Parallel Jacket Transformation in Multi-Mesh Network

Amit Datta
Department of Engineering & Technological Studies,
University of Kalyani
Kalyani,West Bengal 741235
Kalyani, India
e-mail: amitdatta_wb@yahoo.co.in

Mallika De
Dr. Sudhir Chandra Sur Degree Engineering College
540, Dum Dum Road, Kolkata, West Bengal 700074
Kolkata, India
e-mail: demallika@yahoo.com

Abstract— In this paper a parallel algorithm for Jacket transform is proposed in multi mesh architecture having n^4 processing elements. Multi mesh architecture is formed by collection of meshes having n × n structure. These meshes are arranged in n rows and n columns. In this paper, in place generation of the Jacket matrix elements in multi mesh of size n^4 processors has been presented, which is then followed by an algorithm for the Jacket transformation. This parallel algorithm for Jacket transformation of vector of length N has been proposed with O (log √N)) addition time and O (√N) data movement time.

Keywords*-Jacket Matrix, Jacket Transformation, Multi-Mesh, 2D Mesh, Hadamard Transformation*

*****

I. Introduction

The Hadamard matrix and its generalizations are orthogonal matrices [1], [2] with many applications in signal processing, data processing, CDMA multiplexing, CDMA de-multiplexing, cryptography etc. The Jacket matrix motivated by the center weighted Hadamard matrix with an inverse constraint is a special matrix with its inverse matrix being determined by the element wise inverse of the matrix. In particular, several interesting matrices, such as the Hadamard matrix, the Fourier matrix, and the slanted matrix, belong to the Jacket matrix family [4]. In addition, the Jacket matrix is related to many useful matrices, such as, the unitary matrix and the Hermitian matrix, that can be potentially applied in signal processing, data compression, cryptography, orthogonal code design and so on [7].

A Jacket matrix is a square matrix $J = [j_{rs}]$ of order n, whose entries are non-zero and may have values real, complex or from a finite field. Moreover, $JK = KJ = I_n$ where I_n is identity matrix of order n and $K = (j_{rs}^{-1})^T/n$, T stands for transpose and K is the inverse matrix of J. If the transform of the matrix $\boldsymbol{a}$ acted on J is defined by A, then $A = \boldsymbol{a}J$.

In this paper, in place generation of the Jacket matrix elements in multi mesh of size n^4 processors has been presented, which is then followed by an algorithm for the Jacket transformation.

II. Center Weighted Hadamard transform and Jacket transform

The center weighted Hadamard transform (*CWHT*) [3]-[6], like Hadamard transform, requires only real operations. The *CWHT* weights the region of mid-spatial frequencies of the signal where as no such option is available in Hadamard transform (*HT*). The higher order *CWHT* can be formed from the lower order *CWHT* using Kronecker product of fundamental Hadamard matrices with lower order *CWHT*.

Definition 1 (Kronecker product)**:** Let matrix $P = (p_{i,j})_{m\times n}$ be of size $m \times n$ and matrix $Q = (q_{s,t})_{k\times l}$ of size $m \times n$, respectively. The Kronecker product of P and Q, which is denoted by $P \otimes Q$ is a matrix of size $mk \times nl$, i.e.,

$$P \otimes Q = \begin{pmatrix} p_{1,1}\,Q & \cdots & p_{1,n}\,Q \\ \vdots & \ddots & \vdots \\ p_{m,1}\,Q & \cdots & p_{m,n}\,Q \end{pmatrix}$$

The centre weighted Hadamard matrix is a typical case of the Jacket matrix whose inverse matrix is achieved from the element wise inverse of the initial matrix [4].

Definition 2 (Jacket matrix)**:** Let a square matrix of size $N \times N$ be denoted by $J_N = (j_{s,t})_{N\times N}$. The matrix J_N is a Jacket matrix if its inverse matrix can be simply obtained by its element-wise inverse, i.e., for $1 \le s, t \le N$, we obtain

$$J^{-1}_N = \frac{1}{c}\,(1/\,(j_{s,t})^T{}_{N\times N})$$

where, c is a normalized constant such that $J^{-1}_N J_N = J_N J^{-1}_N = I_N$ and the superscript T denotes the matrix transposition operation. In detail, we have,

$$J_N = \begin{pmatrix} j_{1,1} & j_{1,2} & \cdots & j_{1,N} \\ \vdots & \vdots & & \vdots \\ j_{N,1} & j_{N,2} & \cdots & j_{N,N} \end{pmatrix},$$

and its inverse is given by

$$J_N^{-1} = \frac{1}{N}\begin{pmatrix} {j^{-1}}_{1,1} & {j^{-1}}_{2,1} & \cdots & {j^{-1}}_{N,1} \\ \vdots & \vdots & & \vdots \\ {j^{-1}}_{1,N} & {j^{-1}}_{2,N} & \cdots & {j^{-1}}_{N,N} \end{pmatrix}$$

Jacket matrix has the property of the conventional Jacket, where the observed and reverse are more likely to be similar in appearance. Currently, the fractional Jacket transform (*FRJT*) has been applied in a wide range of subjects. *FRJM* is a generalized Jacket matrix with an inverse constraint [8]. The Hadamard and center weighted Hadamard matrices of order $N = 2^k$ is denoted as $[H]_N$ and $[CWH]_N$. The lowest order center weighted Hadamard matrix of size (4 × 4) is defined as follows:

$$[CWH]_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & 2 & -1 \\ 1 & 2 & -2 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

A recursive definition is given below for the generation of higher order center weighted Hadamard matrix from the lower order center weighted Hadamard matrix.

$$[CWH]_N = [CWH]_{N/2} \otimes [H]_2$$

where $\otimes$ is the Kronecker product and $[H]_2$ is the lowest order Hadamard matrix as given below.

$$[H]_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Considering $N = 8$, we can deduce the following

$$[CWH]_8 = [CWH]_4 \otimes [H]_2$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -2 & 2 & -1 \\ 1 & 2 & -2 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \otimes [H]_2$$

$$= \begin{bmatrix} H2 & H2 & H2 & H2 \\ H2 & -2H2 & 2H2 & -H2 \\ H2 & 2H2 & -2H2 & -H2 \\ H2 & -H2 & -H2 & H2 \end{bmatrix}$$

Now, a generalized 4 × 4 Jacket matrix can be represented as,

$$[J]_4 = \begin{bmatrix} a & b & b & a \\ b & -c & c & -b \\ b & c & -c & -b \\ a & -b & -b & a \end{bmatrix}$$

where a, b, c denote the weighted factors. This matrix is weighted in the centre by

$$\begin{bmatrix} -c & c \\ c & -c \end{bmatrix}$$

If $a = b = 1$ and $c = \omega$ (the weight), the above matrix reduces to

$$[J]_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -\omega & \omega & -1 \\ 1 & \omega & -\omega & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

The *CWHT* is obtained by weighting the centre portion of the transform matrix. For $N = 2^r$, the transformed vector A is represented as

$$A_j = \sum_{i=0}^{N-1} (-1)^{<j,i>} (\omega)^{(i_{r-1} \oplus i_{r-2} \oplus \dots \oplus i_0)(j_{r-1} \oplus j_{r-2} \oplus \dots \oplus j_0)} a_i \ , \quad j = 0, 1, 2, \dots, n\text{-}1 \qquad (1)$$

Here, i and j denote the row and column positions in the above matrix J, ω is any real number and it is the weight. $<j, i> = j_{r-1} i_{r-1} \oplus j_{r-2} i_{r-2} \oplus \dots \oplus j_0 i_0$, where $\oplus$ is used to identify modulo two addition.

Example 1: Generation of the elements of the above matrix. Here, $n = 4 = 2^r$ and $r = 2$.

Now, $< j, i > = j_1 i_1 \oplus j_0 i_0$

For, $i = 0 = (00)_2$, $j = 0 = (00)_2$, $< 0, 0> = 0 \oplus 0 = 0$.

The value of the element at position (0, 0) of the Jacket matrix

$$= (-1)^{<0,0>} \omega^{(i_1 \oplus i_0)(j_1 \oplus j_0)} = (-1)^0 \omega^{(0 \oplus 0)(0 \oplus 0)} = 1.$$

In the same way, when $i = 2 = (10)_2$, $j = 2 = (10)_2$, $< 2, 2 > = 1 \oplus 0 = 1$, and the value at position (2, 2) of the Jacket matrix $= (-1)^{<2,2>} \omega^{(i_1 \oplus i_0)(j_1 \oplus j_0)} = (-1)^1 \omega^{(1 \oplus 0)(1 \oplus 0)} = -\omega$.

III. The Multi Mesh (MM) Network

The Multi-Mesh network that was proposed by the authors in [10] is made up of n^2 meshes shown in Fig. 1. In an $n \times n$ mesh, the processors are arranged in n rows and n columns. Such a mesh is used as the basic building block of the Multi-Mesh (MM) network. Here total n^2 such meshes are arranged in the form of an $n \times n$ matrix where each constituent matrix is termed as a block in MM network. In each block there are $4(n-2)$ processors on the four outer boundaries each of which has three neighbours within that block. These are called boundary processors. Also, in each block there are four corner processors each of which has two neighbours within that block. These are called corner processors. The rest of the $(n-2)^2$ processors in every block will be termed as internal processors. Each block in this network is connected to another block by suitable links so that each processor has four links in this network topology.

A processor inside a given block can be uniquely identified by two coordinates. Again blocks are organized as matrix form so each block can be identified by two coordinates, say α and β as $B(\alpha, \beta)$. Thus, each of the n^4 processors in MM can uniquely be identified using a 4-tuple of the coordinate values. The first two coordinates are used to describe the block in which the processor lies and the other two coordinates are used to identify the position of the processor inside that particular block. For example, $P(\alpha, \beta, x, y)$ is a processor lying at the x-th row and y-th column of the block $B(\alpha, \beta)$.Each of these four coordinates has value between 0 to n-1. A special symbol * will be used for any one of these four coordinates to denote the set of all processors with all possible values of the respective coordinates. For example, $P(*, *, 0, 0)$ signifies the set of the top left corner processors of all the n^2 blocks. If the processors $P(\alpha, \beta_1, x_1, y_1)$ are connected to $P(\alpha, \beta_2, x_2, y_2)$ for all values of α, $0 \le \alpha \le n\text{-}1$, we denote these sets of links by an interconnection between the sets $P(*, \beta_1, x_1, y_1)$ and $P(*, \beta_2, x_2, y_2)$. Inter-block connections among the boundary processors are given by the following rules:

Vertical Connection is identified by following rule
$\forall \beta$, $0 \le \beta \le n\text{-}1$, $P(\alpha, \beta, 0, y)$ are connected to $P(y, \beta, n\text{-}1, \alpha)$, where $0 \le y, \alpha \le n\text{-}1$, and

Horizontal Connection is identified by following rule

$\forall$ α, $0 \le \alpha \le n$-1, P (α, β, x, 0) are connected to P (α, x, β, n-1), where $0 \le x, \beta \le n$-1

All these links are two-way connections. Hence, in the multi-mesh network, all processors have a uniform degree of 4. These inter-block connections among the boundary processors are called inter-block links.

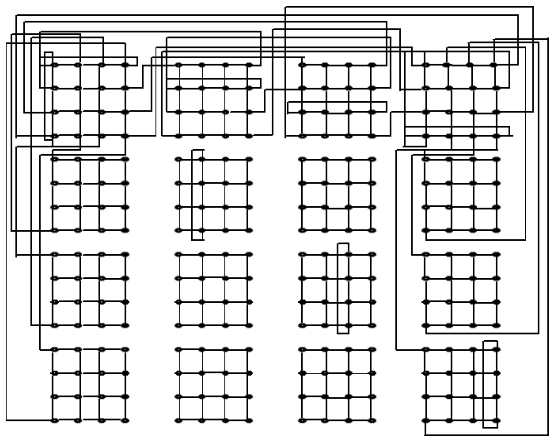


Figure 1. A simple $n \times n$ Multi-Mesh network with n = 4 (all links are not shown).

In a simple $n \times n$ mesh only $(n - 2)^2$ internal processors have degree four, the four corner processors are of degree two and $4(n - 2)$ boundary processors have degree three, as opposed to degree four for all processors on the Multi-mesh. Moreover, the diameter of the network is $2n$ as opposed to $2(n^2 -1)$ for an $n^2 \times n^2$ mesh. For this reason, any real-life applications can be solved on the proposed network more efficiently than on the corresponding mesh with the same number of processors. When time complexity is governed by the diameter of the network, the MM network is more advantageous than mesh. As examples of real-life applications, simple problems like those of calculating the sum, average, minimum, maximum of n^4 data values with $O(n)$ time on the MM network having n^4 processors have been implemented in [10]. For non-trivial problems like sorting of n^4 data values, Discrete Fourier Transform (DFT) and Hadamard Transformation have also been implemented in O (n) time [11], [12]. [13]. In the case of simple $n^2 \times n^2$ mesh each of these problems takes O (n^2) time. The reduced time complexity has been achieved due to the inter block links among the boundary processors of the meshes as defined by the above two rules.

IV. In-place Generation of Jacket Matrix Elements in Multi Mesh

For a two-dimensional mesh the element (i, j) of Jacket matrix is generated using the equation

$$(-1)^{(bin\ (i).\ bin\ (j))}(\omega)^{(i_{r-1}\oplus i_{r-2}\oplus\ldots\oplus i_0)(j_{r-1}\oplus j_{r-2}\oplus\ldots\oplus j_0)}$$

In case of multi mesh of size $n \times n$ with each of the block having $n \times n$ elements, the Jacket elements of a Jacket matrix of size $n^2 \times n^2$ can be generated based on the position of the processor element in the multi mesh. Since, the position of an element in multi mesh involves four parameters α, β, x and y, a Jacket element at (α, β, x, y) can be generated as given below.

$$P\ (\alpha, \beta, x, y) =$$

$$(-1)^{(\mathrm{bin}\ (\alpha)\circ \mathrm{bin}\ (x)).(\mathrm{bin}\ (\beta)\circ \mathrm{bin}\ (y))}\omega^{(X_{r-1}\oplus X_{r-2}\ \ldots\oplus X_0)(Y_{r-1}\oplus Y_{r-2}\ldots\oplus Y_0)} \quad (2)$$

where $X = x + \alpha\, n$ and $Y = y + \beta\, n$ and binary representation of X and Y are $X_{r-1}\ X_{r-2}\ldots X_0$ and $Y_{r-1}\ Y_{r-2}\ldots Y_0$ respectively.

Here, bin (α), bin (β), bin (x) and bin (y), stands for binary representation of α, β, x and y respectively. "∘" is used to denote concatenation and "**.**" stands for dot product of the binary numbers.

In the above equation x and y are the row and column in each of the block matrix in multi mesh and $0 \le x, y \le n$-1 whereas $0 \le X, Y \le n^2 -1$, $n^2 = 2^r$ and $N = n^4$.

Example 2: If n = 4 and α = 2, β = 3, x = 3, y = 0

$$P(2,3,3,0) = (-1)^{(bin\,(2)\,\circ\, bin\,(3))(bin\,(3)\circ\, bin\,(0))}\omega^{(X_3\,\oplus X_2\,\oplus X_1\,\oplus X_0)(Y_3\oplus Y_2\oplus Y_1\oplus Y_0\,)}$$

$$= (-1)^{(1011)\circ(1100)}\ \omega^{(1\oplus 0\oplus 1\oplus 1)(1\oplus 1\oplus 0\oplus 0)}$$
$$= (-1)^{(1.1)+(0.1)+(1.0)+(1.0)}$$
$$= (-1)$$

Similarly, P (3, 3, 3, 3) $= 1.\omega^{(1\oplus 1\oplus 1\oplus 1)(1\oplus 1\oplus 1\oplus 1)} = 1.\omega^0 = 1$

In binary number system, number of bits required to represent each α, β, x, y is same and depends on the value of n. If $n \times$ n mesh is used as a building block to form MM, the bit required will be $\log_2 n$.

V. Jacket Transformation by Matrix Vector Multiplication

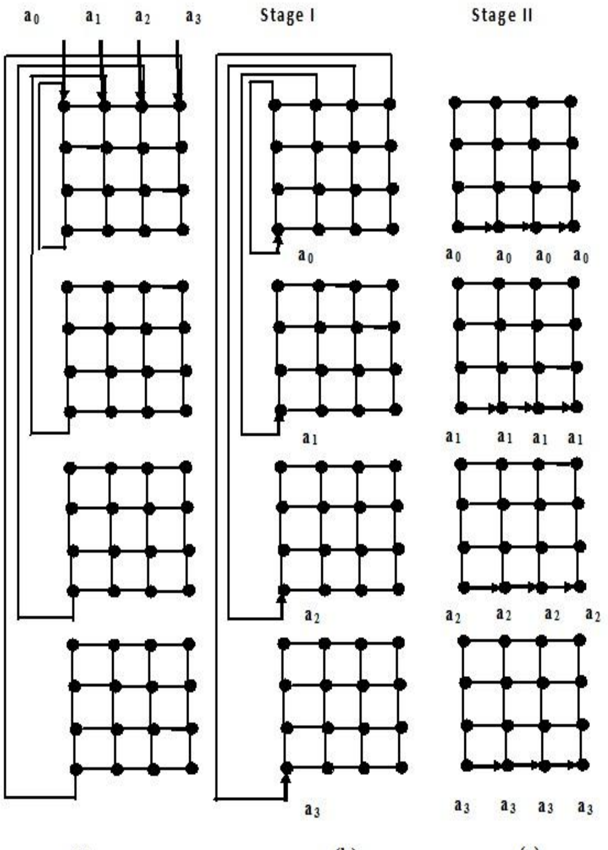
For Jacket matrix elements are generated in place using the formula described in section 4. Now, this $2^m \times 2^m$ Jacket matrix (J_m) is being used for the Jacket transformation of an input real vector $\boldsymbol{a}$ = (a_0, a_1,..., a_{n-1}) of length $n = 2^r$. The transformed vector A = (A_0, $A_1\ldots$ A_{n-1}) is expressed as,

$$\begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{n-1} \end{bmatrix} = [J_m] \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

VI. Parallel Implementation of Jacket Transformation

Parallel implementation of Jacket transformation will now be described using an MM network to achieve the O (n) time complexity.

After the in place generation of matrix elements in all the nodes/processing elements of multi mesh network, the next step is to input the vector ***a*** of length *n* to the multi mesh through upper boundary of the MM network and propagate to other processors of the network as shown in Fig. 2. Only first four components of vector ***a*** are shown in this figure as only the first column of the MM network is shown for the data movement. Vector ***a*** is propagated along intra block links as well as inter block links as shown in Fig. 2. Next step is to perform the matrix multiplication of vector ***a*** and the Jacket matrix elements those already have been generated in place as per the equation (2) given above. An element of transformed vector can be expressed as $\boldsymbol{c}_{i\ j}$ = $[J_m]_{i\ j}$ × $\boldsymbol{a}_j$. To get the transformed vector it is required to perform $A_i \ = \sum_{j=0}^{n^2-1} c_{ij}$ for each *i*, where, all c_{ij}'s are to be brought in a single block M (*i*/*n*, *i*%*n*), as they are now scattered in i^{th} rows of *n* different blocks. To bring the related elements in a single block *n* left shifts are performed along horizontal inter block links of the MM as shown in Fig. 3.



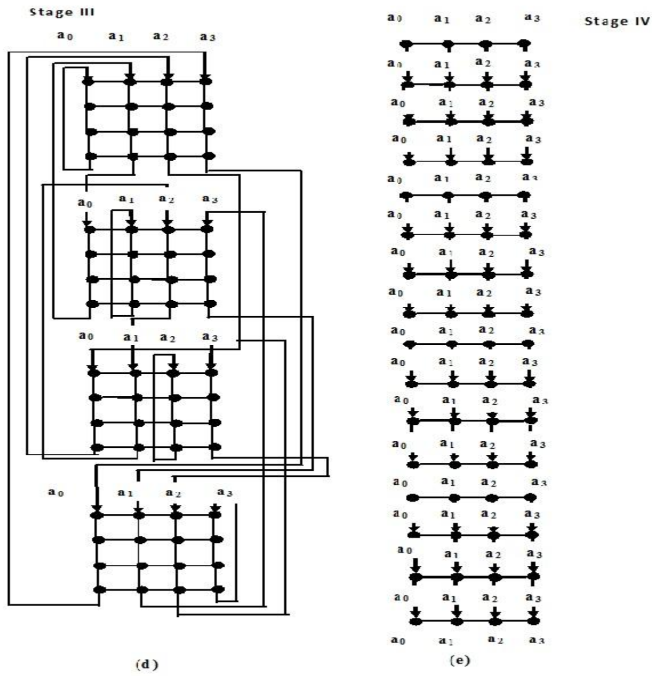


Figure 2. Data movement of vector ***a*** in blocks *M* (*, 0) of a 4 × 4 MM. (a) Input of vector ***a*** is done through the upper boundary of the block *M* (0, 0) (b) ***a*** values are moved vertically to other blocks by vertical inter block links (c) ***a*** values are moved horizontally in the last row of each block in parallel (d) ***a*** values at the last row of each block are moved vertically through vertical inter block links (e) ***a*** values are propagated along column direction in each block in parallel

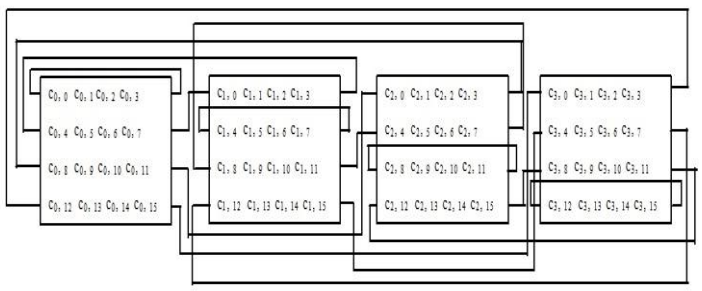

C0, 0 C0, 1 C0, 2 C0, 3
C0, 4 C0, 5 C0, 6 C0, 7
C0, 8 C0, 9 C0, 10 C0, 11
C0, 12 C0, 13 C0, 14 C0, 15
C1, 0 C1, 1 C1, 2 C1, 3
C1, 4 C1, 5 C1, 6 C1, 7
C1, 8 C1, 9 C1, 10 C1, 11
C1, 12 C1, 13 C1, 14 C1, 15
C2, 0 C2, 1 C2, 2 C2, 3
C2, 4 C2, 5 C2, 6 C2, 7
C2, 8 C2, 9 C2, 10 C2, 11
C2, 12 C2, 13 C2, 14 C2, 15
C3, 0 C3, 1 C3, 2 C3, 3
C3, 4 C3, 5 C3, 6 C3, 7
C3, 8 C3, 9 C3, 10 C3, 11
C3, 12 C3, 13 C3, 14 C3, 15

Figure 3. Contents of blocks *M* (0, *) after *n* (=4) steps data movement along the horizontal inters block links in a 4 ×4 MM

Now each of the blocks in multi mesh contains n^2 values which are summed along column wise followed by row wise at the first row of each mesh in parallel as shown in Fig. 4. After this a single step of horizontal data movement along the horizontal inter block links is done in parallel for all meshes in parallel. This horizontal data movement along inter block links bring all the values of MM at the left boundary of it as shown in Fig. 5 (only a single row of meshes is shown in the figure).

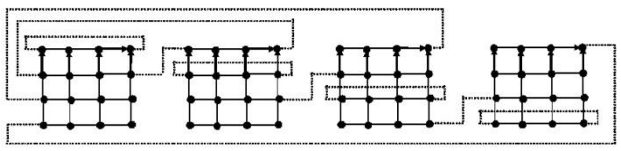


Figure 4. Column sum followed by summation in 0th row of each block in MM for *n* = 4

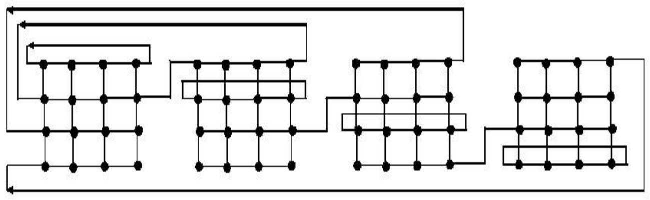


Figure 5. Single step data movement along horizontal inter block link in MM for $n = 4$

VII. Parallel Jacket Transform using MM network

A. Algorithm PJT

1. *Initialization step:* Two separate registers say R1 and R2 are being used for each processor in MM and both are initialized in this step by initializing R1 by the values of Jm, Jacket Matrix, by in place generation of the Jacket matrix elements based on the position of the processor in MM following the formula given in equation (2) and the vector to be transformed i.e. the vector a is pushed through the upper boundary of the MM network in the R2 registers and moved accordingly.

   Step 1: $\forall$ α, β, x and y, $0 \le \alpha, \beta, x, y \le n\text{-}1$ do in parallel

   $$R1(\alpha, \beta, x, y) \leftarrow (-1)^{(bin\ (\alpha)\circ bin\ (x)).(bin\ (\beta)\circ bin\ (yx))}\omega^{(X_{r-1}\oplus X_{r-2})(Y_{r-1}\oplus Y_{r-2})}$$

   where, $X = x + \alpha n$ and $Y = y + \beta n$

   In the above equation x and y are the row and column in each of the block matrix in multi mesh and $0 \le x, y \le n\text{-}1$ whereas $0 \le X, Y \le n^2 -1$, $n^2 = 2^r$ and $N = n^4$.

   Step 2: $\forall$ β and y, $0 \le \alpha, \beta, y \le n\text{-}1$ do in parallel

   $R2(0, \beta, 0, y) \leftarrow a_{\beta n+y}$

2. *Propagation of Vector **a** in MM network:*

   I. $\forall$ β and y, $0 \le \beta, y \le n\text{-}1$ do in parallel

   R2 (y, β, n-1, 0) ← R2 (0, β, 0, y);

   II. $\forall$ α, β, $0 \le \alpha, \beta \le n\text{-}1$ do in parallel

   for i = 1 to n-1 do

   $R2(\alpha, \beta, n\text{-}1, i) \leftarrow R2(\alpha, \beta, n\text{-}1, i\text{-}1)$;

   III. $\forall$ α, β, i, $0 \le \alpha, \beta, i \le n\text{-}1$ do in parallel

   $R2(i, \beta, 0, \alpha) \leftarrow R2(\alpha, \beta, n\text{-}1, i)$;

   IV. $\forall$ α, β, j, $0 \le \alpha, \beta, j \le n\text{-}1$ do in parallel

   for i = 1 to n-1 do

   $R2(\alpha, \beta, i, j) \leftarrow R2(\alpha, \beta, i\text{-}1, j)$;

3. *Generation of partial product in register R1 for each block in parallel:*

   $\forall$ α, β, x and y, $0 \le \alpha, \beta, x, y \le n\text{-}1$ do in parallel

   $R1(\alpha, \beta, x, y) \leftarrow R1(\alpha, \beta, x, y) \times R2(\alpha, \beta, x, y)$;

4. *Data Movement along horizontal inter block links:*

   Horizontal inter block link in MM forms cycle of length 2n between the rth row of the block $M(p, q)$ and the qth row of the block $M(p, r)$ for $q \ne r$, $0 \le p \le n\text{-}1$. For a given α, if data elements in $M(\alpha, *)$ are shifted through n positions along the horizontal cycles, then the pth row elements of $M(\alpha, q)$ will also be shifted to the qth row of $M(\alpha, p)$, $0 \le \alpha \le n\text{-}1$.

   /* Here, `*' indicates all possible values from 0 to n-1, but the same value for it must be used on both sides of the assignment operator */

   $\forall$ α and β, $0 \le \alpha, \beta \le n\text{-}1$ do in parallel

   begin

   I. $R1(\alpha, \beta, *, n\text{-}1) \leftarrow R1(\alpha, *, \beta, 0)$;

   II. for j = n-1 down to 1 do in parallel

   $R1(\alpha, \beta, *, j\text{-}1) \leftarrow R1(\alpha, \beta, *, j)$;

   endfor

   end

   /* Steps I and II are in parallel */

5. *Addition step at each block of MM in parallel:*

   $\forall$ α, β, y $0 \le \alpha, \beta, y \le n\text{-}1$ do in parallel

   /* Sum along each column in each mesh */

```
    begin
    for $j$ = 0 to ($\log_2 n$ - 1)
    begin
        for $i$ = 0 to ($n$ - 1) in step ($2^{j+1}$) do
            $R1(\alpha, \beta, i, y) \leftarrow R1(\alpha, \beta, i, y) + R1(\alpha, \beta, i + 2^j, y)$
    end
    end
```

/* Registers *R1* of processors $P(\alpha, \beta, 0, i)$ contain the column sums for $\forall\ \alpha, \beta, i,\ 0 \le \alpha, \beta, i \le n\text{-}1$ */

```
$\forall\ \alpha, \beta, 0 \le \alpha, \beta \le n-1$ do in parallel
/* Summing along the 0th row in each block*/

begin

$k = 0$;
for $j$ = 0 to ($\log_2 n$ - 1)
begin
    for $i = k$ to ($n$ - 1) in step ($2^{j+1}$) do
        $R1(\alpha, \beta, 0, i + 2^j) \leftarrow R1(\alpha, \beta, 0, i) + R1(\alpha, \beta, 0, i + 2^j)$;
        $k = k + 2^j$;
  end
end
```

/* The sum of n^2 data values of the block is finally brought to the register *R1* of processor $P(\alpha, \beta, 0, n\text{-}1)$ */

6. *Data movement for the final output vector from MM:*

The output vector or the transformed vector *C* is moved to the 0th column of all the blocks $M(\alpha, 0), 0 \le \alpha \le n\text{-}1$.

```
$\forall\ \alpha$ and $\beta, 0 \le \alpha, \beta \le n-1$ do in parallel

    $R1(\alpha, 0, \beta, 0) \leftarrow R1(\alpha, \beta, 0, n-1)$;
/* It is a single step horizontal data movement along inter block links */
```

B. *Simulation of the algorithm PHT*

Simulation program of the parallel algorithm PHT is done using GCC Open MPI library. The resultant transformed output vector for a given input vector was verified with a serial Jacket transformation algorithm.

VIII. Time Complexity of Parallel Jacket Transformation

Constant time say $t1$ is required for initialization. Propagation of vector *B* requires $2n$ steps of data movements in stage 2. Generation of partial products needs constant time $t2$ in stage 3. Data movements in stage 4 require n steps. Stage 5 requires $2(n\text{-}1)$ data movement steps and $2 \log_2 n$ addition steps. Data arrangement i.e. stage 6 is done in single step. So, total number of steps required for the entire process is $t1 + t2 + 5n + 2 \log_2 n$, where $t1$ is constant time for in place generation of Jacket matrix elements in parallel, $t2$ is constant time for generation of partial products in parallel, steps required for data movement is $5n$ and $2 \log_2 n$ is the number of addition steps. So, total time required is $t1 + t2 + 5n + 2\log_2 n = 5N^{1/2} + t + 2\log_2 N^{1/2}$ where $N^2 = n^4$, $t = t1 + t2$ and $t > 0$, which implies time complexity of this algorithm is $O(n)$ or $O(N^{1/2})$.

IX. Time Complexity of Parallel Jacket Transformation

A parallel algorithm for Jacket transformation of vector of length N has been proposed with $O(\log \sqrt{N})$ addition time and $O(\sqrt{N})$ data movement time. Each processor of the MM network, having N^2 processors, generates a single component of the Jacket matrix element depending on the processor position in MM network i.e. each of the components is generated in place. The proposed algorithm has suggested an implementation of the Jacket Transform in a new parallel architecture which is of lower diameter, lower degree and regular interconnections.

Moreover, the MM network is more advantageous than mesh of same size for implementation of Jacket Transform. In case of simple n2 × n2 mesh this problem would have taken O (n2) time. The reduced time complexity has been achieved due to the inter block links among the boundary processors of the meshes.

The proposed algorithm can be modified to calculate the scaled up version of the problem. In the present paper Jacket matrix of size $2^4 \times 2^4$ has been used in the example to execute the proposed algorithm which can be suitably modified to compute Jacket matrix of size $2^{4+1} \times 2^{4+1}$ to transform the input vector of size 2^{4+1} using the same number of processors. In the later case each processor has to compute four partial products resulting in increase of time complexity by a constant.

Acknowledgment

The authors express their sincere gratitude to the faculty members of the Department of Engineering & Technological Studies, University of Kalyani, and West Bengal, India, where the work has been carried out.

REFERENCES

[1] K. J. Horadam: An introduction to cocyclic generalized Hadamard matrices, Discrete Applied Math, 102 (2000), 115-131.

[2] K. J. Horadam: A Generalised Hadamard Transform, IEEE International Symposium on Information Theory, Adelaide, Australia, September 4-9, 2005.

[3] Moon Lee: The Center Weighted Hadamard Transform, IEEE Transactions on Circuits and Systems, Vol. 36, No. 9, pp 1247-1249, September 1989.

[4] Moon Ho Lee: A new reverse jacket transform based on Hadamard matrix, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 47, pp. 39-47, 2000.

[5] Moon Ho Lee: Information Theory, 2000. Proceedings, IEEE International Symposium on Information Theory, pp 471, 2000.

[6] C. P. Fan and J. F. Yang: Fast center weighted Hadamard transform algorithm, IEEE Transactions Circuits Syst. II, vol. 45, pp 436-441, Mar. 1998.

[7] R. A. Hom, C. R. Johnson: Topics in Matrix analysis, Cambridge University Press, New York, 1991.

[8] Yun Mao, Jun Peng, Ying Guo and Moon Ho Lee: On the fast fractional Jacket transform, Circuit syst Signal Process, (2014), 33. 1491-1505.

[9] D. Park and Moon Ho Lee : Jacket matrix bases recursive Fourier analysis and its applications, (2015), In Tech, DOI: 10.5772/59353.

[10] D. Das, M.De and B.P.Snha: A new network topology with multiple meshes, *IEEE Transactions on Computers*, *48* (5), 1999, 536-551.

[11] M. De, D. Das and B. P. Sinha: An efficient sorting algorithm on the Multi-mesh network, *IEEE Transactions on Computers*, *46* (10), 1997, 1132-1137.

[12] S. De, A. Datta, A. B. Bhattacharya and M. De: Fast Parallel Algorithm for Discrete Fourier Transform in Multi-Mesh Network, *Journal of Parallel and Distributed Computing and Network*, ISSN (Online) 1925-5543, ISSN (Print): 1925-5535, ACTA press, pp.1-15, 2014,DOI: 10.2316/Journal.211.2014.4.211- 1012.

[13] A. Datta, S. Das and M. De: Parallel Hadamard Transform in Multi Mesh Network, *Proc. International Conference on High Performance Computing and Applications* (ICHPCA 2014), 22nd- 24th Dec., Bhubaneswar, Also in IEEE Digital Explore, DOI: 10.1109/ICHPCA.2014.7045320.