Fast Partial Frequency Spectrum Computation for Real-Time Information Acquisition Systems

Chang-Eup Ha*, Wan-Jin Kim*, Dae-Won Do**, Dong-Hun Lee**, and Hyoung-Nam Kim*
*School of Electrical Engineering, Pusan National University, Busan, Korea
**Agency for Defense Development, Jinhae, Korea
hnkim@pusan.ac.kr

ABSTRACT
In radar and sonar systems, it is important to get a frequency spectrum on real time. To implement a high performance system, the update interval should be as short as possible, but this requirement is not easily achievable due to the limited hardware resource. Although some fast Fourier transform (FFT) algorithms and pruning techniques may be able to reduce computational complexity, the use of them does not guarantee the best solution. To overcome this problem, we propose a pruned generalized FFT (PGSFFT) combining the GSFFT and transform decomposition (TD). Since the PGSFFT takes advantages of both GSFFT and TD, it is possible to reduce the computational complexity of the system. The enhanced computational efficiency leads to the improvement of the system performance. To verify the performance of the proposed method, we analyze the required operations for the PGSFFT and perform simulations. Simulation results show that the proposed PGSFFT has better performance than other FFTs or pruning algorithms.

KEY WORDS
Pruning, Transform decomposition, GSFFT, sliding DFT

1. Introduction

In radar and sonar systems, target information such as distance, bearing, and Doppler shift should be obtained on real time for target detection and estimation. For the real time information acquisition, it seems that time-domain processing is better than frequency-domain processing due to the computational burden. However, since the information which can be extracted from time-domain signal is restricted, the use of frequency-domain processing is inevitable. For frequency-domain processing, the first step is to transform the time-domain signal into its corresponding frequency domain signal. In this step, the widely-used method is the fast Fourier transform (FFT) [1]. It is certain that the FFT is an effective transform method but it may not be exactly true in some situations, for example: an update interval is very short or only partial range of frequency spectrum is required. In detection systems, information update interval should be as short as possible for fast and precise processing, but the computational burden, which is related to the number of executed FFT, arithmetically increases by decreasing the update interval. If the burden is acceptable in the given hardware resource, it may not be a problem. However, if the burden exceeds the system capability, it is hard to process the data in the frequency domain on real time. This short update interval problem may be solved by adopting additional hardware resource but it is not a recommended solution. Especially, in case of the partial use of frequency spectrum, a waste of the hardware resource is inevitable. Radar and active sonar systems radiate a pulse and detect a signal reflected from targets. Even though the received signal is distorted and Doppler shifted, its frequency characteristic does not change abruptly. This fact means that frequency band required for the detection is not a full-band but a part of it. It is required to eliminate unnecessary operations for the reduction of computational load but the FFT is not able to calculate only partial frequency band, resulting in hardware resource waste.

These problems of the short update interval and partial use of the spectrum are closely connected to the system performance and they should be solved to implement an optimized system. Some researchers have dealt with these problems and they have presented some remarkable methods: generalized sliding FFT (SFFT) and pruning techniques such as transform decomposition (TD) [2]-[5]. The GSFFT is one of the computing methods of sequential discrete Fourier transform (DFT) using FFT and it has lower computation complexity than recursive FFT computation. The TD is one of the existing high efficiency pruning methods and it is able to compute only desired frequency bins. While the GSFFT and the TD may be a good solution to overcome the short update interval and partial use of the spectrum, respectively, they may not be when both requirements have to be dealt with in one process. To cope with the situation where two demands are required simultaneously, we propose a new method called the pruned GSFFT (PGSFFT). The PGSFFT is based on the combination of the GSFFT and the TD. The PGSFFT utilizes both of their advantages, and thus it is able to reduce the computational complexity in the given situation.

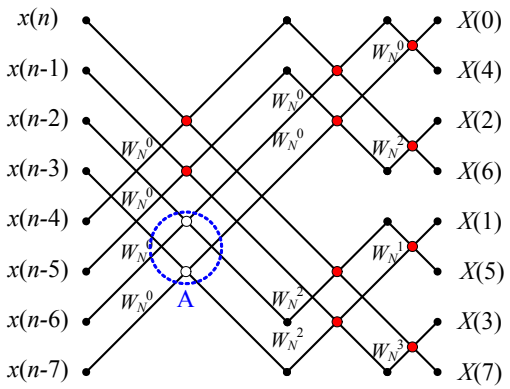


Fig. 1. A general FFT structure at $N = 8$. The dotted circle marked 'A' is the overlapped part of the previous and current FFT computation when the input data hops two samples at a time. Except the butterfly marked 'A', the other butterflies should be computed to get frequency bins.

The rest of this paper is organized as follows. In section 2, we briefly describe the GSFFT and the TD which compose the proposed algorithm. The description on our proposed method and its computational complexity analysis are presented in section 3. In section 4, we show the computational efficiency of PGSFFT compared to other FFT and pruning techniques. Finally in section 5, we conclude the paper.

2. Overview of the GSFFT and TD

2.1 Generalized Sliding FFT

The sliding DFT (SDFT) is a computing method to sequentially get frequency-domain signal whenever the time-limited rectangular window is moved one sample [6]. The concept of the SDFT is that the DFT of the current input sequence in an N-points window can be obtained just with one complex multiplication and two complex additions when the one-sample-advanced DFT is known. Especially if the input sequence length N is a power of two, the SDFT can be shown as N-point FFT which requires N complex multiplications. From this idea, the sliding FFT (SFFT) is developed [2], [7]. The SFFT is very efficient method compared to recursive FFT computations but the remained problem is that the SFFT is only useful when input sequence slides one sample at a time.

The generalized SFFT (GSFFT) is a method which hops power of two samples and the GSFFT makes it possible to eliminate the redundancies between consecutive slides. To help the understanding, an example is illustrated in fig. 1. If the FFT computation is performed every two samples, the overlapped part between the previous and current window is dotted circle which is marked by 'A' in fig. 1. In the GSFFT, the 'A' marked butterfly is not computed because it has been already computed and saved to the memory in the previous calculation. Eliminating the

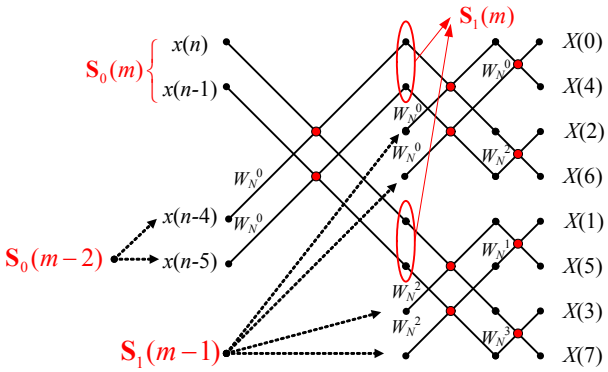


Fig. 2. A structure of generalized sliding FFT at $N = 8$. The dotted line means that data which are computed and saved at the previous computation is loaded from the memory.

redundancy part in fig. 1, the general FFT structure can be modified as fig. 2.

For further explanation on the GSFFT, let $\mathbf{U}(m)$ be input signals which is moved into the window at the current iteration, then and $\mathbf{U}(m)$ is defined as follows [7]

$$\mathbf{U}(m) = [x(n)\ x(n-1)\ \ldots\ x(n-2^k+1)]^T, \quad (1)$$

where $x(n)$ is input signal. k is the number of nodes which do not use a previously saved data and it is logarithm of M to base 2. Using (1), the input vector $\mathbf{x}(m)$ in the window at m-th iteration can be expressed by [7]

$$\begin{aligned}\mathbf{x}(m) &= [\mathbf{U}^T(m)\ \mathbf{U}^T(m-1)\ \ldots\ \mathbf{U}^T(m-2^q+1)]^T \\ &= [x(n)\ x(n-1)\ \ldots\ x(n-N+1)]^T\end{aligned}, \quad (2)$$

where q is the number of nodes which use a previously saved data and it is the logarithm of N/M to base 2. The state vector $\mathbf{S}_i(m)$ at i-th node is defined as and i is the node index whose range is $0 \le i < k + q$. For $0 \le i < q$, $\mathbf{S}_i(m)$ is expressed by [7]

$$\mathbf{S}_{i+1}(m) = \left(\mathbf{P}_{i,k}\left(\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\otimes\mathbf{I}_{2^{k+i}}\right)\right)\begin{bmatrix}\mathbf{S}_i(m)\\ \mathbf{V}_{i,k}\mathbf{S}_i(m-2^{q-i-1})\end{bmatrix}, \quad (3)$$

where

$$\begin{cases}\mathbf{S}_0(m) = \mathbf{U}(m)\\ \mathbf{P}_{i,k} = \mathbf{P}_i \otimes \mathbf{I}_{2^k}\\ \mathbf{V}_{i,k} = \mathbf{V}_i \otimes \mathbf{I}_{2^k}\\ \mathbf{P}_i = [\mathbf{e}_{\text{odd}}\ \mathbf{e}_{\text{even}}]\\ \mathbf{V}_i = \text{diag}(W_N^{\sigma(0)}, W_N^{\sigma(1)}, \ldots, W_N^{\sigma(2^i-1)})\end{cases}. \quad (4)$$

Here, $\mathbf{I}_2{}^k$ denotes the 2^k by 2^k identity matrix and $\otimes$ is the Kroneker product. $\mathbf{P}_i$ denotes a 2^{i+1} by 2^{i+1} permutation matrix where $\mathbf{e}_{\text{odd}}$ is $[\mathbf{e}_1\ \mathbf{e}_3\ \ldots\ \mathbf{e}_{2^{i+1}-1}]$ and $\mathbf{e}_{\text{even}}$ is $[\mathbf{e}_2\ \mathbf{e}_4\ \ldots\ \mathbf{e}_{2^{i+1}}]$ and $\mathbf{e}_r$ means the r-th column of the 2^{i+1} by 2^{i+1} identity matrix. $\mathbf{V}_i$ is a diagonal matrix which is

composed of twiddle factor W_N and $\sigma(r)$ is the bit reverse of r. For $q \le i < k+q$, $\mathbf{S}_i(m)$ is expressed by [7]

$$\mathbf{S}_i(m) = [S_{i,0}(m)\ S_{i,1}(m)\ S_{i,N-1}(m)]^T, \quad (5)$$

and the next state is evaluated as follows

$$\begin{bmatrix} \mathbf{S}_{i+1,t}(m) \\ \mathbf{S}_{i+1,t+2^{k+q-i-1}}(m) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \mathbf{S}_{i,t}(m) \\ \mathbf{V}_{i,t}\mathbf{S}_{i,t+2^{k+q-i-1}}(m) \end{bmatrix}, \quad (6)$$

where $\mathbf{V}_{i,t}$ is $W_N^{\sigma_i(t)}$ and $\sigma_i(t)$ is equal to bit reverse of $(t-1) \bmod N / 2^{i+1}$.

The above equations, especially (3) and (6), show that the computational complexity of the GSFFT depends on the updated interval M and the window size N. To clarify the effect by various M and N, it is needed to quantify the computational complexity of the GSFFT. The number of complex multiplication is equal to the sum of the number of all twiddle factors. In (3), the number of required twiddle factors is 2^{i+k} at each node and their sum is

$$\sum_{i=0}^{i=q-1} 2^{i+k} = 2^k(2^0 + 2^1 + \cdots + 2^{q-1}) = 2^{q+k} - 2^k = N - M. \quad (7)$$

Similarly, the sum of twiddle factors of (6) based on general radix 2 FFT computation is

$$\sum_{i=q}^{i=q+k-1} \frac{N}{2} = \frac{kN}{2} = \frac{N}{2}\log_2 M. \quad (8)$$

From (6) and (7), the total number of complex multiplications for the GSFFT is $(N/2)\log_2 M + N - M$. One complex multiplication is composed of 2 real additions and 4 real multiplications and 4 real additions per butterfly are required. The number of real additions and multiplications is $3N\log_2 M + 6N - 6M$ and $2N\log_2 M + 4N - 4M$ and, respectively. Therefore, the total number of operations is $5N\log_2 M + 10N - 10M$.

2.2 Transform Decomposition

The GSFFT can reduce the computational complexity by omitting redundant calculations which are caused by the overlapped data samples between the previous and current window. However, when the desired part of the frequency spectrum is computed, it is not possible to use the GSFFT for reducing the unnecessary calculations. This problem is due to the fact that the GSFFT is able to produce full frequency spectrum. The pruning techniques are a method which eliminates the computation by the unessential input or output, and thus they may be a good solution to overcome the disadvantage of GSFFT. There exists many pruning methods but we only deal with the transform decomposition (TD) which is known to have high computation efficiency.

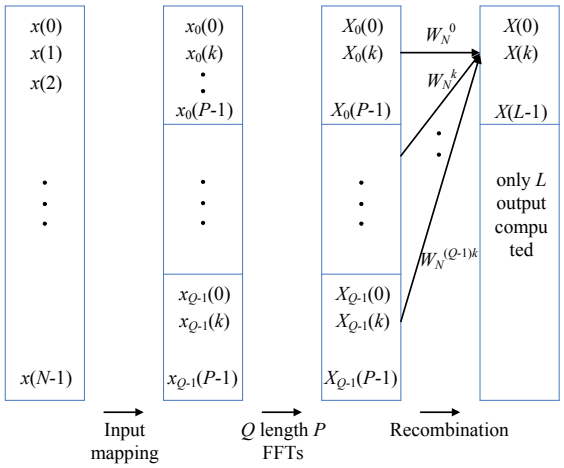


Fig. 3. The block diagram of the transform decomposition.

The TD technique is based on the divide-and-conquer approach of the Cooley-Tukey FFT algorithm. The main idea of divide-and- conquer approach is that the DFT can be divided into small DFTs and obtained by combining their results. Fig. 3 shows the block diagram of the TD for output pruning. The TD is largely composed of three procedures: realignment of input sequence, Q P-point FFTs and recombination of the results. The detailed description of the TD is as follows [3].

The DFT is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad k = 0,1,\ldots,N-1. \quad (9)$$

In the first step of the TD, the input data sequence whose length is N is restructured to Q subsets that each subset has P input data. Then the time index n can be written as

$$n = Qn_1 + n_2, \text{ where } \begin{cases} n_1 = 0,1,\ldots,P-1 \\ n_2 = 0,1,\ldots,Q-1 \end{cases}, \quad (10)$$

and we can rewrite (9) as follow

$$\begin{aligned} X(k) &= \sum_{n_2=0}^{Q-1} \sum_{n_1=0}^{P-1} x(n_1Q + n_2) W_N^{(n_1Q+n_2)k} \\ &= \sum_{n_2=0}^{Q-1} \left[\sum_{n_1=0}^{P-1} x(n_1Q + n_2) W_N^{n_1 <k>_p} \right] W_N^{n_2 k} \end{aligned}, \quad (11)$$

where $<\ >_p$ denotes the modulo-P operator. It is possible to divide (11) into two equations and they are

$$X(k) = \sum_{n_2=0}^{Q-1} X_{n_2}(<k>_p) W_N^{n_2 k} \quad (12)$$

and

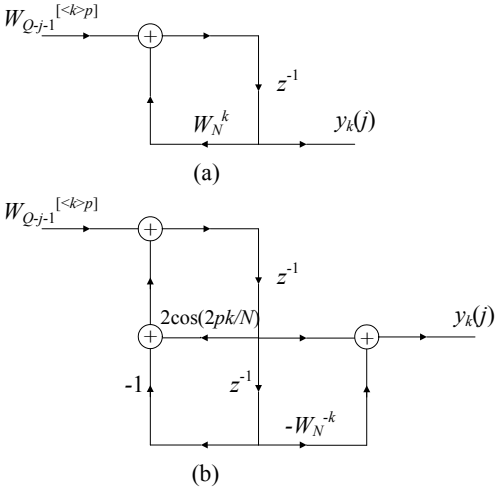


Fig. 4. (a) Flow graph of the first-order Goertzel algorithm; (b) Flow graph of the second-order Goertzel algorithm.

$$X_{n_2}(j) = \sum_{n_1=0}^{P-1} x(n_1 Q + n_2) W_P^{n_1 j} = \sum_{n_1=0}^{P-1} x_{n_2}(n_1) W_P^{n_1 j}, \quad (13)$$

where the range of j is from 0 to $P-1$. The second step of the TD is calculation of Q P-point FFTs of (13). In this step, the computational efficiency of TD varies with the algorithm which is used for the computation of P-point FFT. In general, the spilt radix FFT algorithm is known to have the lowest complex multiplications and additions among the power of two FFT algorithms [8]. However, unlike the general radix 2 or 4 FFT algorithm, the number of butterflies is varied in successive stages. Since this problem makes it difficult to implement the radix algorithm on the hardware [9], we consider only the Cooley-Tukey FFT algorithm. The last step is to combine outputs of (13) using (12). In (12), the combining process is performed by the DFT but the computing method is not restricted to the DFT. The Goertzel algorithm based on an IIR filtering approach is one of the methods to improve computational efficiency [10] and it can be derived from (12) as follows

$$\begin{aligned} X(k) &= \sum_{n_2=0}^{Q-1} X_{n_2}(< k >_p) W_N^{n_2 k} \\ &= \sum_{m=0}^{Q-1} X_{Q-m-1}(< k >_p)(W_N^k)^{Q-m-1} \end{aligned}, \quad (14)$$

where $m = Q - n_2 - 1$. Then let $y_k(j)$ as

$$y_k(j) = \sum_{m=0}^{j-1} X_{Q-m-1}(< k >_p)(W_N^k)^{j-m-1}. \quad (15)$$

From (14) and (15), $X(k)$ can be described as

$$X(k) = y_k(j)\big|_{j=Q}. \quad (16)$$

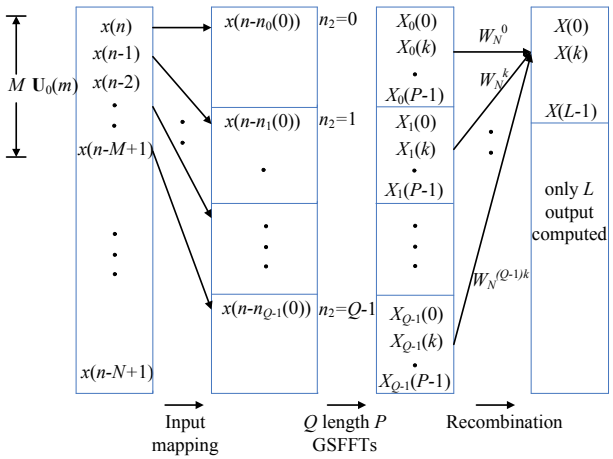


Fig. 5. Pruned generalized sliding FFT (PGSFFT)

This equation means that $X(k)$ is considered as the Q-th output of the convolution between the sequence $(W_N^k)^{j-1}$ and $X_{Q-j-1}(< k >)_p$. Fig. 4 (a) and (b) show the flow graph of (15) and its modified version, respectively. If the flow graph of Fig. 4 (b) is applied to the TD, it is able to reduce 4 multiplications to 2 multiplications per iteration.

To evaluate the computational complexity of the TD, we assumed that the radix 2 FFT algorithm and the modified Goertzel algorithm of fig. 4 (b) are used for minimizing the computational complexity. If the length of the desired frequency bins is L, the number of real multiplications for TD is $2N\log_2 P + 2LQ + 2L$. The number of real additions is $3N\log_2 P + 4L(Q-1) + 4L$ and the total number of operations is $5N\log_2 P + 6LQ + 2L$.

3. Pruned GSFFT

The concept of the proposed pruned GSFFT is based on the idea that the P-point FFT of the TD is possible to be replaced with other FFT algorithms. Since the aim of this paper is to present the method which extracts the desired frequency band while sliding, we propose the efficient sliding output pruning technique by combining the GSFFT and the TD. That is, the GSFFT is used for P-point FFT instead of the Cooley-Tukey algorithm. This proposed method is called pruned GSFFT (PGSFFT) and fig. 5 shows how the PGSFFT works.

Letting M be the update interval and N be the window length. In the GSFFT, we can freely select M if it is only satisfied the condition that power of two. In the PGSFFT, assuming that N is the product of P and Q, M has to meet the additional condition as follows

$$M = Qx \quad \text{where } x = 2^d, 0 \le d \le \log_2 N. \quad (17)$$

The updated inputs of PGSFFT at the m-th iteration is presented in (1) and they are distributed to the n_2-th P-point GSFFT as follows

$$\mathbf{S}_0^{n_2}(m) = \left[x(n-R(0)) \;\; x(n-R(1)) \;\; \ldots \;\; x\left(n-R\left(\frac{M}{Q}-1\right)\right)\right]^T . \quad (18)$$

where

$$R(n_1) = Qn_1 + n_2, \text{ where } \begin{cases} n_1 = 0,1,\ldots,M/Q-1 \\ n_2 = 0,1,\ldots,Q-1 \end{cases}, \quad (19)$$

Each P-point GSFFT block can be computed as described in (3) and (6). Each output of the P-point GSFFT block is equal to (13) and (15) can be applied to each output of the P-point GSFFT block to compute only a subset of frequency points.

To evaluate the computational complexity of the proposed method, we analyzed the number of required operations. The number of real multiplications for Q times P-point GSFFT block with updated input length M/Q is $Q\cdot[2N\log_2(M/Q) + 4P - 4(M/Q)]$. If we assume that L is the length of the output subset, the number of required real multiplications for the PGSFFT can be obtained by

$$\begin{aligned} &\mathrm{MUL}_{\mathrm{PGSFFT}} \\ &= Q\left(2P\log_2\left(\frac{M}{Q}\right)+4P-4\left(\frac{M}{Q}\right)\right)+2L(Q+1). \quad (20) \\ &= 2N\log_2\left(\frac{MP}{N}\right)+4N-4M+\frac{2LN}{P}+2L \end{aligned}$$

The total number of real additions required for the proposed PGSFFT is

$$\begin{aligned} &\mathrm{ADD}_{\mathrm{PGSFFT}} \\ &= Q\left(3P\log_2\left(\frac{M}{Q}\right)+6P-6\left(\frac{M}{Q}\right)\right)+4LQ. \quad (21) \\ &= 3N\log_2\left(\frac{MP}{N}\right)+6N-6M+\frac{4LN}{P} \end{aligned}$$

Finally the total number of real operations can be expressed as follows

$$\begin{aligned} &\mathrm{OPR}_{\mathrm{PGSFFT}} \\ &= 5N\log_2\left(\frac{MP}{N}\right)+10N-10M+\frac{6LN}{P}+2L. \quad (22) \end{aligned}$$

Optimum value of P which is corresponding to the minimal $\mathrm{OPR}_{\mathrm{PGSFFT}}$ can be calculated by taking the derivative of $\mathrm{OPR}_{\mathrm{PGSFFT}}$ for P. P should satisfy the condition of $\frac{\partial \mathrm{OPR}_{\mathrm{PGSFFT}}}{\partial P} = 0$ and be the closest power of two and less than N.

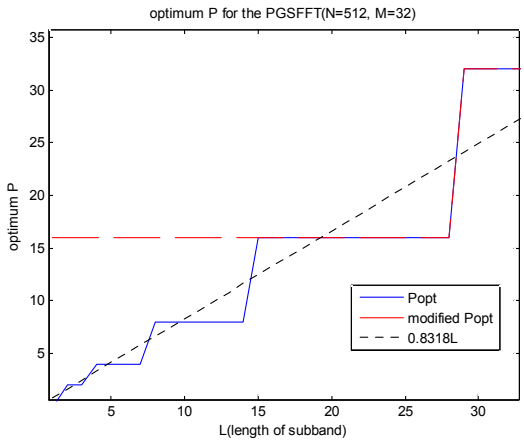


Fig. 6. Optimum value of P and modified optimum P for the case N=512 and M=32. Ideal optimum value of P(solid line) under N/M must be set to N/M(dashed line). In this case, optimum value of P under 16 must be set to 16.

Table1. Computational complexity of various algorithms with PGSFFT

Algorithm	Real operations
Cooley-Tukey [1]	$5N\log_2 N$
Transform Decomposition (Cooley-Tukey) [3]	$5N\log_2 P+\frac{6LN}{P}+2L$
Pruned FFT (Skinner) [5]	$5N\log_2 L+2N-2L$
GSFFT [2]	$5N\log_2 M+10N-10M$
PGSFFT	$5N\log_2(\frac{MP}{N})+10N\text{-}10M+\frac{6LN}{P}+2L$

$$P_{opt} = \left\lfloor P\Big|_{\frac{\partial \mathrm{OPR}_{\mathrm{PGSFFT}}}{\partial P}=0} \right\rfloor = \left\lfloor \frac{6}{5}\ln 2L \right\rfloor = \lfloor 0.8318L \rfloor, \quad (23)$$

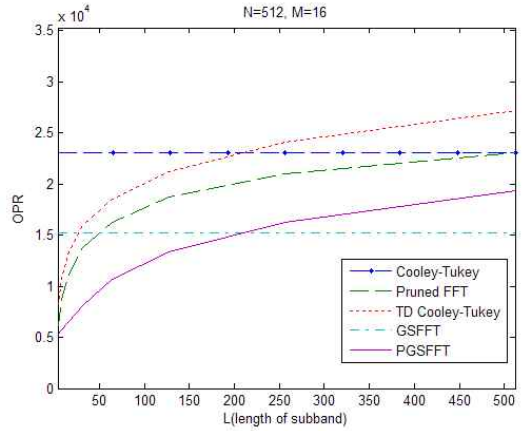
From (17) and N=PQ, P_{opt} should satisfy following equation.

$$P_{\mathrm{opt}} = \frac{N}{M}x \text{ where } x = 2^d, 0 \le d \le \log_2 N. \quad (24)$$

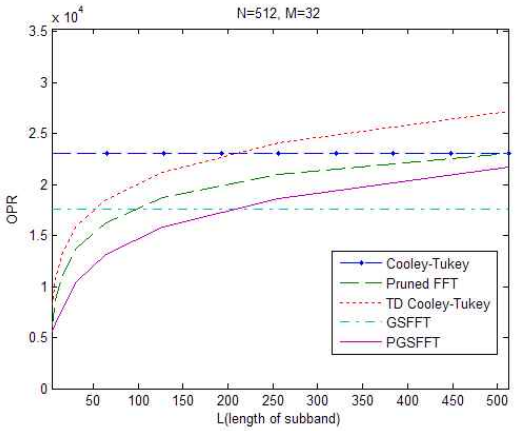
In (24), if P_{opt} is less than N/M, P_{opt} is set to N/M as shown in fig. 6.

4. Computational efficiency of PGSFFT

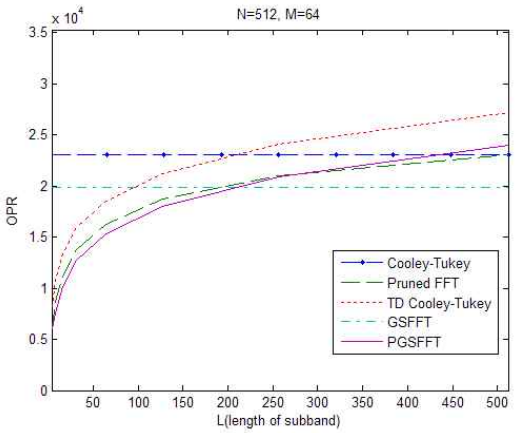
To verify the computational efficiency of the proposed PGSFFT algorithm, we compared various algorithms and it is presented in table 1. In the table 1, we can see the computational complexity of pruning FFT algorithms such as the TD and the Skinner's method is dependent on the subset length L. The GSFFT which is one of the sliding DFT algorithms is dependent on the updated interval M. The optimum value of P_{opt} is used for the transform decomposition using Cooley-Tukey for P-point



(a)



(b)



(c)

Fig. 6. Computational complexity of PGSFFT with various algorithms: (a) M=16; (b) M=32; (c) M=64.

FFT and PGSFFT. PGSFFT is dependent on both M and L. To compare the computational efficiency of algorithms, we fixed the parameters N and M and we performed a computer simulation. Here we note that P_{opt} is given to the TD and PGSFFT using (24). The simulation results are presented in figs. 6 and 7.

When the updated interval of M is less than 32, the computational complexity of the PGSFFT is less than other algorithms until the subset length reaches to

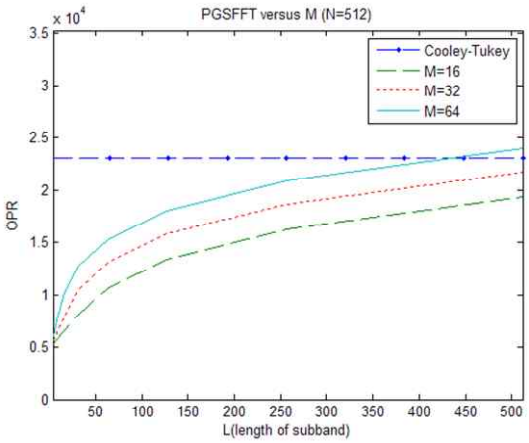


Fig.7. Comparison of the total number of operations required for the PGSFFT versus a various updated sample length (N=512).

approximately 200 samples as shown in fig. 6 (a) and (b). However, As M increases, fig. 6 (c) shows that the computational gain is closed to that of the pruned FFT. If M exceeds 64, PGSFFT has no advantage compared to other algorithms. This trend is due to the increase of butterflies which has to be computed in P-point GSFFT of PGSFFT. This fact means that PGSFFT dose not guarantee the computational efficiency regardless to M and there exists the upper bound of M. To ensure the computational efficiency of the PGSFFT, the M should be selected after comparing the efficiency of PGSFFT and other FFT algorithms by the system requirement.

5. Conclusion

We presented a PGSFFT combining GSFFT and transform decomposition. By the mathematical analysis on the computational complexity, we showed that the number of required operation for the proposed PGSFFT is less than other pruning or sliding DFT algorithms when the update interval M is less than one eighth of window size N (or buffer size) and subband length L is less than N/3. Since the PGSFFT computes the updated input signal while sliding, it is able to show the desired frequency spectrum on real time with relatively lower computational complexity than every-sample FFT. This advantage is directly linked to the performance improvement of the system. Since the PGSFFT can be applied to the system which uses FFT algorithm such as radar, sonar, frequency-domain beamformer, and real-time audio system, it may be contributed to the hardware optimization.

Acknowledgement

This work has been supported by the Agency for Defence Development, South Korea, “Research on the optimum signal processing technique for the wide-band SONAR system” 2009-2011.

References

[1] A.V. Oppenheim, *Discrete-Time Signal Processing,* 2nd ed.(Upper Saddle River, NJ: Prentice-Hall, 1996).

[2] B. Farhang-Boroujeny, S. Gazor, Generalized sliding FFT and its application to implementation of block LMS adaptive filters, *IEEE Trans. Signal Proc., 42*(3), 1994, 532-538.

[3] H. V. Sorensen, C. Sidney, Efficient Computation of the DFT with Only a Subset of Input or Output Points, *IEEE Trans. Signal Processing, 41*(3), 1993, 1184-1199.

[4] J. D. Markel, FFT pruning. *IEEE Trans. Audio Electroacoust., 19*(4), 1971, 305-311.

[5] D.P. Skinner, Pruning the decimation in-time FFT algorithm for computing a few DFT points. *IEEE Trans. Acoust., Speech, Signal Processing,24*(2), 1976, 193-194.

[6] E. Jacobsen, R. Lyons, The sliding DFT, *IEEE Signal Proc. Magazine, 20*(2), 2003, 74-80.

[7] S. Gazor, B. Farhang-Boroujeny, A state space approach for efficient implementation of block lms adaptive filters. *Proc. ICCS/ISITA. conf. Commun. Syst.*, Singapore, 1992, 808-812.

[8] H.V. Sorensen, M. Heideman, C. Burrus, On computing the split-radix FFT, *IEEE Trans. Acoust., Speech, Signal Processing, 34*(1), 1986, 152-156.

[9] M.A. Richards, On hardware implementation of the split radix FFT, *IEEE Trans. Acoust., Speech, Signal Processing,* 36(10), 2002, 1575-1581.

[10] G. Goertzel, An algorithm for the evaluation of finite trigonometric series. *American Math. Month., 65*, 1958, 34-35.