


A Novel Accelerating Algorithm and Its Implement in Real Sequence FFT

Hong-gui Deng^{1,2}, Sheng-wei Guo¹, Jian Duan³ 

¹Xinjiang Nonferrous Central South University Joint Research Institute, Urumqi, 830000, China

²Department of Electronics and Information, Central South University, Changsha, 410083, China

³State Key Laboratory of High-performance Complex Manufacturing, Central South University, Changsha 410083, China

Abstract: We propose an improved FFT algorithm, which costs only a half of the calculation time compared with the conventional FFT if the input data are real numbers. The algorithm is optimized by dividing $2N$ data points into 2 separated groups through parity. The odd part and even part of the $2N$ data points are used as the real part and imaginary part of a new complex data sequence with N data points. After FFT of this new data sequence, the FFT of the original $2N$ data points can be calculated through formulations. From our experiment based on FPGA, this new implementation is more effective than conventional FFT by saving half of calculation time.

Keywords: FFT, real sequence, FPGA

0 Introduction

Discrete Fourier Transform (DFT), which is employed to transform the signal from the time domain to the frequency domain, is a key tool in digit signal processing, especially in the areas of acoustics, electronics and imaging processing, which are driven by the trend of the increased using of filtration, frequency spectral thinning, and spectrum estimation.^[1-3]

One of the most efficient methods of performing this transformation is the Fast Fourier transform (FFT), which was proposed by Cooley and Tukey in 1965. The Simplest and most common form of FFT is the radix-2 butterfly algorithm. Each butterfly consists of multipliers and adders that calculate two input points and give two output points based on the coefficients chosen from a sine table. The amounts of multiplication computation in N -points DFT are reduced from N^2 to $(N/2) \cdot \log_2 N$, which greatly reduce the computing time of the DFT.^[4,5]

Unfortunately, conventional FFT only considers

that the input data is a complex data sequences. However, real data sequences are very common in engineering fields.^[6,7] The typical solution for this problem is to fill the imaginary part with zeros, which make computing efficiency relative low, A new method is introduced in this paper based on the symmetry and periodicity of Fourier Transform (FT) to improve the computing efficiency. In the algorithm, a $2N$ -point sequence is divided into two separated groups by the parity and for a new complex N -point sequence. We can obtain the FT of the original $2N$ -point data through calculating this N -point complex sequence to save the computing time.

We can implement high-speed and real-time digital signal processing on FPGA. Based on the improved algorithm, our experiment shows that it can save both memory and computing time.

1 The principle of present real sequence algorithm

Let $\{X(0), X(1), \dots, X(N-1)\}$ be Fourier



Jian Duan (Correspondence)

duanjian@csu.edu.cn

Transform (FT) of the $\{x(0), x(1), \dots, x(N-1)\}$

where $W_N = e^{-j(2\pi/N)}$. Then

$$x(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk}, k = 0, 1, \dots, N-1 \quad (1)$$

Let $X_r(k)$ and $X_i(k)$ be the real and imaginary parts of $X(k) = X_r(k) + jX_i(k)$.

FT can transform the data sequence $x(0), x(1), \dots, x(N-1)$ (time domain sample) to the data sequence $X(0), X(1), \dots, X(N-1)$, which are defined in frequency domain. According to the definition, $x(n)$ is a complex sequence. In a practical practice, if input is a real sequence, the traditional way is to fill the imaginary parts with zeros.

We describe a new algorithm, which can save a half of computing time if the input data sequence is a real data sequence. Provided that $x(n)$ is a 2N-points real sequence, it can be divided into 2 separated groups by the parity. The data with odd serial numbers can form a sequence $g(n)$, and the data with even serial numbers can be used to build another new sequence $h(n)$. Both $g(n)$ and $h(n)$ are N-points real sequences. $G(k)$ and $H(k)$ are Fourier transform sequences of $g(n)$ and $h(n)$ by FFT, respectively. According to the FFT Radix-2 algorithm of Cooley-Tukey, we have:

$$X(k) = G(k) + W_N^k H(k) \quad k = 0, 1, \dots, N-1 \quad (2)$$

$$X(k+N) = G(k) - W_N^k H(k) \quad k = 0, 1, \dots, N-1 \quad (3)$$

In the process of calculating $G(k)$ and $H(k)$ by DFT, these formulas can be employed to divide $x(n)$ into 2 groups, continue to implement in the same way until they are 2-points sequences. In the process of computation, the 2-points are calculated by DFT first, stage by stage until getting the results of original sequence by DFT. The amounts of multiplication computation of this method are $N \cdot \log_2 2N$ times, (2N-points sequence)

2 The principle of the accelerating algorithm for real sequence

In equations (2) and (3), for a division, it is required to calculate the transformed data sequences $G(k)$ and $H(k)$. By observing the symmetry and periodicity of Fourier Transform, a new method is adopted to transform 2N-points real sequences into an N-points complex sequence. The amount of computation can be reduced by half. The principle is given as follows:

Assuming an N-point finite complex sequence $y(n) = g(n) + ih(n)$ and $Y(k)$ is transformed from $y(n)$ by FT defined as:

$$Y(k) = \sum_{n=0}^{N-1} (g(n) + ih(n))e^{-j2\pi(nk/N)} \quad k = 0, 1, \dots, N-1 \quad (4)$$

Substituting k by $N-k$, we have:

$$Y(N-k) = \sum_{n=0}^{N-1} (g(n) + ih(n))e^{j2\pi(nk/N)} \quad k = 0, 1, \dots, N-1 \quad (5)$$

Then, expanding $e^{j\alpha} = \cos(\alpha) + j\sin(\alpha)$ we have

$$\begin{aligned} Y(N-k) + Y(k) &= 2 \sum_{n=0}^{N-1} (g(n) + ih(n)) \cos(2\pi nk / N) \\ Y(N-k) - Y(k) &= 2j \sum_{n=0}^{N-1} (g(n) + ih(n)) \sin(2\pi nk / N) \end{aligned} \quad k = 0, 1, \dots, N-1 \quad (6)$$

Since $Y_r(k)$ and $Y_i(k)$ are the real and imaginary parts of $Y(k) = Y_r(k) + jY_i(k)$, it follows that

$$\begin{aligned} Y_r(k) + Y_r(N-k) &= 2 \sum_{n=0}^{N-1} g(n) \cos(2\pi nk / N) = 2G_r(k) \\ Y_i(k) + Y_i(N-k) &= 2 \sum_{n=0}^{N-1} h(n) \cos(2\pi nk / N) = 2H_r(k) \quad k=0, 1, \dots, N-1 \quad (7) \\ Y_r(k) - Y_r(N-k) &= 2 \sum_{n=0}^{N-1} h(n) \sin(2\pi nk / N) = -2H_i(k) \\ Y_i(k) - Y_i(N-k) &= 2 \sum_{n=0}^{N-1} g(n) \sin(2\pi nk / N) = 2G_i(k) \quad k=0, 1, \dots, N-1 \quad (8) \end{aligned}$$

Since $G(k) = G_r(k) + jG_i(k)$ and $H(k) = H_r(k) + jH_i(k)$, we have

$$\begin{aligned} G(k) &= \frac{1}{2} [(Y_r(N-k) + Y_r(k)) + j(Y_i(N-k) - Y_i(k))] \\ H(k) &= \frac{1}{2} [(Y_i(N-k) + Y_i(k)) - j(Y_r(N-k) - Y_r(k))] \quad k=0, 1, \dots, N-1 \quad (9) \end{aligned}$$

If $Y(k)$, which is transformed from $y(n)$ by FFT, can be obtained, $G(k)$ and $H(k)$ can be obtained by separation. According to equation (2) and (3), the transformed sequence $X(N)$ can also be obtained. In the Cooley-Tukey algorithm, the complexity of transforming an N -point complex sequence by FFT is similar to the transformation of N -point real sequence by DFT, which is lower than to the transformation of $2N$ -point real sequence by DFT. Therefore the computational efficiency will be higher.

3 The Realization of real sequence algorithm

Generally, there are two platforms are typically employed to perform FFT: DSP and FPGA. Both of them have advantages and disadvantages. Specifically, the operational speed of DSP is slower than that of FPGA. The interface of DSP is not flexible, and the enormous storage demanded during the operation is not efficient. In addition, DSP requires a special external interface to control the chip and RAM that increase the complexity. On the other hand, the technology is more mature, and the exploitation cost is lower than that of FPGA. As a result, most of hardware realizations for FFT are to take advantage of DSP. However, with the development of micro electronics and FPGA, both the operational speed and the capacity of FPGA have been significantly improved, and multiplier units are embedded into some FPGA devices. As FPGA surpasses DSP in terms of volume, operational speed and flexibility, it has become the ideal choice of the hardware realization for FFT algorithms. The FPGA device EP1S20F484C5 of Stratix series of Altera Corporation is used as an example in this paper.

3.1 The frame of real sequence processor

FPGA-based FFT processor includes input memory unit 1, output memory unit 2,

computation module, control module, and ROM factor table, as showed in figure 1. In this design, input is a real sequence. Thus, $2N$ -point real sequence should be transformed into an N -points complex sequence for FFT and finally form $2N$ -point real sequence is restored by separating odd and even sequence.

3.2 Parallel processor and same address operation

By employing Cooley-Tukey's DFT Radix-2 algorithm, $\log_2 n$ steps of the butterfly computation is required to transform an N -point

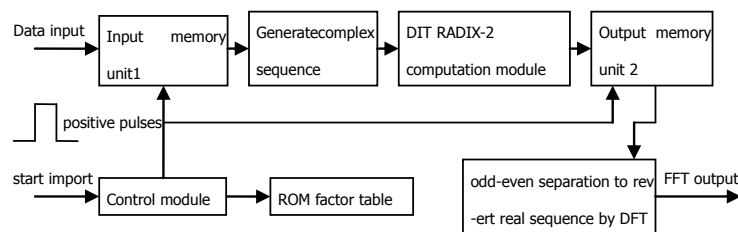


Figure 1: FFT real sequence row processor hardware frame

sequence by FFT, and each step is composed of $n/2$ butterfly units.

The butterfly units are executed step by step. At each step, $n/2$ units are independent and they can be executed in parallel. In this way, the calculation only requires $n/2$ butterfly units, which saves internal memory of FPGA and accelerates the operating speed because of the parallel computing. Meanwhile, the execution of each butterfly unit is not relevant to that of other butterfly units. Each butterfly can be computed at the same address. And output data can still be saved at the same units with the input data. This will save storage space and let searching addresses easily.

3.3 Creation of W factor

In FFT, rotator factor is referred as,

$$W_N^{nk} = \cos\left(\frac{2\pi nk}{N}\right) - j \sin\left(\frac{2\pi nk}{N}\right)$$

There are two methods to generate rotator factor by:

a) Directly calculating. The value of rotator factor is calculated each time when is required. This method can save ROM memory.

b) Creating a table for W factor. Because W_N^{nk} only has n different values, the values of W_N^{nk} can be calculated in advance and saved in the ROM of FPGA. The values can be searched when they are requested. This method can accelerate operating speed. In some real-time signal processing system, operating speed of FFT is very important. Therefore, this method may have a wide application.

3.4 Complex multiplication units and the improvement

The Input data $x = x_r + jx_i$ multiplies rotator

factor $W = \cos\theta + j\sin\theta$, where $\theta = \frac{2\pi nk}{N}$

$$y_r = x_r \cos\theta - x_i \sin\theta$$

(10)

$$y_i = x_i \cos\theta - x_r \sin\theta$$

(11)

Generally, in a calculation period, 4 real number multiplier units and 2 real numbers adder units are required. Equations (10) and (11) can be modified in this paper to save the amount of computation as follows:

$$y_r = (x_r + x_i) \cos\theta - x_i(\sin\theta + \cos\theta)$$

(12)

$$y_i = (x_r + x_i) \cos\theta + x_r(\sin\theta - \cos\theta)$$

(13)

Three values: $\cos\theta$, $\sin\theta + \cos\theta$, $\sin\theta - \cos\theta$ can be calculated in advance and be saved in ROM. According to (12) and (13), a normal complex multiplication can be realized only by three real number multiplications and three real number additions. Because area taken by a real number

adder unit is much smaller than that of real number multiplication, operating speed of the adder unit is much faster than that of multiplier unit. It will significantly improve the operating speed and save the memory space.

4 Quality Comparison

Table 1 FFT hardware realization quality comparison

Processor	Type	Data format	Clock frequency	Processing Time
BD SP9124	DSP	16 bit block point floating	60MHz	54μs
PDSP16510	DSP	24 bit block point floating	40MHz	98μs
Reference 8	ASE	16 bit point fixed	80MHz	40μs
TSMCQ 18um	ASE	32 bit point floating	200MHz	27.6μs
This design Stratx	FPGA	32 bit point floating	200MHz	11.52μs

Table 1 is the comparison of some hardware realizations of FFT, parallel stream line butterfly unit of FPGA device EP1S20F484C5 of Stratix serious can work steadily on the clock of 200MHz. $\log_2 N$ steps in N-points FFT are required. Each stage has N/2 butterfly units. In this design, the improved odd-even separated algorithm is employed. A real sequence with 1024-points is used as an example, it can transform a 1024 real sequence into two 512-points complex sequences, and there are total 9 steps, and each step requires 256 butterfly calculations. About $T = 256 \times 9 \times 5ns = 11.520\mu s$ is required to finish an FFT of the 1024-point real sequence.

5 Conclusion

Based on two major features of FFT, symmetry and periodicity, we proposed an improved accelerating FFT algorithm for real sequence inputs. The algorithm separates a 2-N point into two sequences by parity, then generates a new complex sequence which takes the odd and even serial numbers of the former sequence as the real and imaginary parts. Subsequently, the implement of the algorithm based on FPGA is described and the experiment results describe a memory saving and a gain in the performance of operational speed compared with present methods. Hence, the new algorithm will breed significant influence in engineering practice.

Acknowledgements

This work was financially supported by Natural Science Foundation of Xinjiang Uygur Autonomous Region, China (Grant No. 2013211A035).

Reference

- [1] S Sukhsawas, K Benkrid. A High-level Implementation of a High Performance Pipeline FFT on Virtex-E FPGAs *IEEE Computer Society Annual Symposium on VLSI*, p 229-230, 2004
- [2] Kumhom P, Johnson J.R, Nagvajara P. Design, Optimization, and Implementation of a Universal FFT Processor. Proceedings of 13th Annual IEEE International ASIC/SOC Conference (Cat. No.00TH8541), p 182-187, 2000
- [3] Chen he, Zhao zhong-wu. ASIC design of floating-point FFT pro-cessor[J] *Journal of University of Beijing for Science & Technology:English Version* 2004,13(4): 389-393.
- [4] LU Jianfeng. Real time Fourier Transform syetem design based on double DSP[J] *Photoelectric engineering*, 2005,32(11): 93-96.
- [5] Radhouane, R, Liu, P.; Modlin, C. Minimizing the memory requirement for continuous flow FFT implementation: continuous flow mixed mode FFT (CFMM-FFT) 2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353), p 116-134 vol.1, 2000
- [6] GAN Liang-cai, DING Ya-hu. CPLD implementation of 1024-point FFT of short-wave wideband channel simulator[J].*Electricity Journal:English version*, 2006,15(4): 692-696.
- [7] Cepisca, C., Grigorescu, S.D., Covrig, M.,Andrei, H.. About the Efficiency of Real Time Sequences FFT Computing. Proceedings of the 2007 IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, DDECS, p 211-214, 2007
- [8] M. Rawshi, M.Wojtyski. Distributed arithmetic based implementation of Fourier transform targeted at FPGA architectures[C]. Proceedings of the 14th International Conference "Mixed Design of Integrated Circuits and Systems", MIXDES 2007, p 152-156, 2007
- [9] LIU Guodong,CHEN Boxiao,CHEN Duofang. FPGA design of FFT processor[J]. *Aviation calculation technology*, 2004,34(3): 101-104.
- [10] KUO JC,WEN CH,WUAY. Implementation of a programmable 64-2048-pointFFT/IFFT processor forOFDM based communication sys-tems[C]. Proceeding of IEEE International Symposium on Circuits and Systems. 2003: 121-124.
- [11] Wang, Yuke;Tang, Yiyan Jiang; Yingtao; Chung, Jin-Gyun; Song, Sang-Seob; Lim, Myoung-Seob. Novel memory reference reduction methods for FFT implementations on DSP processors. *IEEE Transactions on Signal Processing*, 2007, 55(5):2338-2349.