows: Section 2 gives a brief review of the existing method of pruning the FHT flow graph. Section 3 introduces the TD method of pruning the FHT flow graph. Then, its computational complexity is analyzed and compared with the existing method. Section 4 gives the implementation details and Section 5 highlights the conclusions drawn.

* Corresponding author. Tel.: +91 44 2257 6410; fax: +91 44 2257 0509.
E-mail addresses: prabhu_kmm@hotmail.com (K.M.M. Prabhu), gurugene7@yahoo.com (G. Ghurumuruhan).

0141-9331/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2005.10.002

2. Pruning the FHT flow graph

2.1. Narayanan and Prabhu (NAP) pruning algorithm

For clarity, we briefly review the NAP method [5]. The pruned flow graph, unlike its FFT counterpart, does not repeat itself in the subsequent stages [1–3]. In the case of FFT, a block pruned at any stage gives rise to two similar blocks in the next stage. The new blocks thus formed are half the size of the previous blocks. However, such a compact structure does not exist in the case of FHT. This makes its analysis difficult [5].

In the NAP pruning algorithm, pruning the first stage gives rise to two blocks, each being half the size of its original. The difference between the FFT and FHT flow graphs is that, in the latter, the smaller blocks are not identical. In [5] they are labelled as type-A and type-B blocks. It had been further shown in [5] that:

(a) The block at stage 1 is of type-A.
(b) A type-A block when pruned gives rise to one type-A block and one type-B block.
(c) A type-B block when pruned gives rise to two type-B blocks.
(d) Stage L is the last stage that can be pruned, where 2^L is the number of output DHT points required.

The blocks obtained by pruning the last stage are slightly different in structure and they are labelled as type-C and type-D blocks [5]. The explanation of each block is clearly given in [5].

We define the efficiency of a pruning algorithm PR, as:

$$\eta_{\mathrm{PR}}(L) = 1 - \frac{N_{\mathrm{PR}}(L)}{N_{\mathrm{DIR}}},$$

where $N_{\mathrm{PR}}(L)$ is the number of multiplications needed by the pruning algorithm, given that the number of output points needed is 2^L and N_{DIR} is the number of multiplications needed by the algorithm without pruning.

The efficiency of the NAP algorithm is shown to be [5]:

$$\eta_{\mathrm{NAP}} = \frac{(\nu - L)(N + 2^{L+1} - 2) + 2^{L+1} - 2N + 2}{N\nu - 3N + 4}. \quad (1)$$

This is plotted in Fig. 1. A more detailed analysis of the NAP algorithm can be found in [5].

3. Transform decomposition

The idea of transform decomposition (TD) method was first introduced by Sorensen and Burrus [4] for pruning the FFT flow graphs. In this section, we describe the method of pruning the FHT flow graph (using the TD method), when the number of output points is limited. A 2D mapping on n $(0 \le n \le N-1)$ is defined as in [6]:

$$n = n_1 + N_1 n_2, \quad 0 \le n_1 \le N_1 - 1, \quad 0 \le n_2 \le N_2 - 1 \quad (2)$$

where $N = N_1 N_2$. This mapping is used to convert a one-dimensional sequence into a two-dimensional sequence. The discrete Hartley transform (DHT) of an N-point sequence, $h(n)$, is given by

$$H(k) = \sum_{n=0}^{N-1} h(n)\mathrm{cas}\left(\frac{2\pi kn}{N}\right) \quad (3)$$

where $\mathrm{cas}(x) = \cos(x) + \sin(x)$. Applying (2) in (3) we get,

$$H(k) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} h(n_1 + N_1 n_2)\mathrm{cas}\left(\frac{2\pi kn_1}{N} + \frac{2\pi kn_2}{N_2}\right). \quad (4)$$

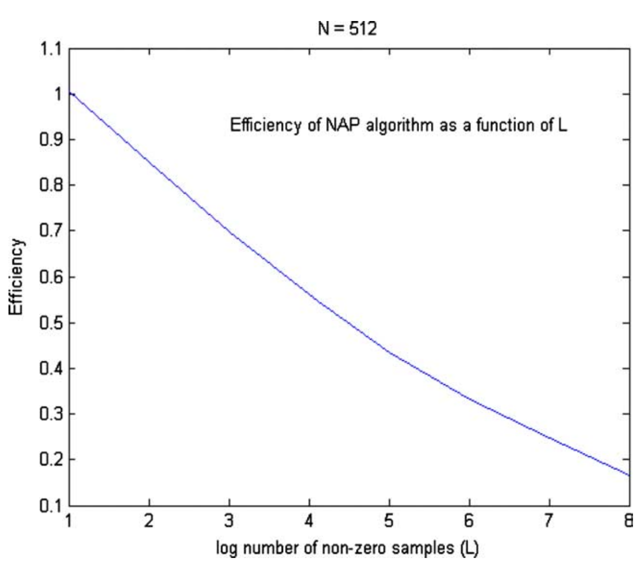

Fig. 1. Efficiency of the NAP pruning algorithm as a function of the number of non-zero (2^L).

By denoting $h(n_1 + N_1 n_2)$ as $\hat{h}(n_2;n_1)$, (4) can be rewritten as

$$H(k) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} \hat{h}(n_2;n_1)\left[\mathrm{cas}\left(\frac{2\pi kn_2}{N_2}\right)\cos\left(\frac{2\pi kn_1}{N}\right) + \mathrm{cas}\left(\frac{-2\pi kn_2}{N_2}\right)\sin\left(\frac{2\pi kn_1}{N}\right)\right] \quad (5),$$

where we have used the fact that $\mathrm{cas}(\alpha + \beta) = \mathrm{cas}(\alpha)\cos(\beta) + \mathrm{cas}(-\alpha)\sin(\beta)$.

Let us denote the N_2 point DHT of $\hat{h}(n_2;n_1)$ as $\hat{H}(k;n_1)$, i.e. let

$$\hat{H}(k;n_1) = \sum_{n_2=0}^{N_2-1} \hat{h}(n_2;n_1)\mathrm{cas}\left(\frac{2\pi kn_2}{N_2}\right), \quad 0 \le n_1 \le N_1 - 1, \quad (6)$$

$0 \le k \le N_2 - 1$.

Using (6), Eq. (5) can be rewritten as,

$$H(k) = \sum_{n_1=0}^{N_1-1}\left[\hat{H}((k)_{N_2};n_1)\cos\left(\frac{2\pi kn_1}{N}\right) + \hat{H}((-k)_{N_2};n_1)\sin\left(\frac{2\pi kn_1}{N}\right)\right], \quad 0 \le k \le N-1, \quad (7)$$

where $(k)_N$ denotes k (mod N).

This completes the derivation of the TD method. The transform decomposition (TD) method of pruning the FHT algorithm can be summarized as follows:

(1) Let $h(n)$ be an N-point sequence, where $N = N_1 N_2$. Divide $h(n)$ into N_1 sub-sequences, each of length N_2, using the mapping given in (2).
(2) Find the DHT of each of the N_1 smaller sequences using any efficient FHT algorithm.
(3) Combine the smaller size DHTs as per (7) to produce the required output points (See Fig. 1).

To show how this method reduces the number of multiplications in calculating a band of the DHT spectra, we carry out the computational complexity analysis of the TD method of pruning the FHT flow graph.

Let $C(N_2)$ be the number of multiplications needed to perform each of the N_2-point DHTs in step (2) of the TD method. Therefore, we require a total of

$$M_2 = N_1 C(N_2)$$

multiplications in step (2) of our algorithm. We also perform

$$M_3 = 2^{L+1}(N_1 - 2)$$

multiplications to compute 2^L values of $H(k)$ in step (3) of the algorithm. Hence, the total number of multiplications required to compute the DHT points using the TD method is given by (Fig. 2):

$$T_L(N) = M_2 + M_3 = N_1 C(N_2) + 2^{L+1}(N_1 - 2)$$

$$= N_1 C\left(\frac{N}{N_1}\right) + 2^{L+1}(N_1 - 2).$$