

Radix-4 Factorizations for the FFT with Ordered Input and Output

Vikrant¹, Ritesh Vyas², Sandeep Goyat³, Jitender Kumar⁴, Sandeep Kaushal⁵

YMCA University of Science & Technology, Faridabad (Haryana), India

Vikrant_bmit@rediffmail.com¹, ritesh.vyas157@gmail.com², sandeep.kandela85@gmail.com³, er.singlajitender@gmail.com⁴
sandeepkaushal@in.com⁵

Abstract: In this work we derive two families of radix-4 factorizations for the FFT (Fast Fourier Transform) that have the property that both inputs and outputs are addressed in natural order. These factorizations are obtained from another two families of radix-2 algorithms that have the same property. The radix-4 algorithms obtained have the same mathematical complexity (number of multiplications and additions) that Cooley-Tukey radix-4 algorithms but avoid de bit-reversal ordering applied to the input or at the output.

Index Terms- Fast Fourier Algorithms, Fast algorithms

I. INTRODUCTION:

The discrete Fast Fourier Transform algorithm, well known by the acronym FFT, has found a big deal of applications in engineering since it was first discovered by Gauss [1] and rediscovered by Cooley and Tukey [2] in the 1960s. In fact, under the acronym FFT we have a wide variety of algorithms and therefore there is a large bibliography about the field. Only to give a set of relevant references: there are algorithms referred to as higher radix [3][4], mixed-radix [5], prime-factor [6][19], Winograd [7], split-radix[8][9][20][21], identical geometry from stage-to-stage FFT [12], recursive [10], combination of decimation-in-time and the decimation-in-frequency [11], among many variants. An interesting overview on the estate of the art of FFT could be found in [13] and in [30].

Today, one of the interests in FFT research algorithms is to reduce its arithmetic complexity by minimizing the total number of real multiplications and additions as has been done recently in [21]. However, it is interesting to note that the performance of FFT on computers is determined by many other factors such as cache or central processing unit pipeline optimization; this is, the hardware in which the algorithm is computed.

Some matrix representations for FFT and other fast discrete signal transforms are found in [14], [15], [16], [17] and [18]. Following the matrix notation a fast algorithm can be thought as a sparse factorization of the transform matrix in which the new organization of operations reduces the complexity of the direct full matrix vector multiplication problem (of order N^2) drastically (to order $N \log_2 N$). But, as a result, the calculation of the FFT in terms of sparse factors provided by almost all the factorizations, the output vector appears disordered. So, in practice, many FFT algorithms need some input or output data

permutation and the bitreversa ordering is the one that most frequently appears. In some algorithms the permutation is applied at the input and in others it is applied at the output. Although the bit-reversal ordering has a very efficient hardware implementation, its software implementation has been recent improved in [22 - 27]. In most applications the order must be re-established by performing a permutation of the elements.

On concerning FFT ordered algorithms, in [29] they are proposed to be used in vector processors. In reference [28] appear two flow graphs for an eight point FFT sorted algorithms according to Stockham. In [28] the authors change the Cooley-Tukey flow graphs to obtain the other algorithms presented in it by applying flow graph transform rules; method which is easy only in some cases and above all when the flow graph has reduced dimensions. The Stockham algorithms are not derived in [28] in which a unique reference links to a private communication. In [30] input-output FFT ordered algorithms are derived from the Cooley-Tokey factorizations inserting a -different - permutation matrix between factors. In [31] two recursive properties involving matrix F_N and $F_{N/2}$ are presented and by iterating them and with a little of algebra, the sorted radix-2 algorithms are easily obtained. In this work, we extend the radix-2 results from [31] to obtain radix-4 solutions. To do this we have considered two ways. One is based on the extension of the recursion properties given in [31] to relate the matrices F_N and $F_{N/4}$ and then, using a similar method, to derive the factorization. The other way we have chosen is based on the idea that a radix-4 factor can be written in terms of two consecutive radix -2 factors. This way provides more detailed information about the structure of the radix -4 basic operation, sometimes called dragonfly, in order of performing them efficiently. A dragonfly operation computes groups of four output elements from groups of 4 input elements. Our goal is to obtain radix-4 factorizations that avoid the reordering operation.

In section 2 we briefly present the used notation. In section 3 we present the radix-2 sorted algorithms derived in [31]. In section 4 we derive the radix-4 sorted algorithms and finally we give some conclusions.

II. USED NOTATION

Given that in this work we always manage square matrices in what follows, an $N \times N$ square matrix is denoted by a bold capital letter with subscript N . Number N is always a power of two.

The elements of matrix A_N positioned at the row m and the column n are denoted by a_{mn} . Sometimes we will use the notation $A_N = \{a_{mn}\}$.

A column vector is represented by a bold small letter and, since its length can always be known from the context in this paper, its subscript indicates the position of the column in a matrix.

The $N \times N$ identity matrix is denoted by I_N . Then I_N can be written by its column vectors e_i as $I_N = [e_1 e_2 \dots e_n]$. With O_N we denote the $N \times N$ zero matrix.

The matrix P_N is the $N \times N$ even-odd permutation matrix. P_N in terms of the previously defined vectors e_i takes the form $P_N = [e_1 e_3 \dots e_{n-1} e_2 e_4 \dots e_n]$.

Most of the times we will use the Kronecker product to show a particular matrix structure. The symbol \otimes stands for the right Kronecker product and, for arbitrary square matrices A_M and B_N , the Kronecker product $A_M \otimes B_N$ is an $M_N \times M_N$ matrix that can be written using the elements a_{mn} of matrix A_M and B_N as:

$$A_M \otimes B_N = \begin{bmatrix} a_{11} B_N & \dots & a_{1M} B_N \\ \dots & \dots & \dots \\ a_{M1} B_N & \dots & a_{MM} B_N \end{bmatrix} \quad (1)$$

Next, we recall some useful properties involving the Kronecker product. We have:

$$I_{2^{n1}} \otimes I_{2^{n2}} = I_{2^{n1+n2}} \quad (2)$$

$$(A_M \otimes B_N)(C_M \otimes D_N) = A_M C_M \otimes B_N D_N \quad (3)$$

Note that superscript n in a matrix means the power n of this matrix. Finally, the factorization of an arbitrary matrix M_N in terms of n factors (or stages) $E_N(i)$ is written as follows:

$$M_N = \prod_{i=1}^n E_N(i) = E_N(n) \dots E_N(2) E_N(1) \quad (4)$$

III. FFT RADIX-2 ALGORITHMS WITH ORDERED INPUT AND OUTPUT DATA

Consider $N=2^n$ and j the square root of -1 . The Fourier transform matrix F_N is defined as:

$$F_N = \left\{ e^{-j \frac{2\pi}{N} pq} \right\} \quad p, q = 0 : N-1, \quad (5)$$

Being $x = [x(1) \dots x(N)]^T$ the ordered input vector, the ordered transformed vector $y = [Y(1) \dots Y(N)]^T$ is obtained by performing the operation $y = F_N x$

To obtain radix-2 FFT algorithms with input and output ordered data in [31] were introduced two recursion properties involving matrix F_N and matrix $F_{N/2}$.

Let B_{2^i} denote the matrix defined by:

$$B_{2^n} = \begin{bmatrix} I_{2^{n-1}} & A_{2^{n-1}} \\ I_{2^{n-1}} & -A_{2^{n-1}} \end{bmatrix} \quad (6)$$

Where

$$A_N = \text{diag} \left\{ e^{-j \frac{\pi}{N} p^2} \right\} \quad p = 0 : N-1. \quad (7)$$

is a diagonal matrix. Equation (6) can also be written as:

$$B_{2^n} = H_{2^n} \begin{bmatrix} I_{2^{n-1}} & O_{2^{n-1}} \\ O_{2^{n-1}} & A_{2^{n-1}} \end{bmatrix} \quad (8)$$

being

$$H_{2^n} = \begin{bmatrix} I_{2^{n-1}} & I_{2^{n-1}} \\ I_{2^{n-1}} & -I_{2^{n-1}} \end{bmatrix} \quad (9)$$

Then, the two recursions are:

$$F_{2^n} = B_{2^n} P_{2^n} (F_{2^{n-1}} \otimes I_2) \quad (10)$$

$$F_{2^n} = (F_{2^{n-1}} \otimes I_2) P_{2^n}^T B_{2^n}^T \quad (11)$$

By iterating (10) and (11) in [31] the next two sets of solutions are obtained.

From (10) the full factorization takes the form:

$$F_{2^n} = \prod_{i=1}^n (B_{2^i} P_{2^i} \otimes I_{2^{n-i}}) \quad (12)$$

with factors (stages) taking the form:

$$B_{2^i} P_{2^i} \otimes I_{2^{n-i}} = H_{2^i} \begin{bmatrix} I_{2^{i-1}} & O_{2^{i-1}} \\ O_{2^{i-1}} & A_{2^{i-1}} \end{bmatrix} P_{2^i} \otimes I_{2^{n-i}} \quad (13)$$

From (11) the full factorization takes the form:

$$\mathbf{F}_{2^n} = \prod_{i=1}^n \left(\mathbf{P}_{2^{n-i+1}}^T \mathbf{B}_{2^{n-i+1}}^T \otimes \mathbf{I}_{2^{i-1}} \right) \quad (14)$$

with factors taking the form:

$$\mathbf{P}_{2^{n-i+1}}^T \mathbf{B}_{2^{n-i+1}}^T \otimes \mathbf{I}_{2^{i-1}} = \mathbf{P}_{2^{n-i+1}}^T \begin{bmatrix} \mathbf{I}_{2^{n-i}} & \mathbf{0}_{2^{n-i}} \\ \mathbf{0}_{2^{n-i}} & \mathbf{A}_{2^{n-i}} \end{bmatrix} \mathbf{H}_{2^{n-i+1}} \otimes \mathbf{I}_{2^{i-1}} \quad (15)$$

Note that in (12) and (14) the results are presented in an ordered form and no permutation matrices appear at the beginning or at the end of the factor chain.

In both cases the basic operation that the factors perform is called (radix-2) butterfly and it is represented graphically in fig. 1.



Fig. 1. Radix-2 butterfly representation showing the dependence between two inputs and two outputs.

Fig.2 stands for the interconnection pattern of a FFT of N=8 point in terms of butterflies using the factorization given in (12) and Fig. 3 stands for the interconnection pattern of a FFT of N=8 point in terms of butterflies using the factorization given in (14).

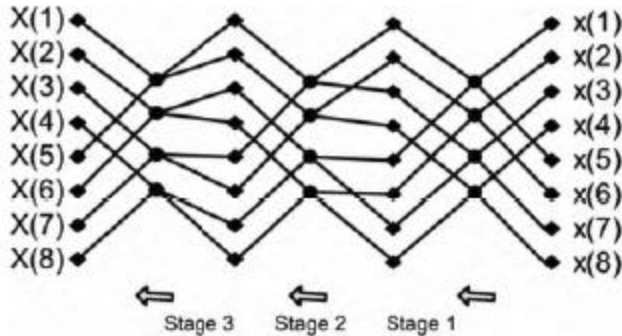


Fig.2. Interconnection pattern representation for the first factorization in terms of butterflies given N=8. Inputs and outputs are addressed in natural order.

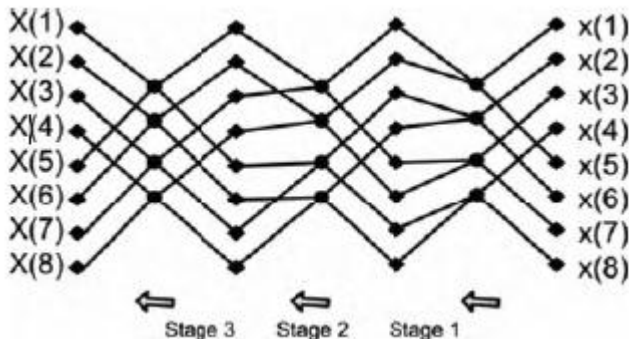


Fig.3. Interconnection pattern representation for the second factorization in terms of butterflies given N=8. Inputs and outputs are addressed in natural order.

IV. FFT RADIX-4 ALGORITHMS WITH ORDERED INPUT AND OUTPUT DATA

If N, the length of the transform, is a power of 4 we can obtain radix-4 decompositions. Of course, if N is a power of 4 it is also a power of 2.

In a more general point of view, take $R = 2^F$ (being R the radix of the decomposition) and consider that N satisfies that $N = 2^n = R^m$. Then any discrete Fourier transform of size N admits a radix-R factorization with F radix-R factors $E'(i)$ that can easily be written as a product of F successive radix 2 factors $E(i)$:

$$\mathbf{E}'_N(g) = \prod_{f=1}^F \mathbf{E}_N(F(g-1) + f) \quad (16)$$

The radix-R factorization, using the notation introduced in (16), becomes:

$$\prod_{g=1}^m \mathbf{E}'_N(i) \quad (17)$$

Then in the radix-4 case in which we are interested we have $R=4$ and $F=2$. Given (12) and (16) the new radix-4 factorization takes the form:

$$\mathbf{F}_N = \mathbf{F}_{2^n} = \mathbf{F}_{4^m} = \prod_{g=1}^m \left(\prod_{f=1}^2 \left(\mathbf{B}_{2^{2(g-1)+f}} \mathbf{P}_{2^{2(g-1)+f}} \otimes \mathbf{I}_{2^{n-(2(g-1)+f)}} \right) \right) \quad (18)$$

If we analyze the radix-4 factors in (18) written as products of two consecutive radix-2 ones and, by applying the properties (2) and (3), we have:

$$\begin{aligned} & \prod_{f=1}^2 \left(\mathbf{B}_{2^{2(g-1)+f}} \mathbf{P}_{2^{2(g-1)+f}} \otimes \mathbf{I}_{2^{n-(2(g-1)+f)}} \right) = \\ & \left(\mathbf{B}_{2^{2(g-1)+2}} \mathbf{P}_{2^{2(g-1)+2}} \otimes \mathbf{I}_{2^{n-(2(g-1)+2)}} \right) \times \\ & \left(\mathbf{B}_{2^{2(g-1)+1}} \mathbf{P}_{2^{2(g-1)+1}} \otimes \mathbf{I}_{2^{n-(2(g-1)+1)}} \right) = \\ & \left(\mathbf{B}_{2^{2(g-1)+2}} \mathbf{P}_{2^{2(g-1)+2}} \otimes \mathbf{I}_{2^{n-(2(g-1)+2)}} \right) \times \\ & \left(\mathbf{B}_{2^{2(g-1)+1}} \mathbf{P}_{2^{2(g-1)+1}} \otimes \mathbf{I}_2 \otimes \mathbf{I}_{2^{n-(2(g-1)+2)}} \right) = \\ & \mathbf{B}_{2^{2(g-1)+2}} \mathbf{P}_{2^{2(g-1)+2}} \left(\mathbf{B}_{2^{2(g-1)+1}} \mathbf{P}_{2^{2(g-1)+1}} \otimes \mathbf{I}_2 \right) \otimes \mathbf{I}_{2^{n-(2(g-1)+2)}} \end{aligned} \quad (19)$$

So, the first family of radix-4 factorization takes the form:

$$\mathbf{F}_{4^m} = \prod_{g=1}^m \mathbf{B}_{2^{2g}} \mathbf{P}_{2^{2g}} (\mathbf{B}_{2^{2g-1}} \mathbf{P}_{2^{2g-1}} \otimes \mathbf{I}_2) \otimes \mathbf{I}_{2^{n-2g}} \quad (20)$$

The other family that can be found, now from (14) and (16), is

$$\mathbf{F}_N = \mathbf{F}_{2^n} = \mathbf{F}_{4^m} = \prod_{g=1}^m \left(\prod_{f=1}^2 (\mathbf{P}_{2^{n-(2(g-1)+f)+1}}^T \mathbf{B}_{2^{n-(2(g-1)+f)+1}}^T \otimes \mathbf{I}_{2^{2(g-1)+f-1}}) \right), \quad (21)$$

and if we rewrite the radix-4 factors in (21) using properties (2) and (3), we have:

$$\begin{aligned} & \prod_{f=1}^2 (\mathbf{P}_{2^{n-(2(g-1)+f)+1}}^T \mathbf{B}_{2^{n-(2(g-1)+f)+1}}^T \otimes \mathbf{I}_{2^{2(g-1)+f-1}}) = \\ & (\mathbf{P}_{2^{n-2g+1}}^T \mathbf{B}_{2^{n-2g+1}}^T \otimes \mathbf{I}_{2^{2g-1}}) (\mathbf{P}_{2^{n-2g+2}}^T \mathbf{B}_{2^{n-2g+2}}^T \otimes \mathbf{I}_{2^{2g-2}}) = \\ & (\mathbf{P}_{2^{n-2g+1}}^T \mathbf{B}_{2^{n-2g+1}}^T \otimes \mathbf{I}_2 \otimes \mathbf{I}_{2^{2g-2}}) (\mathbf{P}_{2^{n-2g+2}}^T \mathbf{B}_{2^{n-2g+2}}^T \otimes \mathbf{I}_{2^{2g-2}}) \\ & = (\mathbf{P}_{2^{n-2g+1}}^T \mathbf{B}_{2^{n-2g+1}}^T \otimes \mathbf{I}_2) \mathbf{P}_{2^{n-2g+2}}^T \mathbf{B}_{2^{n-2g+2}}^T \otimes \mathbf{I}_{2^{2g-2}} \end{aligned} \quad (22)$$

So we obtain the second family of factorizations as:

$$\mathbf{F}_{4^m} = \prod_{g=1}^m (\mathbf{P}_{2^{n-2g+1}}^T \mathbf{B}_{2^{n-2g+1}}^T \otimes \mathbf{I}_2) \mathbf{P}_{2^{n-2g+2}}^T \mathbf{B}_{2^{n-2g+2}}^T \otimes \mathbf{I}_{2^{2g-2}} \quad (23)$$

What is interesting from expressions (20) and (23) is that, using the representation of matrices B given in (8), we can obtain a very efficient dragonfly with the minimum number of complex multiplications.

In these cases the basic operation that the factors perform is called radix-4 butterfly or dragonfly and it is represented graphically in fig A.

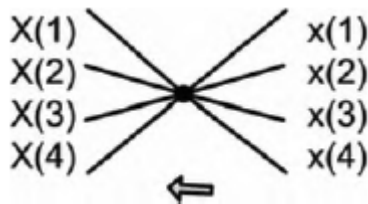


Fig.4. Radix-4 butterfly represent at ion showing the dependence between four inputs and four outputs.

A way to see the relation between inputs and outputs stage to-stage could be done by analyzing the positions of the non-zero elements of each factor. Then, from the indices m, n of the non-zero elements in each sparse matrix representing a stage (or factor), we can observe that the n input element is needed to calculate the m output element in the i-th stage. We can

observe that four elements are needed at the input to calculate four outputs in the basic operation graphically represented in fig A. To better see this, as an example, consider the expression (20) when m=2 and n=4. We have:

$$\begin{aligned} \mathbf{F}_{16} &= \\ & (\mathbf{B}_{2^4} \mathbf{P}_{2^4} (\mathbf{B}_{2^3} \mathbf{P}_{2^3} \otimes \mathbf{I}_2) \otimes \mathbf{I}_{2^0}) (\mathbf{B}_{2^2} \mathbf{P}_{2^2} (\mathbf{B}_{2^1} \mathbf{P}_{2^1} \otimes \mathbf{I}_2) \otimes \mathbf{I}_{2^2}) \\ & = (\mathbf{B}_{2^4} \mathbf{P}_{2^4} (\mathbf{B}_{2^3} \mathbf{P}_{2^3} \otimes \mathbf{I}_2)) (\mathbf{B}_{2^2} \mathbf{P}_{2^2} (\mathbf{B}_{2^1} \mathbf{P}_{2^1} \otimes \mathbf{I}_2) \otimes \mathbf{I}_{2^2}) \end{aligned}$$

To show the mapping between the mathematics and the graphs lets only consider de first stage of F16. To obtain it step by step, initially considerer matrix $\mathbf{B}_2 \mathbf{P}_2$, this is:

$$\mathbf{B}_2 \mathbf{P}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

which computes two outputs from two inputs. The operation

$$\mathbf{B}_2 \mathbf{P}_2 \otimes \mathbf{I}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

Do not change the input-output relation. Next product also preserves the 2 input-2 output relation

$$\mathbf{B}_4 \mathbf{P}_4 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -j \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & j \end{bmatrix}$$

But the term

$$\mathbf{B}_4 \mathbf{P}_4 (\mathbf{B}_2 \mathbf{P}_2 \otimes \mathbf{I}_2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Clearly changes the input-output relation (4 inputs/ 4 outputs) in the way represented in fig 4. Finally, the stage will be:

$$B_4 P_4 (B_2 P_2 \otimes I_2) \otimes I_4 =$$

1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	-j	0	0	0	-1	0	0	0	j	0	0	0
0	1	0	0	0	-j	0	0	0	-1	0	0	0	j	0	0
0	0	1	0	0	0	-j	0	0	0	-1	0	0	0	j	0
0	0	0	1	0	0	0	-j	0	0	0	-1	0	0	0	j
1	0	0	0	-1	0	0	0	1	0	0	0	-1	0	0	0
0	1	0	0	0	-1	0	0	0	1	0	0	0	-1	0	0
0	0	1	0	0	0	-1	0	0	0	1	0	0	0	-1	0
0	0	0	1	0	0	0	-1	0	0	0	1	0	0	0	-1
1	0	0	0	j	0	0	0	-1	0	0	0	-j	0	0	0
0	1	0	0	0	j	0	0	0	-1	0	0	0	-j	0	0
0	0	1	0	0	0	j	0	0	0	-1	0	0	0	-j	0
0	0	0	1	0	0	0	j	0	0	0	-1	0	0	0	-j

where we can see four identical radix-4 butterflies. This factor (or stage) is represented as stage I in fig. I. In the case of doing the same operations with the second factor of F_{16} we will obtain the stage 2.

Fig. 5 stands for the interconnection pattern of a FFT of $N=16$ points in terms of dragonflies using the factorization given in (20) and Fig. 6 stands for the interconnection pattern of a FFT of $N=16$ points in terms of dragonflies using the factorization given in (23). In both cases we have 2 stages. Note that both topologies are symmetric.

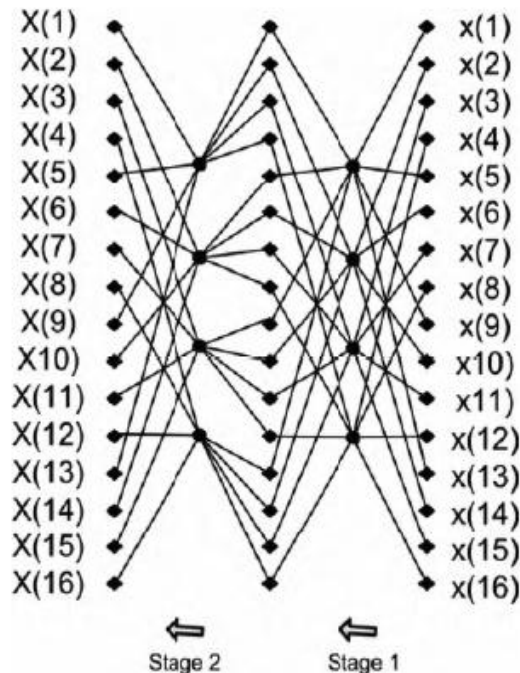


Fig. 5. Interconnection pattern representation for radix -4 first factorization in terms of dragonflies given $N= 16$. Inputs and outputs are addressed in natural order.

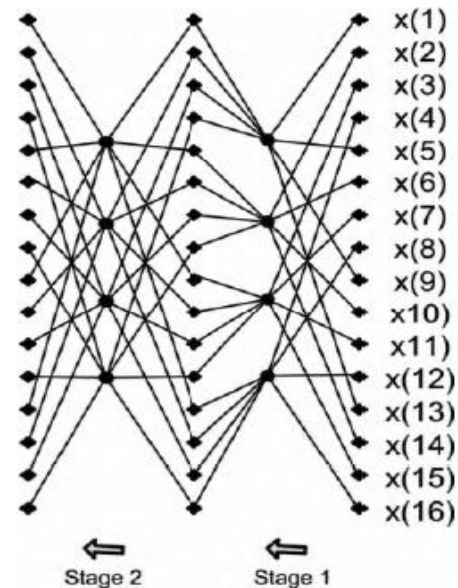


Fig. 6. Interconnect ion pattern representation for radix-4 second factorization in terms of dragonflies given $N=16$. Inputs and outputs are addressed in natural order

V. CONCLUSIONS

This work extends the radix-2 families of factorizations presented in [31] that have the property that both inputs and outputs are addressed in natural order to obtain another two radix-4 factorizations with the same property. The algorithms obtained have the same complexity in terms of floating-point operations that the well programmed CooleyTukey radix-4 algorithms but avoiding the bit-reversal ordering applied at the input samples or at the output. To achieve a full radix-4 decomposition N , the length of the transform, must be a power of four. We observe also that the two solutions have a symmetric interconnection pattern.

REFERENCES

- [1] M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the History of the FFT," IEEE Acoustics, Speech, and Signal Processing Magazine, vol. 1, pp. 14-21, Oct. 1984.
- [2] J. W. Cooley, I. W. Tukey "An Algorithm for the Machine Calculation of Complex Fourier Series". Math. Of Computations vol. 19, p.p. 297-301, Apr. 1965.
- [3] G. D. Bergland, "A Radix-Eight Fast-Fourier Transform Subroutine for Real-Valued Series," IEEE Trans. Audio Electroacoust, vol. 17, no. 2, pp. 138-144, June 1969.
- [4] D. Takahashi, "A Radix-16 FFT Algorithm Suitable for Multiply-Add Instruction based on Goedecker Method" Intern. Conference on Acoustics, Speech, and Signal Processing, ICASSP-2003, vol. 2, pp. II - 665-668, 6-10 April 2003.
- [5] R. C. Singleton, "An Algorithm for Computing the Mixed Radix Fast Fourier Transform" IEEE Trans. Audio Electroacoust. vol. 1, no. 2, pp. 93-103, June 1969.

- [6] D. PI. Kolba and T. W. Parks, "A Prime Factor FFT Algorithm Using High-Speed Convolution," IEEE Trans. Acoust., Speech, Signal Processing, vol. 25, no. 4, pp. 2812-94, Aug. 1977.
- [7] S. Winograd, "On Computing the Discrete Fourier Transform," Math. Comput., vol. 32, no. 141, pp. 175-199, Jan. 1978.
- [8] H. V. Sorensen and C. S. Burrus, "A New Efficient Algorithm for Computing a Few DFT Points," IEEE Trans. Acoust., Speech, Signal Proc., vol. 35, no. 6, pp. 849-863, June 1987.
- [9] D. Takahashi, "An Extended Split-Radix FFT Algorithm," IEEE Signal Processing Letters, vol. 8, no. 5, pp. 145-147, May 2001.
- [10] A. R. Varkonyi-Koczy, "A Recursive Fast Fourier Transform Algorithm," IEEE Trans. Circuits and Systems, II, vol. 42, pp. 614-616, Sep. 1995.
- [11] A. Saidi, "Decimation-in-Time-Frequency FFT Algorithm," Proc. IEEE International Conf. on Acoustics, Speech, and Signal Processing, vol. 3, pp. 453-456, 19-22 April 1994.
- [12] M. C. Pease, "An adaptation of the fast Fourier transform for parallel processing". J. Assoc. Comput. vol. 15, pp. 252-264, April 1968.
- [13] P. Duhamel and M. Vetterli "Fast Fourier transforms: A tutorial review and a state of the art" Signal Process., vol. 19, pp. 259-299, 1990.
- [14] A. Glassman, "A generalization of the fast Fourier transform," IEEE Trans. Comput., vol. C-19, pp. 105-116, Feb. 1970.
- [15] M. Drubin, "Kronecker product factorization of the FFT matrix," IEEE Trans. Comput., vol. C-20, pp. 590-593, May 1971.
- [16] H. Sloate, "Matrix Representations for Sorting and the Fast Fourier Transform" IEEE Trans.on Circuits and Systems, vol. 21, no. 1, pp. 109-116, January 1974
- [17] Granata, M. Conner, R. Tolimieri, "Recursive Fast Algorithms and the Role of the Tensor Product" IEEE Trans.on Signal Proc., vol.40, no. 12, pp. 2921-2930, Dec 1992.
- [18] S. Egnér, M. Puschel, "Automatic Generation of Fast Discrete Signal Transforms" IEEE Trans.on Signal Proc., vol. 49, no. 9, pp. 1992-2002, Dec. 2001.
- [19] S. C. Chan and K. L. Ho, "On indexing the prime-factor fast Fourier transform algorithm" IEEE Trans. Circuits and Systems, vol. 38, no. 8, pp. 951-953, 1991.
- [20] P. Duhamel, H. Hollmann, "Split-radix FFT algorithm," Electron. Lett. vol. 20, no. 1, pp. 14-16, 1984.
- [21] S. G. Johnson and M. Frigo, "A modified split-radix FFT with fewer arithmetic operations", IEEE Trans. Signal Processing, vol. 55, no. 1, pp. 111-119, 2007.
- [22] R.J. Polge; B.K. Bhagavan, I.M. Carswell, "Fast Computational Algorithms for Bit Reversal" IEEE Transactions on Computers, vol. C-23, no. 1, pp. 1-9, Jan. 1974.
- [23] S. Walker, "A new bit reversal algorithm"; IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 38, no. 8, pp. 1472-1473, Aug. 1990.
- [24] A. Biswas, "Bit reversal in FFT from matrix viewpoint IEEE Transactions on Signal Processing", vol. 39, no. 6, pp. 1415-1418, June 1991.
- [25] A. Edelman, S. Heller, S. Lennart Johnsson, "Index transformation algorithms in a linear algebra framework" IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 12, pp. 1302-1309, Dec. 1994.
- [26] K. Drouiche, "A new efficient computational algorithm for bit reversal mapping", IEEE Transactions on Signal Processing, vol. 49, no. 1, pp. 251-254, Jan. 2001
- [27] Prado; "A new fast bit-reversal permutation algorithm based on a symmetry", IEEE Signal Processing Letters, vol. 11, no. 12, pp. 933-936, Dec. 2004.
- [28] W. T. Cochrane, I. W. Cooley, I. W. Favin, D. L. Helms, R.A. Kaenel, W. W. Lang, G. C. Mailing, D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the fast Fourier transform?", IEEE Trans. Audio Electroacoust., vol. 15, pp. 45-55, 1967.
- [29] P. N. Swartztrauber, "FFT Algorithms for Vector Computers", Parallel Computing, col. 1, pp. 45-63, 1984.
- [30] C. Van Loan, "Computational Frameworks for the Fast Fourier Transform", SIAM Press, Philadelphia, PA, 1992.
- [31] P. Marti-Puig, "Two Families of Radix-2 FFT Algorithms with Ordered Input and Output Data ", IEEE Signal Processing Letters, vol. 16, no. 2, pp. 65-68, Feb. 2009.
- [32] P. Marti-Puig, R.R Bolano " Radix-4 FFT Algorithms with ordered Input and Output Data DSP 2009