Mini Review

Open Access

CrossMark

Fuzzy optimization FFT circuit for image processing

Abstract

In the FPGA implementation of FFT, a fuzzy approach is presented to reduce the multiplication numbers of twiddle factors and memory space, which speeds up the butterflies' computation. Also, the design of address mapping can get position of data without calculation. Simultaneously, in combination of using the pipeline structure, the speed of the FFT for FPGA implementation can be increased. Moreover, the modules have been simulated by timing and verified by data to judge the correctness of the design. This design could be applied in signal processing and image processing.

Volume 4 Issue 4 - 2018

Zehong (Jimmy) Cao
EEG & BCI & Computational Intelligence, China

Correspondence: Zehong (Jimmy) Cao, EEG & BCI & Computational Intelligence, China, Email zhcaonctu@gmail.com

Received: July 30, 2018 | **Published:** August 09, 2018

Introduction

FAST Fourier Transform (FFT) is an effective and fast discrete Fourier transform (DFT) algorithm, which is the core of digital signal processing algorithms. FFT is widely used in radar, communications, image processing, signal detection and other fields, most of those fields call for the FFT processor with high speed and high precision real-time processing performance. In order to simplify the calculation and shorten operation time to one or two magnitudes, the ideology of FFT algorithm is sequentially dividing the N point DFT into short sequences of DFT for calculating. The introduction of Verilog Hardware Description Language (HDL) provided a modeling and simulation environment for fast prototyping digital circuits and systems on FPGA. After analyzing the radix-4 FFT algorithm, this project advanced a 64-point FFT processor implementation, which has many advantages.[1,2] This paper studies the content as follows: first, describe the FFT algorithm. Then, introduce the principle of FPGA architecture, performance and characteristics, combined with the theory of FFT algorithms to determine the FPGA devices as a FFT algorithm basis. After that, using the design tools Quartus□, Modelsim and hardware description language Verilog to achieve the above algorithm. Moreover, analysis of existing design ideas implementation, and compare the various implementations. I find the design structure of the FFT processor, using pipeline structure, and generating address and twiddle factor quickly. Finally, simulate each module, and debug it.[3] For the FFT processor, the analysis method is that generate a random signal as the input signal. First, simulate the data signal using FFT function in Mat lab. Second, put the input signal to the FFT processor, and observed data calculated. Finally, it is compared with Mat lab simulation results. The FFT algorithm is applied in image processing.[4,5]

The FFT algorithm and system structure

DFT calculation

For the one-dimensional DFT, x(n) is a N numbers sequence, the DFT is defined as follows:

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j2\pi nk/N}=\sum_{n=0}^{N-1}x(n)W_N^{kn}\quad 0\le k\le N-1$$

Where this formula achieves transformation from the time domain to the frequency domain.

Cooley-Tukey algorithm

The Cooley-Tukey algorithm is used to calculate the multi-point DFT.[6] For N-Point DFT, the number of complex multiplication is equal toN^2. Obviously, if N-point DFT is decomposed into several short sequences DFT, the complex multiplication in DFT will be reduced greatly. For example, dividing the N-Point DFT into two $N^2/2$-Point DFT, the number of complex multiplication in DFT will be reduced to $N^2/2$. Assume that $N=r_1r_2$ in two-dimensional Cooley-Tukey fast algorithm,[7] calculating the Cooley-Tukey fast algorithm is divided into five steps:

First, the x(n) is rewritten as $x(n_1,n_0)$, using the formula:

$$x(n)=x(r_2n_1+n_0)=x(n_1,n_0),\begin{cases}n_1=0,1,2,...,r_1-1\\ n_0=0,1,2,...,r_2-1\end{cases}$$

Second, dividing r_1-point DFT into r_2 units, it can achieve $X_1(k_0,n_0)$.

Third, N numbers $X_1(k_0,n_0)$ are multiplied by the corresponding twiddle factor $W_N^{k_0n_0}$, which compose $X_1(k_0,n_0)$.

$$X_1(k_0,n_0)=\sum_{n_1=0}^{r_1-1}x(n_1,n_0)W_{r_1}^{n_1k_0},\qquad k_0=0,1,\cdots,r_1-1$$

Forth, dividing r_2-point DFT into r_1 units, it can achieve $X_2(k_0,k_1)$.

Finally, collate sequence and obtain $X_1(k_1,k_0)=X(k)$, where $k=r_1\times k_1+k_0$.

$$X_1(k_1,k_0)=X_2(k_0,k_1)$$

This project is used for a 64-point FFT processor.

$N=r_1\times r_2\times r$, that is $64=4\times4\times4$.

$$x(n)=x(r_1r_2n_2+r_3n_1+n_0)=x(n_2,n_1,n_0),\begin{cases}n_2=0,1,2,....,r_1-1\\ n_1=0,1,2,....,r_2-1\\ n_0=0,1,2,....,r_3-1\end{cases}$$

Then,

$$x(n)=x(16n_2+4n_1+n_0)=x(n_2,n_1,n_0),\begin{cases}n_2=0,1,2,3\\ n_1=0,1,2,3\\ n_0=0,1,2,3\end{cases}$$

FFT transformation from $x(n_2,n_1,n_0)$, $n=16n_2+4n_1+n_0$ to $X(k_0,k_1,k_2)$,

$$k=16k_2+4k_1+k_0.$$

First stage: $x(n_2,n_1,n_0)$, Radix-4 transform to $G(k_0,n_1,n_0)$, twiddle factor $W_{64}^{k_0(4n_1+n_0)}$;

Second stage: $G(k_0,n_1,n_0)$, Radix-4 transform to $H(k_0,k_1,n_0)$,

twiddle factor $W_{64}^{k_1 n_0}$;

Third stage: $H(k_0,k_1,n_0)$, Radix-4 transform to $X(k_0,k_1,k_2)$.

Design system structure

Due to complete a 64-point FFT calculation function for the project, the schemes may be selected from the following table (Table 1).[8] I use statistical results to select the most suitable circuit structure. If I select the radix-2 scheme, the BR2MDC is better relatively due to the 100 utilization rates and its hardware control is simple relatively. However, since the BR2MDC increases the number of the buffer units, the hardware area used is increased greatly. On the other hand, if I selected the radix-8 scheme, the hardware utilization rate of R8SDF is less than before, but its hardware control is very complicated.[9] Thence, the radix-4 scheme is the best choice, and the entire circuit is divided into three levels. Moreover, the hardware utilization rate of R4SDF is few, and its hardware control is not complicated. Therefore, this project uses the radix-4 single-path structure (R4SDF).

Table 1 statistical results to select the most suitable circuit structure

Circuit Structure	Complex Multiplier	Complex Adder	Delay Units	Control Complexity
R2SDF	5	12	31	Simple
R2MDC	5	12	62	Simple
BR2MDC	5	12	190	Simple
R4SDF	2	24	63	Medium
R4MDC	6	24	60	Simple
R8SDF	1	48		Complex
R8MDC	7	48		simple

Experiment results

Figure 1 shows the RTL viewer result. Then, run the Test beach file, and the functional simulation result is described in Figure 2. Finally, check the FFT processor is correct, and Figure 3 shows this FFT processor is applied for image processing.

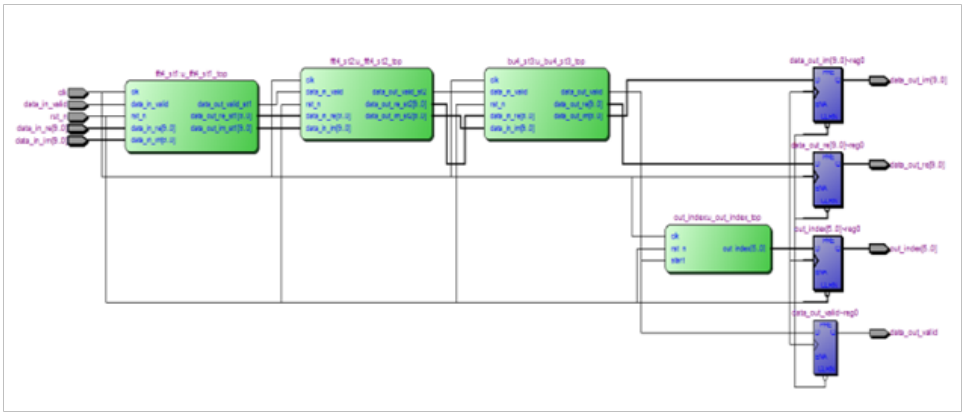

Figure 1 RTL viewer.

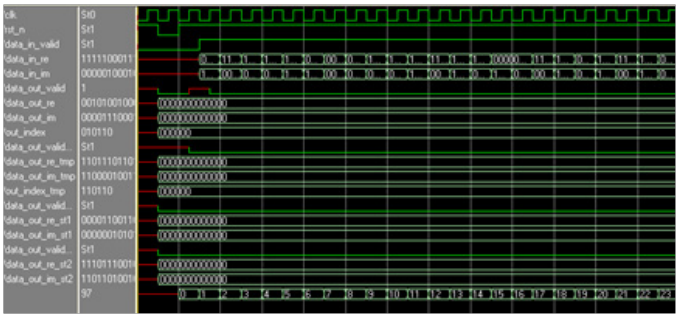

Figure 2 Functional simulations.

Conclusion

This paper analyses the feature of the radix-4 FFT algorithm, advances a pipeline hardware implementation structure, which can reduce consumption of resource ad achieve easily larger points (such as 256 points, 512 points) FFT expansion. The implemented FFT processor meets the high-speed real-time image processing requirements.

The accuracy of addition and multiplication will loss in each level of 64-point FFT. When the values are quantized to 10-bits, the input and output values are 10-bits. Calculating adder and multiplier at each level need to adjust operating bits. In the 64-point processor, there are six levels addition calculation and two levels multiplication

calculation. This loss of precision in the 10-bits operation is not negligible. In the future, a good improving method is that append the adder protecting measurement at each level. For example, increasing 1-bit accuracy, six levels adder will increased 6-bits.

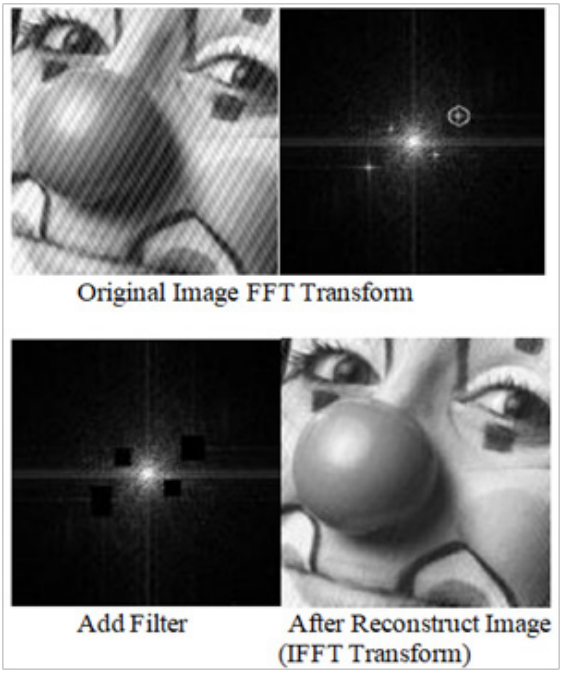

Figure 3 Image processing.

Acknowledgements

None.

Conflict of interest

The authors declare no conflict of interest.

References

1. Ayan Banerjee, AnindyaSundar. FPGA realization of a cordic based FFT processor for biomedical signal processing. *Microprocessors and Microsystems*. 2001;25(3):131−142.
2. Jesus Garcia, Juan A Michell. FPGA realization of a split radix FFT processor. *VLSI Circuits and System*. 2007;6590:65900−65908.
3. LI Xiao-feng,LIU Ming-jie. FPGA Processing Technology on Impact signal. 3rd International Conference on Advanced Computer Theory and Engineering; 2010
4. Shousheng He, Mats Torkelson. A New Approach to Pipeline FFT Processor. Parallel Processing Symposium, Proceedings of IPPS '96, The 10th International; 1996. p. 766−770.
5. Xiao-feng, Chen Long, Wang Shihu. The implementation of high-speed FFT processor based on FPGA. International Conference on Computer, Mechatronics, Control and Electronic Engineering; 2010.
6. Arman Chahardahcherik. Implementing FFT Algorithms on FPGA, *International Journal of Computer Science and Network Security*. 2011;11(11):148−156.
7. Li Wenqi, Wang Xuan, Sun Xiangran. Design of fixed-point high-performance FFT processor. 2nd International Conference on Education Technology and Computer; 2010.
8. Uwe Meyer-Baese, Liu Ling. FPG a implementation of digital signal processing. 2nd ed. Beijing: Tsinghua. University Press; 2006.
9. Jian Li, Feng Liu, Teng Long. Research on pipeline R22SDF FFT, Radar research lab, School of Information Science and Technology; China.