ALGORITHM OF DEFINING 1-D INDEXING FOR M-D MIXED RADIX FFT IMPLEMENTATION

Chwen-Jye Ju, Ph.D.

Sharp Microelectronics Technology, Inc.
5700 NW Pacific Rim Blvd.
Camas, WA 98607, USA

ABSTRACT

Multi-dimensional (M-D) fast Fourier transform (FFT) is an essential algorithm in array signal processing. However, the calculation of M-D indexing and transposition of data matrix required by the M-D FFT are the algorithm's performance killer. The paper will propose a novel M-D to 1-D FFT signal flow graph (SFG) mapping. Thus, the M-D FFT can be efficiently implemented by the unified 1-D indexing and the address generator design can be simplified. In addition, the matrix transposition is no longer necessary. Finally, practical chip design consideration in implementing the algorithm is given.

1. INTRODUCTION

In recent decades, the fast Fourier transform algorithm has been a driving force to the progress of digital signal processing. With the advance of the VLSI technology, the FFT algorithm has been pushed further in solving the multidimensional array signal processing in real-time. However, there is no efficient addressing method for 1-D to M-D FFTs. Therefore, the paper will conquer this problem and propose a unified addressing for 1-D to M-D FFTs. All the M-D indexing can be simplified and implemented by 1-D indexing. The proposed approach has been implemented by many companies in their high-end systems such as radar, medical image recovery, etc.

A novel vector-matrix representation of 1-D to M-D radix-2 FFT algorithms has been discussed in [1,2]. It is shown that the M-D FFT has the same matrix form as the 1-D FFT if both have the same number of data. This implies that the SFG structure of the M-D FFT can be mapped to that of the 1-D FFT. Thus, the unified 1-D indexing can be applied to the M-D FFT. This paper will extend the radix-2 FFT results to the mixed radix FFT case.

For definiteness, this paper only discusses the decimation-in-time digit-reverse-input and normal-output FFT algorithms. Section 2 introduces an easy way of constructing an M-D FFT SFG structure. The required M-D FFT addressing sequences including digit-reverse, data, and twiddle factor are defined in Section 3, 4, and 5. Section 6 investigates the practical design consideration of the algorithm. The unified indexing for 1-D to M-D FFT algorithms has been implemented in the array processor chip set LH9124/LH9320 developed by Sharp Microelectronics Technology [3,4]. It can be seen from the chip set implementation that the proposed M-D FFT approach has tremendous advantages over the traditional M-D FFT approach in both cost and performance.

2. M-D FFT SIGNAL FLOW GRAPH

It is well-known that the twiddle factor matrix of the DFT can be recursively partitioned into the multiplication of the butterfly stage (BS) matrices [5,6]. These matrices can also be represented by cascading butterfly stages of the FFT signal flow graph as shown in Fig. 1. Thus, the SFG structure of the 1-D FFT can be represented by

$$SFG = BS_{11} @ BS_{12} @ \cdots @ BS_{1s_1} \qquad (1)$$

where s_1 denotes the number of FFT stages and "@" is a cascading operator. BS_{1j} can be an arbitrary radix-n_{1j} butterfly stage. Thus, Fig. 1 can be represented by 3@2@2@3.

2.1. 2-D FFT Signal Flow Graph

If the 2-D FFT is implemented by the 1-D FFT, we can select either the row-column or column-row approach. For definiteness in implementation, we will define an array mapping. For a 2-D array with the row length L_1 and column length L_2, the 2-D array mapping for the row-column approach will be $(N_1,N_2) = (L_1,L_2)$ and the column-row approach will be $(N_1,N_2) = (L_2,L_1)$. Thus, the SFG structure of the 2-D FFT can be represented by that of the 1-D FFT with the length $N=N_1*N_2$. If the SFG structure of the N_1-point FFT is

$$SFG_1 = BS_{11} @ BS_{12} @ \cdots @ BS_{1s_1} \qquad (2)$$

and that of the N_2-point FFT is

$$SFG_2 = BS_{21} @ BS_{22} @ \cdots @ BS_{2s_2}, \qquad (3)$$

then the SFG structure of the 2-D FFT will be

$$SFG = SFG_1 @ SFG_2. \qquad (4)$$

Fig. 1 shows the mapped SFG structure of a 6 by 6 2-D FFT. The N_1-point FFT is implemented by one radix-3 stage followed by one radix-2 stage. The N_2-point FFT is implemented by one radix-2 stage followed by one radix-3 stage. Thus, the 2-D FFT can be implemented as the 36-point 1-D FFT with SFG structure 3@2@2@3 if the input, output and twiddle factor sequences are properly defined.

2.2. M-D FFT Signal Flow Graph

The 2-D to 1-D FFT SFG mapping can be directly extended to the M-D case. Set an M-D array $(L_1,L_2,\cdots,L_M)$ with the length of the j-th tuple L_j. There are two approaches, row-column and column-row, to implement the 2-D FFT. However, there are M! approaches to implement the M-D FFT. For definiteness, the order of the M-D FFT implementation will be mapped to the M-D array $(N_1,N_2,\cdots,N_M)$. The total number of points of the mapped 1-D FFT will be

$$N = N_1 * N_2 * \cdots * N_M . \quad (5)$$

The SFG structure of the M-D FFT is represented by

$$SFG = SFG_1 @ SFG_2 @ \cdots @ SFG_M \quad (6)$$

where SFG_i can be further partitioned into

$$SFG_i = BS_{i1} @ BS_{i2} @ \cdots @ BS_{is_i} . \quad (7)$$

3. DIGIT-REVERSE SEQUENCE

It is well-known for the in-place FFT algorithm that if the input array is in normal order, then the output array after FFT operations will be in digit-reverse order and vice versa. This section will investigate how to define addressing for the M-D normal and digit-reverse arrays. Thus, those M-D arrays can be efficiently implemented by 1-D addressing.

3.1. 2-D Digit-Reverse Sequence

Given a 2-D array $[(n_1,n_2)]$, after its discrete Fourier transform we may get another 2-D array $[(k_1,k_2)]$ in normal order as following mapping

$$[(n_1,n_2)] \xrightarrow{2\text{-}D\ DFT} [(k_1,k_2)] . \quad (8)$$

Moreover, for the 2-D array after the 2-D FFT operations, we may get the 2-D array in digit-reverse order as following mapping

$$[(n_1,n_2)] \xrightarrow[R\text{-}C\ 2\text{-}D\ FFT]{C\text{-}R\ 2\text{-}D\ FFT} [(dr(k_1),dr(k_2))] . \quad (9)$$

The following will show what the mapping will be if only 1-D addressing is employed. The 1-D addressing for the 2-D normal array can be defined in the last-tuple-major order as

$$nr(n_1,n_2) = n_1 * N_2 + n_2 = n_1 n_2$$

$$= n_{1s_1} n_{1(s_1-1)} \cdots n_{11} n_{2s_2} \cdots n_{21} . \quad (10)$$

Then, the digit-reverse addressing of the normal 2-D array can be derived as

$$dr(nr(n_1,n_2)) = dr(n_1 n_2) = dr(n_2) * N_1 + dr(n_1)$$

$$= nr(dr(n_2),dr(n_1)) \quad (11)$$

and that of the 2-D digit-reverse array can be derived as

$$dr(nr(dr(n_1),dr(n_2))) = nr(n_2,n_1) . \quad (12)$$

It can be seen from (10), (11) and (12) that if the inputs of the 2-D FFT are defined in the normal (digit-reverse) column-major order, then the outputs of the 2-D FFT will be in the digit-reverse (normal) row-major order. Similarly, if the inputs are in the normal (digit-reverse) row-major order, then the outputs are in the digit-reverse (normal) column-major order. Thus, the 2-D FFT implemented by the unified 1-D addressing get the mapping as follows

$$[(n_1,n_2)] \xrightarrow[1\text{-}D\ FFT]{Unified} [(dr(k_2),dr(k_1))] . \quad (13)$$

The digit-reverse operation is reversible. Thus, we have

$$dr(dr(nr(n_1,n_2))) = n_1 n_2 = nr(n_1,n_2) . \quad (14)$$

3.2. M-D Digit-Reverse Sequence

The 2-D to 1-D indexing mapping can be extended to the M-D case. The M-D element stored in the memory can be defined in the last-tuple-major order as

$$nr(n_1,n_2,\cdots,n_M) = n_1 n_2 \cdots n_{M-1} n_M . \quad (15)$$

The digit-reverse addressing for the last-tuple-major order of the M-D normal array can be derived as

$$dr(nr(n_1,n_2,\cdots,n_M) = dr(n_M) dr(n_{M-1}) \cdots dr(n_1)$$

$$= nr(dr(n_M),dr(n_{M-1}),\cdots,dr(n_1)) \quad (16)$$

and that of the M-D digit-reverse array can be

$$dr(nr(dr(n_1),\cdots,dr(n_M))) = nr(n_M,\cdots,n_1). \quad (17)$$

Therefore, if the inputs of the M-D FFT are defined in the normal (digit-reverse) last-tuple-major order, then the outputs of the 2-D FFT will be in the digit-reverse (normal) first-tuple-major order and vice versa. Similarly, the digit-reverse operation is reversible as

$$dr(dr(nr(n_1,\cdots,n_M))) = nr(n_1,\cdots,n_M) . \quad (18)$$

3.3. Parameter Definition

$$w_{ij} = N_{i+1} * N_{i+2} * \cdots * N_M * n_{i(j+1)}$$

$$* n_{i(j+2)} * \cdots * n_{is_i} \quad (19)$$

$$\hat{w}_{ij} = N_{i+1} * N_{i+2} * \cdots * N_M * n_{ij} * n_{i(j+1)} * \cdots * n_{is_i} \quad (20)$$

$$\bar{w}_{ij} = N_1 * N_2 * \cdots * N_{i-1} * n_{i1} * n_{i2} * \cdots * n_{i(j-1)} \quad (21)$$

$$v_{ij} = n_{i(j+1)} * n_{i(j+2)} * \cdots * n_{is_i} * N_1 * \cdots * N_{i-1} * N_{i+1} * \cdots * N_M \quad (22)$$

$$\bar{v}_{ij} = n_{i1} * n_{i2} * \cdots * n_{i(j-1)} \quad (23)$$

4. DATA SEQUENCE

The data sequence for the mapped M-D FFT will be the same as that for the 1-D FFT in each stage if the total number of data is the same. As shown in Fig. 1, the data sequences for the first and second stages of the row FFT are the same as those for the first and second stages of the 1-D FFT and the first and second stages of the column FFT are the same as those of the third and fourth stages of the 1-D FFT. The addressing algorithm to generate the data sequence for the BS_{ij}-stage of the M-D FFT is listed in the following

```
for (k=0; k ≤ w̄ij−1; k++)
    for (l=0; l ≤ ŵij−1; l++)
        {Output l * w̄ij + k}
```

5. TWIDDLE FACTOR SEQUENCE

In the mapped M-D FFT implementation, the M-D FFT can employ exactly the same data and digit-reverse addressing sequences as the 1-D FFT. However, The twiddle factor sequences will be different except the first dimension as shown in Figs. 1 and 2. The indices of twiddle factors in the figures are indicated upper for the 2-D case and lower for the 1-D case. Nevertheless, with different parameter setting both M-D and 1-D twiddle factor sequences can be generated by the same operation.

The addressing algorithm to generate the twiddle factor sequence for the BS_{1j}-stage of the 1-D FFT is listed as

```
for (k=0; k ≤ w̄1j−1; k++)
    for (l=0; l ≤ w1j−1; l++)
        for (m=0; m < r1j−1; m++)
            { output m * k * w1j }
```

and that for the BS_{ij}-stage of the M-D FFT is listed as

```
for (k=0; k ≤ v̄ij−1; k++)
    for (l=0; l ≤ vij−1; l++)
        for (m=0; m < rij−1; m++)
            { output m * k * vij }
```

Table I lists the parameters required to generate the data and twiddle factor sequences of the 36-point 1-D FFT and 6 by 6 2-D FFT with 3@2@2@3 and 2@3@3@2 SFG structures. Two parameters are required for the data sequence and three parameters are required for the twiddle factor sequence of the stage. With the same number of array points, there is no difference in setting parameters for the data sequences of 1-D and M-D FFTs. However, parameter setting for the twiddle factor sequences of 1-D and M-D FFTs is different.

Table I. Parameter Setting for Data and Twiddle Factor Sequences of 36-Point 1-D and 6 by 6 2-D FFTs

Parameter	3@2@2@3	2@3@3@2
r_{11}	3	2
r_{12}	2	3
r_{13}, r_{21}	2	3
r_{14}, r_{22}	3	2
w_{11}	12	18
w_{12}	6	6
w_{13}	3	2
w_{14}	1	1
$\hat{w}_{11}$	36	36
$\hat{w}_{12}$	12	18
$\hat{w}_{21}$	6	6
$\hat{w}_{22}$	3	2
$\bar{w}_{11}$	1	1
$\bar{w}_{12}$	3	2
$\bar{w}_{21}$	6	6
$\bar{w}_{22}$	12	18
v_{11}	12	18
v_{12}	6	6
v_{21}	18	12
v_{22}	6	6
$\bar{v}_{11}$	1	1
$\bar{v}_{12}$	3	2
$\bar{v}_{21}$	1	1
$\bar{v}_{22}$	2	3

6. ALGORITHM REALIZED BY LH9124/LH9320

This section discusses hardware realization of the proposed algorithm. It is impractical to build butterfly modules for all the radixes in the data path of a chip. Therefore, the execution unit (LH9124) of the SMT's array processor chip set selects radix-2, radix-4, and radix-16 modules [3]. The radix-16 butterfly is too complex to be directly implemented. Thus, the radix-16 is actually implemented by two radix-4 stages and can be finished every 16 cycles [7].

The proposed addressing algorithm is realized by a programmable address generator called LH9320 [4]. It provides the address pattern required by the LH9124. Since the radix-16 butterfly is implemented by two radix-4 stages, the algorithm for generating twiddle factor sequence of the quasi radix-16 stage has to be modified as

```
for ( k=0; k ≤ w̄ij-1 ; k++)
    for ( l=0; l ≤ wij/4-1; l++)
        for ( n=0; n ≤ 3; n++)
            { output n * k * wij }
        for ( n=0; n ≤ 3; n++)
            for ( m=1; m ≤ 3; m++)
                { output m * (n * w̄ij + k) * wij/4 }
```

Table II compares the performance of the 1-D and M-D FFTs. It can be seen with the same number of array points that both 1-D and M-D FFTs have the same performance. With 25 nanoseconds machine cycle time, the 256 by 256 2-D complex FFT can be finished within 6.56 milliseconds.

There are several advantages for the proposed M-D FFT implementation. First, the number of instructions required is greatly reduced. Thus, the program memory is not necessary and the performance can be improved by reducing instruction pipelined overhead. For example, the proposed approach requires only 3 instructions to implement 16 by 16 by 16 3-D FFT, while the traditional approach requires 768 instructions. Second, no data matrix transposition is required because the transposition is already covered in the M-D to 1-D SFG structure mapping. Thus, it is much easier for users to implement the M-D FFT without considering the time-consuming complex matrix transposition. Third, the proposed approach automatically solves the scaling problems occurred in the block floating-point arithmetic. The traditional approach does require extra hardware or software to solve the problem. Fourth, the simplified and unified 1-D addressing not only speeds up the M-D FFT algorithms but also makes the chip function definition easier. Finally, the function instructions makes users save efforts in program coding and debugging.

Table II. Performance of FFTs by LH9124/LH9320

M-D FFT	Ins. No.	Cycles	μsecs
1-D 256 points	2	648	16.2
2-D 16 by 16	2	648	16.2
1-D 4K points	3	12492	312.3
3-D 16 by 16 by 16	3	12492	312.3
1-D 64K points	4	262416	6560.4
2-D 256 by 256	4	262416	6560.4

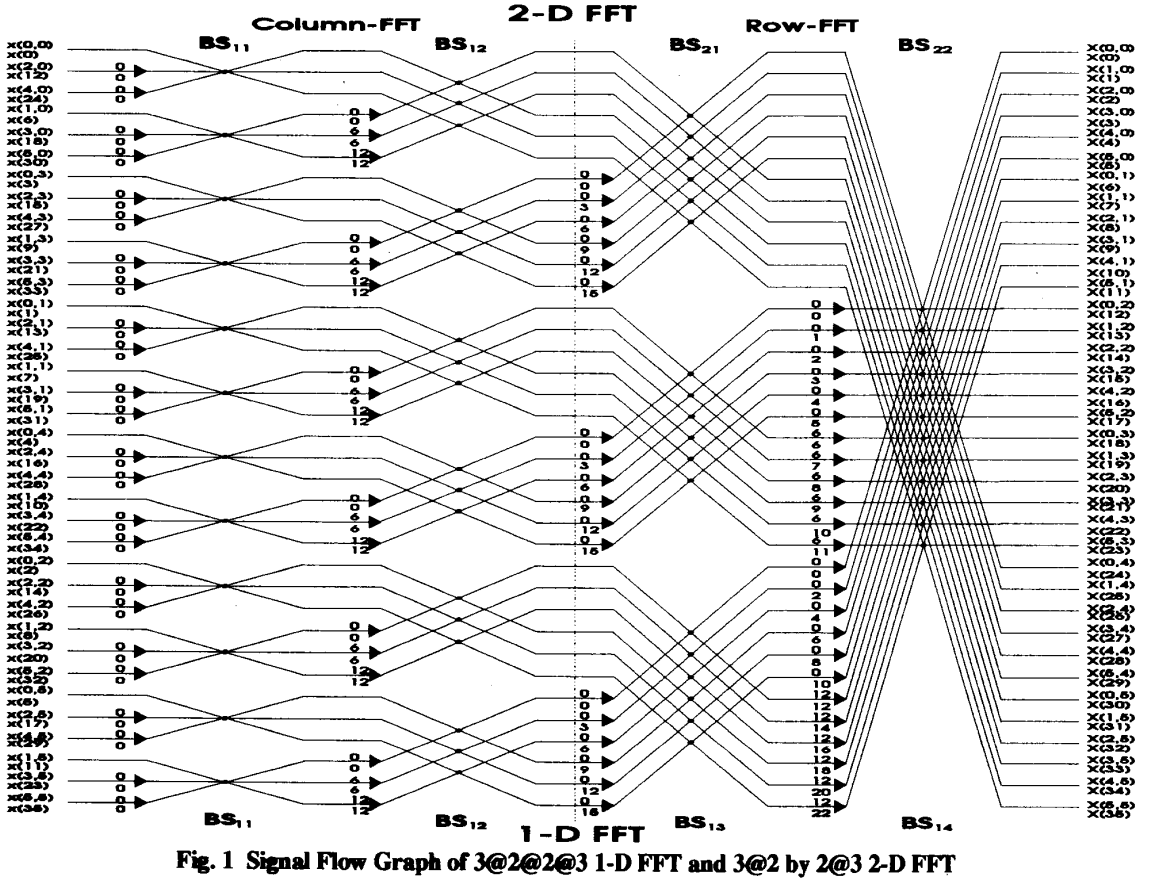

Fig. 1 Signal Flow Graph of 3@2@2@3 1-D FFT and 3@2 by 2@3 2-D FFT

7. CONCLUSIONS

This paper defines the unified 1-D addressing for the M-D FFT implementation. The addressing sequences can be derived from the factorization of the twiddle factor matrix [6]. The discussion only includes the decimation-in-time digit-reverse-input and normal-output FFT algorithms. Essentially all the results extended to other FFT algorithms in a straightforward manner. Algorithms for defining mixed radix 1-D FFT indexing can also be found in [8]

The unified indexing concept of the M-D FFT implementation automatically solves the scaling problem of the block floating-point arithmetic. The concept can also be extended to derive the efficient general DSP algorithms for block floating-point arithmetic such as IIR filtering, adaptive filtering, polyphase filter bank, and multi-channel DSP [9].

ACKNOWLEDGMENT

The author wishes to thank the System and Design groups of Sharp Microelectronics Technology for practically implementing the unified FFT algorithms in the array processing chip set.

REFERENCES

[1] C. Ju and M. Fleming, "Design concept of real-time array signal processors," *Proceeding of the International Conference on Signal Processing Applications and Technology*, Boston, pp.188-197, Nov. 1992.

[2] C. Ju, "Equivalent relationship and unified indexing of FFT algorithms," *Proceeding of International Symposium on Circuits and Systems*, Chicago, May 1993.

[3] *LH9124 Digital Signal Processor User's Guide*, Sharp Electronics Corporation.

[4] *LH9320 Address Generator User's Guide*, Sharp Electronics Corporation.

[5] J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex Fourier series," *Math. Comput.*, vol.19, pp.297-301, Apr. 1965.

[6] C. Ju, "Derivation and realization of fast Fourier transform," unpublished.

[7] C. Ju, *LH9124/LH9320 Fast Fourier Transform Application Note*, Sharp Electronics Corporation.

[8] G. L. DeMuth, "Algorithms for defining mixed radix FFT flow graphs," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, pp.1349-1358, Sept. 1989.

[9] C. Ju, "General DSP algorithms for block floating-point arithmetic," unpublished.

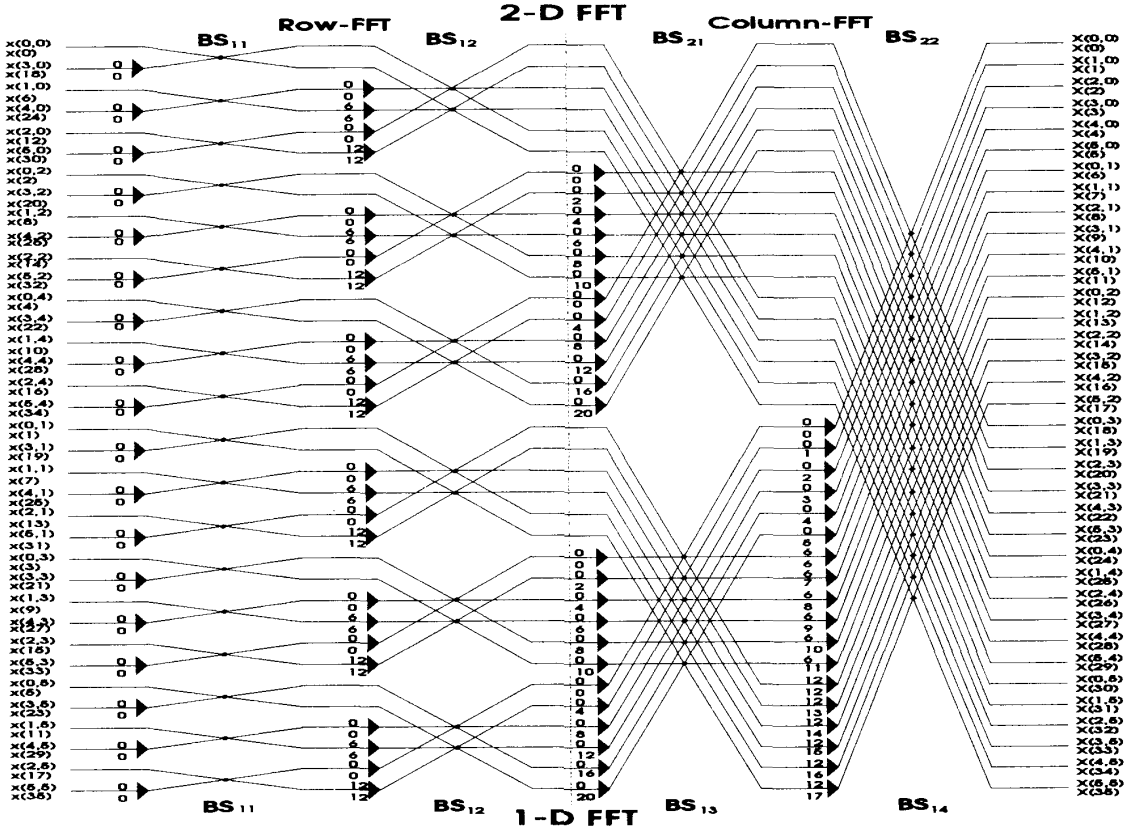

Fig. 2 Signal Flow Graph of 2@3@3@2 1-D FFT and 2@3 by 3@2 2-D FFT