A NOTE ON PARALLEL AND PIPELINE COMPUTATION OF FAST UNITARY TRANSFORMS

by

Bernard J. Fino and V. Ralph Algazi

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

and

Electrical Engineering Department
University of California, Davis

ABSTRACT

This correspondence discusses the parallel and pipeline organization of fast unitary transforms algorithms such as the Fast Fourier Transform and points out the efficiency of a combined parallel-pipeline processor of a transform such as the Haar transform in which (2^n-1) hardware "butterflies" generate a transform of order 2^n every computation cycle.

This work was sponsored by National Aeronautic and ~~Space~~ Administration Grant NGR-05-003-538 and National Science Foundation Grant GK-37282.

8

Algorithms for all fast unitary transforms, such as the Fast Fourier transform (FFT), fast Walsh-Hadamard transform (FWT) and other fast unitary transform [1], require n stages of computation for transforms of order 2^n. Each stage of computation can be in turn decomposed into at most 2^{n-1} "butterflies" [2], each performing a rotation by a matrix of order 2. Some or all of the butterflies at one stage of computation can operate in parallel (see [3], [4] for FFT) and fast unitary transforms have thus a greater potential in applications with the development of low cost parallel circuitry. For example, we show in Fig. 1a the FFT Cooley-Tukey algorithm of order 4 with 2 butterflies in each of its 2 stages of computation. If τ seconds is the time required to perform a butterfly operation, each stage can be performed in τ seconds with the highest possible degree of parallelism which uses 2^{n-1} butterflies. Thus, a transform of order 2^n can be performed in $n\tau$ seconds as compared to $n2^{n-1}\tau$ seconds with sequential computation (which requires only one butterfly).

If a number of successive transforms have to be computed, it is possible to increase further the throughput rate with several transformers working simultaneously, each operating on a different input vector and each possibly at a different stage of computation (see [5] for FFT): this is generally referred to as a pipeline organization. Parallel and pipeline organizations can be combined conveniently with $n2^{n-1}$(at most) butterflies working in parallel and one transform of order 2^n is obtained every τ seconds on the average. Fig. 1b shows a possible organization of the FFT Cooley-Tukey algorithm of order 4. All stages of this pipeline algorithm are identical: the 2 first butterflies perform the first stage of Fig. 1a and the 2 last butterflies perform the second stage. The input vector is entered in the first 4 cells and its FFT transform obtained in the same cells after 2 cycles. This algorithm can be wired-in and will give the transform coefficients in any order but it requires a large amount of hardware and requires the access at its input of two sets of $n2^n$ storage cells.[1]

Some transforms, however, do not require 2^{n-1} butterflies at each stage of computation and then a pipeline algorithm can be implemented with much less hardware. We consider now in particular a pipeline algorithm for the Fast Haar Transform (FHT). Although less known, the FHT is closely related to the FWT [6], has a fast algorithm [7], is certainly a transform of interest for signal encoding [8], [9] and other applications [10]. A pipeline-parallel algorithm for the FHT requires only (2^n-1) butterflies and still produces a transform of order 2^n at every cycle. We show in Fig. 2a the Haar matrix of order 8 and in Fig. 2b a possible organization of the FHT of the same order. The number of butterflies decreases for successive stages and this is the property which can be exploited in a pipeline processor. In Fig. 3, we show a stage of a possible organization of the pipeline FHT of order 8.

Many other transforms can have similar pipeline algorithms with reduced amount of hardware: the Modified generalized discrete transforms [11], the WFH transforms [1], the Slant Haar transforms [12] and other generalized Slant transforms [13]. In all cases, the pipeline-parallel algorithm needed to perform a transform of order 2^n in one cycle is the total number of butterflies appearing in the flow diagram of the algorithm. By contrast, parallel processing requires the maximum number of butterflies needed at any stage.

REFERENCES

[1] B. J. Fino and V. R. Algazi, "A Unified Treatment of Discrete Unitary Transforms with a Fast Algorithm," submitted for publication.

[2] L. R. Rabiner, et al., "Terminology in Digital Signal Processing," IEEE Trans. on Audio and Electro-Acoust., Vol. AU-20, No. 5, pp. 322 - 337, December 1972.

[3] M. C. Pease, "An Adaptation of the Fast Fourier Transform for Parallel Processing," J. ACM, Vol. 15, No. 2, pp. 252 - 264, April 1968.

[4] B. Gold and T. Bially, "Parallelism in Fast Fourier Transform Hardware," IEEE Trans. Audio and Electo-Acoust., Vol. AU-21, No. 1, pp. 5 - 16, February 1973.

[5] M. J. Corinthios, "The Design of a Class of Fast Fourier Transform Computers," IEEE Trans. on Computers, Vol. C-20, No. 6, pp. 617 - 623, June 1971.

[6] B. J. Fino, "Relations Between Haar and Walsh-Hadamard Transforms," Proc. IEEE, Vol. 60, No. 5, pp. 647 - 648, May 1972.

[7] V. J. Rejchrt, "Signal Flowgraph and a Fortran Program for Haar-Fourier Transform," IEEE Trans. on Comput., Vol. C-21, No. 9, pp. 1026 - 1027, September 1972.

[8] B. J. Fino, "Etude expérimentale du codage d'images par les transformations de Haar et Hadamard complexe," Ann. Telecomm., tome 27, pp. 125 - 208, May-June 1972.

[9] W-H. Chen, Slant Transform Image Coding, Tech. Rep. 441, University of Southern California, May 1973.

[10] J. E. Shore, "On the Application of Haar Functions," IEEE Trans. on Comm., Vol. COM-21, No. 3, pp. 209 - 216, March 1973.

[11] K. R. Rao, N. Ahmed and R. B. Schultz, "A Class of Discrete Orthogonal Functions," to be published.

[12] B. J. Fino and V. R. Algazi, "Slant Haar Transform," to be published in Proc. IEEE.

[13] B. J. Fino and V. R. Algazi, "Generalized Slant Transforms," submitted for publication.

FOOTNOTE

[1]The computation can be also performed "in place" with $n2^n$ storage cells only followed by cyclic shifts by 2^n cells.

CAPTIONS

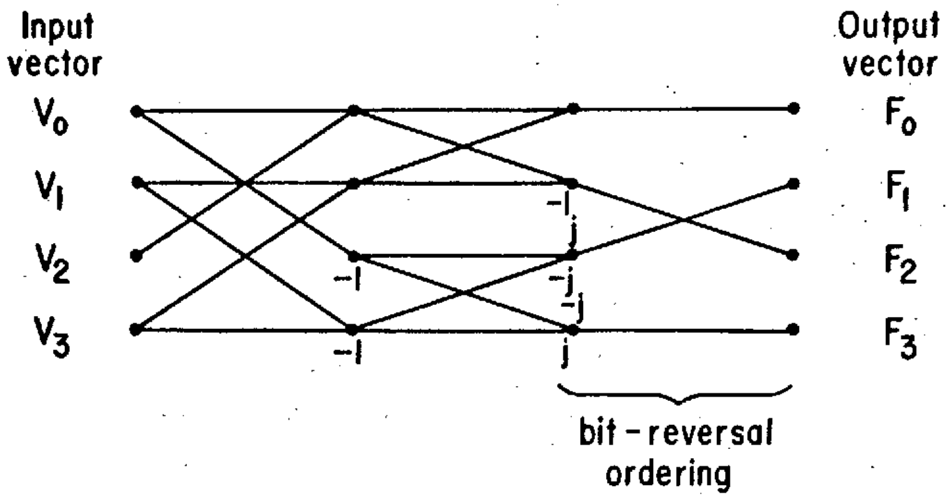
Fig. 1a : FFT Cooley-Tukey Algorithm of order 4

Fig. 1b : Pipeline FFT Cooley-Tukey Algorithm of order 4

Fig. 2a : Haar matrix of order 8

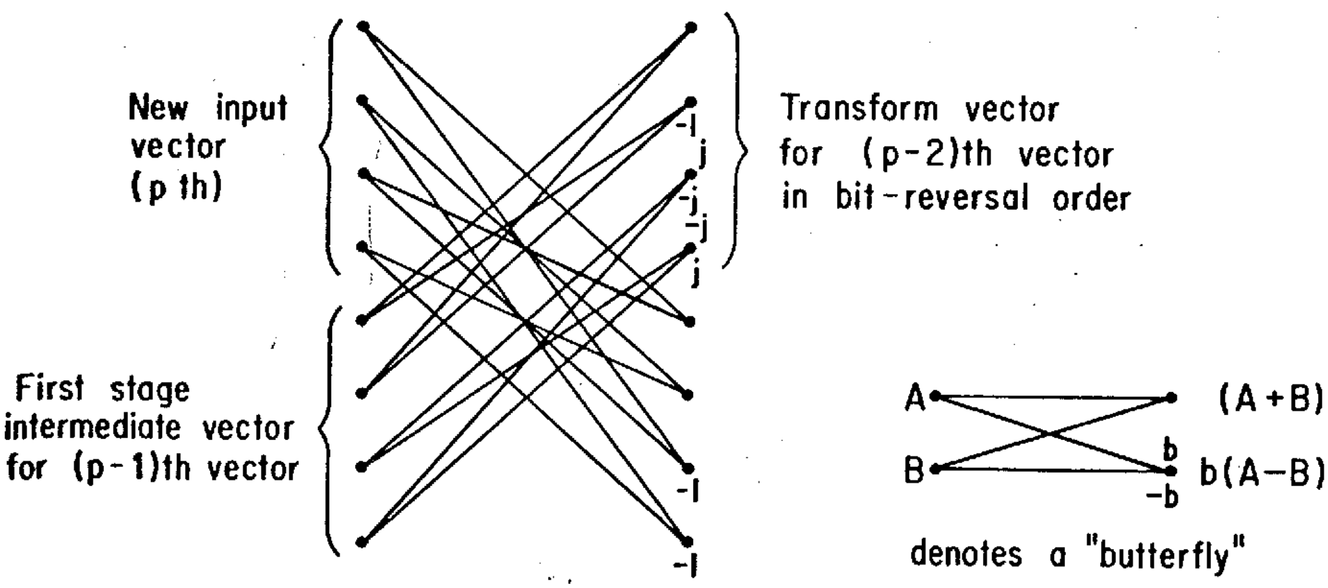
Fig. 2b : Fast Haar Transform of order 8

Fig. 3 : Pipeline Fast Haar Transform of order 8.



Input vector
Output vector
V_0
V_1
V_2
V_3
F_0
F_1
F_2
F_3
-1
j
-j
j
bit - reversal ordering

(a)

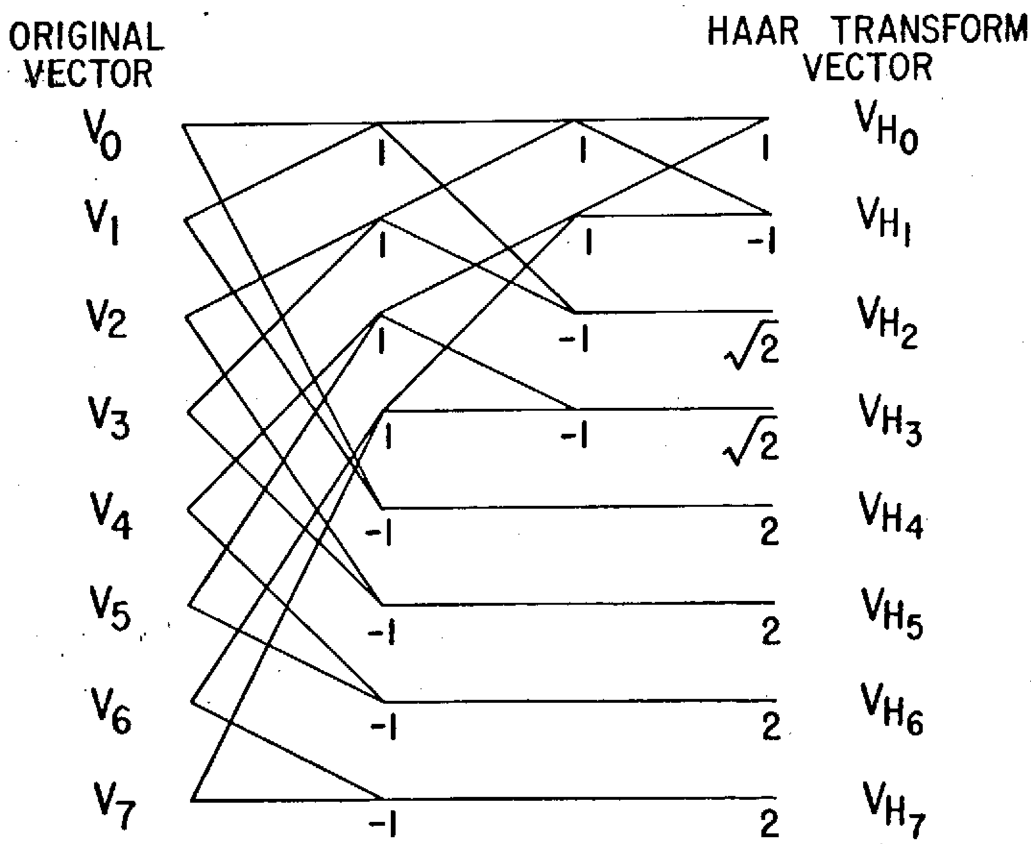


New input vector (p th)
First stage intermediate vector for (p-1)th vector
Transform vector for (p-2)th vector in bit-reversal order
-1
j
-j
A
B
(A+B)
b(A−B)
b
−b
denotes a "butterfly"

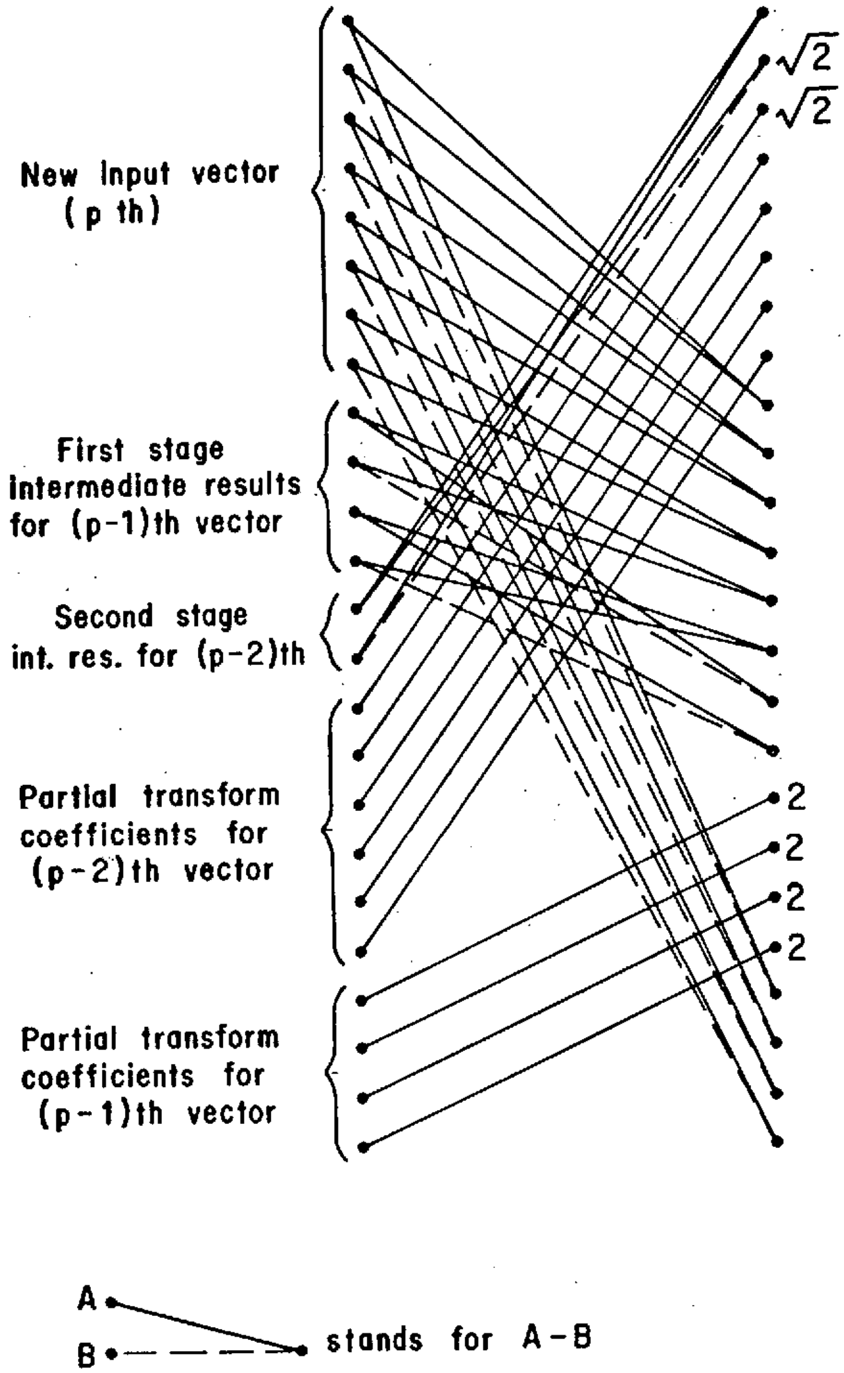
(b)

$$[H_8] = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}$$

(a)



(b)


New input vector
(p th)
First stage
intermediate results
for (p-1)th vector
Second stage
int. res. for (p-2)th
Partial transform
coefficients for
(p-2)th vector
Partial transform
coefficients for
(p-1)th vector
√2
√2
2
2
2
2
A
B
stands for A-B