High Efficient Real-time Calculation Method of here Large FFT with High Overlapping Rate

Xing Gao-xiang¹, Yin Tian-qi²
1 College of Electronic Engineering, Naval University of Engineering, Wuhan, China
2 Institute of Noise and Vibration, Naval University of Engineering, Wuhan, China
nuexgx@163.com¹, whsss06@163.com²

Abstract. In order to meet the needs of large number of overlapping samples in sequential fine feature extraction, this work proposes a frame by frame (segment) iterative algorithm based on the "sparse" feature of incremental data to achieve efficient real-time calculation of large FFT, which breaks through the hardware bottleneck of real-time computing platform such as DSP platform in high-resolution spectral analysis. The proposed algorithm makes use of the sparseness of incremental data to make the computation burden of segment iteration lighter than that of standard FFT, and has the potential of parallel computation. The simulation results show that the new algorithm is more efficient than the standard FFT when the number of segments is large.

1. Introduction

With the increasing refinement of target feature information required by modern information detection system, Radar and Sonar signal processing put forward new requirements for the length of FFT. In other words, it is an urgent need to implement real-time processing [1-6] of large FFT in frame beat. The large FFT can be realized by CPUs, and each hardware manufacturer provides their own mathematical library based on FFTW [7] (FFT in the West). However, the calculation efficiency cannot meet the real-time requirements of Radar and Sonar systems. In the situation of high real-time requirement, the real-time system based on GPU or DSP is usually used. In DSP real-time systems, there are two bottlenecks when FFT with large number of points is completed: (1) the calculation time of single channel FFT with large number of points may be close to or exceed the frame beat, which cannot be realized in real time with a single calculation unit; (2) the FFT algorithm embedded in the current mainstream DSP chip has constraints on the length of FFT, such as the maximum length constraint of single channel FFT of TMS320C6678 multi-core chip is 131072 [8], which leads to difficulty to realize FFT processing of larger number of points.

In order to break through the above bottlenecks, scholars in related fields have carried out a lot of fruitful work. The main idea to solve the problem is to divide the large number of FFT points into several small number of FFT points and implement them in parallel in the way of "resource matching". In reference [9], the long point sequence is divided into several subsequences, each subsequence is FFT with small points, then weighted with rotation factor, and finally the frequency-domain sequence is obtained by summing corresponding elements. The process of rotation factor weighting will increase the storage and calculation amount, which is not conducive to real-time calculation. In reference [10,11], the long point sequence is divided into several subsequences and combined into a two-dimensional matrix. First, the column dimension FFT is done, and then the row dimension FFT is
done after weighting. The calculation amount of this method is smaller than that of reference [9], but it is slightly larger than that of standard FFT, so it has the possibility of real-time calculation. However, when there are a lot of historical data in the long series, the amount of calculation is wasted. According to the definition of standard FFT, the long sequence is divided into several segments, and the principle of realizing large points FFT by segment recurrence is given in reference [12]. However, the FFT of incremental data is still long points, which does not bring much convenience to the real-time calculation.

In this work, the incremental data is modeled as a "sparse" time series with only a few non-zero points in the first part. The "sparse" sequence is calculated by using the principle of segmented FFT. Under a reasonable number of segments, the FFT calculation of incremental data is less than that of standard FFT. This feature makes it convenient for DSP real-time system to realize FFT with large number of points in parallel.

2. Calculation Principle of Sparse Large FFT

2.1. Conventional FFT

Suppose \( N (N \gg 1, N = 2^Q, Q \) is a positive integer) point time series \( x(n), 0 \leq n \leq N - 1 \), whose \( N \) point FFT transformation can be expressed as

\[
X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{2\pi jnk}{N}} = \sum_{n=0}^{N-1} x(n) W_N^{nk}
\]

where \( 0 \leq n \leq N - 1, 0 \leq k \leq N - 1 \), and \( W_N^{nk} = \exp \left\{ -j \frac{2\pi nk}{N} \right\} \) is twiddle factors. Equation (1) can also be expressed as

\[
X(k) = \text{FFT}_N[x(n)]
\]

where \( \text{FFT}_N[\cdot] \) represents FFT processing with \( N \) length.

If the \( N \) value is large, the direct calculation may take a long time or be limited by hardware conditions. For example, the maximum length of FFT algorithm embedded in Dsplib Library of TMS320C6678 multi-core DSP is 131072. At this time, it is necessary to segment the input series and perform parallel calculation for FFT with large number of points to reduce the calculation time and overcome the hardware constraints.

2.2. Segment Recursive Method For Large FFT

Set the FFT processing to take the frame as the beat, and process the accumulated \( N \) points of data in each beat as do the \( N \) point FFT, where the \( N \) is

\[
N = LM
\]

where, \( M \) is the number of points per frame, and \( L \) is the number of frames that each FFT needs to accumulate by frame, and \( L \geq 1 \). When \( L \gg 1 \) and only one frame of data is updated in each FFT processing, there is a large amount of historical data in the processing formula (1), and the amount of repeated calculation is also huge. In [6], the following recursive implementation principles of large FFT are proposed:

When the \( r \)th frame is set, the starting point of the time series of \( N \) point is \( i = rM \), where \( r = 0,1, \cdots \), which is recorded as \( x_i(n), 0 \leq n \leq N - 1 \). When the \( (r + 1) \)th frame is set (sliding, updating \( M \) points), the starting point of the time series of \( N \) points is \( i + M = (r + 1)M \), which is recorded as \( x_{i+M}(n) \). Based on these assumptions, the \( N \) points FFT of frame \( r \) and \( r + 1 \) can be expressed as

\[
X_r(k) = \sum_{n=0}^{N-1} x_i(n) W_N^{nk}
\]

\[
X_{r+M}(k) = \sum_{n=0}^{N-1} x_{i+M}(n) W_N^{nk} = \sum_{n=M}^{N-1} x_i(n) W_N^{(i-M)k} = \left[ X_r(k) + \sum_{w=0}^{M-1} (x_i(n+M) - x_i(n)) W_N^{wk} \right] W_N^{-MK}
\]

(5)
Equation (5) shows that for the FFT with $N \gg 1$, the result of the FFT of the previous frame can be used, that is, the result plus the $N$-points FFT of the difference between the new data and the old data (all $M$-points), and then multiply the coefficient $W_N^{-nk}$.

The difference between the new data and the old data with $M$ points is defined as the incremental data, which is signed as $z(n)$

$$z(n) = x_i(n + N) - x_i(n)$$  \hspace{1cm} (6)

Padding zeros to $z(n)$ to get an $N$ point series $z'(n)$, and the $N$-point FFT of it is

$$Z(k) = \sum_{n=0}^{N-1} z'(n) W_N^{-nk}$$  \hspace{1cm} (7)

It is worth noting that in (7), only the first $M$ points data in the series $z'(n)$ are non-zero. In view of "sparsity", reasonable selection of algorithm can effectively calculate the FFT of the "sparse" $z(n)$.

Further analysis of (5) shows that it is not difficult to find that the calculation of FFT with large number of points can be realized by recurrence of incremental data. Although the FFT of incremental data is still a large number of points, there are a lot of continuous zero values in the incremental data, which will bring convenience to the calculation.

### 2.3. Calculation Principle of Sparse Large FFT

The first $M$ point data of $z(n)$ in (7) is not zero, and the later $N-M$ points are all zero. When $N$ is an integral multiple of $M$, such as $L = N/M$ ($L$ is the number of segments or accumulated frames), the following method can be considered to calculate the FFT of $N$ point sequence $z'(n)$.

The $N$ point $z'(n)$ is divided into $L$ segments, each segment is $M$ in length, with $N = LM$. At this time, the $L$ segment data can be expressed as follows:

$$Frm0: z(m), \quad m = 0, 1, \cdots, M - 1$$

$$\vdots$$

$$FrmL-1: z(m+(L-1)M), m = 0, 1, \cdots, M - 1$$

Combining the equation (1) and equation (8) of FFT, the equation (7) has the following expression

$$Z(k) = \sum_{m=0}^{M-1} z(m) W_N^{-mk} + \sum_{m=0}^{M-1} z(m+M) W_N^{(m+M)k} + \cdots + \sum_{m=0}^{M-1} z(m+(L-1)M) W_N^{(m+(L-1)M)k} = \sum_{l=0}^{L-1} \sum_{m=0}^{M-1} z(m+LM) W_N^{(m+LM)k}$$  \hspace{1cm} (9)

where $0 \leq k \leq N - 1$.

The purpose of the subsection of (8) is to decompose a large number of points into a number of combinations of small number of points. The ultimate goal of (9) is to execute $M$ point FFT in each segment, and then combine them to get $N$ point FFT. The frequency $k$ is defined as follows:

Set $k = hL + i, 0 \leq h \leq M - 1, 0 \leq i \leq L - 1$, the equation (9) becomes

$$Z(hL+i) = \sum_{l=0}^{L-1} \sum_{m=0}^{M-1} z(m+LM) W_N^{(m+LM)(hL+i)} = \sum_{l=0}^{L-1} \sum_{m=0}^{M-1} z(m+LM) W_L^{hi} W_M^{mi} W_N^{-nk}$$  \hspace{1cm} (10)

The (10) can be regarded as FFT processing for a two-dimensional matrix as shown in figure 1. It can be operated by row first and then column, or by column first, with different processing complexity. They are discussed respectively below.

### 2.3.1. Row FFT followed by Column Sum

The so-called "Row FFT followed by Column Sum" really works on the $z(m + LM)$ in (9), whose row represents a different data segment (frame). The FFT is performed according to each segment, and then column processing is performed followed in the “Row FFT followed by Column Sum” as follows:
\[
Z(hL+i) = \sum_{l=0}^{L-1} \sum_{m=0}^{M-1} z(m + lM) W_N^{(m+LM)l} W_M^{mh} = \sum_{l=0}^{L-1} \sum_{m=0}^{M-1} g(m + lM) W_M^{mh} = \sum_{l=0}^{L-1} G(h,l,i) = Z(h,i) \tag{11}
\]

where \( g(m + lM) = z(m + lM) W_N^{(m+LM)i} \), \( G(h,l,i) = \sum_{m=0}^{M-1} g(m + lM) W_M^{mh} \), \( 0 \leq i \leq L - 1 \). The (11) shows that the large FFT can be done in each segment after weighted segment by segment, and then the corresponding elements of each segment are added. The specific steps are as follows:

1) Divided the \( N \) points data into \( L \) segments, signed by \( z(l, m) \), \( 0 \leq l \leq L - 1, 0 \leq m \leq M - 1 \). The segmented data \( z(l, m) \) is a two-dimensional matrix, as shown in figure 1.

2) Use the twiddle factors \( W_N^{(m+LM)i} \) \( (0 \leq i \leq L - 1) \) to weight the matrix \( z(m + lM) \). Note that the variable \( i \) is introduced here to change the original two-dimensional matrix into a three-dimensional matrix \( g(m + lM) \), as shown in figure 2.

3) After the \( M \) points FFT for matrix \( z(l, m, i) \) by lines, frequency domain three-dimensional matrix \( G(h, i, l) \) is obtained, as shown in figure 3.

4) Summing the segment dimension of \( G(h, i, l) \), the \( Z(h,i) \) is obtained, as shown in figure 4.

5) The formula \( k = hL + i \) can index all frequency points, as shown in the curve in figure 4.

### 2.3.2. Column FFT followed by Row Processing

Make the following deformation for equation (10):

\[
Z(hL+i) = \sum_{l=0}^{L-1} \left( \sum_{m=0}^{M-1} z(m + lM) W_N^{ml} \right) W_M^{mh} = \sum_{l=0}^{L-1} G'(m,i) W_M^{mh} = Z(h,i) \tag{12}
\]

The detail implementation steps of (12) are as follows:

1) Divided the \( N \) points data into \( L \) segments, signed by \( z(l, m) \), as shown in figure 1.

2) Do \( M \) times \( L \) points column direction FFT and then get \( Z'(m, i) \), as shown in figure 5.

3) The same dimension matrix \( G'(m, i) \) is obtained by weighting \( Z'(m, i) \), that is, Hadamard operation with the twiddle factors \( W_N^{ml} \).

4) The matrix \( Z(h, i) \) is obtained by performing the \( M \) points row direction FFT transform on the matrix \( G'(m, i) \).

5) The formula \( k = hL + i \) can index all frequency points, as shown in the curve in figure 4.

### 2.3.3. Discussion on the complexity of two implementation methods

It can be found in comparative analysis between (11) and (12) that:
1) The weighting by twiddle factors in (11) will change the original two-dimensional matrix into three-dimensional matrix and increase the data storage and calculation amount.

2) The (12) has no three-dimensional change, so it has more advantages from the storage point of view. However, its column dimension FFT constrains the power of the number of segments to 2, and the number of segments of the FFT is a bit large.

Calculation burden $B$ analysis: the calculation burden $B_{rc}$ of (11) (subscript indicates first row and then column)

$$B_{rc} = LNB_M + LN \log_2 M + LN \left( \frac{M-1}{M} + \frac{L-1}{L} \right) B_A$$

(13)

where the $B_M$ denotes the calculation burden of complex multiplication and the $B_A$ denotes the calculation burden of complex addition.

The calculation burden $B_{cr}$ of (12) (subscript indicates first column and then row)

$$B_{cr} = NB_M + N \log_2 M + N \log_2 L$$

(14)

The above formula shows that the calculation burden of (12) is slightly larger than that of standard FFT in (1). At this cost, the large FFT is implemented. Obviously, the calculation burden of (12) is far less than that of (11), that is, $B_{cr} \approx L B_{rc}$, where $L$ is the number of segments.

2.3.4. Discussion on the complexity of two implementation methods

Although the calculation burden of (12) is slightly larger than that of (1), it is not conducive to separate processing by segment (frame). However, there is a special case when the input data meets the "sparse" condition.

It can be seen from (8) that in the sparse state described in (7) the $z(m + lM) = z(m)$ is not zero when $l = 0$, and other values are all zero. In this case, (11) and (12) are respectively deformed into

$$Z_{eq11}(hL + i) = \sum_{m=0}^{M-1} z(m)W_N^{mi}W_M^{mh}, \quad l = 0$$

(15)

$$Z_{eq12}(hL + i) = \sum_{m=0}^{M-1} z(m)W_N^{mi}W_M^{mh}, \quad l = 0$$

(16)

Equation (15) and (16) are the same, that is to say, for the case that only the first segment of data is non-zero, the large FFT can be implemented as follows:

1) Firstly weighting the data $z(m)$ when $l = 0$ by twiddle factors, we can get $z'(m, i) = z(m)W_N^{mi}$, which is shown in figure 5.

2) Do the $M$ points FFT by rows of $z'(m, i)$ and get a new matrix $Z(hL + i)$.

Here, the computation burden can be estimated by

$$B_{sp} = NB_M + N \log_2 M$$

(17)

Compared with (14), the computation burden of the sparse case noted by (17) is $N \log_2 L$ less.

The large FFT with high overlapping rate data can be calculated efficiently by combining (15), (16) and (5).

2.3.5. Application of the sparse FFT

1) The first application is the case where hardware capabilities are restricted. For example, the length of FFT algorithm embedded in dsplib library is limited to 131072 for the TMS320C6678 multi-core chip. When a longer number of FFT points is needed to improve the resolution of frequency energy, the proposed algorithm is not limited by the length of FFT of the hardware, and can be calculated separately by frame, which is convenient for parallel implementation.
2) The second application is the case where large FFT needed to be completed within short heartbeat, such as sonar high-resolution spectrum analysis, radar microsecond level high-resolution frequency domain processing and so on.

3. Performance evaluation

3.1. Calculation Accuracy

Suppose we want to achieve a frequency resolution of 0.06 Hz and the sampling rate is set at 8000Hz, then we need a 131072 points FFT to achieve this resolution.

For comparison convenience, the original FFT is marked as "FFT_ori", the method of "row FFT followed by column sum" is noted as "FFT_r_c", and the two-dimensional FFT is noted as "FFT_c_r".

Simulation parameters: A continuous wave (CW) signal with central frequency 2200Hz is Sampled with frequency 8000 Hz. FFT length is set to 131072. The data for FFT is divided into 32 segments with 4096 points each. The FFT amplitude spectrum of the three algorithms involved in the comparison is shown in figure 6. Figure 6 shows that the spectra of the three algorithms are consistent.

Take "fft_ori" sequence as the true value, and calculate the mean square errors of "FFT_r_c" and "FFT_c_r" respectively, as shown in Table 1.

| Order | Algorithm type | mean square errors |
|-------|----------------|--------------------|
| 1     | FFT_r_c        | 4.0706e-20         |
| 2     | FFT_c_r        | 5.7091e-23         |

3.2. Calculation Burden analysis

The calculation burden of the 131072 points “FFT_ori” is 2228224.

The calculation burden of the “FFT_c_r” is 2752512, which is slight larger than that of “FFT_ori”.

To analyze the change of calculation burden vs segments number, set the number of segments to the power of 2. In this case, 9 values are set, which are 1, 2, 4, 8, 16, 32, 64, 128 and 256. The curve of calculation burden with the number of segments for "FFT_r_c" and "FFT_c_r" is shown in figure 7.

The (17) shows that the calculation burden of "sparse" condition is less than that of "FFT_c_r", and it is a function of the segment number \(L\). Figure 8 shows the curve of the calculation burden of "sparse" condition vs the number of segments, which is signed as "FFT_spr", and compared with that of "FFT_ori", "FFT_c_r".

Figure 8 shows that the computation burden of "FFT_spr" is slightly smaller than that of "FFT_c_r", and the difference gradually increases with the number of segments \(L\), and
the difference is \( N \log_2 L \). It is worth noting that there is an intersection between the computation burden of "FFT_spr" and that of standard FFT. In this special case, the intersection number is \( L = 16 \). This means that when the number of segments is less than 16, the computation burden of "FFT_spr" is greater than that of the standard FFT, and vice versa.

Further compare the computation burden of (17) and standard FFT. It is not difficult to find that when \( \log_2 L = 4 \) the following formula holds:

\[
NB_M + N \log_2 M = N \log_2 N
\]  

Figure 8 further illustrates that when the number of segments is large, the computation burden of "sparse" FFT will be less than that of standard FFT, that is, the large FFT of (16), (17) and (6) have advantages in computation burden, and parallel computing can be used to further shorten the calculation time.

4. Conclusions

In this paper, an efficient calculation of large FFT in real-time system is studied. On the basis of segment (by frame) iterative calculation method, an efficient calculation method of sparse large number FFT is proposed, which is suitable for high overlapping rate data. Theoretical derivation and numerical simulation show that when the number of segments is greater than 16, the computation burden of sparse large number FFT is less than that of standard FFT. The proposed sparse large FFT can solve the problem of repeated calculation of a large number of data in real-time system, and has a wide application prospect in high-resolution spectrum analysis of radar and sonar.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant No.51679247).

References

[1] Qian Bing-feng, SUN Yi-ze, “Research and implementation of VLFFT algorithm using SMP system”, Journal of University of Electronic Science and Technology of China, vol.48, n1, pp.26-31, Jan.2019. (In Chinese)

[2] WEI Xing, HUANG Zhi-hong, YANG Hai-gang,”High throughput dual-mode reconfigurable floating-point FFT processor”, Journal of Electronics & Information Technology, vol.40, n12, pp.3042-3050, Dec.2018. (In Chinese)

[3] LIU Li, GAO Mei-guo, ZHOU Run, “Algorithm and implementation of a large-point FFT under the master-slave parallel multi-processor architecture”, Systems Engineering and Electronics, vol.25, n10, pp.1193-1196, Oct.2003. (In Chinese)

[4] YUAN Qi, YANG Kang, ZHOU Jian-jiang, “Implementation of large points FFT based on C6678 multi-core DSP”, Electronic Measurement Technology, vol.38, n2, pp.74-80, Feb.2015. (In Chinese)

[5] GAO Li-ning, MA Xiao, LIU Teng-fei, “Research and implementation of ultra large sequences FFT optimized algorithm”, Journal of Electronics & Information Technology, vol.36, n4, pp.998-1002, April 2014. (In Chinese)
[6] Wen-Fang Yen, ShingChern D You, Yung-Chao Chang, “Real-time FFT with pre-calculation”, Computers and Electrical Engineering, vol.35, pp. 435-440, 2009.

[7] Frigo M, Johnson S G. “FFTW3.2”, 2012-12-01, www.fftw.org.

[8] “DSPLIB user’s manual(c66x) release 3.4.0.0”, 2014, www.ti.org (http://processors.wiki.ti.com/index.php/Software_libraries).

[9] Annamaria R Varkonyi-Koczy, “A recursive fast Fourier transformation algorithm”, IEEE Trans. on Circuits and System-II: Analog and Digital Signal Processing, vol.49, n9, pp.614-616, Sep.1995.

[10] Xiaohui Li, Ellen Blinka, “Very large FFT for TMS320C6678 processors (SPRY277)”, Texas Instruments, 2015.

[11] D. Takahashi, “High-performance Parallel FFT algorithms for the Hitachi SR8000”, in Proceedings of the Fourth International Conference/ Exhibition on High-Performance Computing in the Asia-Pacific Region. 2000.

[12] A Van der Byl, M R. Inggs, “Recursive sliding discrete Fourier transform with oversampled data”, Digital Signal Processing, vol.25, pp. 275-279, 2014.