SRC TR 88 - 39

# OPTIMAL ARCHITECTURES for MULTIDIMENSIONAL TRANSFORMS

by

Chaitali Chakrabarti
and
Joseph Ja'Ja'

# Optimal Architectures for Multidimensional Transforms [1]

Chaitali Chakrabarti
Department of Electrical Engineering
Systems Research Center
University of Maryland
College Park, MD. 20742

Joseph JáJá
Department of Electrical Engineering
Institute for Advanced Computer Studies
Systems Research Center
University of Maryland
College Park, MD. 20742

## Abstract

Multidimensional transforms have widespread applications in computer vision, pattern analysis and image processing. The only existing optimal architecture for computing multidimensional DFT on data of size $n = N^d$ requires very large rotator units of area $O(n^2)$ and pipeline-time $O(\log n)$. In this paper we propose a family of optimal architectures with area-time trade-offs for computing multidimensional transforms. The large rotator unit is replaced by a combination of a small rotator unit, a transpose unit and a block rotator unit. The combination has an area of $O(N^{d+2a})$ and a pipeline time of $O(N^{\frac{d}{2}-a} \log n)$, for $0 < a \leq d/2$. We apply this scheme to design optimal architectures for two-dimensional DFT, DHT and DCT. The computation is made efficient by mapping each of the one-dimensional transforms involved into two dimensions.

[1]Supported in part by NSA Contract No. MDA-904-85H-0015, NSF Grant No. DCR-86-00378 and by the Systems Research Center Contract No. OIR-85-00108

# 1 Introduction

Multidimensional transforms are a powerful tool for analyzing multidimensional signals. The 2-D Discrete Fourier Transform (DFT) is widely used in spectrum analysis, speech processing and image processing. The 3-D and 4-D DFTs are used to represent dynamic patterns in computer vision and pattern analysis. Since the number of computations involved in such transforms is very large, optimal architectures with efficient computational schemes are needed.

There exists an architecture [GS] for computing multidimensional DFT on a single file of $n = N^d$ elements whose $AT_p^2$ performance achieves the known lower bound, where $A$ is the area and $T_p$ is the pipeline time. The design consists of $N^{d-1}$ DFT(N) computation units and a rotator of size $O(n^2)$ for data permutation. In this paper we show that if the input is in the form of a 2-D array of size $N^{d/2+a} \times N^{d/2-a}$, $0 < a \leq d/2$, then we can design a family of optimal architectures for different values of $a$. The design of [GS] is a member of this family when $a = d/2$. There are two cases depending on whether A) $a$ is an integer or B) $a$ is not an integer but $N^a$ is an integer. Our design for Case A consists of $N^{d/2+a-1}$ DFT(N) arrays, a transpose unit of size $O(N^d \log n)$ and a rotator of size $O(N^{d+2a})$. Case B, which is slightly more complicated, requires an additional block rotator unit of size $O(N^{d+2a})$. The maximum pipeline time for both the cases is $O(N^{d/2-a} \log n)$. Thus for low values of $a$ the area would be small, whereas for large values of $a$ the pipeline time would be small. In addition all these designs satisfy the $AT_p^2$ lower bound.

The rest of the paper is organized as follows. In Section 2 we state the definition and lower bounds for the computation of multidimensional linear transforms. Non optimal and optimal architectures that exist in the literature are briefly discussed in Section 3. Section 4 deals with the family of optimal architectures for different values of $a$. In Section 5 we propose optimal as well as efficient designs for computing 2-D DFT, Discrete Hartley Transform (DHT) and Discrete Cosine Transform (DCT).

Section 6 summarizes the whole paper.

## 2 Preliminaries

Let $n$ be the total number of data elements that are to be organized in a $d$-dimensional data cube. Each element has to be represented by $d$ indexes $n_1, n_2, \ldots, n_d$. Any $d$-dimensional linear transform can be defined by

$$X(k_1, k_2, \ldots, k_d) = \sum_{n_d} \cdots \sum_{n_2} \sum_{n_1} x(n_1, n_2, \ldots n_d)\alpha_1(n_1, k_1)\alpha_2(n_2, k_2)\ldots\alpha_d(n_d, k_d) \tag{1}$$

where $\alpha_i$s are the transform functions, $0 \leq k_i, n_i \leq N_i - 1$ for $1 \leq i \leq d$ and $n = N_1 N_2 \ldots N_d$. For instance, $\alpha_i(n_i, k_i) = exp(-j\frac{2\pi}{N_i} n_i k_i)$, $1 \leq i \leq d$ for $d$-dimensional DFT. In order to simplify our analysis, we assume

1. $N_1 = N_2 = \cdots = N_d = N$ and $n = N^d$
2. $N$ is a power of 2, that is, $N = 2^m$.

We proceed to state along the lines of [GS] the lower bound on $AT^2$ for multidimensional DFT, DHT and DCT. The pipeline time $T_p$ is used to describe the time performance of any circuit. We assume that if $n$ be the problem size, then O(log$n$) bits are sufficient to represent the value of a variable. Vuillemin [Vu] has shown that $AT^2 = \Omega(I^2)$ for any chip computing a transitive problem of size $I$. For multidimensional DFT, DHT and DCT, the information content of a problem $I$ is $\Omega(n \log n)$. Thus the lower bound of $AT_p^2$ is $\Omega(N^2 \log^2 n)$.

## 3 Existing architectures

In this section we briefly discuss the various optimal and non-optimal architectures for transforms with dimension $d \geq 2$. There exist schemes for computing 1-D as well as multidimensional DFT in the literature. However, there are no schemes for computing multidimensional DHT and DCT. Once we know how to optimally compute 1-D DHT and DCT, optimal computation of multidimensional DHT and

DCT would be along the same lines as that of multidimensional DFT.

The schemes for computing 2-D DFT are based on computing 1-D DFT on columns followed by 1-D DFT computation on rows. These schemes require either a separate array transpose unit [Ch], [OJ] or an internal scheme for transposing the data [Zh]. Chowdhury et.al.[Ch] have designed a RAM array transposer (RAMAT) which uses $N^2$ RAM cells of size $O(\log n)$ bits each. The area of this unit is $O(N^2 \log n)$ and this dominates the area of the design. By using efficient DFT(N) circuits which require $O(N \log n)$ time, $AT_p^2$ of the design becomes $O(N^4 log^3 n)$. This is $O(\log n)$ away from the optimal. The scheme is illustrated in Fig.1.

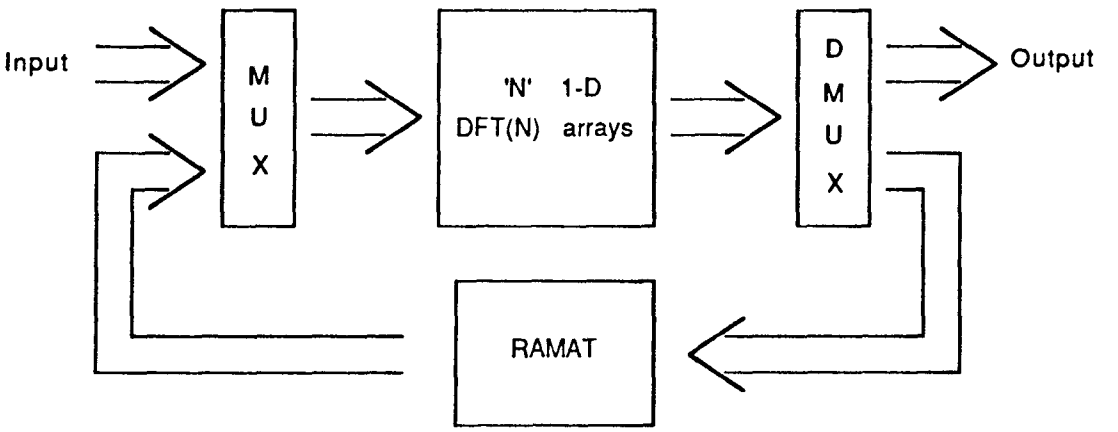



Fig.1 Architecture using RAMAT

In another design [OJ], the RAM cells are replaced by $O(\log n)$ bit shift registers. The data movement is regulated by control inputs, the generation of which is complicated. We have improved upon the design of [OJ] by simplifying the control considerably. For all these array transposer designs, the minimum area possible is $O(N^2 \log n)$. Zhang [Zh] has designed a mesh connected systolic array, each cell of which is capable of computing DFT in the vertical as well as in the horizontal direction. $AT_p^2$ for this design is also $O(N^4 \log^3 n)$. Bilardi [BS] was the first one to come up with an optimal DFT circuit. The scheme consists of orthogonal tree networks (OT) for the row and column computations and a transposer of size $O(n^2)$. An overall time complexity of

$O(\log n)$ guarantees the $AT_p^2$ lower bound. DFT(n) can be computed optimally [PV] for a large range of $T$, $T \in [\Omega(log^2 n), O(\sqrt{n \log n})]$, if interconnections based on cube connected cycles are used. However the minimum computation time of $T = O(\log n)$ cannot be achieved by this scheme. Bilardi [BS] has proposed optimal schemes for computing 2-D DFT for any T in the range $[\Omega(\log n), O(\sqrt{n \log n})]$. This is possible by organizing the input in the form of a 2-D array with 's' wavefronts and 'n/s' input lines. The pipelined transposer unit with an area of $O(n^2/s^2 + n \log n)$ and time complexity of $O(s \log n)$ can be approximated to $O(n^2/s^2)$ for $1 \leq s \leq \sqrt{n/\log n}$. All designs in this range of 's' achieve the $AT^2$ lower bound.

Not much work has been done in the field of VLSI architectures for multidimensional transforms. Gertner and Shamash [GS] have recently proposed an optimal architecture for multidimensional Fourier transforms. Their design as shown in Fig.2

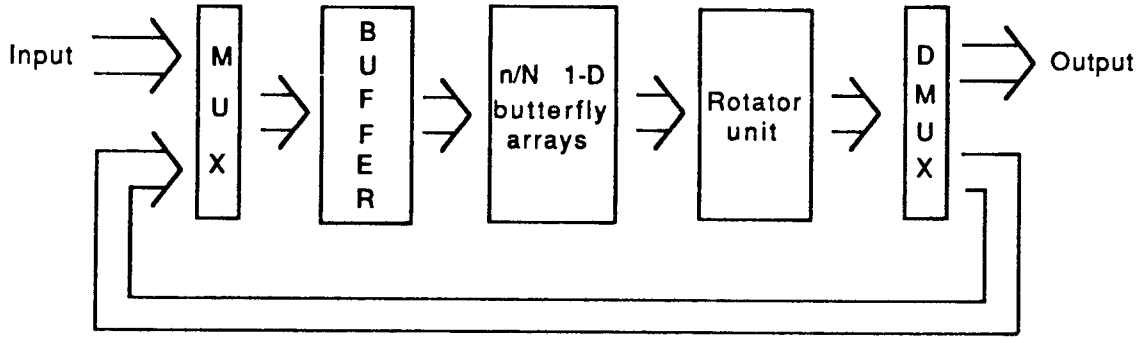



Fig.2 Architecture using rotator

consists of $N^{d-1}$ arrays for computing 1-D DFT(N) and a rotation network array or rotator for permuting the data. We shall next discuss the theory and layout of the rotator. The rotator is based on a network which performs rotation over an index in one step. Thus if the data is represented by $x(n_1, n_2, \ldots, n_d)$, then after passing through the rotator once it appears as $x(n_2, \ldots, n_d, n_1)$ and after passing it once more it appears as $x(n_3, n_4, \ldots, n_2)$. Since $O(\log n)$ bits are used to represent the data, every rotation is equivalent to $\log n/d$ cyclic shifts of its binary representation.

The data enter the rotator in $n$ rows. It is then rotated with the help of load-shift cells placed at the connection of a row input and its column necklace. A necklace is defined as a collection of nodes which are formed on rotation of the indexes. For example, for d = 3 and $\log n = 6$, the necklace generated by 000011 is 000011 - 001100 - 110000. Fig.3 illustrates a rotator for $n = 16$ and $d = 2$. The number of columns

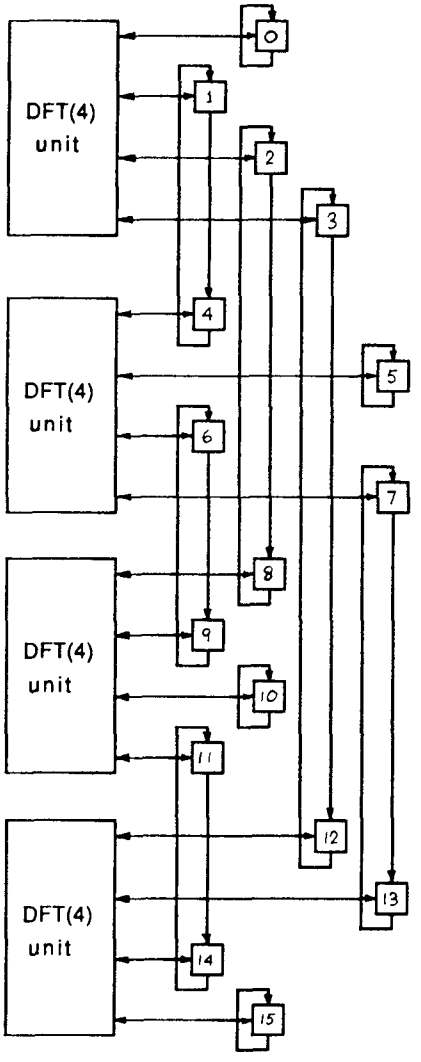



Fig.3 Layout of 4× 4 2-D DFT

required to layout the rotator is O($n$). The area complexity is O($n^2$) and the pipeline time complexity is O($\log n$). Thus this scheme achieves the $AT_p^2$ lower bound.

The large size of the rotator unit and the large number of 1-D DFT(N)

arrays makes the design of [GS] unattractive. For moderately large $n$, the size of the interconnection would be huge. This would also mean complicated inter-chip and intra-chip connections. We can reduce the area of the design drastically at the expense of a larger pipeline time and still achieve the $AT_p^2$ lower bound. In the next section we will show that there exists a family of optimal architectures, of which the design of [GS] is a member.

## 4 Family of optimal architectures

In the design by Gertner and Shamash [GS], a single file of $n = N^d$ elements are processed by $N^{d-1}$DFT(N) circuits. The data is then rotated and fed back to the DFT computation block (see Fig.2). This process is repeated $d$ times. In our design

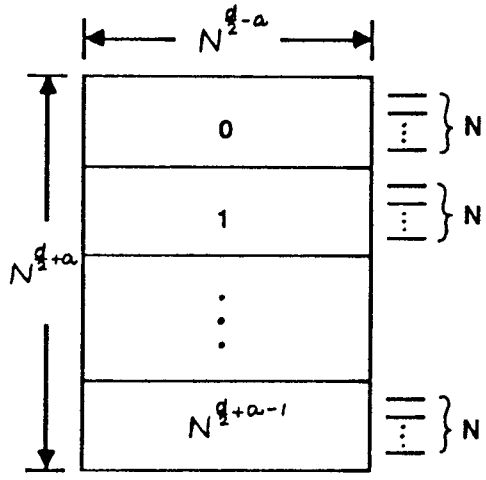



Fig.4 2-D data organization of $n = N^d$ elements

the data is organized in a 2-D array with $N^{d/2+a}$ rows and $N^{d/2-a}$ columns as shown in Fig.4. The bounds of $a$ are $0 < a \leq d/2$. Without loss of generality we assume that $d$ is divisible by 2. When $a = d/2$, the 2-D array collapses into a single column of $N^d$ elements. This is the case investigated by [GS]. When $d$ is not divisible by 2, the data can be organized into a 2-D array with $N^{\lceil d/2 \rceil + a}$ rows and $N^{\lfloor d/2 \rfloor - a}$ columns. The advantage of having a 2-D data block is that the same DFT computation block can be used to compute DFT of multiple columns. We shall investigate two cases

here :

A. When $a$ is an integer

B. When $a$ is not an integer but $N^a$ is an integer.

Case A :

In this case DFT(N) is computed $d/2+a$ times along the columns followed by $d/2-a$ times along the rows. The number of DFT(N) circuits required is $N^{d/2+a-1}$ compared to $N^{d-1}$ of [GS]. The data is passed through a rotator unit ROT1. The theory and layout of this unit are very similar to that of the rotator of [GS] discussed in Sec.3. Since the number of input lines to the rotator is $N^{d/2+a}$, the size of the rotator is $O(N^{d+2a})$. The data is circulated $d/2+a$ times through ROT1, transposed and then circulated $d/2-a$ times through ROT2. ROT2 is a rotator unit very similar to ROT1 but with $N^{d/2-a}$ input lines. Note that ROT2 can be eliminated if the transpose unit is constructed such that ROT2 has the same number of input lines as ROT1.

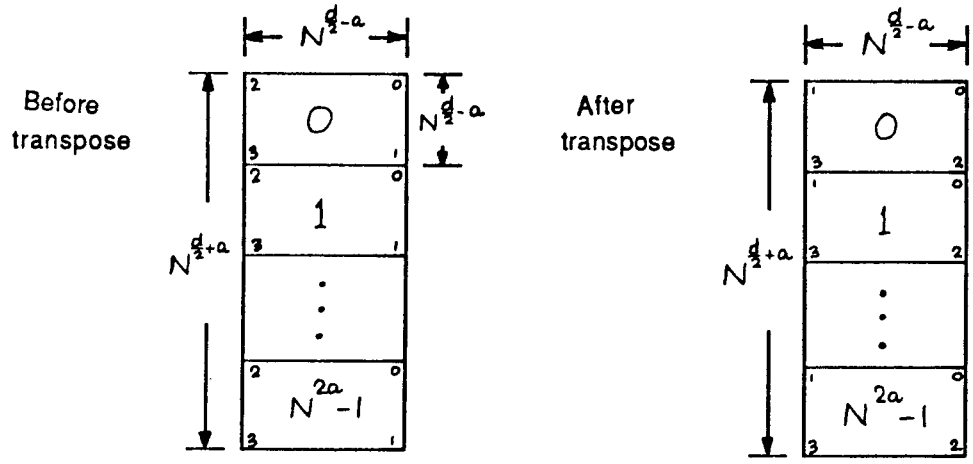



Fig.5 Data organization before and after transpose

The procedure is as follows. We can think of the data block to consist of $N^{2a}$ subblocks, each of size $N^{d/2-a} \times N^{d/2-a}$. Thus if we transpose the subblocks parallely, the size of the data block remains the same. In that case there is no need for a second rotator unit. The transpose unit consists of $N^{2a}$ subunits. Each transpose subunit operates on a data subblock. Since the transpose subunits operate in parallel, $T_p$ for the transpose unit is $O(N^{d/2-a} \log n)$. The total area of the transpose

unit is $O(N^d \log n)$. Fig.5 shows the data blocks before and after the transpose. We claim that we have not lost any information by transposing the data in subblocks. The reason is as follows. DFT(N) has to be computed $d/2 - a$ times over each row (see Fig.4). By transposing a subblock of size $N^{d/2-a} \times N^{d/2-a}$, each row of length $N^{d/2-a}$ is converted into a column of length $N^{d/2-a}$. Thus the consecutiveness of the elements in a row get converted into the consecutiveness of the elements in a column. Thus computation of DFT(N) $d/2 - a$ times over each column of the transposed data block gives the correct result. The block diagram for our design for Case A is shown in Fig.6.

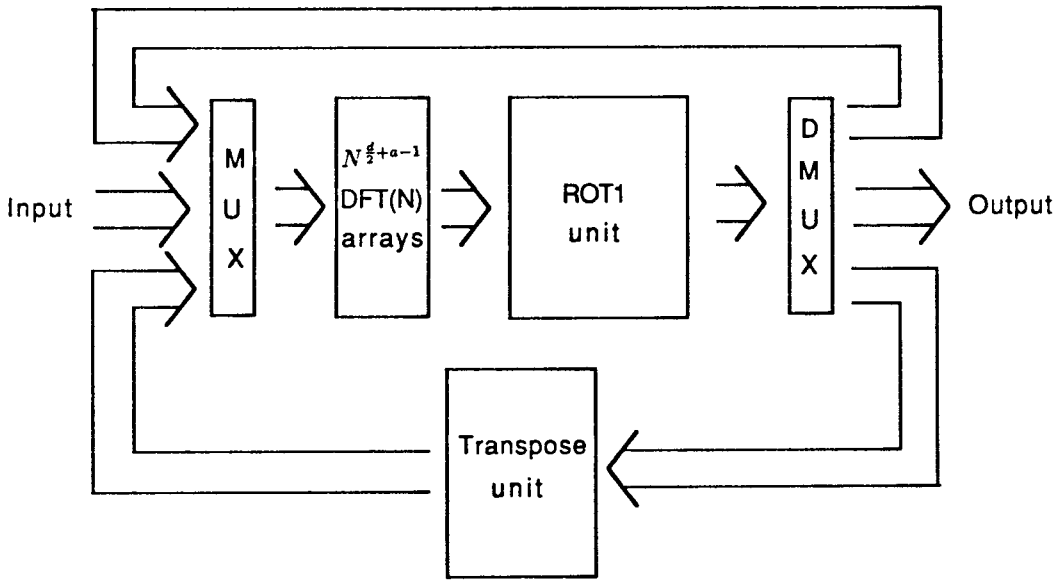



Fig.6 Architecture for Case A

The area complexity of the input MUX and the output DMUX are $O(N^{d/2+a})$. If we use the scheme of [OJ] in the computation of DFT(N), the area of the DFT(N) computation block would be $N^{d/2+a-1} \times O(N^2) = O(N^{d/2+a+1})$. The area of ROT1 is $O(N^{d+2a})$ and that of the transpose unit is $O(N^d \log n)$. The maximum pipeline time for this design is the time taken to transpose the data. $T_{p_{max}} = O(N^{d/2-a} \log n)$. In order that the design by optimal, $A_{max}$ has to be $O(N^{d+2a})$. Thus our design is optimal for those values of $a$ for which $N^d \log n \leq N^{d+2a}$. This means that there is

an additional constraint of $a \geq \frac{1}{2}\log_N \log n$. For all practical cases, $\frac{1}{2}\log_N(\log n) \leq 1$. Thus for all values of $a$ in the range $1 \leq a \leq d/2$, we can design $d$-dimensional DFTs optimally.

Case B :

This case is slightly more complicated than Case A. This is because when $a$ is not an integer, $d/2 + a$ and $d/2 - a$ are non-integers. Our scheme for Case A has to be modified appropriately.

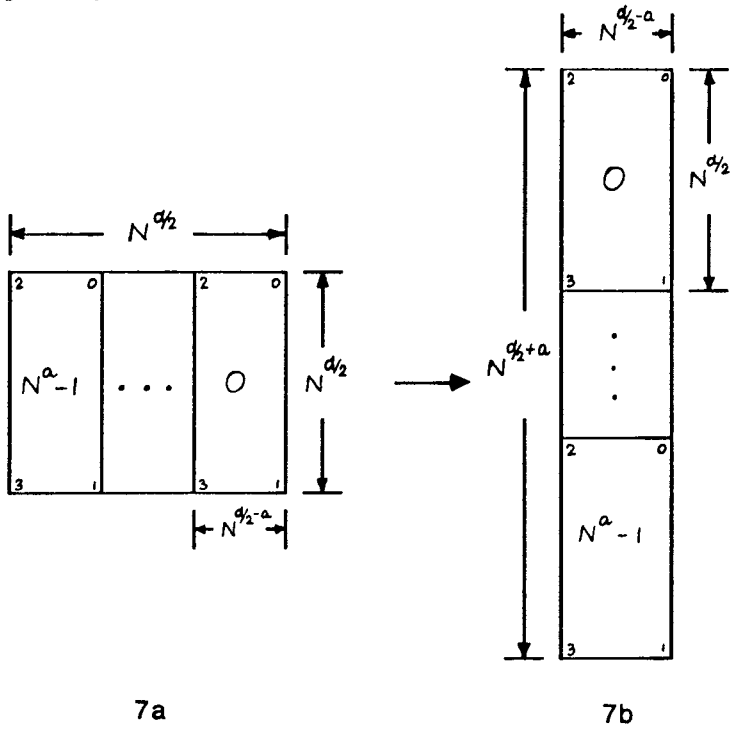



Fig.7 Data block organization for Case B

Let $a = i + f$, where $i$ is an integer and $f$ is a simple fraction. We can think of the data to be organized in a 2-D array of size $N^{d/2} \times N^{d/2}$. The array can be partitioned vertically into $N^a$ blocks each of size $N^{d/2} \times N^{d/2-a}$ as shown in Fig.7a. If the blocks are placed one after the other, the data organization is transformed into the form illustrated in Fig.7b. It is to be noted that every row of the data of Fig.7a has been split into $N^a$ parts and occur in $N^a$ different blocks of Fig.7b. Since the consecutiveness of the elements in a column are not lost in Fig.7b, computation

of DFT(N) of the columns is straightforward. Each of the $N^a$ blocks is circulated $d/2$ times through a DFT(N) computation block and a ROT1 unit. The number of DFT(N) circuits in each block is $N^{d/2-1}$. The rotator unit consists of $N^a$ ROT1 units

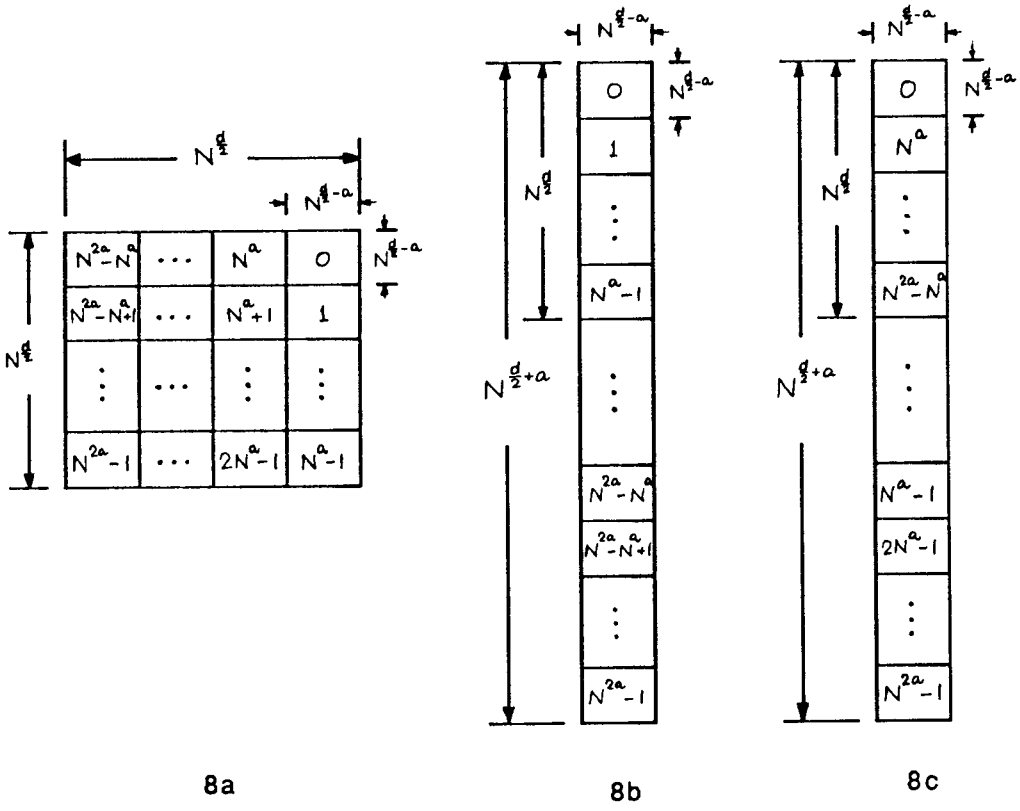



Fig.8a Partitioning data of size $n = N^d$ into $N^{2a}$ subblocks

8b Subblock organization

8c Reordering of subblocks after passing through BROT

each of size $O(N^d)$. Thus the total area occupied by the rotator unit is $N^a \times O(N^d)$ = $O(N^{d+a})$. The data is then passed through a transpose unit and a block rotator unit. The latter is necessary in order that the consecutiveness of the elements in the rows of Fig.7a are transformed into the consecutiveness of the elements in the columns of Fig.7b. We shall illustrate the procedure now. The data block of Fig.8a is partitioned into $N^{2a}$ subblocks each of size $N^{d/2-a} \times N^{d/2-a}$. These subblocks are arranged as shown in Fig.8b. The transpose unit consists of $N^{2a}$ subunits. Each

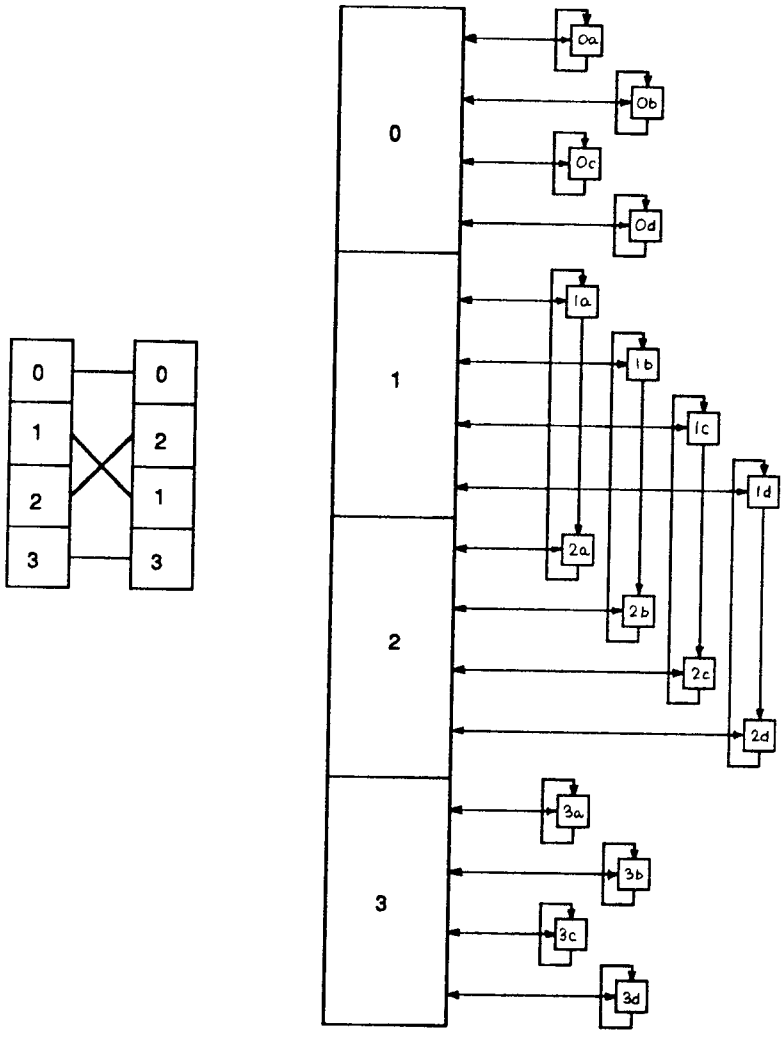



Fig.9 Layout of BROT with 4 subblocks and 4 elements/subblock

transpose subunit operates on a data subblock. The area of the transpose unit is $N^{2a}O(N^{d-2a}\log n) = \mathrm{O}(N^d \log n)$. Since the subunits operate parallely, the pipeline time complexity is $\mathrm{O}(N^{d/2-a}\log n)$. After transpose, though the rows and columns of a subblock are interchanged, the subblock ordering remains the same. The data is then passed through a block rotator unit BROT. Fig.8c illustrates the ordering of the subblocks after passing through BROT. The function of BROT is to rotate a data subblock such that the relative position between the elements in the subblock do not change. The layout of BROT is similar but not identical to ROT1. Since BROT

rotates the subblocks only once, every necklace in the layout of BROT would have at most 2 elements. In fact, all the elements of $N^{2a} - N^a$ subblocks form necklaces with one other element and all the rest form necklaces with themselves. The total number of columns required to rotate the subblocks is $O(N^{d/2+a})$. The area of BROT is thus $O(N^{d+2a})$. Fig.9 illustrates the layout of BROT with 4 subblocks and 4 elements/subblock. The block diagram for the design for Case B is shown in Fig.10.

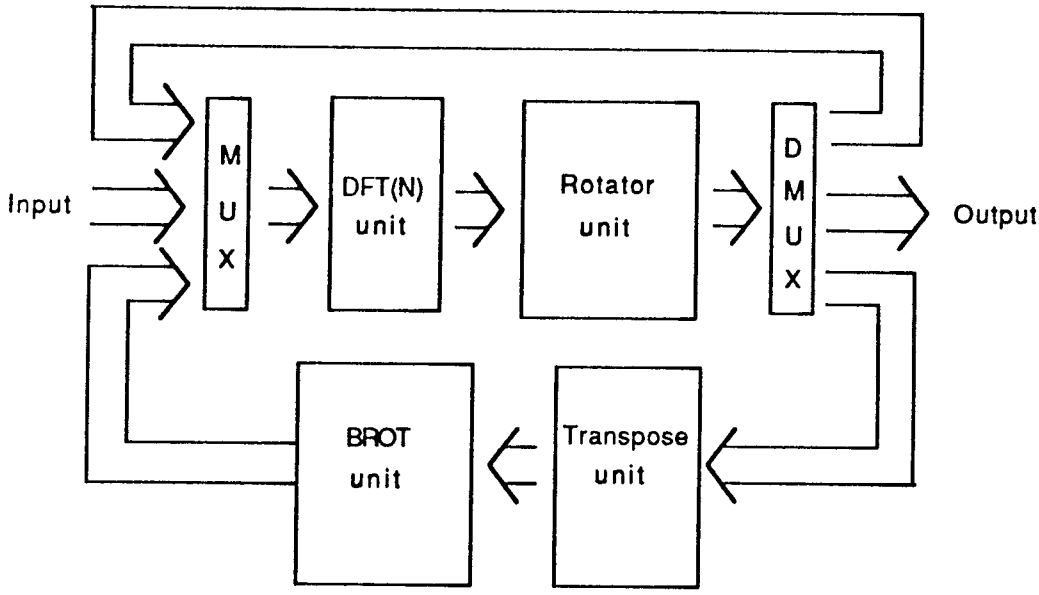



Fig.10 Architecture for Case B

The area of this design is dominated by the block rotator unit with area $O(N^{d+2a})$ for $\frac{1}{2}\log_N(\log n) \leq a \leq d/2$. The maximum pipeline time is the time taken to transpose the data. $T_{p_{max}} = O(N^{d/2-a}\log n)$. Thus $AT_p^2 = O(N^{2d}\log^2 n) = O(n^2\log^2 n)$ for all $a$ such that $\frac{1}{2}\log_N(\log n) \leq a \leq d/2$ and $N^a$ is an integer.

An interesting issue is the size of this family of optimal architectures. This happens to be a function of the number of values $a$ can take. For $N = 2^m$, the values that $f$ can take are $[\frac{1}{m}, \frac{2}{m}, \ldots, \frac{m-1}{m}]$ and the values that $i$ can take are $\hat{i} \leq i \leq d/2$, where $\hat{i} = \lceil \frac{1}{2}\log_N(\log n)\rceil$. Let $\hat{f}$ be the value of $f$ which is greater than and closest to $\frac{1}{2}\log_N(\log n)$. The number of possible values of $a$ is then $m(\frac{d}{2} - \hat{i} - \hat{f} + 1) + 1$.

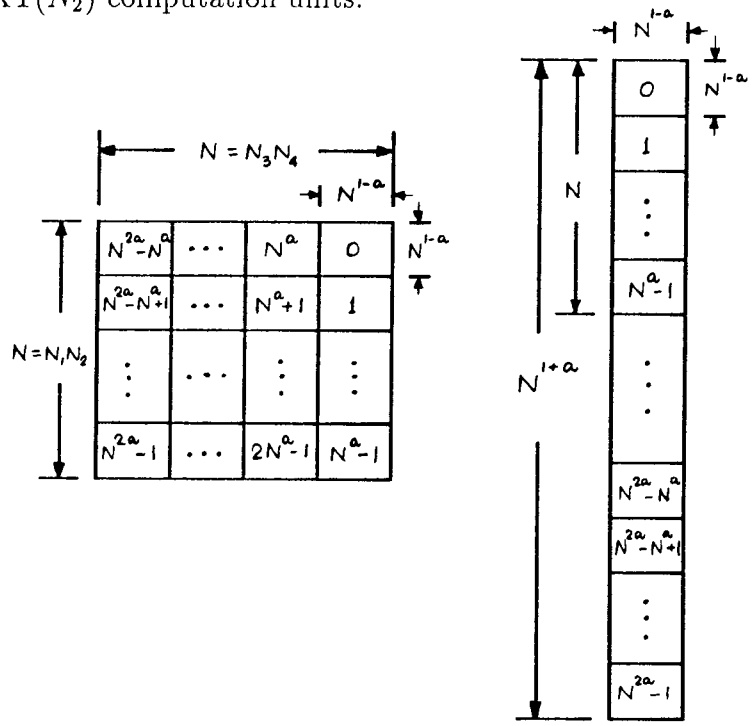
This is also the number of members of the family of optimal architectures.

## 5 Applications to specific transforms

In this section, we apply the method developed in Sec.4 to design architectures for some specific transforms like 2-D DFT, DHT and DCT. If the input data is in a 2-D array of size $N \times N$, then a straightforward way is to compute 1-D transform on the columns followed by 1-D transform on the rows. It is well known that the efficient way of computing a 1-D transform of $N$ points is to map it into a 2-D transform of $N_1 \times N_2 = N$ points and then compute the 1-D transforms over $N_1$ and $N_2$ points. So a 2-D transform over $N \times N$ points would be mapped into a 4-D transform over $N_1 \times N_2 \times N_3 \times N_4$, where $N_1N_2 = N_3N_4 = N$ and $N_1 = N_3$; $N_2 = N_4$. It is to be noted that the size of the family of optimal architectures does not change because of the 2-D to 4-D conversion. The bounds for $T_p$ are still $[O(\log n), O(\sqrt{n/\log n})]$ though the efficiency of the 1-D computation increases. Another point of interest is the sequence of the dimensions over which the transforms can be evaluated. In the case of 4-D DFT, the sequence is not at all important. DFT can be evaluated over the 3rd and 1st dimension followed by the 4th and 2nd dimension. In the case of 4-D DHT or DCT, evaluation over the 1st(3rd) dimension has to be followed by evaluation over the 2nd(4th) dimension. This is because whereas evaluation of DFT over one dimension is independent of the evaluation over the others, it is not so in DHT and DCT [CJ].

The 2-D $N \times N$ input array (as shown in Fig.11a) is decomposed into subblocks of size $N^{1-a} \times N^{1-a}$ and rearranged into an array of size $N^{1+a} \times N^{1-a}$ as shown in Fig.11b. Instead of referring to DFT, DHT and DCT separately, we refer to this group of transforms as DXT. Fig.12 illustrates our scheme for computing DXT. The data is first passed through $N^{1+a}/N_1 = N_2N^a$ DXT($N_1$) computation units. The rotator unit consists of $N^a$ ROT1 units, each of area O($N^2$). The rotated data is

passed through $N^{1+a}/N^2 = N_1 N^a$ DXT($N_2$) computation units. In case of DHT or DCT, the second computation unit takes care of any adjustments that have to be made. The data is then passed through the transpose and the block rotator units. Since $N_3 = N_1$ and $N_4 = N_2$, the data is circulated through the same DXT($N_1$), rotator and DXT($N_2$) computation units.




**11a** **11b**

Fig.11a Partitioning data of size $n = N^2$ into $N^{2a}$ subblocks

11b Subblock organization

In our scheme, there exist a family of optimal architectures with area $O(N^{2+2a})$ and pipeline time $O(N^{1-a} \log n)$, $\frac{1}{2} \log_N(\log n) \leq a \leq 1$. If we modify our design of the rotator unit such that it consists of one ROT1 unit with $N^{1+a}$ input lines and $O(N^{2+2a})$ area, and if $a$ is chosen to be $\log_N N_2$, then we can do away with the BROT unit. For most cases $a = \log_N N_2$ is greater than the lower bound of $a$. For this value of $a$, $N^{1-a} = N_1 = N_3$ and DXT($N_3$) can be computed directly after passing through the transpose unit. This is a considerable gain in terms of absolute area. We would refer to this design as the most practical of all optimal designs for 2-D DXT.

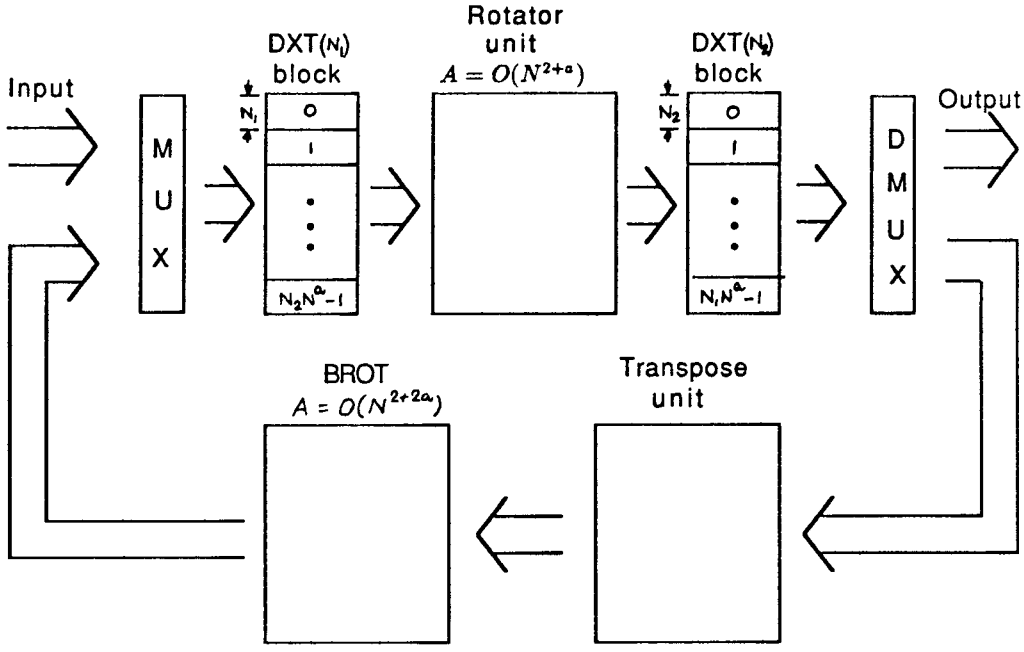



Fig.12 Architecture for 2-D DXT

## 6 Conclusion

In this paper we have proposed a family of optimal architectures for computing multidimensional transforms such as those of DFT, DHT and DCT. We have discussed the computation of multidimensional DFT in detail. The architectures for multidimensional DHT and DCT are very similar, the only difference being in the computation block of the 1-D transform. Since we know how to optimally compute 1-D DHT and DCT [CJ], optimal computation of multidimensional DHT and DCT is no different from that of DFT.

Gertner and Shamash [GS] have proposed an optimal architecture for computing multidimensional DFT on a single file of $n = N^d$ elements. The design consists of $N^{d-1}$ DFT(N) computation arrays and a rotator for permuting the data. The rotator with an area of $O(n^2)$ dominates the area of the design. The pipeline time complexity is $O(\log n)$. Thus this design achieves the $AT_p^2$ lower bound. We have shown that

if the input is in the form of a 2-D array of size $N^{d/2+a} \times N^{d/2-a}$, $0 < a \leq d/2$, we can design a family of optimal architectures for different values of $a$. The design of [GS] is a member of this family with $a = d/2$. We have studied two cases, A when $a$ is an integer and B when $a$ is not an integer but $N^a$ is an integer. The design for Case A consists of $N^{d/2+a-1}$ DFT(N) computation arrays, a transpose unit of size $(N^d \log n)$ and a rotator of size $O(N^{d+2a})$. $T_{p_{max}}$ for this design is $O(N^{d/2-a} \log n)$. Case B is slightly more complicated and consists of an additional block rotator unit BROT with area $O(N^{d+2a})$. For all these designs the lower bound of $AT_p^2$ is achieved.

The size of this family of optimal architectures is a function of the number of values $a$ can take. For $N = 16$ and $d = 4$, the number of members of this family is 7. We choose a particular member depending on whether minimization of area or minimization of pipeline time is more important. A design with low values of $a$ would have less area, whereas a design with large values of $a$ would require less time.

We know that the efficient way of computing 1-D linear transform over $N$ points is to map it into a 2-D transform of $N = N_1 N_2$ points and then compute 1-D transform over each dimension. Similarly an efficient way of computing 2-D transform of $N \times N$ points would be to map it into a 4-D transform and then compute 1-D transform over each dimension. We have designed optimal architectures for computing such transforms. The most practical design in this family of optimal architectures occurs when $a$ is chosen to be $\log_N N_2$. In this case even though $a$ is not an integer, we do not need the block rotator unit. This is a considerable saving in terms of absolute area.

## References


[BS] Bilardi,G.,Sarrafzadeh,M.,"Optimal VLSI circuits for the Discrete Fourier Transform", Advances in Computing Research, vol.4, pp.87-101.

[Ch] Chowdary, N.U.,et.al.,"A high speed two dimensional FFT processor", in Proc.Int.Conf.Acous.,Speech,Signal Processing, San Diego,CA,1984, pp.4.11.1-4.11.4.

[CJ] Chakrabarti,C.,JáJá,J.,"Optimal Systolic Designs for the Computation of the Discrete Hartley and the Discrete Cosine Transforms", UMIACS-TR-88-14, Feb'88.

[GS] Gertner,I.,Shamash,M.,"VLSI Architectures for Multidimensional Fourier Transform Processing", IEEE Trans. on Computers, vol.C-36, no.11, Nov'87, pp.1265-1274.

[PV] Preparata,F.P.,Vuillemin,J.E.,"Area-time optimal VLSI networks for computing integer multiplication and discrete Fourier transform", Proc.ICALP, Haifa,Israel, July'81, pp.29-40.

[Vu] Vuillemin,J."A combinatorial limit to the computing power of VLSI", IEEE Trans. on Computer, vol.C-32, Mar'83, pp.294-300.

[Zh] Zhang,C.H.,"Multi-dimensional systolic networks for discrete Fourier transform", Proc. 11th Annu.Int.Symp.Computer Architecture, Ann Arbor, MI, 1984, pp.21-27.