A New Approach for Parallel Discrete Fourier Transform in Multi Mesh Network

Amit Datta and Mallika De

Abstract A simple parallel algorithm is proposed in this paper to compute Discrete Fourier Transform (DFT) coefficients of N points on Multi-Mesh (MM) architecture having N^2 processors where each of the processor generates a single element of the Fourier matrix based on the node position in the network i.e. Fourier elements are generated in place. The algorithm given here transforms a vector of length N in constant time multiplication and O ($\sqrt{N}$) addition and data communication time.

Keywords Multi mesh · Discrete Fourier transform · Parallel discrete Fourier transform · 2D mesh

1 Introduction

Discrete Fourier transforms (DFTs) [1, 2] are used extensively in a range of computing applications like audio processing, image processing, medical imaging etc.

A number of parallel algorithms developed for DFT/FFT in different architectures can be found in the literature [2–7].

Using an extension of the common factor algorithm authors in [3] have shown that a simple planar 2-dimensional systolic array having $N = M^2$ processing

A. Datta (✉)
Research Scholar, Department of Engineering and Technological Studies, University of Kalyani, Kalyani, India
e-mail: amitdatta_wb@yahoo.co.in

M. De
Department of Engineering and Technological Studies, University of Kalyani, Kalyani, India
e-mail: demallika@yahoo.com

G. P. Biswas and S. Mukhopadhyay (eds.), *Recent Advances in Information Technology*, Advances in Intelligent Systems and Computing 266, DOI: 10.1007/978-81-322-1856-2_10,

elements can be used to compute DFT of size N in $2\ M + 1$ steps of pipelined operations, achieving the area-time complexity $AT^2 = O\ (N^2\ log^3\ N)$.

In this paper we present an entirely different but simple direct matrix–vector multiplication approach for computing DFT in Multi-Mesh architecture. The time complexity of the algorithm is $O\ (\sqrt{N})$ for N point DFT. The algorithm is implemented in Multi-Mesh (MM) architecture [7–9] having N^2 processors where each processor is of 4 degree. As shown in Fig. 1, there are N meshes of size $\sqrt{N} \times \sqrt{N}$ each, which themselves are again arranged in $\sqrt{}$N rows and $\sqrt{N}$ columns so that there will be N^2 processors in the MM network.

The important aspect of the proposed parallel algorithm is that it requires constant time multiplication. The order of time complexity is mainly governed by the data communication and addition times. The Fourier components are generated in place in parallel for each processor using the processor position co-ordinates and require three multiplication steps and two addition steps. The input of the vector to be transformed and their propagation requires $(2n + 1)$ communication steps. The partial products are generated by a single multiplication step. After which $2(n - 1)$ additions and $(n + 1)$ data communication steps are required to get the transformed vector.

2 Discrete Fourier Transform

The DFT [1, 2] of a continuous function $a\ (t)$ is given by

$$A_f = \sum_0^{N-1} a_t e^{\frac{2\pi ifk}{N}},\ 0 \leq f \leq N - 1$$

where, $i = \sqrt{-1}$. The variable t is used to represent time, and f is used to represent frequency. The Fourier transform is used to convert a function of time into a function of frequency.

3 The Multi Mesh (MM) Network

First, the MM network proposed by the authors [7] is described. The basic building block of the MM network is n × n mesh.

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 M (α, β). Thus, each of the n^4 processors in MM can be uniquely identified using a 4-tuple. If the boundary processors of P (α, β_1) are connected to boundary processors of P (α, β_2) for all values of α, $0 \leq \alpha \leq n - 1$, we denote these sets of links by an interconnection between the sets P $(*, \beta_1)$ and P $(*, \beta_2)$. Inter-block vertical connection is identified by following rule.

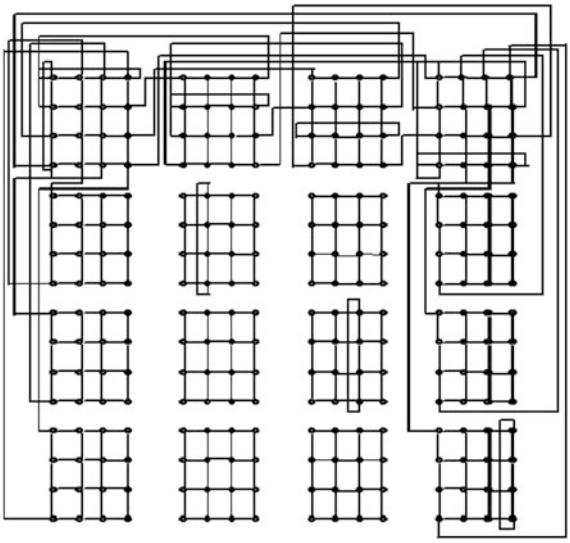

Fig. 1 A simple $n \times n$ multi-mesh network with $n = 4$ (all links are not shown)

$\forall\ \beta,\ 0 \leq \beta \leq n - 1$, P $(\alpha, \beta, 0, y)$ are connected to P $(y, \beta, n - 1, \alpha)$, where $0 \leq y, \alpha \leq n - 1$, and inter block horizontal connection is identified by following rule.

$\forall\ \alpha,\ 0 \leq \alpha \leq n - 1$, P $(\alpha, \beta, x, 0)$ are connected to P $(\alpha, x, \beta, n - 1)$, where $0 \leq x, \beta \leq n - 1$.

4 Fourier Transformation Using Matrix–Vector Multiplication

Transformation of a matrix A of size $N \times N$ by a vector B of size N is given by $C = A \times B$, where C is the transformed vector of length N.

4.1 Parallel Implementation of Fourier Transform Using MM Network

The given matrix A is partitioned into $n \times n$ sub-matrices $A_{\alpha, \beta}$ where $0 \leq \alpha, \beta \leq n - 1$. In this way there are n^2 sub-matrices which are initially mapped to M (α, β), $0 \leq \alpha, \beta \leq n - 1$, each of which contains n^2 processors P $(\alpha, \beta, *, *)$. This '*' indicates all possible values from 0 to $n - 1$. Here each processor holds a single component of matrix A and all these values are generated in place according to the position of the node P (α, β, x, y), $0 \leq \alpha, \beta, x, y \leq n - 1$. For example,

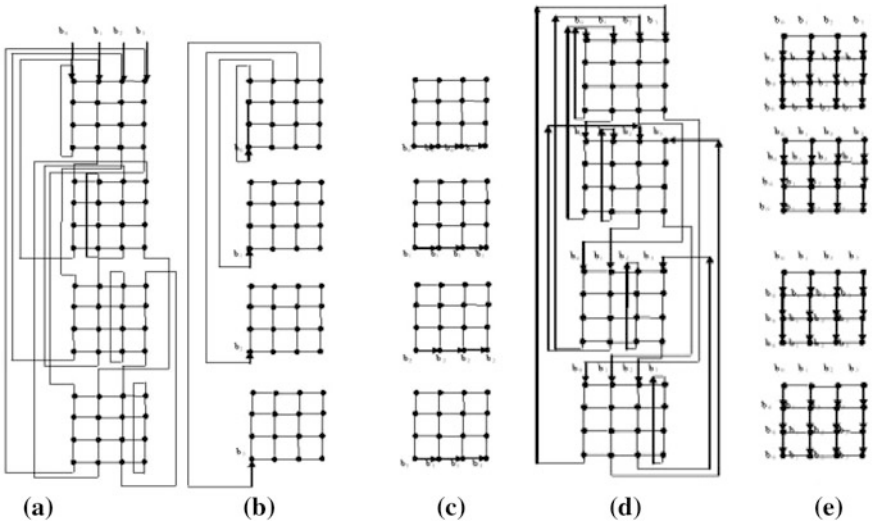

Fig. 2 Data movement steps of vector B in blocks M (*, 0) of a 4 × 4 MM: **a** Input through the *upper* boundary of the block M (0, 0). **b** Input values are moved to other blocks through *vertical inter block links*. **c** Propagation along row in each block. **d** Last row of data is moved along *vertical inter block links*. **e** Propagation along column direction within block

$\omega^{(\alpha n + x)(\beta n + y)}$ is the value of A at the node position P (α, β, x, y) and is calculated as $\cos\frac{(\alpha n+x)(\beta n+y)2\pi}{n} - i\sin\frac{(\alpha n+x)(\beta n+y)2\pi}{n}$ using the built in complex functionality $\omega = e^{\frac{2\pi i}{n}} = \cos\frac{2\pi}{n} - i\sin\frac{2\pi}{n}$. Next B values are input through the upper boundary of the MM network and are moved to all the other positions in the MM network. The movement of the B values are shown in Fig. 2 where only first four components of B vector are shown for the first column of blocks of a 4 × 4 Multi-Mesh.

Once the B values are populated and A values are calculated at the corresponding node, the next step towards completion of Fourier Transform is to perform multiplication at each node.

To calculate the $C_i \sum_{j=0}^{n^2-1} c_{ij}$ for each i, where $c_{ij} = a_{ij} \times b_j$, all c_{ij} 's are to be brought in a single block M (i/n, $i\%n$), as they are now scattered in ith rows of n different blocks. To achieve this n shift is performed along the horizontal inter block links of the MM network as shown in Fig. 3 given below.

Now the n^2 values of each mesh are summed along the column direction then horizontally along the first row of each block to make an element of the result vector C. After this step one more single step data movement is performed along the horizontal inter block links to bring the values of all the blocks M (α, 0), $0 \leq \alpha \leq n - 1$, at the left boundaries as shown in and Fig. 4. The final output is obtained at the left boundary of the MM network.

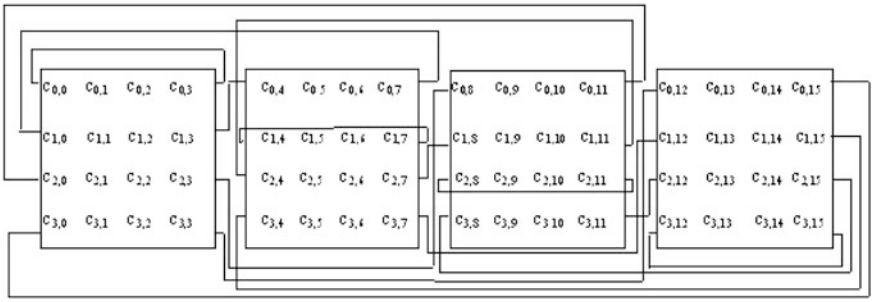

Fig. 3 Contents of blocks M (0, *) after n (=4) steps data movement along the *horizontal inter block links* in a 4 × 4 MM

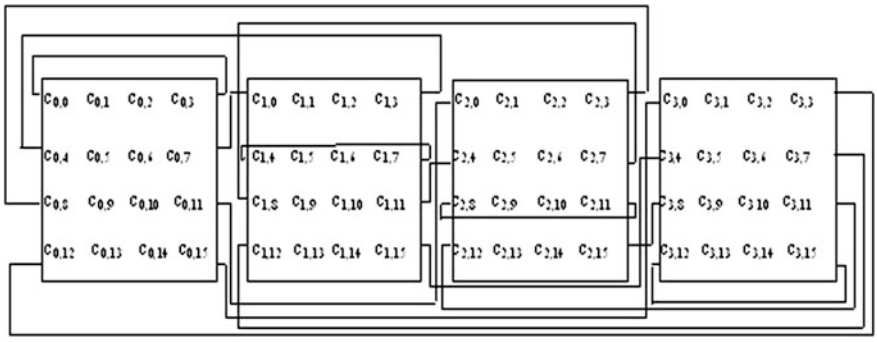

Fig. 4 Column sum followed by 0th row sum in each block of MM for $n = 4$ (*solid lines* with *arrowhead* indicates the direction of additions)

5 Parallel Discrete Fourier Transform Using MM Network

5.1 Algorithm PDFT

A. **Initialization step**

This step is subdivided into two parts. Register R1 s contain the initial values of matrix A, whereas registers R2 s contain the initial values of vector B.

Step 1: $\forall\ \alpha, \beta, x$ and y, $0 \leq \alpha, \beta, x, y \leq n - 1$ do in parallel
R1 $(\alpha, \beta, x, y) \leftarrow$

Step 2: $\forall\ \beta$ and y, $0 \leq \alpha, \beta, y \leq n - 1$ do in parallel
R2 $(0, \beta, 0, y) \leftarrow$

B. **Propagation of B vector to the other rows of MM network**

I. $\forall\ \beta$ and y, $0 \leq \beta, y \leq n - 1$ do in parallel
R2 $(y, \beta, n, 0) \leftarrow$ R2 $(0, \beta, 0, y)$;

II. $\forall\ \alpha, \beta, 0 \leq \alpha, \beta \leq n-1$ do in parallel
for $i = 1$ to $n-1$ do
R2 $(\alpha, \beta, n-1, i) \leftarrow$ R1 $(\alpha, \beta, n-1, i-1)$;
III. $\forall\ \alpha, \beta, i, 0 \leq \alpha, \beta, i \leq n-1$ do in parallel
R2 $(i, \beta, 0, \alpha) \leftarrow$ R2 $(\alpha, \beta, n-1, i)$;
IV. $\forall\ \alpha, \beta, j, 0 \leq \alpha, \beta, j \leq n-1$ do in parallel
for $i = 1$ to $n-1$ do
R1 $(\alpha, \beta, i, j) \leftarrow$ R2$(\alpha, \beta, i-1, j)$;

C. **Multiplication Step**

$\forall\ \alpha, \beta, x$ and y, $0 \leq \alpha, \beta, x, y \leq n-1$ do in parallel
R1 $(\alpha, \beta, x, y) \leftarrow$ R1 $(\alpha, \beta, x, y) \times$ R2 (α, β, x, y);

D. **Data Movement Step**

In the MM network horizontal inter block links form cycles of length $2n$ between the k th row of the block $M(i, j)$ and the j th row of the block $M(i, k)$ for $j \neq k$, $0 \leq i \leq n-1$. For a given α, if data elements in $M(\alpha, *)$ are shifted through n positions along the horizontal cycles, then the ith row elements of $M(\alpha, j)$ will also be shifted to the jth row of $B(\alpha, i)$, $0 \leq \alpha \leq n-1$.

/* 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 */

$\forall\ \alpha$ and $\beta, 0 \leq \alpha, \beta \leq n$-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 */

E. Addition Step

$\forall\ \alpha, \beta$ and $y, 0 \leq \alpha, \beta, y \leq n$-1 do in parallel
begin
for $i = n$-1 down to 1 do
R1 $(\alpha, \beta, i\text{-}1, y) \leftarrow$ R1 $(\alpha, \beta, i\text{-}1, y)$ + R1 (α, β, i, y) ;
/* R1 $(\alpha, \beta, 0, y)$ contain the column sums */
for $j = 0$ to n-2 do
R1 $(\alpha, \beta, 0, j+1) \leftarrow$ R1 $(\alpha, \beta, 0, j+1)$ + R1 $(\alpha, \beta, 0, j)$;
/* Summing along the 0^{th} row in each block, the sum of the n^2 data values of the block is finally brought to the processor P $(\alpha, \beta, 0, n\text{-}1)$ */
end

Table 1 Time complexities of PDFT computation in various architectures are given below

Architecture	Number of nodes	Length of vector	Time complexity
2D mesh [2]	N	N	$N + 2\ N^{3/2}$
Multi dimentional mesh	N	N	$O\ (\log^2 N)$
Hypercube [6]	N	N	$O\ (logN)$
Binary tree [5]	$O\ (N)$	N	$O\ (NlogN)$
Multi mesh [7]	N	$N^{0.4}$	$O\ (N^{1/4})$
Star graph [6]	$n!$	$n!$	$O\ (n^2)$
Multi mesh (proposed)	N	$N^{1/2}$	$O\ (N^{1/4})$

F. Data Arrangement Step

In this step the output data vector C is moved to the 0^{th} column of all the blocks $B\ (\alpha, 0)$, $0 \leq \alpha \leq n\text{-}1$.

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

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

/* It is a single step horizontal data movement along inter block links */

6 Time Complexity of Parallel DFT

The number of steps required for initialization stage A is 6 and for propagation stage is $2n$. Generation of partial products needs single step. Data movement stage D requires n steps. Addition stage E requires $2(n - 1)$. Data arrangement stage F is done in single step. So, total number of steps required for the entire process is $5n + 6$ or $5\ N^{1/2} + 6$ where $N^2 = n^4$ which implies time complexity of this algorithm is $O\ (n)$ or $O\ (N^{1/2})$.

7 Comparative Study of Time Complexities of Parallel DFT in Multi-Mesh and Other Architectures

The time complexities of parallel DFT computation in various architectures have been tabulated in Table 1.

For the purpose of comparison, we are considering same mesh size $(n \times n)$ and same total number of nodes (n^4) for both MM and Multi-dimensional mesh. The diameters of MM and 4-D mesh are $2n$ and $4\ (n - 1)$ respectively, and degree of each node in MM is uniformly 4, whereas the node degree of each internal node is 8 for 4-D mesh. If smaller meshes are used to construct the multi-dimensional mesh, the node degree will increase. Moreover, the time complexity of proposed DFT computation in MM outperforms multi-dimensional mesh for practical sizes of meshes, i.e., mesh sizes less than 2048×2048.

In Hypercube better time complexity is achieved through its high node degree and number of links with increasing dimensions.

8 Conclusion and Future Work

A parallel algorithm for DFT of vector of length N ($=n^2$) has been proposed with constant multiplication time and O ($\sqrt{N}$) addition and data communication time. Each processor of the MM network having N^2 ($=n^4$) processors, generates a single component of the Fourier matrix depending on the processor position in MM network i.e. each of the components is generated in place.

The authors in [7] have implemented $n^{1.42} \times n^{1.42}$ matrix-by-matrix multiplication in $O(n)$ time using n^4 processors (same as that used in the proposed work) of MM and indicated that DFT of vector length $n^{1.42}$ can be computed in the same way. In the present work, DFT of vector length n^2 is computed as linear transformation in $O(n)$ time using n^4 processors of MM. The Fourier matrix is calculated in-place at each node of the MM, so there is no need to input the Fourier matrix through the left boundary of each mesh (block) as was done in [7], and vector to be transformed was input only through the upper boundary of the MM whereas the input was made through the upper boundary of each mesh (block) in [7].

As a future work, we may consider a generalized MM network where total number of processors in MM network is $m^2 \times n^2$ having $m \times n$ number of processors in each block of the MM network for $m, n \geq 3$.

References

1. Cooley, J.M., Tukey, J.W.: An algorithm for the machine computation of the complex Fourier series. Math. Comput. **19**, 297–301 (1965)
2. Strong, J.P.: The Fourier transform on mesh connected processing arrays such as massively parallel processors. CAPAIDM **19**, 190–196 (1986)
3. Shousheng, H., Torkelson, M.: A systolic array implementation of common factor algorithm to compute DFT. In: International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN), pp. 374–381 (1994)
4. Sinha, B.P., Mukherjee, A.: Parallel sorting algorithm using multiway merge and its implementation on a multi-mesh network. J. Parallel Distrib. Comput. **60**, 891–960 (2000)
5. Jaja, J.: An Introduction to Parallel Algorithms. Addison-Wesley (1992)
6. Fragopoulou, P., Akl, S.G.: A parallel algorithm for computing Fourier transform on the star graph. Trans. Parallel Distrib. Syst. **5**(5), 525–531 (1994)
7. Das, D., De, M., Sinha, B.P.: A new network topology with multiple meshes. IEEE Trans. Comput. **48**(5), 536–551 (1999)
8. De, M., Das, D., Sinha, B.P.: An efficient sorting algorithm on the multi-mesh network. IEEE Trans. Comput. **46**(10), 1132–1137 (1997)
9. Sinha, B. P.: Multi-mesh an efficient topology for parallel processing. In: Proceedings of 9th International Parallel Processing Symposium, (IPPS), pp. 17–21 (1995)