Implementation of Symbol Timing Synchronization for MB-OFDM UWB Systems on FPGA

YANG Xiao, XUE Jing, WANG Lei, ZHANG Tong
School of Automation, Northwestern Polytechnical University
XI'AN, China
aiaiqqwo@gmail.com

***Abstract*—This paper proposes a design of symbol timing synchronization for MB-OFDM systems with low hardware consumption. Most of the existing algorithms for symbol timing synchronization cost a huge number of operations on cross-correlation or energy computing, so that, a great amount of multipliers and logic resources would be consumed if these algorithms are implemented on FPGA. The scheme proposed and implemented in this paper firstly locates the strongest muti-path by sign cross-correlation which has a low implementation complexity, and then finds the start of FFT window by utilizing an existing cross-correlation based algorithm inside a boundary, this boundary can be confirmed by the location of the strongest muti-path. The module for symbol timing synchronization has been designed in Quartus II, and the functional simulation result obtained shows correctness of the module.**

Keywords- UWB; MB-OFDM; timing synchronization; FPGA

I. INTRODUCTION

Multiband Orthogonal Frequency Division Multiplexing (MB-OFDM) is one of the three dominating transmission technologies proposed for UWB [1]. For short range high data rate transmission, MB-OFDM technology which has a high frequency spectrum utilization and strong capacity of combating with multi-path interference gives a better performance than the other two technologies, that is DS-CDMA and Impulse based UWB. The MB-OFDM system divides the 7500MHz spectrum from 3.2GHz to 10.6GHz allocated by federal communication commission (FCC) for UWB devices, into 14 bands with a bandwidth of 528MHz, data information is transmitted in each band by OFDM modulation.

Timing synchronization is a critical and indispensable step in any OFDM systems. Timing error can break the orthogonality of the OFDM subcarriers and cause inter symbol interference (ISI) as well as inter carrier interference (ICI), which degrade the performance of system dramatically. In this paper, a low complexity design for symbol timing synchronization that estimates the start location of FFT window is implemented on FPGA. Research on UWB channel [2] shows that, in some cases, the strongest multi-path is not necessarily the first received multi-path. Because of that, the conventional timing synchronization schemes proposed for narrow band OFDM systems are ill-suited for MB-OFDM systems. In recent years, more and more research focused on symbol timing synchronization for MB-OFDM systems. Most of these algorithms can give satisfactory results, however they also cost a huge number of operations. For this reason, these algorithms are difficult to be implemented. In this paper, we proposed a structure for dealing with computational complexity by narrowing down the scope of cross-correlation computing, a large quantity of unnecessary computation can be omitted.

This paper is organized as follows: Section II describes the system model for MB-OFDM system. Section III presents the scheme and flow chart of symbol timing synchronization in this paper. Section IV presents the structure of the implementation and functional simulation for the symbol timing synchronization module. Finally, Section V concludes the paper.

II. SYSTEM DESCRIPTION

A. Standard Preamble

The RF signal of MB-OFDM systems is transmitted across three sub bands by frequency hopping. A symbol is defined as an OFDM symbol (IFFT output with a length of N_{FFT} samples) plus a zero-padded suffix (N_{ZPS} samples). The WiMedia Alliance [5] specified that N_{FFT} and N_{ZPS} be of values 128 and 37. Dedicated preambles are used for timing synchronization, carrier frequency offset recovery and channel estimation. The preamble is defined to be a real baseband signal, which shall be inserted into the real portion of complex baseband signal. The standard preamble is consisted of 24 repeated synchronization symbols which are 128-point time-domain real pseudo-random sequences plus a 37 samples zero-padded suffix and 6 channel estimation symbols (CE). The 24 repeated symbols are respectively 21 packet synchronization sequences (PS) and 3 frame synchronization sequences (FS). Figure 1 shows the structure of the standard preamble for TFCs1 [3][5].

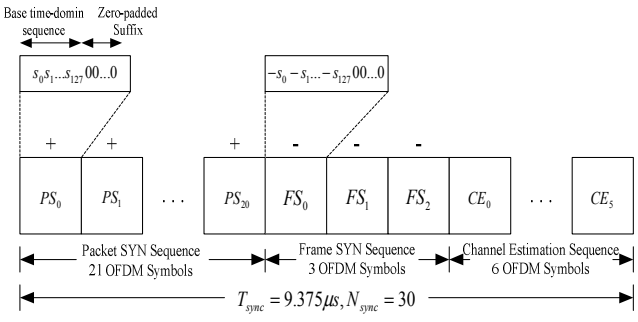

Figure 1. Block diagram of the standard preamble for TFCs1.

This work was supported by the astronautical support foundation of China through a grant provide by the China Aerospace Science& Industry Corporation in 2009(grant NO. 2009HTXGD).

B. UWB Channel module

The IEEE 802.15.3a task group has adopted a modified Saleh-Valenzuela (S-V) channel module for UWB [2]. The muti-path channel impulse response which is cluster based can be expressed as:

$$h(t) = X \sum_{j=0}^{J} \sum_{k=0}^{K} a_{k,j} \delta(t - T_j - \tau_{k,j}) \quad (1)$$

Where $a_{k,j}$ is channel coefficient for kth ray of jth cluster, T_j is delay of jth cluster, $\tau_{k,j}$ is delay of kth ray related to jth cluster arrival time, X is log-normal shadowing for the channel.

Let $t(n)$ be the time-domain sequence to be transmitted, $h(n)$ be the baseband equivalent of sample spaced impulse response of the UWB channel. Then the time-domain received sample can be expressed as:

$$r(d) = \sum_{i=0}^{l-1} t(d - i - \theta) h(i) + n(d) \quad (2)$$

Where d is the time integer index, θ is the timing offset, $r(d)$ is the received signal, l is the channel length and $n(d)$ is a complex AWGN random variable with zero mean.

III. SCHEME OF SYMBOL TIMING SYNCHRONIZATION

A. Complexity analysis

Cross-correlation function [4][6][8] utilized in most of the existing timing synchronization algorithms, can be expressed as:

$$C(d) = \sum_{\substack{i=d \\ j=0}}^{\substack{i=d+N-1 \\ j=N-1}} r(i) s(j)^* \quad (3)$$

Where $\vec{r}$ is the received signal, $\vec{s} = \{s_0 s_1, \ldots s_{N-1}\}, N = 128$ is the known training sequence i.e. the preamble pattern, $*$ denotes to complex conjugate operation. 256 multipliers will be consumed if all the values of $\vec{C}$ are computed, and that's a large consumption.

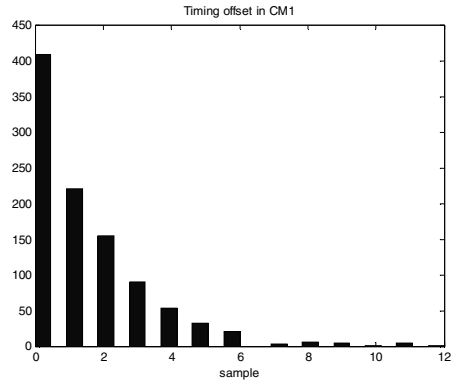

Figure 2. Timing offset of the strongest muti-path in CM1.

By studying the timing offset of the strongest muti-path to the first received muti-path, we can find a boundary inside which the values of $\vec{C}$ are necessary to compute and the others not, figure 2 and figure 3 show the timing offset of the strongest muti-path in CM1 and CM2, each of the channel is simulated for 1000 times. The results show that the first received muti-path must inside a boundary which is 13 samples before the strongest muti-path. In this way, the symbol timing synchronization algorithms based on cross-correlation could be implemented with only 28 multipliers if the location of the strongest muti-path was obtained.

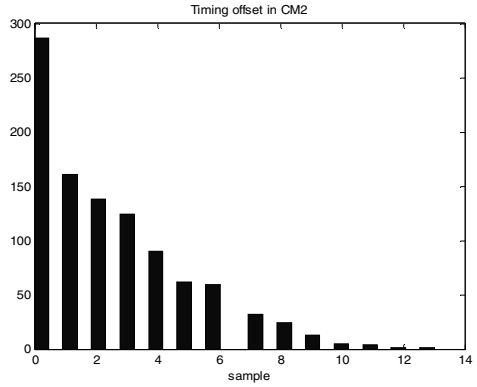

Figure 3. Timing offset of the strongest muti-path in CM2.

To cut down the resource consumption, we need to detect the strongest muti-path in the first received symbol of the preamble with low complexity. The preamble symbol is a real pseudo-random sequence and the sign of the sequence also have a fine auto-correlativity which is showed in figure 4, the cross-correlation of sign sequence can be expressed as:

$$C_{sign}(d) = \sum_{\substack{i=d \\ j=0}}^{\substack{i=d+N-1 \\ j=N-1}} r_{sign}(i) s_{sign}(j)^* \quad (4)$$

Where $\vec{C}_{sign}$ is the sign cross-correlation value, $\vec{r}_{sign}$ is sign of the received signal and $\vec{s}_{sign}$ is sign of the known training sequence. Because of its fine auto-correlativity, sign cross-correlation can be utilized for strongest muti-path detection. The strongest muti-path can be located if the absolute value of $C_{sign}(d)$ is greater than the subsequent 37 samples i.e. the longest muti path delay and greater than a predetermined threshold η which is used for data arrival detection.

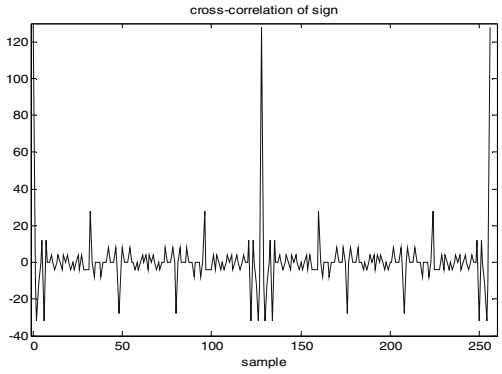

Figure 4. Cross-correlation of sign of the preamble symbol signal.

B. *Scheme for symbol timing synchronization*

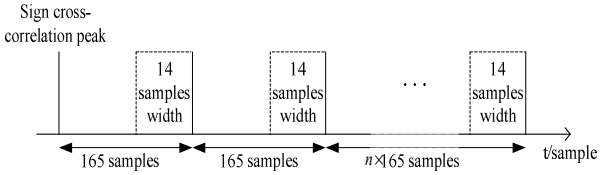

Figure 5. Boundary of every preamble symbol.

The channel impulse response (CIR) is assumed to be time-invariant within one data frame [3], therefore, after the peak detection by the first preamble symbol, location of cross-correlation peaks of the following symbols, which have the same timing offset as the strongest muti-path, can be confirmed by a period of 165 ($N_{FFT} + N_{ZPS}$) samples. The boundary that mentioned before can be confirmed by the location of correlation peaks, figure 5 shows the boundary that is 13 samples before cross-correlation peak of every preamble symbol.

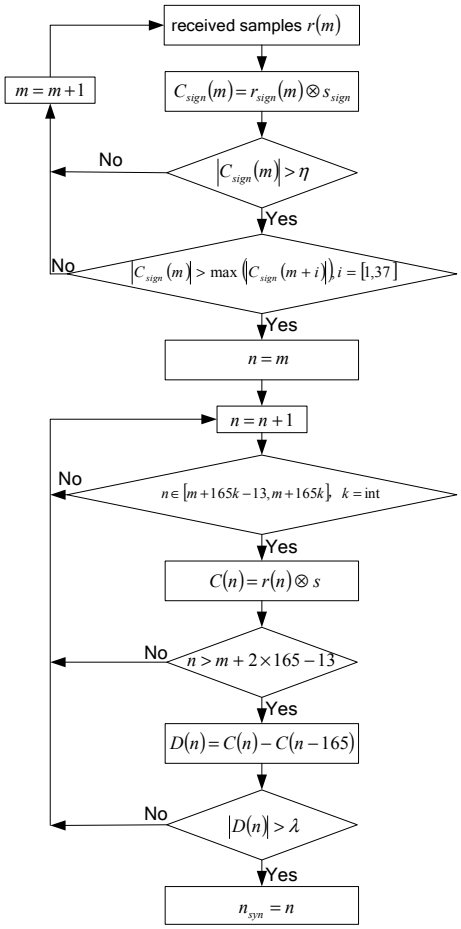

Figure 6. Scheme for symbol timing synchronization.

Once the boundary was confirmed, symbol timing synchronization can be achieved by cross-correlation based algorithms with low resource consumption. The algorithm named FTA was proposed in [6], which compare $C^*(d)C(d-165)$ against a predetermined threshold, and find the FFT start. Paper [4] proposed an algorithm that is similar to the FTA, this algorithm can be expressed as:

$$D(n) = C(n) - C(n-165) \tag{5}$$

If both the samples belong to the same polarity, the difference value $D(n)$ will give only the associated noise magnitude. When they belong to the different polarity, $D(n)$ reaches to a considerable magnitude. If the absolute value of $D(n)$ exceeds a predetermined threshold λ , then the start of FFT window is n . These two algorithms can reach to almost the same synchronization performance, so, in this paper we select the algorithm proposed in [4] for its lower consumption. Figure 6 shows the scheme for synchronization.

IV. IMPLEMENTATION

This implementation of symbol timing synchronization consists of three sub modules: the sign correlation module which is used for peak detection, the correlator module which is used for cross-correlation computing, and the difference computing module which is used for locating the start of FFT window. Figure 8 shows the structure of these three sub modules.

Preamble symbol is defined to be a real baseband signal, and the carrier frequency offset (CFO) has negligible effect on timing synchronization [6], therefore, only the real portion is utilized in this implementation for symbol timing synchronization. The sign correlation module computes the sign correlation values and locates the peak (3). Once the location of the strongest muti-path is confirmed, the correlator will compute the cross-correlation values when the received data reached to the boundary. Figure 7 shows the structure of the correlator which is consisted of 15 registers (D0-D14), 15 multipliers and a 128 point 16-bit width ROM, the known preamble sequences are stored in the ROM. When the input data are inside the boundary, the module will compute the cross-correlation values of these data.

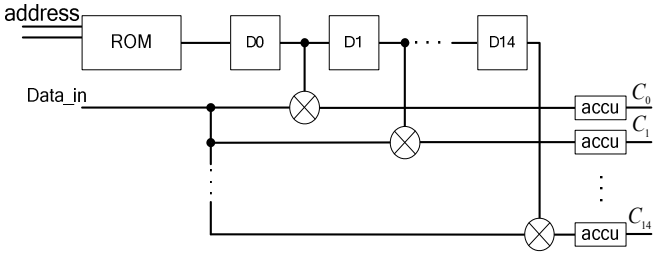

Figure 7. Structure of the correlator.

If more than two symbol's cross-correlation values are obtained, difference computing module can calculate the difference between cross-correlation values and find the start of FFT window. Functional simulation for the correlator is showed in figure 9, the input data are 16-bits width preamble symbol with 1 sign bit, 1 integer bit and 14 fraction bits, and the output C is 24 bits width with 1 sign bit, 10 integer bits and 13 fraction bits, the clock period is $10ns$. Because the input data are same to the stored preamble data, the output of C should reach to a peak value while the output is at the first sample of one symbol. The result shows that the C reached to

a peak value of 127.989 ($1048482/2^{13}$) when the output is at the first sample of one symbol.

Functional simulation for symbol timing synchronization is showed in figure 10, the input data are 9 packet synchronization symbols and 3 frame synchronization symbols with negative polarity, the 'fft_flag' rose up at the first sample of frame synchronization symbols, and that is the start of FFT window.

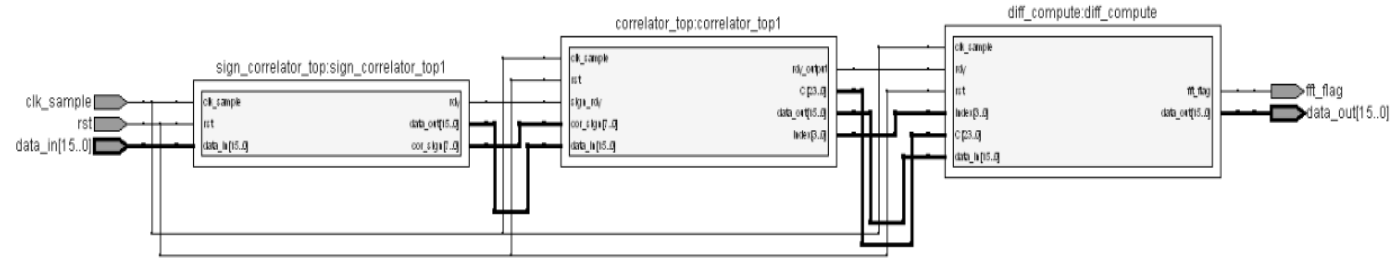

Figure 8. Structure of the implementation.

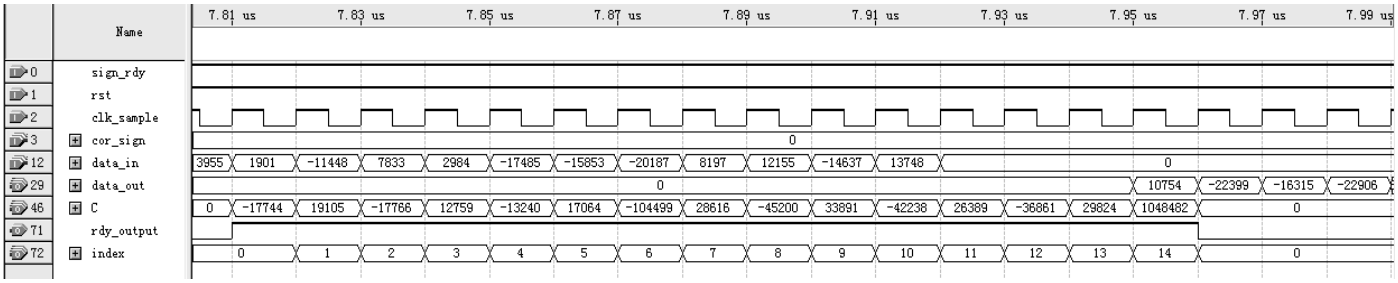

Figure 9. Functional simulation for the correlator.

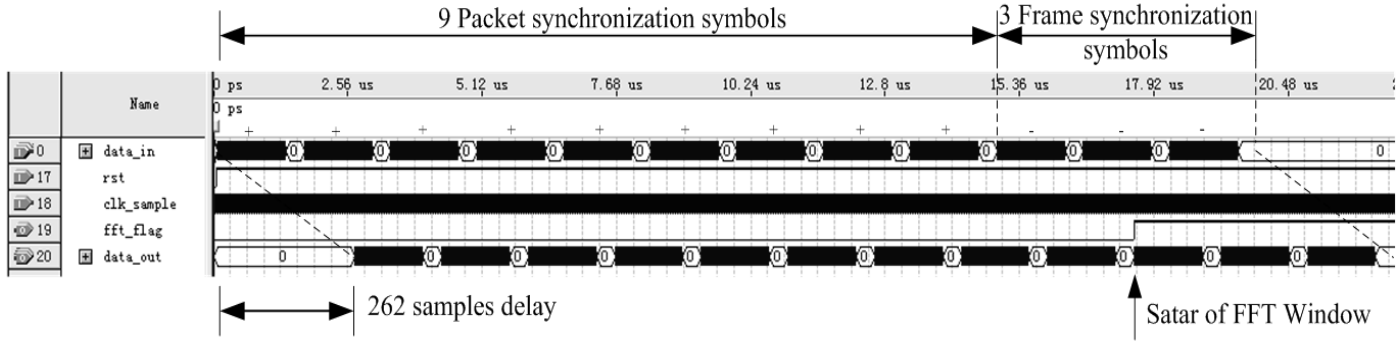

Figure 10. Functional simulation for symbol timing synchronization.

V. CONCLUSION

In this paper, a low complexity implementation of symbol timing synchronization for MB-OFDM systems is presented. Only 15 multipliers are consumed in this implementation for cross-correlation computing, this number would be at least 128 if the algorithm proposed in [4] is implemented on FPGA. The functional simulation result shows that the module is correct. In addition, the other modules for synchronization in MB-OFDM systems are also needed to be implemented, which is the future researching work to be done.

REFERENCES

[1] M. Gabriella, D. Benedetto and G. Giancola, Understanding Ultra Wide Band radio Fundamentals , House of Electronics Industry, May 2005.

[2] A. Batra, J. balakrishnan, G. R. Aiello, J. R. Foerster and A.Dabak, "Design of a Multiband OFDM System for Realistic UWB Channel Environments," IEEE Trans. on Microwave Theory and Techniques, vol. 52, issue. 9, pp. 2123-2138, September 2004.

[3] ECMA-368, "High Rate Ultra Wideband PHY and MAC standard," December 2005.

[4] D. Sen, S. Chakrabarti, "Symbol timing synchronization for ultra-wideband (UWB) multi-band OFDM (MB-OFDM) systems," IEEE Conference on COMSWARE. pp. 200, January 2008.

[5] WiMedia Alliance, "Mutiband OFDM physical layer specification: Final deliverable 1.5," August 11, 2009.

[6] C. W. Yak, Z. Lei, S. Chattong and T. T. Tjhung, "Timing synchronization and frequency offset estimation for Ultra-Wideband (UWB) Multi-Band OFDM systems," IEEE Symposium on PIMRC. vol. 1, pp.471, July 2006.

[7] Z. Z. Ye, C. Duan, P. V. Orlik , "A Synchronization Design for UWB-Based Wireless Multimedia Systems," IEEE Trans. on Broadcasting, vol. 56, issue. 2, pp.221-225, June 2010.

[8] X. Y. Wang, Z. C. Zhang, C. Y. Chen, "A robust time synchronization scheme for MB-OFDM UWB system," IEEE Conference on ICSPS. vol. 3, pp. 529-532, July 2010.