International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.3, May-June 2012 pp-657-660 ISSN: 2249-6645

Design of high speed FFT Processor Based on FPGA

G. Shafirulla*1, M. Subbareddy #2

Department of Electronics and Communication, Vaagdevi institute of Technology & Science, Proddutur, A.P.
Department of Electronics and Communication, Vaagdevi institute of Technology & Science, Proddutur, A.P

Abstract: It is important to develop a high-performance FFT processor to meet the requirements of real time and low cost in many different systems. So a radix-2 pipelined FFT processor based on Field Programmable Gate Array (FPGA) for Wireless Local Area Networks (WLAN) is Proposed. Unlike being stored in the traditional ROM, the twiddle factors in our pipelined FFT processor can be accessed directly. A novel simple address mapping scheme is also proposed. The FFT processor has two pipelines, one is in the execution of complex multiplication of the butterfly unit, and the other is between the RAM modules, which read input data, store temporary variables of butterfly unit and output the final results. Finally, the pipelined 64-point FFT processor can be completely implemented within only 67 clock cycles.

Keywords- FFT; FPGA; address mapping

I. INTRODUCTION

Fast Fourier Transform (FFT) processor is widely used in different applications, such as WLAN, image process, spectrum measurements, radar and multimedia communication services [1]. However, the FFT algorithm is a demanding task and it must be precisely designed to get an efficient implementation. If the FFT processor is made flexible and fast enough, a portable device equipped with wireless transmission system is feasible. Therefore, an efficient FFT processor is required for real-time operations [2] and designing a fast FFT processor is a matter of great significance.

In the past twenty years, FPGA has developed rapidly and gradually become universal. Compared with design flow of traditional ASIC, designs based on FPGA have the advantages of flexibility and high performance price ratio. Many researchers have studied on pipelined FFT based on FPGA [3], [4], [5]. For instance, in [3], they proposed an approach to design an FFT processor for wireless applications, but his design has too many clock cycles and isn't fast enough. In comparison to their designs, we propose a simple and feasible pipelined implementation of a 32-bit 64-point FFT processor based on FPGA for WLAN.

This paper is organized as follows. In the next section, we introduce a basic radix-2 FFT algorithm to briefly discuss which decimation is better to the system, a three-multiplication method, and a novel address mapping scheme which reduces delay and increases the speed of the system. In section III, the pipelined FFT architecture is proposed and each unit is also illustrated. Section IV is the implementation of the 64-point FFT processor based on FPGA, and hardware resources are explicitly listed out. The last section gives the conclusion.

II. FFT ALGORITHM AND ADDRESS MAPPING SCHEME

A. Radix-2 FFT Algorithm

The FFT algorithm can compute the Discrete Fourier Transform (DFT) effectively. Given a sequence {x(n)} of N complex numbers, we can compute its DFT, another sequence {X(k)} of N complex numbers, according to the following formula [6]

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

And according to the different way to decimate, it can be divided into two types, DIF (Decimation in Frequency) and DIT (Decimation in Time). The DIF algorithm is easier to design than DIT. And considering the finite word length effect, DIF has much more advantages than DIT, such as reducing the additive noise, which is introduced by the multiplication when it is implemented with the fixed point [7] and reducing the complexity of the whole system. Consequently, we use the DIF algorithm to design radix-2 FFT module and most of current FFT processors are also based on this algorithm [8].

B. Three-multiplication Method

It's undeniable that complex multiplication is the dominant factor affecting the speed and the throughput of FFT processor. Computing a complex multiplication requires four real multipliers and two real adders. As we all know, the hardware area of a real multiplier is larger than that of a real adder in FPGA. So we should do ours best to convert the complex multiplication into addition and subtraction to optimize the whole performance as high as possible. Having taken into account all operands are 32-bit complex numbers, the difference of two inputs $X_m(i)$ and $X_m(j)$ can be expressed by $Z_1 = x_1+jy_1$ and the twiddle factor $W_N^{nk}=\exp(\frac{2n}{N"}\pi)$ can be expressed by $Z_2 = x_2+jy_2$. So the product of them can be also expressed by z □ □x □ □jy . That's to say, the 16-bit real part x of the product is equivalent to $x_1\ x_2-y_1y_2$ and the 16-bit imaginary part y is equivalent to $x_1\ y_2-x_2y_1$ Therefore, we can transform the product z easily as the following equations

$$x=x_1(x_2+y_2)-y_2(x_1+y_1) \qquad (2)$$

$$y=x_1(x_2+y_2)-x_2(x_1-y_1) \qquad (3)$$

Obviously, using this factorization scheme, the system has some advantages [9]. The number of real multiplications is reduced from four to three. And addition has less consumption than multiplication. So the system power consumption is also reduced. In this study, we can save sixteen embedded multiplier 9-bit elements in FPGA. As for this 64-point FFT processor, the numerical values of x_2+y_2, x_1-y_1, x_1+y_1, x_2 and y_2 can be gotten before they participate in the real multiplication.

C. The Novel Address Mapping Scheme

In this paper, the block size of our system is 64 points. Having considered the properties of radix-2 64-point FFT, it needs to read 8 operands from memories at a time so as to achieve a high-speed FFT. As we all know, parallel accessing data is crucial to a system [10]. Thus, these 8 operands in our design are located in different row or column of memory blocks and this arrangement ensures that 8 conflict-free memory accesses can be performed in parallel. Initially, we use 8 32-bit dual-port memories to store 64 operands in sequence. And then a new linear shift conflict-free address mapping scheme is adopted to change the addresses of operands. The primary two-dimensional addresses of operands will be mapped to new ones. For instance, we assume that the original two-dimensional coordinate is (a, b), in which a and b represent the address of the data in one memory and the number of the 8 memories, respectively. Then, we obtain a new conflict-free address

(A, B) by means of the following equations

$$A=b, \qquad (4)$$

$$B=(a+b)\%8. \qquad (5)$$

In our design, it can be ensured that no memory location is read from or written to at the same time, and this new mapping scheme is feasible, effective and simple.

III. THE PIPELINED FFT PROCESSOR ARCHITECTURE

For high throughput systems, pipelined architecture is a good choice, and it is also an ideal method to implement high-speed long-size FFT owing to its regular structure
And simple control. The performance of pipelined FFT processor can be improved by optimizing the structure and saving hardware resources. The block diagram of our proposed FFT processor is illustrated in Fig.1. It consists of four essential units. Control unit, the kernel of the FFT processor, harmonizes the whole system. Butterfly unit (BU), which has three-stage pipelined structure, carries out the complex multiplication. Two dual-port RAMs are used to store and output data. And AGU, the abbreviation

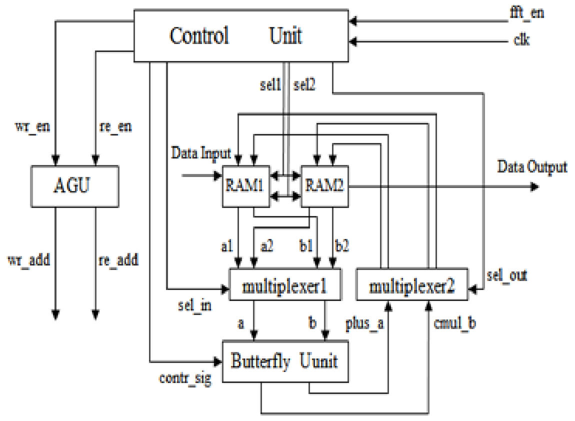

Figure 1. The pipelined FFT architecture.

of address generator unit, produces 8 3-bit read addresses and write addresses.

A. Control Unit

Control unit, which generates all control signals for the whole system, is responsible for operation control of the processor. A 48-bit signal w_con controls the whole FFT processor. And this signal w_con generates two parameters, write_en and read_en, to control AGU. It also generates sel1 and sel2 signals to select data from two RAMs, each of which is made up of 8 32-bit registers. The BU and the remaining parts are controlled by w_con as well. This control unit harmonizes all steps of the FFT processor based on a 7-bit counter.

B. The DIF Butterfly Unit

For FFT algorithm, the central component is the BU that calculates the sum and difference of two input data, and plays an extremely important role in computing the product of the difference and twiddle factors. We only use 11 factors w_{64}^{0} , w_{64}^{1} , w_{64}^{2} , w_{64}^{3} , w_{64}^{4} , w_{64}^{5} , w_{64}^{6} , w_{64}^{7} , w_{64}^{8} , w_{64}^{16} , w_{64}^{24} to express all 32 factors that we need in this FFT processor owing to the fact that the twiddle factor w_{N}^{nk} can be separated into two components. For instance, w_{64}^{28} can be derived from the product of w_{64}^{4} andw_{64}^{24} . Moreover, the value of w_{64}^{0} is a constant 1, and for w_{64}^{16} only a negative sign is needed to add to the real part of the relevant data, and then to inverse the real part and imaginary part. So we can eliminate these two factors, that's to say, actually we just need 9 twiddle factors. In addition, we use 16-bit fixed point decimal to express these 9 twiddle factors. Although the fixed point decimal arithmetic isn't precise enough, it can satisfy the requirements of general systems.

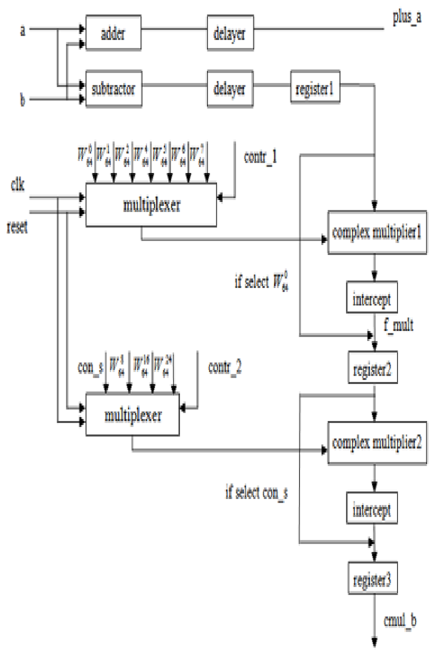

Figure 2. Block diagram of BU.

Due to the fact that the multiplication of twiddle factors and corresponding data is very important, three-stage pipeline structure is used for the complex multiplication to obtain a high speed computation. The architecture of BU is shown in Fig. 2.

In the BU, both of the complex inputs are 32 bits, including 16-bit real part and 16-bit imaginary part. The sum of them needs to be scaled down by a factor of 2 to avoid arithmetic overflow, and the same operation is applied to the difference of them. On the other hand, the factor 0 W64 and the first parameter con_s of the second Multiplexer are not involved in the complex multiplier, and they can be used as a constant 1, just as Fig.2 has depicted. Thus, the power consumption of the complex multiplier can be reduced and the hardware resources will be saved. There're some points to be emphasized. The difference and twiddle factors are both 32 bits, so the result of the first complex multiplier will be 64 bits. But because we adopt the fixed point decimal computation, we should intercept it to a 32-bit parameter f_mult as the input of the second complex multiplier.

The most remarkable advantage in this unit is that we Use 3 32-bit registers to realize the three-stage pipeline of butterfly transform, using register1 to store the difference, register2 to store the intercepted result of second stage f_mult and register3 to store the final result cmul_b.

C. The RAM Unit

The RAM1 and RAM2 are made up of 8 32-bit registers respectively. And data is always written to the outside memories from RAM2, and it is always read to RAM1 from the outside memories. Then let us introduce the key algorithm used in this unit. Considering the properties of 64-point FFT, we can use the radix-2 DIF 8-point FFT as a whole unit, so there

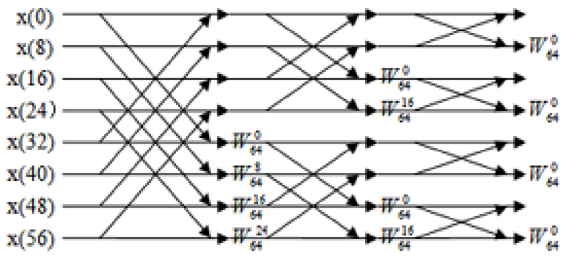

Figure 3. The example of radix-2 8-point FFT.

are only two stages to accomplish the 64-point FFT. And these two stages are identical to the six stages of the standard radix-2 DIF 64-point FFT. System parallel reads 8 32-bit operands from outer memories to RAM1 at a time, and we need only read sixteen times. An example of this Algorithm is shown in Fig.3.

The pipeline in RAM units is briefly discussed as follows. Firstly, system reads 8 operands from outer Memories and writes them to RAM1, and then the results will be stored in RAM2 after they are computed. Meanwhile, system reads the next 8 operands. Subsequently, system will access operands from RAM1 or RAM2 to compute these 16 operands. Furthermore, within a single clock four butterfly computations will be executed simultaneously. The final results of these 16 operands will be all stored in RAM2, and then they are written to outer memories. At the same time, another 8 operands will be read. Accordingly, only 76 clock cycles are needed to complete this radix-2 DIF 64-point FFT.

D. AGU

Compared with other units, AGU is also quite important. It will create 8 read and 8 write addresses, which determine the data access to outer memories.

In this FFT processor, we adopt the in-place computation method so as to make it a more simplified and faster system. That's to say, we write the results into where they are read. In contrast to the sequence of 64 input operands configured by address mapping, the final output sequence of this FFT processor are in bit-reversed order and need to be adjusted to normal order. And we can Make these appropriate adjustments before the FFT computation. So, although we adjust the sequence of inputs or outputs, the performance of our FFT processor won't be degraded.

IV. HARDWARE RESOURCES

The functional simulation and timing simulation are successfully made. The main hardware resources of this design are given as follows. The device is the EP2C70F896C6 of Cyclone II family. The total logic elements are 562/68,416 (8%), the total pins are 563/622(91%) and the total embedded multiplier 9-bit elements are 48/300(16%). Meanwhile, the clock frequency is 31.69MHz. As in [3], they proposed a fixed-point l6-bit 64-point FFT processor with 92 clock cycles in total, but our clock cycles is 67. And our FFT processor has a higher speed and lower power consumption.

V. CONCLUSION

This paper proposes a novel radix-2 FFT processor based on FPGA for WLAN, using Verilog HDL as hardware description language and Quartus II as design and synthesis tool. To achieve high-throughput, pipelined architectures have been used in the butterfly unit and the dual-port RAM. The dedicated parallel-pipelined FFT processor architecture can process input data at high speed, and the whole system performance can be greatly improved due to adopting a novel simple address mapping scheme. For radix-two system, this mapping scheme is better and simpler than most of others. The design is implemented on a FPGA chip. And this pipelined FFT completes a complex 64-point FFT within 2.1μs. The hardware testing result explains that it can meet the requirements of the WLAN.

REFERENCES

[1] J. A. C. Bingham, “Multicarrier modulation for data transmission: an idea whose time has come,” IEEE Communication Magazine, vol. 28, no. 5, pp. 5-14, May 1990.

[2] J. Palicot and C. Roland, “FFT: A basic function for a reconfigurable receiver,” 10th International conference on Telecommunications, vol. 1, pp. 898-902, March 2003.

[3] Min Jiang, Bing Yang, Yiling Fu, et al., “Design Of FFT processor with Low Power complex multiplier for OFDM-based high-speed wireless applications,” International Symposium on Communications and Information Technology, vol. 2, pp. 639-641, Oct. 2004.

[4] Kai Zhong, Hui He, and Guangxi Zhu, “An ultra-high speed FFT processor,” International Symposium on Signals, Circuits and Systems, vol. 1, pp. 37 -40, July 2003.

[5] Hongjiang He and Hui Guo, “The Realization of FFT Algorithm based on FPGA Co-processor,” Second International Symposium on Intelligent Information Technology Application, vol. 3, pp. 239-243, Dec. 2008.

[6] J. G. Proakis and D. G. Manolakis, “Introduction to Digital Signal Processing,” New York: Macmillan, 1988.

[7] R. B. Perlow and T. C. Denk, “Finite word length design for VLSI FFT processors,” Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, vol. 2, pp. 1227– 1231, Nov. 2001.

[8] J. W. Cooky and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” Math. of Comp., vol. 19, No. 90, pp. 297-301, April 1965.

[9] S. Oraintara, Y. J. Chen, and T. Q. Nguyen, “Integer fast Fourier transform,” IEEE Trans. Acoustics, Speech, Signal Processing, vol. 50, No. 3, pp. 607-618, March 2002.

[10] J. H. Takala, T. S. Jarvinen, and H. T. Sorokin, "A Conflict-free parallel memory access scheme for FFT processors," International Symposium on Circuits and Systems, vol. 4, pp. 524-527, May 2003.