Split Vector-Radix-2/8 2-D Fast Fourier Transform

Soo-Chang Pei, *Fellow, IEEE,* and Wei-Yu Chen

***Abstract*—This letter presents an efficient split vector-radix-2/8 fast Fourier transform (FFT) algorithm. The split vector-radix-2/8 FFT algorithm saves 14% real multiplications and has much lower arithmetic complexity than the split vector-radix-2/4 FFT algorithm. Moreover, this algorithm reduces 25% data loads and stores compared with the split vector-radix-2/4 FFT algorithm.**

***Index Terms*—Split-radix-2/4, split-radix-2/8, two-dimensional (2-D) fast Fourier transform (FFT), vector-radix.**

I. Introduction

THE TWO-DIMENSIONAL (2-D) discrete Fourier transform (DFT) is an important tool in digital image processing. The vector-radix fast Fourier transform (FFT) algorithm [1] is an efficient method on computing the 2-D DFT. This method computes the twiddle factor multiplications in both rows and columns simultaneously. It can reduce 25% of complex multiplication over the row-column method, and it also avoids the matrix transpose operation.

The one-dimensional (1-D) split-radix-2/4 FFT has been used to develop a 2-D FFT algorithm, the split vector-radix-2/4 2-D FFT [2]–[6], which greatly reduces the computation complexity as compared with the conventional radix-(2×2) 2-D FFT. The split vector-radix-$2/2^n$ FFT based on the polynomial transform has been discussed in [5]. Without concerning all the trivial multiplications, the arithmetic complexity of the split vector-radix-2/8 FFT in [5] is roughly estimated and nearly the same as the split vector-radix-2/4 FFT algorithm. In this letter, the split-radix-2/8 FFT algorithm in [7] is applied to 2-D FFT to develop a split vector-radix-2/8 FFT algorithm. The implementation and evaluation of this split vector-radix-2/8 FFT algorithm will be discussed in detail. We will show that it saves 14% real multiplications and has much lower arithmetic complexity than the split vector-radix-2/4 FFT algorithm, and this algorithm can reduce the number of data loads and stores.

II. Split Vector-Radix-2/8 2-D FFT Algorithm

The 2-D DFT of an $N_1\times N_2$ 2-D data $x(n_1,n_2)$ is defined as

$$X(k_1,k_2)=\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)W_{N_1}^{n_1k_1}W_{N_2}^{n_2k_2} \tag{1}$$

Manuscript received July 8, 2003; revised August 21, 2003. This work was supported in part by the National Science Council of Taiwan, R.O.C. under Contract NSC 91-2219-E-002-044 and in part by the Ministry of Education under Contract 89-E-FA06-2-4. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Xiang-Gen Xia.

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan 10617, R.O.C. (e-mail: pei@cc.ee.ntu.edu.tw).

Digital Object Identifier 10.1109/LSP.2004.826652

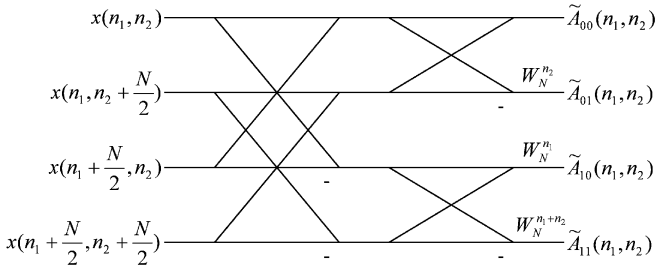

Fig. 1. Butterfly for DIF vector radix 2-D FFT.

where

$$W_{N_i}=\exp\left(-\frac{j2\pi}{N_i}\right),\qquad k_i=0,1,\ldots,N_i-1,\ i=1,2.$$

For simplicity, we assume $N_1=N_2=N$ in this letter.

The decimation-in-frequency (DIF) vector-radix-(2×2) FFT decomposes the $N\times N$ 2-D FFT in (1) into four $(N/2)\times(N/2)$ 2-D FFTs. The 2-D DFT formulation in (1) can be written as

$$X(2k_1+a,2k_2+b)=\sum_{n_1=0}^{(N/2)-1}\sum_{n_2=0}^{(N/2)-1}\widetilde{A}_{ab}(n_1,n_2)W_{N/2}^{n_1k_1+n_2k_2}$$
$$k_1,k_2=0,1,\ldots,\frac{N}{2}-1\ \ a,b=0,1 \tag{2}$$

$$\begin{aligned}A_{ab}(n_1,n_2)=&x(n_1,n_2)+(-1)^a x\left(n_1+\frac{N}{2},n_2\right)\\&+(-1)^b x\left(n_1,n_2+\frac{N}{2}\right)\\&+(-1)^{a+b}x\left(n_1+\frac{N}{2},n_2+\frac{N}{2}\right)\\ \widetilde{A}_{ab}(n_1,n_2)=&A_{ab}(n_1,n_2)W_N^{an_1+bn_2}.\end{aligned} \tag{3}$$

The corresponding butterfly is shown in Fig. 1. The $N\times N$ 2-D FFT is broken into four $(N/2)\times(N/2)$ 2-D FFTs.

The 1-D split-radix-2/8 FFT algorithm has been presented in [7]. Now, we apply it on 2-D vector-radix FFT. We can do some substitutions in (1) as follows:

$$\begin{aligned}&k_1=8k_1'+a,\quad k_2=8k_2'+b,\\&k_i'=0,\ldots,\frac{N}{8}-1,\quad a,b=0,\ldots,7\\&n_1=n_1'+\alpha'\cdot\frac{N}{8},\quad n_2=n_2'+\beta'\cdot\frac{N}{8},\\&n_i'=0,\ldots,\frac{N}{8}-1,\quad \alpha',\beta'=0,\ldots,7.\end{aligned}$$

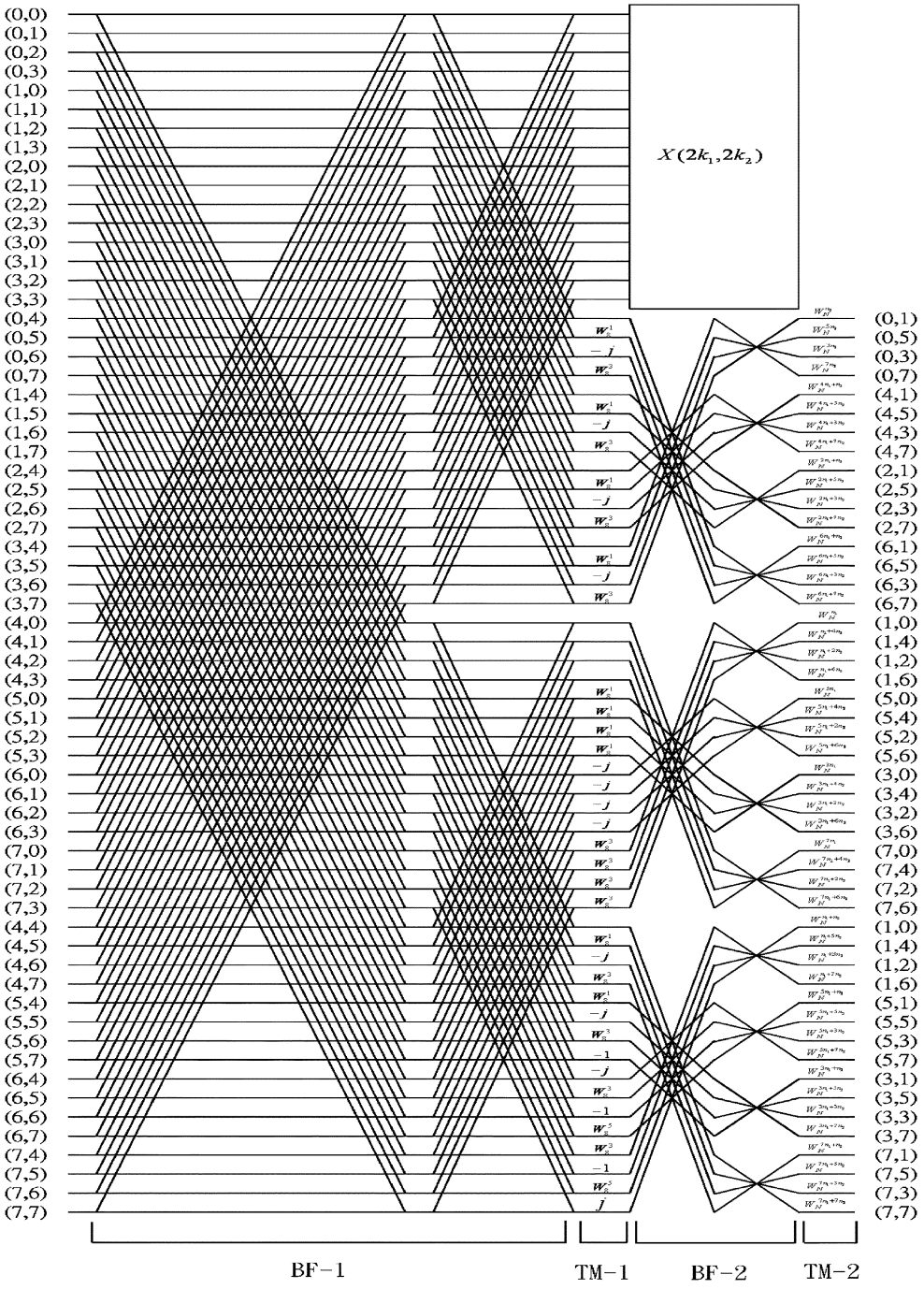

Fig. 2. Architecture of SVRFFT-2/8 decomposition.

Then, (1) can be rewritten as shown in (4), at the bottom of the page. Using (3) and $W_8^4 = -1$, we can obtain the derived formulation of the split vector-radix-2/8 FFT as shown in (5), at the bottom of the page. For simplification, we define

$$X(8k_1'+a,8k_2'+b)=\sum_{n_1'=0}^{(N/8)-1}\sum_{n_2'=0}^{(N/8)-1}\left[\sum_{\alpha'=0}^{7}\sum_{\beta'=0}^{7}x\left(n_1'+\alpha'\cdot\frac{N}{8},n_2'+\beta'\cdot\frac{N}{8}\right)\cdot W_8^{a\alpha'+b\beta'}\right]\cdot W_N^{an_1'+bn_2'}\cdot W_{N/8}^{n_1k_1+n_2k_2} \quad (4)$$

$$X(8k_1+a,8k_2+b)=\sum_{n_1=0}^{(N/8)-1}\sum_{n_2=0}^{(N/8)-1}\left[\sum_{\alpha=0}^{3}\sum_{\beta=0}^{3}A_{ab}\left(n_1+\alpha\cdot\frac{N}{8},n_2+\beta\cdot\frac{N}{8}\right)\cdot W_8^{a\alpha+b\beta}\right]\cdot W_N^{an_1+bn_2}\cdot W_{N/8}^{n_1k_1+n_2k_2} \quad (5)$$

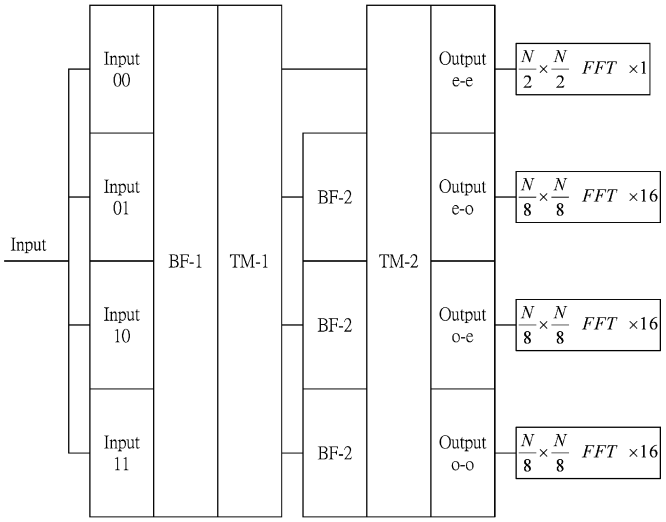

Fig. 3. Simplified version of Fig. 2.

$$H_{ab}(n_1,n_2)=\left[\sum_{\alpha=0}^{3}\sum_{\beta=0}^{3}W_8^{a\alpha+b\beta}\cdot A_{ab}\left(n_1+\alpha\cdot\frac{N}{8},n_2+\beta\cdot\frac{N}{8}\right)\right]$$
$$\widetilde{H}_{ab}(n_1,n_2)=H_{ab}(n_1,n_2)\cdot W_N^{an_1+bn_2}$$
$$k_1,k_2=0,\ldots,\frac{N}{8}-1,\quad a,b=0,\ldots,7. \tag{6}$$

Then, (5) can be written as

$$X(8k_1+a,8k_2+b)=\sum_{n_1=0}^{(N/8)-1}\sum_{n_2=0}^{(N/8)-1}\widetilde{H}_{ab}(n_1,n_2)\cdot W_{N/8}^{n_1k_1+n_2k_2} \tag{7}$$

and we assume that at least one of a,b is odd, since we have used radix-(2×2) for the even–even part in (2) with $a=b=0$. The corresponding butterfly has 64 input nodes and 64 output nodes. The resultant architecture and the simplified version are shown in Figs. 2 and 3, respectively. We have divided the butterflies into several blocks. The BF-1 block of the butterflies in Fig. 3 is a vector-radix-(2×2) FFT butterfly, which contains 16 butterflies as shown in Fig. 1, and the BF-2 block is a vector-radix-(4×4) FFT butterfly. The BF-1 decomposes the 2-D FFT into four parts. Then, the three BF-2 blocks further decompose the even-odd, odd-even, odd-odd parts into 48 $(N/8)\times(N/8)$ 2-D FFTs.

Consider the twiddle factor multiplication TM-1 between BF-1 and BF-2. It is observed that all the multiplications in TM-1 belong to two kinds of special multiplications. The further discussion is made in Section III. In addition, 48 $(N/8)\times(N/8)$ 2-D FFTs usually needs much fewer multiplications than 3 $(N/2)\times(N/2)$ 2-D FFTs. The reason is that the number of multiplications usually increases as $N^2\log N$ rather than N^4. These properties greatly reduce the number of real multiplication in the butterflies.

At the TM-2 block of the butterflies in Fig. 3, we have to multiply $H_{ab}(n_1,n_2)$ by the twiddle factor $W_N^{an_1+bn_2}$ to get the corresponding $\widetilde{H}_{ab}(n_1,n_2)$. According to (7), we can compute $X(8k_1+a,8k_2+b)$ from $\widetilde{H}_{ab}(n_1,n_2)$ using $(N/8)\times(N/8)$ 2-D FFT. But in the even-even part, we will use $(N/2)\times(N/2)$ 2-D FFT to compute $X(2k_1,2k_2)$ from the output. The input and output allocations are shown in Fig. 4. For input node (i_1,i_2) of block "Input pq" in Fig. 4, the corresponding input is $x(n_1+i_1\times(N/8)+p\times(N/2),n_2+i_2\times(N/8)+q\times(N/2))$, where $n_1,n_2=0,1,\ldots,(N/8)-1$, and $p,q=0,1$. And in Fig. 4, we take "Output e-o" block as an example. At the output node (u,v) for "Output e-o" block, we will get $\widetilde{H}_{ab}(n_1,n_2)$, where $a=e_u$, and $b=o_v$. The symbols "e" and "o" represent for "even" and "odd". And $\{e_0,e_1,e_2,e_3\}=\{0,4,2,6\}$, $\{o_0,o_1,o_2,o_3\}=\{1,5,3,7\}$. By assigning the corresponding e_i or o_i to a and b, we can get the correct output allocation for the other output blocks. We will get $\widetilde{H}_{ab}(n_1,n_2)$ at the output of "Output e-o," "Output o-e," and "Output o-o." At the output of "Output e-e" block, we will get $A_{00}(n_1+u\times(N/8),n_2+v\times(N/8))$, which could be combined with each other to form $\widetilde{A}_{00}(n_1,n_2)$ in Fig. 1.

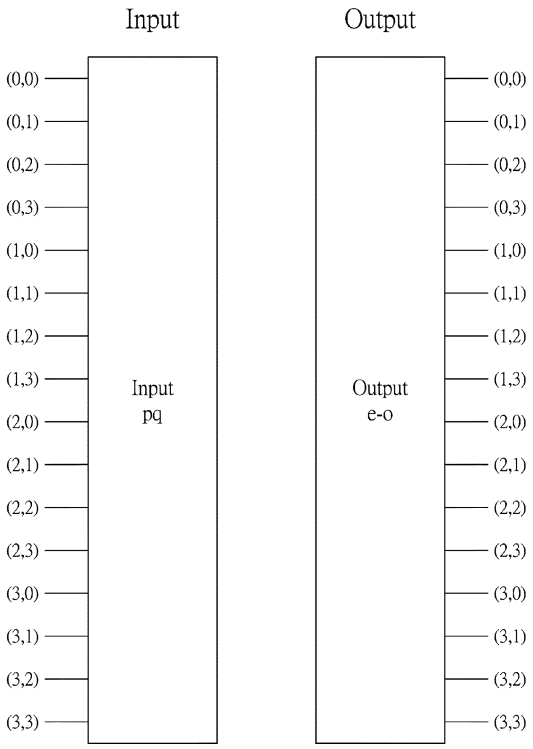

Fig. 4. Split vector-radix-2/8 2-D input/output allocations.

III. Complexity Evaluation

Let $M(N,N)$ and $A(N,N)$ be the number of real multiplications and real additions needed when computing the complex 2-D FFT using the split vector-radix-2/8 algorithm.

As we mentioned in Section II, there are some special multiplications in the butterfly. The first kind are the trivial multiplications that include multiplication with ± 1, and $\pm j$. This kind of multiplication could be done without any real multiplication and real addition. The other kind includes multiplication with $e^{\pm j\pi/4}$ and $e^{\pm j3\pi/4}$, which needs only two real multiplications and two real additions. Take these reduction into account and consider some common terms between different nodes. We obtain

$$M(N,N)=M\left(\frac{N}{2},\frac{N}{2}\right)+48M\left(\frac{N}{8},\frac{N}{8}\right)+\frac{15}{4}N^2-\frac{11}{8}N-780 \tag{8}$$

TABLE I
NUMBER OF REAL MULTIPLICATIONS AND ADDITIONS TO COMPUTE A 2-D COMPLEX DFT

N	VRFFT-2		VRFFT-4		SVRFFT-2/4 [4][5][6]		SVRFFT-2/8	
	Mults	Adds	Mults	Adds	Mults	Adds	Mults	Adds
8	48	816			48	816	48	816
16	768	4608	768	4512	720	4496	528	4432
32	5760	23936			4176	22928	3544	22672
64	34560	118016	26880	112384	24720	111568	20340	111412
128	185856	561664			123216	525712	106168	524224
256	936960	2606080	672768	2444288	614928	2421072	520908	2414620
512	4528128	11864064			2876880	10956816	2478784	10972888
1024	21245952	53219328	14684160	49459200	13395600	48920528	11504820	48979732

TABLE II
NUMBER OF REAL MULTIPLICATIONS PLUS ADDITIONS TO COMPUTE A 2-D COMPLEX DFT

N	VRFFT-2	VRFFT-4	SVRFFT-2/4 [4][5][6]	SVRFFT-2/8
	M+A	M+A	M+A	M+A
8	864		864	864
16	5376	5280	5216	4960
32	29696		27104	26216
64	152576	139264	136288	131752
128	747520		648928	630392
256	3543040	3117056	3036000	2935528
512	16392192		13833696	13451672
1024	74465280	64143360	62316128	60484552

$$A(N,N)=A\left(\frac{N}{2},\frac{N}{2}\right)+48A\left(\frac{N}{8},\frac{N}{8}\right)+\frac{49}{4}N^2-\frac{3}{8}N-580. \tag{9}$$

When considering the arithmetic complexity, we use four real multiplications and two real additions scheme for one complex multiplication. And the split vector-radix-2/4 algorithm in all the tables is the one in [4]–[6].

Table I lists the number of real multiplications and additions needed for the algorithm, and Table II shows the total number of real multiplication plus real addition. The 2-D transforms up to length (16×16) are specially programmed for the SVR-2/8 FFT to remove some trivial or special multiplications. We obtain that the arithmetic complexity of the SVR-2/8 FFT is much lower than SVR-2/4 FFT algorithm. For $\mathrm{N} = 1024$, SVR-2/8 FFT can reduce 14% of real multiplications.

Now consider the number of data loads and stores for several vector-radix FFT algorithms. Table III shows the number of loads, and stores for vector-radix-(2×2), vector-radix-(4×4), SVR-2/4, and SVR-2/8 butterflies. When calculating the number of loads and stores, we assume that enough registers are available to perform the entire butterfly. In Table IV, the number of loads and stores used by each algorithm is shown. The lower order terms have been omitted. The SVR-2/8 FFT algorithm requires fewer loads and stores than the other algorithms. In comparison with SVR-2/4 FFT algorithm, the SVR-2/8 FFT algorithm reduces 25% of the data loads and stores.

TABLE III
NUMBER OF DATA LOADS AND STORES FOR GENERAL BUTTERFLIES USED IN 2-D DFT ALGORITHMS. THE NUMBER OF LOADS INCLUDES LOADING ALL OF THE CONSTANTS

Algorithm	Loads	Stores
VRFFT-2	14	8
VRFFT-4	62	32
SVRFFT-2/4 [4][5][6]	56	32
SVRFFT-2/8	224	128

TABLE IV
NUMBER OF LOADS AND STORES DIVIDED BY $N^2 \log_2 N$ USED BY 2-D DFT ALGORITHMS TO COMPUTE $N \times N$ POINT 2-D DFT. LOWER ORDER TERMS HAVE BEEN OMITTED

Algorithm	Loads	Stores
VRFFT-2	7/2	2
VRFFT-4	31/16	1
SVRFFT-2/4 [4][5][6]	7/3	4/3
SVRFFT-2/8	7/4	1

IV. CONCLUSION

An efficient split vector-radix-2/8 FFT algorithm has been developed. The computation complexity of this method is much lower than the split vector-radix-2/4 FFT. On the other hand, the split vector-radix-2/8 FFT algorithm saves 14% real multiplications and 25% data loads and stores compared with the split vector-radix-2/4 FFT algorithm. Therefore, it may be preferable to use the split vector-radix-2/8 FFT rather than split vector-radix-2/4 FFT.

REFERENCES

[1] D. B. Harris and J. H. McClellan, "Vector-radix fast Fourier transform," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, May 1977, pp. 548–551.
[2] S. C. Pei and J. L. Wu, "Split vector-radix 2D fast Fourier transform," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 978–980, Aug. 1987.
[3] Z. J. Mou and P. Duhamel, "In-place butterfly-style FFT of 2-D real sequences," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1642–1650, Oct. 1988.
[4] H. R. Wu and F. J. Paoloni, "On the two-dimensional vector-radix FFT algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1302–1304, Aug. 1989.
[5] S. C. Chan, "Fast transform algorithms and their applications," Ph.D. dissertation, Univ. Hong Kong, 1992.
[6] S. C. Chan and K. L. Ho, "Split vector-radix fast Fourier transform," *IEEE Trans. Signal Processing*, vol. 40, pp. 2029–2039, Aug. 1992.
[7] D. Takahashi, "An extended split-radix FFT algorithm," *IEEE Signal Processing Lett.*, vol. 8, pp. 145–147, May 2001.