ハイパフォーマンス
コンピューティング 83－5
（2000．10．6）

拡張split-radix FFTアルゴリズム

高　橋　大　介†

本論文では，拡張split-radix FFTアルゴリズムを提案する．この拡張split-radix FFTアルゴリズムは従来のsplit-radix FFTアルゴリズムに比べて乗算回数が少なくなる．さらに，拡張split-radix FFTは従来のsplit-radix FFT，基数4のFFTに比べて，ロードおよびストア回数が少なくなるという利点がある．

An Extended Split-Radix FFT Algorithm

DAISUKE TAKAHASHI†

An extended split-radix FFT algorithm is proposed. The extended split-radix FFT algorithm requires fewer multiplications than the conventional split-radix FFT algorithm. Moreover, this algorithm has the advantage of fewer loads and stores than either the conventional split-radix FFT algorithm or the radix-4 FFT algorithm.

1. はじめに

高速Fourier変換（fast Fourier transform，以下FFT）[1]は，離散Fourier変換（discrete Fourier transform，以下DFT）を高速に計算するアルゴリズムとして，科学技術計算において今日広く用いられているアルゴリズムである．

FFTアルゴリズムの一つであるsplit-radix FFTアルゴリズム[2]は，1984年にDuhamelとHollmannによって提案されたFFTアルゴリズムであり，演算回数が最も少ないFFTアルゴリズムとして知られている．

split-radix FFTアルゴリズムでは，基数4のFFTアルゴリズムにおいて，偶数番目のデータに対しては基数4の分解を適用しても，基数2の分解に比べてひねり係数（twiddle factor）[3]の乗算回数が削減されないという事実に着目している[4]．つまり，偶数番目のデータに対しては基数2の分解を，奇数番目のデータに対しては基数4の分解をそれぞれ適用し，分解を進めていくことで余分な演算を省くというのが，split-radix FFTアルゴリズムの基本的なアイデアである．

このような工夫を行うことで，split-radix FFTアルゴリズムでは，基数2，4，8のFFTアルゴリズムよりも少ない演算量でFFTを計算できる．しかし，メモリアクセス回数の観点から考えると，split-radix FFTのメモリアクセス回数は基数2のFFTと基数4のFFTのメモリアクセス回数の間に位置することになり，その結果基数4，8のFFTに比べてメモリアクセス回数が多くなるという欠点があった．

FFTでは，基数を大きくすることによって，メモリアクセス回数が削減できることが知られている[5]．したがって，従来のsplit-radix FFTアルゴリズムにおいて，偶数番目のデータに対しては基数2の分解を，奇数番目のデータに対しては基数8の分解[6]をそれぞれ適用することで，メモリアクセス回数をさらに削減できると予想される．

本論文では，split-radix FFTの考え方を拡張して，基数2と基数8の分解を組み合わせた場合に拡張split-radix FFTアルゴリズムが構成できることを示す．さらに，この拡張split-radix FFTは，従来のsplit-radix FFTおよび基数4のFFTに比べて乗算回数だけでなく，ロードおよびストア回数も削減されることも示す．

2. 従来のsplit-radix FFTアルゴリズム

split-radix FFTアルゴリズムの基本的なアイデアは，偶数番目のデータに対しては基数2の分解を，奇数番目のデータに対しては基数4の分解をそれぞれ適用することにある．つまり，split-radix FFTアルゴリズムは1/2の長さの1つのDFTと，1/4の長さの2つのDFTを統合する考え方に基づいている．

N点DFTは次式で定義される．

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk}, \quad k = 0, \cdots, N-1 \qquad (1)$$

† 埼玉大学大学院理工学研究科
Graduate School of Science and Engineering, Saitama University

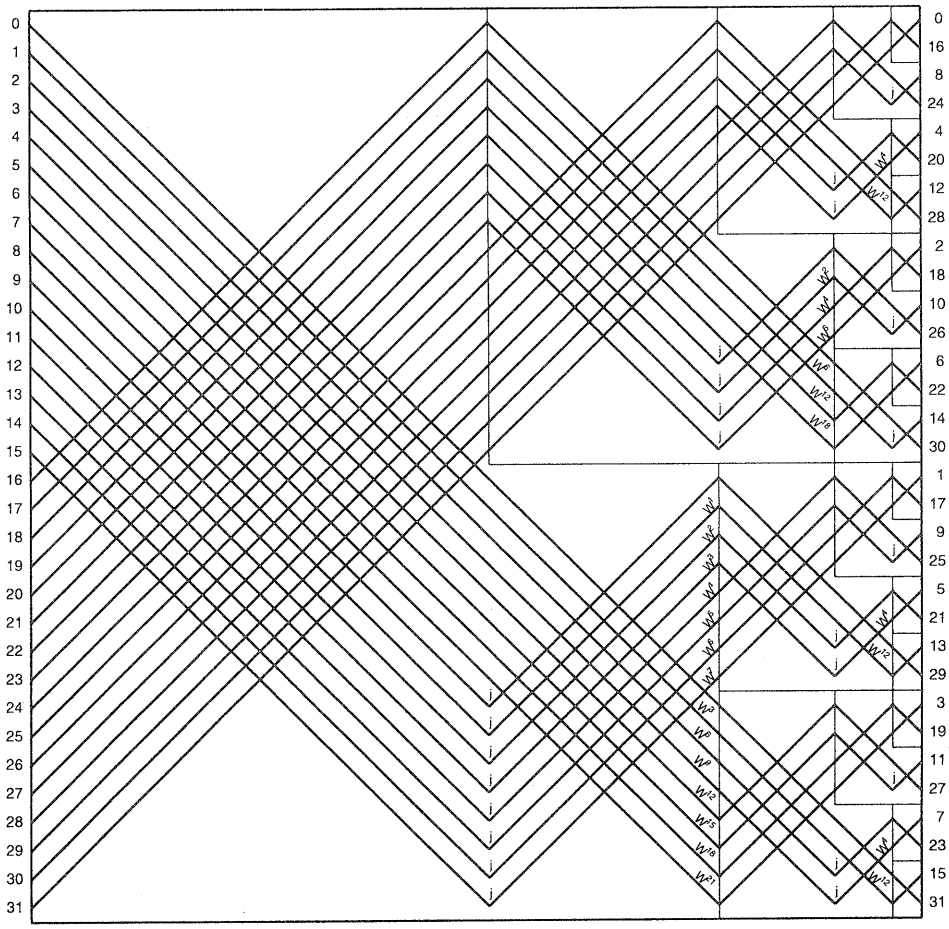

図 1　$N = 32$ の場合の従来の split-radix FFT アルゴリズム（DIF）

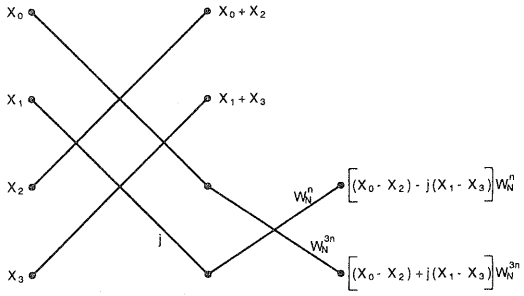

図 2　従来の split-radix FFT のバタフライ演算（DIF）

ここで，$W_N = \exp(-j2\pi/N)$，$j = \sqrt{-1}$ であり，X_k と x_n は複素数である．

式 (1) において，$N = 2^m$ であるとき，

$$X_{2k} = \sum_{n=0}^{N/2-1} \left[x_n + x_{n+N/2}\right] W_N^{2nk} \qquad (2)$$

が偶数番目のデータに対して成り立ち，

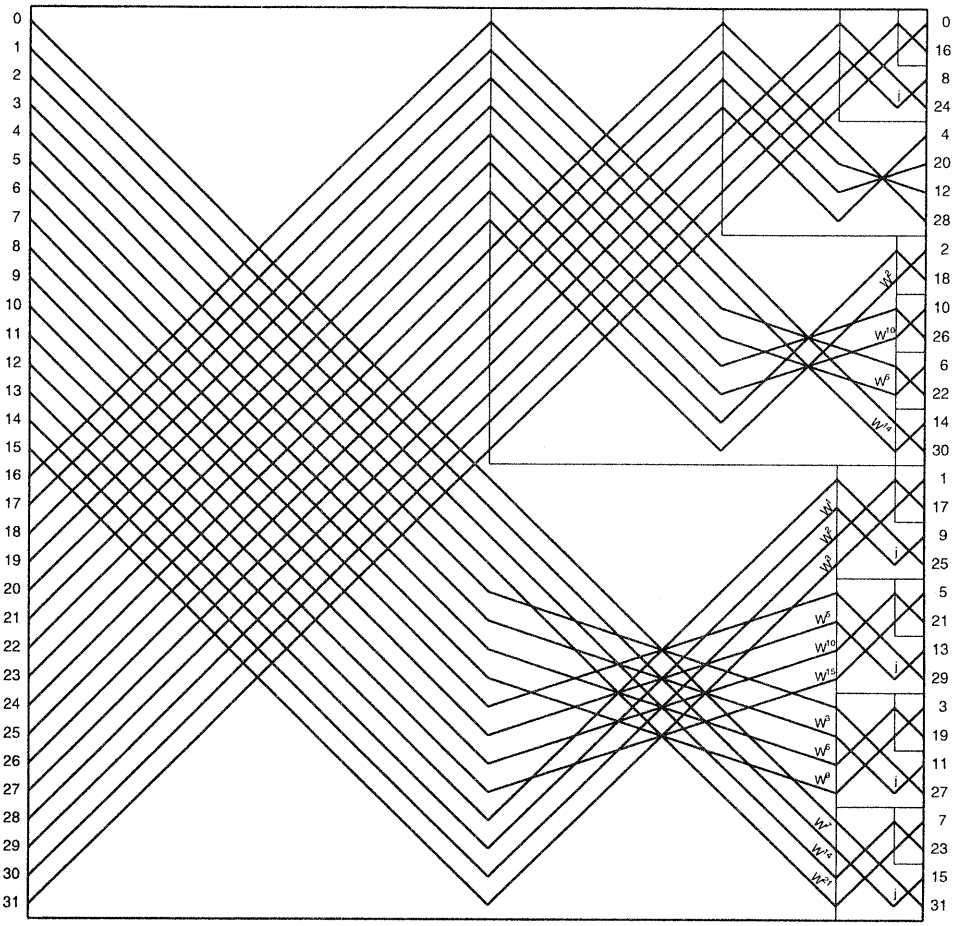

図 3　$N = 32$ の場合の拡張 split-radix FFT アルゴリズム（DIF）

$$X_{4k+1} = \sum_{n=0}^{N/4-1} \Big[\big(x_n - x_{n+N/2}\big) - j\big(x_{n+N/4} - x_{n+3N/4}\big) \Big] W_N^n W_N^{4nk} \tag{3}$$

$$X_{4k+3} = \sum_{n=0}^{N/4-1} \Big[\big(x_n - x_{n+N/2}\big) + j\big(x_{n+N/4} - x_{n+3N/4}\big) \Big] W_N^{3n} W_N^{4nk} \tag{4}$$

が奇数番目のデータに対して成り立つ．

$N = 32$ の場合の従来の周波数間引き（decimation-in-frequency，以下 DIF）split-radix FFT アルゴリズムを**図 1** に，従来の split-radix FFT のバタフライ演算（DIF）を**図 2** に示す．

図 1 に示すように，従来の split-radix FFT アルゴリズムではバタフライ演算 1 回につき，変換が 1 段進む上半分（基数 2 の部分）と，2 段進む下半分（基数 4 の部分）から構成される L 字型のバタフライ演算となる．図 1 において，縦の細い線で区切られている箇所で，データのロードおよびストアが必要になる．

つまり，従来の split-radix FFT アルゴリズムでは，バタフライ演算の上半分では基数 2 の FFT と同等のメモリアクセス回数が，下半分では基数 4 の FFT と同等のメモリアクセス回数が必要になる．これから分かるように，split-radix FFT アルゴリズムのメモリアクセス回数は，基数 2 の FFT と基数 4 の FFT のメモリアクセス回数の間に位置することになる．

基数が大きいほどメモリアクセスが少なくなることを考えると，split-radix FFT では，基数 2 の FFT に比べるとメモリアクセス回数は少なくなっているもの

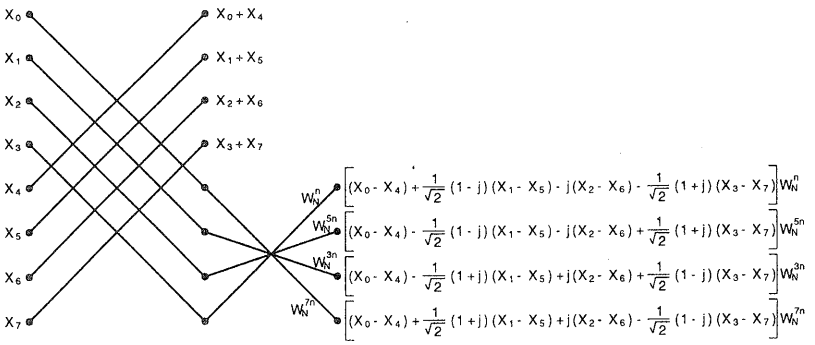

図 4　拡張 split-radix FFT のバタフライ演算（DIF）

の，基数 4，8 の FFT に比べてメモリアクセス回数が多くなる．

最近のプロセッサでは，基数 4 の FFT でもメモリバンド幅が不足する場合があることが指摘されている [7] ことを考えると，従来の split-radix FFT は演算回数の観点では基数 4，8 の FFT より有利であるが，メモリアクセス回数の観点では逆に不利であることが分かる．

3. 拡張 split-radix FFT アルゴリズム

本論文で提案する拡張 split-radix FFT アルゴリズムの基本的なアイデアは，偶数番目のデータに対しては基数 2 の分解を適用するという点については従来の split-radix FFT アルゴリズムと同様であるが，奇数番目のデータに対しては基数 4 ではなく，基数 8 の分解を適用するという点が異なっている．

奇数番目のデータに対して，基数 8 の分解を用いることにより，基数 4 の分解を用いた場合に比べて乗算回数が削減され，さらにメモリアクセス回数も削減することができる．結局，拡張 split-radix FFT アルゴリズムでは 1/2 の長さの 1 つの DFT と，1/8 の長さの 4 つの DFT を統合していることになる．

式 (1) において，$N = 2^m$ であるとき，

$$X_{2k} = \sum_{n=0}^{N/2-1} \left[x_n + x_{n+N/2}\right] W_N^{2nk} \tag{5}$$

が偶数番目のデータに対して成り立ち，

$$\begin{aligned} X_{8k+1} &= \sum_{n=0}^{N/8-1} \Bigg[\left(x_n - x_{n+N/2}\right) \\ &\quad + \frac{1}{\sqrt{2}}(1-j)\left(x_{n+N/8} - x_{n+5N/8}\right) \\ &\quad - j\left(x_{n+N/4} - x_{n+3N/4}\right) \\ &\quad - \frac{1}{\sqrt{2}}(1+j)\left(x_{n+3N/8} - x_{n+7N/8}\right) \Bigg] W_N^n W_N^{8nk} \end{aligned} \tag{6}$$

$$\begin{aligned} X_{8k+3} &= \sum_{n=0}^{N/8-1} \Bigg[\left(x_n - x_{n+N/2}\right) \\ &\quad - \frac{1}{\sqrt{2}}(1+j)\left(x_{n+N/8} - x_{n+5N/8}\right) \\ &\quad + j\left(x_{n+N/4} - x_{n+3N/4}\right) \\ &\quad + \frac{1}{\sqrt{2}}(1-j)\left(x_{n+3N/8} - x_{n+7N/8}\right) \Bigg] W_N^{3n} W_N^{8nk} \end{aligned} \tag{7}$$

$$\begin{aligned} X_{8k+5} &= \sum_{n=0}^{N/8-1} \Bigg[\left(x_n - x_{n+N/2}\right) \\ &\quad - \frac{1}{\sqrt{2}}(1-j)\left(x_{n+N/8} - x_{n+5N/8}\right) \\ &\quad - j\left(x_{n+N/4} - x_{n+3N/4}\right) \\ &\quad + \frac{1}{\sqrt{2}}(1+j)\left(x_{n+3N/8} - x_{n+7N/8}\right) \Bigg] W_N^{5n} W_N^{8nk} \end{aligned} \tag{8}$$

$$\begin{aligned} X_{8k+7} &= \sum_{n=0}^{N/8-1} \Bigg[\left(x_n - x_{n+N/2}\right) \\ &\quad + \frac{1}{\sqrt{2}}(1+j)\left(x_{n+N/8} - x_{n+5N/8}\right) \\ &\quad + j\left(x_{n+N/4} - x_{n+3N/4}\right) \\ &\quad - \frac{1}{\sqrt{2}}(1-j)\left(x_{n+3N/8} - x_{n+7N/8}\right) \Bigg] W_N^{7n} W_N^{8nk} \end{aligned} \tag{9}$$

が奇数番目のデータに対して成り立つ．

拡張 split-radix 周波数間引き（DIF）分解は，最初のステージで N 点 DFT を 1 つの $N/2$ 点 DFT と 4 つの $N/8$ 点 DFT に置き換える．N 点 DFT において，このような分解を最後から 3 番目のステージまで

表 1　N 点の複素数データに対する FFT アルゴリズムにおける些細でない実数の乗算回数および加算回数

Algorithm	Multiplications	Additions
Radix-2	$2N\log_2 N - 7N + 12$	$3N\log_2 N - 3N + 4$
Radix-4	$(3/2)N\log_2 N - 5N + 8$	$(11/4)N\log_2 N - (13/6)N + (8/3)$
Radix-8	$(4/3)N\log_2 N - (59/14)N + (40/7)$	$(11/4)N\log_2 N - (57/28)N + (16/7)$
Conventional split-radix[8]	$(4/3)N\log_2 N - (38/9)N + 6$ $+ (2/9)(-1)^{\log_2 N}$	$(8/3)N\log_2 N - (16/9)N + 2$ $- (2/9)(-1)^{\log_2 N}$
Extended split-radix	$(5/4)N\log_2 N - (57/16)N + 4$ $- (7/(32N))(\alpha^{\log_2 N} + \beta^{\log_2 N})$ $- (j13\sqrt{7}/(224N))(\alpha^{\log_2 N} - \beta^{\log_2 N})$	$(11/4)N\log_2 N - (31/16)N + 2$ $- (1/(32N))(\alpha^{\log_2 N} + \beta^{\log_2 N})$ $- (j11\sqrt{7}/(224N))(\alpha^{\log_2 N} - \beta^{\log_2 N})$

表 2　N 点の複素数データに対する FFT アルゴリズムにおける些細でない実数の乗算回数および加算回数

N	radix-2		radix-4		radix-8		Conventional split-radix[8]		Extended split-radix	
	Mults	Adds	Mults	Adds	Mults	Adds	Mults	Adds	Mults	Adds
8	4	52			4	52	4	52	4	52
16	28	148	24	144			24	144	28	148
32	108	388					84	372	92	380
64	332	964	264	920	248	928	248	912	252	932
128	908	2308					660	2164	668	2220
256	2316	5380	1800	5080			1656	5008	1660	5140
512	5644	12292			3992	11632	3988	11380	3932	11676
1024	13324	27652	10248	25944			9336	25488	9148	26180
2048	30732	61444					21396	56436	20892	57996
4096	69644	135172	53256	126296	48280	126832	48248	123792	46844	127220

表 3　基数 2, 4, 8 の FFT, そして従来の split-radix FFT および拡張 split-radix FFT の各バタフライ演算における実数のロード・ストア回数, 乗算回数および加算回数

Algorithm	Loads	Stores	Mults	Adds
Radix-2	6	4	4	6
Radix-4	14	8	12	22
Radix-8	30	16	32	66
Conventional split-radix	12	8	8	16
Extended split-radix	24	16	20	44

表 4　N 点 FFT における漸近的なロード・ストア回数, 乗算回数および加算回数（比較のために, $N\log_2 N$ で割っている）

Algorithm	Loads	Stores	Mults	Adds
Radix-2	3	2	2	3
Radix-4	7/4	1	3/2	11/4
Radix-8	5/4	2/3	4/3	11/4
Conventional split-radix	2	4/3	4/3	8/3
Extended split-radix	3/2	1	5/4	11/4

継続的に行う. 最後から 2 番目のステージでは, いくつかの従来の split-radix バタフライ演算（ひねり係数の乗算なし）が必要になる. 最後のステージでは, いくつかの基数 2 のバタフライ演算が行われる.

$N = 32$ の場合の拡張 split-radix FFT アルゴリズム（DIF）を図 3 に, 拡張 split-radix FFT のバタフライ演算（DIF）を図 4 に示す.

図 3 に示すように, 拡張 split-radix FFT アルゴリズムではバタフライ演算 1 回につき, 変換が 1 段進む上半分（基数 2 の部分）と, 3 段進む下半分（基数 8 の部分）から構成される L 字型のバタフライ演算となる. したがって, アルゴリズムは従来の split-radix FFT アルゴリズムに比べてやや複雑になる. しかし, 図 3 は図 1 と比べると, 縦の細い線の本数が少なくなっており, 拡張 split-radix FFT のメモリアクセス回数は従来の split-radixFFT に比べて少なくなっていることが分かる.

4. 評　　価

$M(N)$ および $A(N)$ を N 点 FFT を行う際の実数の乗算回数および加算回数とする. なお本論文では, 複素数の積は 4 回の実数の乗算と 2 回の実数の加算により行うと仮定する.

式 (5)~(9) より，次式を得る．

$$M(N) = M(N/2) + 4M(N/8) + (5/2)N - 16$$
$$A(N) = A(N/2) + 4A(N/8) + (11/2)N - 8$$

上式に初期条件 $M(2) = 0$, $M(4) = 0$, $M(8) = 4$ および $A(2) = 4$, $A(4) = 16$, $A(8) = 52$ を与え，さらに N について解くことにより，次式を得る．

$$M(N) = (5/4)N\log_2 N - (57/16)N + 4 - (7/(32N))(\alpha^{\log_2 N} + \beta^{\log_2 N}) - (j13\sqrt{7}/(224N))(\alpha^{\log_2 N} - \beta^{\log_2 N})$$
$$A(N) = (11/4)N\log_2 N - (31/16)N + 2 - (1/(32N))(\alpha^{\log_2 N} + \beta^{\log_2 N}) - (j11\sqrt{7}/(224N))(\alpha^{\log_2 N} - \beta^{\log_2 N})$$

ここで，$\alpha = -1 + j\sqrt{7}$, $\beta = -1 - j\sqrt{7}$ である．

N 点の複素数データに対する基数 2，4，8 の FFT，そして従来の split-radix FFT および拡張 split-radix FFT における些細でない（± 1, $\pm j$ による乗算を除外した）実数の乗算回数および加算回数を**表 1** と**表 2** に示す．表 1 から分かるように，拡張 split-radix FFT アルゴリズムは，基数 4 の FFT に比べて漸近的な実数の乗算回数が約 17%削減されており，また基数 8 の FFT に比べて漸近的な実数の乗算回数が約 6%削減されている．

従来の split-radix FFT アルゴリズムと比較すると，拡張 split-radix FFT アルゴリズムは漸近的な実数の乗算回数が約 6%削減されていることが分かる．一方，拡張 split-radix FFT アルゴリズムの漸近的な実数の加算回数は従来の split-radix FFT アルゴリズムに比べて約 3%増加している．結局，拡張 split-radix FFT アルゴリズムの漸近的な実数の乗算回数と加算回数の合計は，従来の split-radix FFT アルゴリズムと同じ $4n\log_2 n$ になる．

基数 2，4，8 の FFT，そして従来の split-radix FFT および拡張 split-radix FFT の各バタフライ演算における実数のロード・ストア回数，乗算回数および加算回数を**表 3** に示す．ロード・ストア回数の算出にあたっては，それぞれの FFT アルゴリズムのバタフライ演算をすべて浮動小数点レジスタ上で行えるだけのレジスタ数があると仮定している．表 3 から，拡張 split-radix FFT では，バタフライ演算内の演算回数が基数 4 の FFT の約 2 倍に，基数 8 の FFT の約 2/3 倍になっていることが分かる．

表 4 に，それぞれの FFT アルゴリズムの漸近的なロード・ストア回数，乗算回数および加算回数を示す．表 4 ではひねり係数のロード回数が含まれていることに注意する．拡張 split-radix FFT は基数 4 の FFT および従来の split-radix FFT に比べて少ないロード・ストア回数で済むことが分かる．特に，従来の split-radix FFT と比較すると，拡張 split-radix FFT は漸近的に 25%のロード・ストア回数を削減している．

5. ま と め

本論文では，拡張 split-radix FFT アルゴリズムを提案した．この拡張 split-radix FFT アルゴリズムは従来の split-radix FFT アルゴリズムに比べて乗算回数が少なくなること，そして従来の split-radix FFT，基数 4 の FFT に比べて，ロードおよびストア回数が少なくなることを示した．拡張 split-radix FFT の効率の良い実装手法および評価が今後の課題である．

謝辞 本研究の一部は，文部省科学研究費補助金奨励研究（A）（課題番号 12780190）の支援を受けた．

参 考 文 献

1) Cooley, J. W. and Tukey, J. W.: An Algorithm for the Machine Calculation of Complex Fourier Series, *Math. Comput.*, Vol.19, pp.297–301 (1965).
2) Duhamel, P. and Hollmann, H.: Split-Radix FFT Algorithm, *Electron. Lett.*, Vol. 20, pp. 14–16 (1984).
3) Brigham, E. O.: *The Fast Fourier Transform and its Applications*, Prentice-Hall, Englewood Cliffs, NJ (1988).
4) Duhamel, P.: Implementation of "Split-Radix" FFT Algorithms for Complex, Real, and Real-Symmetric Data, *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-34, pp. 285–295 (1986).
5) Van Loan, C.: *Computational Frameworks for the Fast Fourier Transform*, SIAM Press, Philadelphia, PA (1992).
6) Bergland, G.D.: A Fast Fourier Transform Algorithm Using Base 8 Iterations, *Math. Comput.*, Vol. 22, pp. 275–279 (1968).
7) 高橋大介，金田康正: 積和演算命令に向いた 8 基底 FFT カーネルの提案，情報処理学会論文誌，Vol. 41, pp. 2018–2026 (2000).
8) Sorensen, H. V., Heideman, M. T. and Burrus, C.S.: On Computing the Split-Radix FFT, *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-34, pp. 152–156 (1986).