Unified Matrix Processor Design for FCT-IV and FST-IV Hartley Based Transforms

Dr. Qadri Hamarsheh
Philadelphia University

Abstract - A new Unified Fast Shifted Cosine and Sine Transform Matrix Processor based on fast Hartley Transform is presented and described. It is also shown that the decimation-in-frequency (DIF) FCT-IV and FST-IV can be computed using the same building blocks which are used for computing decimation-in-time FCT-IV and FST-IV. A Unified Fast Shifted Cosine and Sine Transform Matrix Processor can be designed to compute either the DCT-IV or DST-IV depending on a single line control signal.

1. Introduction

In general case, $N$-point Discrete Cosine and Sine Transforms (DCT and DST) in matrix form are given by the following formulae (1.1) or (1.2):

$$y_N = P_N x_N,$$  \hspace{1cm} (1.1)

$$y(k) = \sum_{n=0}^{N-1} x(n) P_N(n,k),$$  \hspace{1cm} (1.2)

Where $x_N = x(0), x(1), \ldots, x(N-1)$ and $y_N = y(0), y(1), \ldots, y(N-1)$ represent two vectors that define $N$-point input (spatial) and output (spectrum) sequences, and $P_N = [p_N(n,k)]$, $n,k = 0,1,2,\ldots,N-1$ represents quadratic $N \times N$-point matrix that gives the transforms coefficients.

For DCT-IV, transform matrix $p_N(n,k)$ ($CIV_N$) is defined as follows:

$$CIV_N = \sqrt{2/N} \left[ q_k q_n \cos \left( \pi \left( \frac{2n+1}{2} \right) \left( \frac{2k+1}{2} \right) / (4N) \right) \right], \hspace{1cm} n,k = 0,1,2,\ldots,N-1,$$  \hspace{1cm} (1.3)

Where $q_0 = 1$, $q_n = 1/\sqrt{2}$ when $n \neq 0$ [1].

Analog variant for Sine DST-IV transform matrix has the following formula:

$$SIV_N = \sqrt{2/N} \left[ q_k q_n \sin \left( \pi \left( \frac{2n+1}{2} \right) \left( \frac{2k+1}{2} \right) / (4N) \right) \right], \hspace{1cm} n,k = 0,1,2,\ldots,N-1,$$  \hspace{1cm} (1.4)

Where $q_{N-1} = 1/\sqrt{2}$ and $q_n = 1$ when $n \neq N-1$ [1].

Matrices (1.3) and (1.4) describe shifted simultaneously in time and frequency domains and have symmetric property, which means that direct and inverse Matrices are equal. The following relations are hold for these matrices:

$$CIV_N = CIV^{-1}_N, \hspace{1cm} SIV_N = SIV^{-1}_N$$  \hspace{1cm} (1.5)

The benefits of DCT-IV and DST-IV can be summarized as follows:

- Their near optimal concentration of the signal energy in a narrow frequency band (low band), and therefore near optimal data compression.
- Shifted DCT-IV and DST-IV are approximated to the Karuhnen-Loeve transform by the criteria of the quadratic median error among the orthogonal transforms. For that reason, DCT-IV and DST-IV are the most popular transform techniques for image compression and are adopted on various
coding schemes to construct effective data compression standards: JPEG, MPEG, JFI and MPEG2 [2, 3].

- DCT-IV and DST-IV are real type transforms which work only with real signals; hence they reduce the computation complexity.
- DCT-IV transforms are used to avoid artifacts stemming from the block boundaries.

According to the construction method, FCT-IV and FST-IV are classified to:

- Direct algorithms, based on DCT and DST matrix factorization directly [4, 5, 6, 7].
- Indirect algorithms where fast algorithms of another transforms are used to calculate these algorithms [4, 8, 9, 10, 11, 12, 13].
- Recursive algorithms [14].
- Algorithms represented by systolic structures.

In the proposed processor, the indirect algorithms of Fast Shifted Cosine and Sine Transforms use fast Hartley transform as the base transform, so let talk a little about fast algorithms of Hartley transforms and how we can use these algorithms to calculate our destination algorithms.


The discrete version of the Hartley transform for the real sequence $x(n)$ can be written explicitly as:

$$H(k) = \sum_{n=0}^{N-1} x(n) (C_n^k + S_n^k), \quad k = 0,1,\ldots,N - 1 \quad (2.1)$$

And the inverse transform can be written explicitly as:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) (C_n^k + S_n^k), \quad n = 0,1,\ldots,N - 1 \quad (2.2)$$

Where $x(n)$ denotes the input sequence of elements, $C_n^r = \cos(2\pi r/N)$ and $S_n^r = \sin(2\pi r/N)$.

The advantages of DHT are:

- Direct and inverse transforms are defined in real numbers domains; the difference feature from the DFT is that it transforms real inputs to real outputs, with no intrinsic involvement of complex numbers.
- Direct and inverse transforms have a symmetry property, which means that we can use universal algorithm (program or device) to calculate the spectrum of the signals.
- There is a very simple way to convert DHT to DFT (Discrete Fourier Transform) [4].
- There are a wide range of Fast Hartley Algorithms: Radix-2, Radix-4, Radix-24 FHT, etc [4].
- They have computational advantages over the discrete Fourier transform.

The decomposition formula for Radix-2 FHT with decimation-in-time (DIT)-FHT2 (which is used in the proposed processor) can be written as [4]:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) (C_n^k + S_n^k), \quad n = 0,1,\ldots,N - 1 \quad (2.2)$$
\[ H(k) = H_1(k) + H_2(k)C_N^k + H_2(N-k)S_N^k, \quad k=0,1,...,N-1, \]  

(2.3)

Where \( H(k) = DHT_N\{x(n)\} \), \( H_p(k) = DHT_{N/2}\{x(2n+p-1)\} \) and \( p=1,2 \).

The FHT2 Graph for \( N = 16 \) is shown in figure (2.1). The FHT2 base operations (simple and vector rotation) are shown in figure (2.2).

![FHT2 Graph for N=16](image)

Figure 2.1 FHT2 Graph for N=16

![FHT2 base operations](image)

Figure 2.2 FHT2 base operations: a) simple b) vector rotation

In equation (2.4) the FHT2 computation complexity is calculated, taking into account, that the operation “vector rotation” is calculated effectively using 3 additions and 3 multiplications of real numbers [4].

\[ \mu_m(FHT2) = 2^{m-2}(3m-10)+4 ; \alpha_m(FHT2) = 2^{m-2}(7m-10)+4, \]  

(2.4)

3. Indirect Calculation of FCT-IV and FST-IV

The fast algorithm to calculate \( N \) point DCT-IV (FCT-IV \( L_N^{IV}(k) \) ) [15] can be summarized as follows:

Step1. Perform a permutation of the input sequence \( x(n) \) of \( N \) points to obtain a new sequence \( a(n) \) using the following equation:

\[ a(n) = x(2n), \quad a(N-1-n) = -x(2n+1), \quad n = 0,1,...,N/2-1. \]  

(3.1)

Step2. Compute \( N/2-1 \) butterflies operations with \( a(n) \) sequence to obtain a new sequence \( y_2(n) \) using the following formula:

\[ y_2(n) = a(n)C_{2N}^n + a(N-n)S_{2N}^n. \]
\[ y_2(N-n) = a(n)S^n_{2N} - a(N-n)C^n_{2N}, \quad n = 1, 2, ..., N / 2 - 1; \]
\[ y_2(0) = a(0), \quad y_2(N / 2) = a(N / 2). \quad \text{(3.2)} \]

Step 3. Fulfillment of \( N \) – point DHT for \( y_2(n) \) sequence: \( Y_2(k) = DHT_N \{ y_2(n) \}. \)

Step 4. Using operations "vector rotation" to get the required values of DHT:
\[
L_N^IV(k) = Y_2(k)R_{8N}^-(2k + 1) + Y_2(N - 1 - k)R_{8N}^+(2k + 1),
\]
\[
L_N^IV(N - 1 - k) = Y_2(k)R_{8N}^+(2k + 1) - Y_2(N - 1 - k)R_{8N}^-(2k + 1),
\]
\[ k = 0, 1, ..., N / 2 - 1. \quad \text{(3.3)} \]

Step 5. End of FCT-IVH algorithm.

To calculate \( N \) – point DST-IV (FST-IVH \( Q_N^IV(k) \)) [15] we do the following:

Step 1. Sequence permutation of \( x(n) \):
\[ a(n) = x(2n), \quad a(N - 1 - n) = x(2n + 1), \quad n = 0, 1, ..., N / 2 - 1. \quad \text{(3.4)} \]

Step 2. Fulfillment of operations "vector rotation" with the \( a(n) \) sequence to get \( y_2(n) \):
\[ y_2(n) = a(n)C^n_{2N} + a(N - n)S^n_{2N}, \]
\[ y_2(N-n) = a(n)S^n_{2N} - a(N-n)C^n_{2N}, \quad n = 0, 1, ..., N / 2 - 1; \]
\[ y_2(0) = a(0), \quad y_2(N / 2) = a(N / 2). \quad \text{(3.5)} \]

Step 3. Fulfillment of \( N \) – point DHT for \( y_2(n) \) sequence: \( Y_2(k) = DHT_N \{ y_2(n) \}. \)

Step 4: Using operations "vector rotation" to get the required values of DHT:
\[
Q_N^IV(k) = Y_2(k)R_{8N}^-(2k + 1) + Y_2(N - 1 - k)R_{8N}^+(2k + 1),
\]
\[
Q_N^IV(N - 1 - k) = Y_2(k)R_{8N}^+(2k + 1) - Y_2(N - 1 - k)R_{8N}^-(2k + 1),
\]
\[ k = 0, 1, ..., N / 2 - 1. \quad \text{(3.6)} \]

Step 5. End of FST-IVH algorithm.

We can conclude that DCT-IV and DST-IV are symmetric transforms, so if we invert the FCT-IVH and FST-IVH we will obtain the FCT-IVH and FST-IVH algorithms with decimation-in-frequency (DIF).

We can see that, the DIT \( N \) – point DCT-IV and DST-IV based on fast algorithm Hartley contain the following steps:

1. Retrograde indexing the elements of the input sequence
2. First step of vectors rotation (computation of FCT-IV or FST-IV butterflies).
3. \( N \) – point DHT calculation

With DIF algorithms we will have the inverse order of above steps.

The computation complexity for the obtained algorithms is:

\[
\mu^N_N \left( FCT \right) = 2^{m-1} (m + 3) - 2
\]

\[
\alpha^N_N \left( FCT \right) = 2^{m-1} (3m + 1) + 2
\]

\[
\mu^N_N \left( FCT \right) = \mu^N_N \left( FST \right), \ \alpha^N_N \left( FCT \right) = \alpha^N_N \left( FST \right) \quad (3.7)
\]

In equation (3.7), we assume that, the vector rotation can be calculated through 3 real additions and 3 real multiplications, Computation complexity for N point FHT2 equals: [4]

\[
\mu_n \left( FHT \right) = 2^{m-1} (m - 3) + 2; \ \alpha_n \left( FHT \right) = 2^{m-1} (3m - 5) + 6. \quad (3.8)
\]

And vector rotation step in all FCT-IV and FST-IV algorithms contains \(3 \cdot 2^{m-1}\) multiplications and \(3 \cdot 2^{m-1}\) additions.

4. FCT-IVH Matrix Processor Architecture

According to the algorithm FCT-IVH in the previous section, figure 4.1 illustrates the general architecture of the matrix processor (MP).

![Figure 4.1 System Block Diagram for FCT-IVH Matrix Processor](image-url)
FCT-IVH Matrix Processor contains three sub blocks: memory, operational unit and control unit.

Sorted Memory Block (SMB) is used for storing, sorting and indexing the input values from input bus \( x(n) \) in order to give output the same rate that input data; this block performs this process of data using the formula 3.1.

Control Unit Block (CUB) contains Address Generation Unit (AGU) and Control Unit (CU). These units contain control lines that select the memory or I/O and cause them to perform a read or write operation, address processor memory space and choose the length of transform (\( Kn \) control signal).

Operational Unit Block (OUB) contains the following sub blocks:

- Inverter Block, this computation block is used for inversing the sign using formula 3.1.
- Vector Rotation Block VR1, first vector rotation block fulfills the rotation before Hartley, it’s a butterflies’ computation block of adders and subtractors, addition and subtractions are straightway performed as defined in 3.2.
- FHT Processor that fulfills the N-point DHT.
- Vector Rotation Block VR2, second vector rotation block fulfills the rotation after Hartley, it’s a butterflies’ computation block of adders and subtractors, addition and subtractions are straightway performed as defined in 3.3., this block is called FHT→FCT-IV Converter block, because using this block the required values of FCT are calculated.

The internal structures of VR1 and VR2 are shown in figures 4.2 and 4.3. These structures contain \((N/2-1)\) Computation blocks \(vr1, vr2 \ldots vr_{N/2-1}\) for VR1 and \(N/2\) Computation blocks for VR2.

In general, each block computes the vector rotation using the following formula:

\[
Y_1 = X_1 \cdot C_k + X_2 \cdot S_k \\
Y_2 = X_1 \cdot S_k - X_2 \cdot C_k \quad (4.1)
\]

Where: \(X_1, X_2\) - input of the base operation; \(Y_1, Y_2\) - output of the base operation, and \(C_k, S_k\) - phase multiplier factors values.
Figure 4.2 VR1 Structure

Figure 4.3 VR2 Structure
5. Unified FCST_IVH Processor

Figure 4.4 represents the proposed unified processor. Comparing figure 4.1 and figure 4.4 we can see that the FCT-IVH Matrix Processor and FCST-IVH Matrix Processor share the same building blocks therefore, we can design a FCT / FST processor with additional multiplexer and a single line control signal (S in figure 4.4). The S flag determines if the processor computes FCT or FST (formula 3.1 or formula 3.4): S =0 it computes FCT-IVH, S =1 it computes FST-IVH. The output of multiplexer is connected directly to VR1 in Sine Transform case, or the output is connected to the inverter block in Cosine Transform case.

6. Conclusions

This paper has been dedicated to describe and represent the Unified FCST-IVH Matrix Processor based on fast Hartley Transform, in this paper the characteristics of the indirectly FCT-IV and FST-IV algorithms’ realization using Hartley Transforms on the base of matrix processors are presented, the proposed matrix processor consists of highly regular, parallel elements which are suitable for VLSI implementation, this Matrix Processor projects the graph of the algorithm, so the execution time of the algorithm is equal to the sum of the execution time for all algorithm stages.

It is also shown that different FCT-IV and FST-IV algorithms can be computed using the same building blocks.
References


