A 2048-Point FFT Processor Based on Twiddle Factor Table Reduction*

Ji-Hoon Kim and In-Cheol Park

School of EECS, Division of EE, KAIST
Daejeon 305-701, Korea
{jhkim, icpark}@ics.kaist.ac.kr

In long-point fast Fourier transform (FFT) processing, the table size is large enough to occupy significant area and power consumption. The proposed algorithm can reduce the required table size to half, compared to the radix-2^2 algorithm, while retaining the simple structure. To verify the proposed algorithm, a 2048-point pipelined FFT processor is designed using a 0.13 μm CMOS process. By combining the proposed algorithm and the radix-2^2 algorithm, the table size is reduced to 34% and 51% compared to the radix-2 and radix-2^2 algorithms, respectively. The FFT processor achieves a signal-to-quantization-noise ratio (SQNR) of 52.8 dB.

Keywords: FFT, discrete Fourier transform (DFT), pipelined processing

FFT is a major signal processing block being widely used in communication systems, especially in orthogonal frequency division multiplexing (OFDM) systems such as digital video broadcasting and WiMAX (IEEE 802.16). As such a system requires long-point FFT computation, usually more than 1024 points, it is desirable to reduce computational complexity as well as hardware complexity.

To reduce the computational complexity of FFT processing, various FFT algorithms have been proposed such as radix-2^2 [1] and radix-2^3 [2] as well as radix-2 algorithms. Although the previous algorithms could reduce the computational hardware resources such as multipliers, they did not seriously take into account the number of twiddle factors required in FFT processing. In the implementation of a long-point FFT processor, however, the tables become large enough to occupy significant area and power consumption [3].

The proposed algorithm can be derived by applying the radix-2 decimation-in-frequency (DIF) decomposition two times. The N-point DFT of a sequence $x(n)$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad 0 \le k < N \tag{1}$$

where $W_N = exp(-j2\pi/N)$. The two decompositions can be expressed if n and k are replaced with 3-dimensional linear index maps shown below.

$$n = n_1 \cdot N/2 + n_2 \cdot N/4 + n_3, \quad k = k_1 + 2k_2 + 4k_3 \tag{2}$$

Using the above index maps, (1) can be rewritten as

$$X(k) = X(k_1 + 2k_2 + 4k_3) = \sum_{n_3=0}^{N/4-1} \sum_{n_2=0}^{1} \{B(n_2 \cdot N/4 + n_3, k_1) W_N^{(\frac{N}{4}n_2 + n_3)k_1}\} W_N^{(\frac{N}{4}n_2 + n_3)(2k_2 + 4k_3)} = \sum_{n_3=0}^{N/4-1} [H(k_1, k_2, n_3) W_N^{2m(k_1 + 2k_2)}] W_N^{4n_3 k_3} \tag{3}$$

where H(•) defined below.

$$H(k_1, k_2, n_3) = \begin{cases} B(n_3, k_1) + (-1)^{k_2} (-j)^{k_1} B(n_3 + N/4, k_1) & \text{if } n_3 \text{ is even} \\ [W_N^{k_1} B(n_3, k_1) + (-1)^{k_2} (-j)^{k_1} W_N^{k_1} B(n_3 + N/4, k_1)] \cdot W_N^{2k_2} & \text{if } n_3 \text{ is odd} \end{cases} \tag{4}$$

In (3), B(•) represents the following butterfly structure.

$$B(n_2 \cdot N/4 + n_3, k_1) = x(n_2 \cdot N/4 + n_3) + (-1)^{k_1} x(n_2 \cdot N/4 + n_3 + N/2) \tag{5}$$

As k_1 is either 0 or 1, (4) indicates that the butterfly has a trivial multiplication of $-j$ at the input side if n_3 is even and constant multiplications of W_N^1 if n_3 is odd. By performing the constant multiplications at the input side, all the exponents in the twiddle factors and to be multiplied after the butterfly operation in (3) become even values. This implies that, in the proposed algorithm, the table required in every second stage needs to store only the twiddle factors of which exponents are even. However, in other FFT algorithms, most of the twiddle factor tables have twiddle factors associated with both even and odd exponents.

* This work was supported by Institute of Information Technology Assessment through the ITRC and by IC Design Education Center (IDEC)

The number of twiddle factors required in a stage is determined by two factors. The first factor is the maximum exponent value (*MEV*) of twiddle factors and the other is the greatest common divisor (G.C.D) value among the exponents of the twiddle factors, that is, *minimum stride*. Considering these two factors, the number of required twiddle factors, $N_{required}$, can be represented as follows.

$$N_{required} = MEV / minimum\ stride \tag{6}$$

However, the ROM size, N_{ROM}, is usually larger than $N_{required}$ since the number of ROM entries is usually constrained to a power of two. To reduce the ROM table size, in general, non-trivial twiddle factors should be regularly distributed. This regularity can be determined by the value of minimum stride. Table I shows the number of ROM entries required in 16-point FFT processing shown in Fig. 1 for the various FFT algorithms. Since the value of the *minimum stride* in the proposed algorithm is bigger than one, we can reduce the total number of entries of the ROM at least by a factor of two compared to the famous radix-2^2 algorithm that has a twiddle factor table in every second stage. Considering the $\pi/2$ symmetric property of the twiddle factors in counting the number of entries, the total number of table entries required for the case of 8192-point and 2048-point FFT processing are compared in Table II. As denoted in Table 2, the proposed algorithm can effectively reduce the number of table entries compared to the other FFT algorithms.

In the proposed algorithm, the reduction of table entries is achieved at the cost of an additional constant multiplier per every second stage. The complexity of the constant multiplier depends on the number of non-zero bits in the binary representation of the constant. To minimize the number of non-zero bits, rounded constants are expressed in the minimal signed digit (MSD) representation as shown in Table III. Due to the sparse non-zero bits in the sine and cosine values, the constant multipliers can be implemented with a few adders. In case of W_{2048}^{1} multiplication, the corresponding constant multiplier can be implemented with only four adders as shown in Fig. 2. The number of adders needed for implementing the constant multiplier is specified in Table III. The constant multiplier for W_{8}^{1} is used in every third stage in the radix-2^3 algorithm.

In pipelined radix-2 DIF FFT processing, the number of required twiddle factors is largest at the first stage and reduced gradually at the successive stages. Since the proposed algorithm is very effective in long-point FFT processing due to the low complexity of the additional constant multiplier, the first 6 stages of the total 11 stages in 2048-point FFT are processed by the proposed algorithm and the remaining stages are processed by the radix-2^2 algorithm. In the designed FFT processor, the dynamic data scaling method, a sort of semi-floating point representation, is adopted to lower the complexity of the computational units for enough SQNR performance without increasing the internal wordlength gradually [4]. Internal wordlength of the computational units are set to 13bits and input is represented in 12 bits. The bit-width of the twiddle factors is set to 12 bits and the longer wordlength is not cost efficient as the SQNR performance is not increased notably but the cost of multipliers and tables are increased significantly [5]. The address generation of the table in the proposed algorithm is very similar to that of the radix-2^2 algorithm. The proposed processor achieves a SQNR of 52.8 dB and the latency is 2057 cycles for 2048-point FFT processing. Also, the proposed FFT processor can process the 1024/512/256-point FFT by changing the input location and inverse FFT (IFFT) by changing the sign of the imaginary part of the twiddle factors.

A 2048-point FFT processor of which single-path delay feedback structure is shown in Fig. 3 was synthesized with a 0.13 μm 6-Metal CMOS standard cell library. The proposed FFT processor occupies 2.1mm^2 and the gate count is 75,809 excluding memories and ROMs. The FIFO buffers are implemented using RAM memories, and small-sized memories are replaced with registers and logic circuitry. By using a few additional adders, the proposed FFT processor reduces the required table size hugely. The layout of the proposed FFT processor is shown in Fig. 4.

[1] S. He and M. Torkelson, "Design and Implementation of a 1024-point pipeline FFT processor," in Proc. IEEE Custom Integrated Circuits. Conf., pp. 131–134, May 1998

[2] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation," in Proc. IEEE URSI Int. Symp. Signals, Syst. Electron., pp. 257–262, Sept. 1998.

[3] W. Li and L. Wanhammar, "A pipeline FFT processor," in Proc. IEEE Workshop on Signal Processing Systems, pp. 654–662, Oct. 1999.

[4] T. Lenart and V. Owall, "A 2048 Complex Point FFT Processor using a Novel Data Scaling Approach," in Proc. of IEEE International Symposium on Circuits and Systems, pp. IV-45–IV-48, May 2003.

[5] S. Johansson, S. He and P. Nilsson, "Wordlength Optimization of a Pipelined FFT Processor," in 42nd Midwest Symposium on Circuits and Systems, pp. 501–503, Aug. 1999.

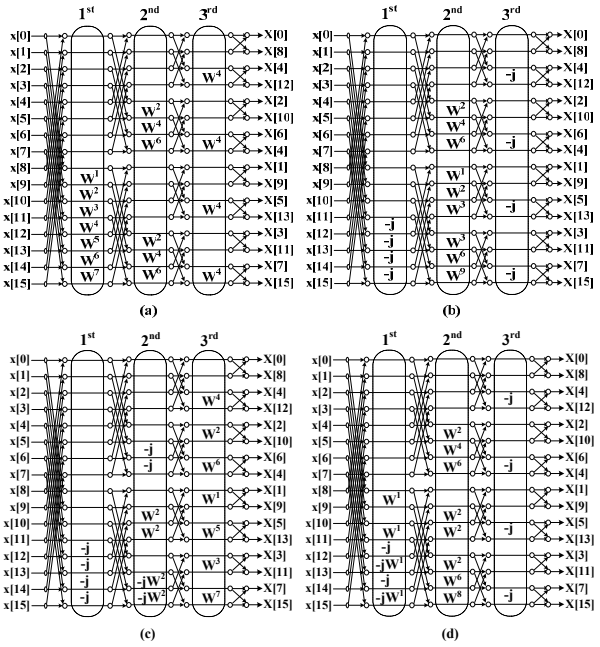

Fig. 1 Signal flow graphs for 16-point FFT. (a) Radix-2 algorithm. (b) Radix-2^2 algorithm. (c) Radix-2^3 algorithm. (d) Proposed algorithm.

TABLE I. DECOMPOSITION FACTORS COMPARISON[1]

	Factor	Radix-2	Radix-2^2	Radix-2^3	Proposed
1st	*minimum stride*[2]	1	N.A.	N.A.	N.A.
	MEV	7	4	4	5
	N_{ROM}	8	1	1	1
2nd	*minimum stride*	2	1	N.A.	2
	MEV	6	9	6	8
	N_{ROM}	4	16	1	4
3rd	*minimum stride*	N.A.	N.A.	1	N.A.
	MEV	4	4	7	4
	N_{ROM}	1	1	8	1

[1] No symmetric property is considered.

[2] N.A means the constant multiplier or rotator exists.

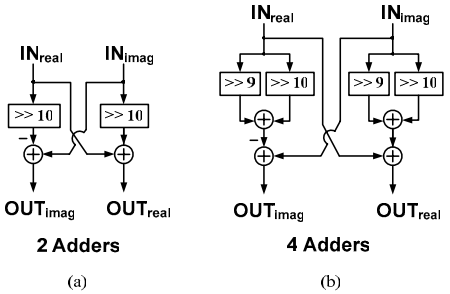

Fig. 2 Low-complexity constant multipliers for (a) W^1_{8192} multiplication and (b) W^1_{2048} multiplication.

TABLE II. COMPLEXITY COMPARISON WITH OTHER FFT ALGORITHMS

		Radix-2	Radix-2^2	Radix-2^3 / Radix-4+2	Proposed + Radix-2^2
8192-point	TN[3]	4092 (150%)	2728 (100%)	2340 (85.8%)	1368 (50.1%)
	CM[4]	18	18	72	38
	GM[5]	40	20	16	20
2048-point	TN	1020 (150%)	680 (100%)	584 (85.9%)	344 (50.6%)
	CM	18	18	54	36
	GM	32	16	12	16

[3] Total # of entries in twiddle factor tables

[4] Total # of adders in constant multipliers

[5] Total # of general multipliers

TABLE III. COMPLEXITY OF CONSTANT MULTIPLIEERS

Constant Operand	MSD Representation	# of required adders
cos(2π/8192)	1.00000000000	**2**
sin(2π/8192)	0.00000000010	
cos(2π/2048)	1.00000000000	**4**
sin(2π/2048)	0.00000000110	
cos(2π/512)	1.00000000000	**6**
sin(2π/512)	0.00000011001	
cos(2π/128)	$1.0000000010\bar{1}0$	**8**
sin(2π/128)	0.00001100100	
cos(2π/8)	0.10110101000	**18**
sin(2π/8)	0.10110101000	

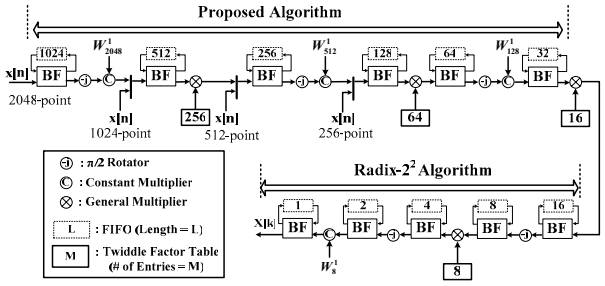

Fig. 3 2048/1024/512/256-point pipelined FFT processor

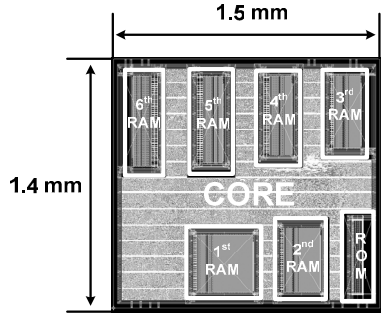

Fig. 4 Layout of the proposed FFT processor