FPGA Implementation of an OFDM-WLAN Synchronizer

K Wang, J Singh and M Faulkner

Centre for Telecommunications and Microelectronics
Victoria University
PO Box 14428
Melbourne City, Vic 8001
Email : {kaiwang, jack.singh, mf}@sci.vu.edu.au

Abstract – **In this paper, we present a timing and frequency synchronization scheme and its FPGA implementation for IEEE 802.11a WLAN systems. In the scheme, an efficient double auto-correlation method based on short training symbols is used for timing synchronization. The performance of the proposed method is comparable or even superior to that of the conventional timing synchronization method under multipath fading channels. By averaging the correlation over four short training symbols, the accuracy of frequency synchronization using short training symbols can be improved to a level that the fine frequency synchronization process using long training symbols in the conventional scheme would not be needed. Thus both timing and frequency synchronization can be achieved using short training symbols alone to reduce computational complexity and overhead. Furthermore, the hardware architecture of the proposed synchronization scheme is developed. The synchronizer is mainly made up of correlator, angle calculator and peak detector, which are implemented by an iterative process, a CORDIC circuit and a finite state machine, respectively. Such an architecture results in low implementation complexity and low computational latency.**

I. INTRODUCTION

The Orthogonal Frequency Division Multiplex (OFDM) is an effective modulation technique used in high bit-rate wireless communications. It is adopted for wireless local area networks (WLANs), e.g., IEEE 802.11a and high performance LAN type 2 (HIPERLAN/2). However, OFDM systems are extremely sensitive to receiver synchronization imperfection [1], which can cause the degradation of system performance. Hence, OFDM synchronization has autonomously evolved into an interesting and active research area, where several approaches [2-4] have been proposed based on preamble symbols or cyclic prefix and pilot subcarriers. For burst mode transmission such as WLANs, the method of using preamble symbols [3] is preferred for fast timing and frequency synchronization. The conventional symbol timing synchronization schemes for IEEE802.11a WLAN systems use short training symbols to estimate a coarse symbol time via auto-correlation and then use long symbols to find a fine symbol time via cross-correlation as in [5]. For frequency synchronization in IEEE802.11a, the coarse frequency offset estimate is acquired from short training symbols and the fine frequency offset estimate is obtained from long training symbols.

OFDM systems can be implemented in programmable digital signal processors (DSPs), field programmable gate arrays (FPGAs) or application specific integrated circuits (ASICs). DSP implementation is advantageous in being flexible to modify the algorithm under implementation. A preamble based OFDM-WLAN synchronization algorithm is implemented in a programmable digital signal processor [5], where the synchronization operates at a rate of about 1/8 of real time. Although this process only operates at the beginning of every packet, it is straightforward to buffer incoming samples during the synchronization process. Apparently, this implementation incurs a relatively long latency.

FPGA/ASIC implementation has advantages of fast, small latency and low power. A cyclic prefix (CP) based OFDM synchronization algorithm used for continuous transmission is implemented in an ASIC [6], which contains 32 kbits RAM and 5500 gates and performs 1300MIPS at 25MHz clock.

In this paper, we present an efficient timing and frequency synchronization scheme based on the IEEE 802.11a preamble structure. The timing synchronization is achieved by a double auto-correlation method using short training symbols only. The performance of this approach is comparable or even superior to that of conventional fine timing synchronization [7]. In the process of finding the carrier frequency offset, the required frequency synchronization performance can be achieved by averaging the auto-correlation over four short training symbols. An FPGA implementation of the above timing and frequency synchronization scheme is discussed in the paper. Major components of the synchronizer under implementation include correlator, angle calculator and peak detector. The use of iteration

Proceedings of the Second IEEE International Workshop on Electronic Design, Test and Applications (DELTA'04)
0-7695-2081-2/04 $ 20.00 © 2004 IEEE

and CORDIC circuit leads to low implementation complexity and low computational latency.

The paper is organized as follows. Section II briefly describes the IEEE 802.11a baseband signal model and preamble structure. Section III presents the timing and frequency synchronization algorithms. Section IV explains the hardware architecture and building blocks. Section V shows the implementation results. The paper is concluded in section VI.

II. AN IEEE802.11A SIGNAL MODEL

In IEEE 802.11a, the OFDM modulation method is used for data transmission. Consider an OFDM system employing N subcarriers for the transmission of parallel data streams of width N_u, where $N - N_u$ subcarriers (virtual carriers) at the perimeter of the spectrum are regarded as the guard band.

At the transmitter, the data stream is mapped into N complex symbols in the frequency domain, including null data symbols for virtual subcarriers. These N complex symbols are modulated on to the N subcarriers by using Inverse Fast Fourier Transform (IFFT) to get a time domain complex OFDM symbol, which is represented as

$$x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(k) e^{\frac{j2\pi kn}{N}}, \qquad n = 0,1,...,N-1 \tag{1}$$

where $X(k)$ denotes the data symbol in subcarrier k, $x(n)$ is the nth sample of the OFDM symbol. The last N_g samples of the IFFT outputs are copied and added to form the cyclic prefix at the beginning of each OFDM symbol. In IEEE 802.11a, $N = 64$, $N_u = 52$ (including 4 pilot subcarriers) and $N_g = 16$. The time domain symbols are interpolated, D/A converted, mixed with a carrier and transmitted.

At the receiver, the received signals are down-converted, filtered, A/D converted and decimated to reconstruct the baseband signals. The received signals transmitted through multipath channels are expressed by

$$r(n) = \sum_{i=0}^{N_h-1} x(n-\theta-i) h(i) e^{\frac{j2\pi\varepsilon n}{N}} + w(n) \tag{2}$$

where $h(i)$ denotes the sampled complex channel impulse response (CIR), N_h is the length of CIR, θ is the timing offset, ε is the relative carrier frequency offset, and $w(n)$ represents complex white Gaussian noise. With the CIR confined to the cyclic prefix length, after removing the cyclic prefix the received signals are demodulated via FFT. The demodulated signal in subcarrier k is given by

$$Y(k) = X(k)H(k)\frac{\sin(\pi\varepsilon)}{\pi\varepsilon} e^{j\pi\varepsilon} e^{\frac{-j2\pi k\theta}{N}} + ICI(k) + W \tag{3}$$

where $H(k)$ denotes the channel transfer function, $e^{\frac{-j2\pi k\theta}{N}}$ is the phase rotation introduced by timing offset θ, $e^{j\pi\varepsilon}$ is the phase rotation introduced by carrier frequency offset, $\frac{\sin(\pi\varepsilon)}{\pi\varepsilon}$ is the amplitude attenuation caused by carrier frequency offset, $ICI(k)$ is the inter-channel interference in sub-carrier k caused by carrier frequency offset, and W represents the noise term. If the timing offset is not in the range of cyclic prefix, it will cause inter-symbol interference (ISI) and inter-channel interference.

In IEEE 802.11a [8], the training sequence is transmitted at the beginning of each frame to help the receiver accomplish synchronization and channel estimation. The training sequence consists of ten short training symbols, each of which is 0.8 μs, followed by two long training symbols, each of which is 3.2 μs plus a 1.6 μs prefix in front of the two long training symbols. The short training symbols are responsible for signal and packet detection, Automatic Gain Control (AGC) level setting, coarse timing synchronization and coarse carrier frequency offset correction. The long training symbols are used for fine carrier frequency offset and channel estimation.

III. TIMING AND FREQUENCY SYNCHRONIZATION ALGORITHMS

A. Timing synchronization

Due to the repeated short training symbol structure specified in IEEE802.11a, we can readily use auto-correlation for timing synchronization. In order to overcome the uncertainty in the timing metric plateau and increase estimation accuracy, we compute two normalized auto-correlation timing metrics. The first metric $M_1(\theta)$ is defined as the correlation between the received signal and itself with a delay of one short symbol N_s, where $N_s = 16$. The second metric $M_2(\theta)$ is defined as the correlation between the received signal and itself with a delay of two short symbols. Thus, $M_1(\theta)$ and $M_2(\theta)$ are written as

$$M_1(\theta) = \sum_{m=0}^{N_s-1} r(\theta+m) \times r^*(\theta+m+N_s) \tag{4}$$

$$M_2(\theta) = \sum_{m=0}^{N_s-1} r(\theta+m) \times r^*(\theta+m+2N_s) \tag{5}$$

By subtracting $M_2(\theta)$ from $M_1(\theta)$, we obtain a triangular shaped timing metric. By searching the maximum value of the difference $|M_1(\theta)|-|M_2(\theta)|$, we can detect a peak that indicates the start of the 9th short symbol. Hence, the timing estimate $\hat{\theta}$ is

$$\hat{\theta} = \arg\max_{\theta}(|M_1(\theta)|-|M_2(\theta)|) \tag{6}$$

The auto-correlation in equations (4) and (5) can be calculated iteratively. The performance of this estimator can be further improved by, for example, averaging over the time stamps of more short symbols acquired by calculating more auto-correlation metrics with different time delay.

The symbol timing estimate $\hat{\theta}$ could be earlier or later than the true time. If $\hat{\theta}$ is earlier than the true time, part of the cyclic prefix of the current symbol is taken as data, thus causing no interference. If $\hat{\theta}$ is later than the true time, part of the cyclic prefix of next symbol is taken as data, which introduces ISI. However, the ISI can be easily avoided by shifting ahead the estimated symbol time.

It is shown in [7] that sufficient estimation accuracy can be achieved by just using short training symbols alone when SNR is high.

B. *Frequency synchronization*

At the receiver, the main difference between two consecutive short training symbols is the phase difference caused by carrier frequency offset. For short training symbols, ignoring noise $w(n)$ in equation (2), we have

$$r(n)r^*(n+N_s) = \left|\sum_{i=0}^{N_h-1} x(n-\theta-i)h(i)\right|^2 e^{\frac{-j2\pi\varepsilon N_s}{N}} \tag{7}$$

So the phase of the auto-correlation above between two consecutive short training symbols is proportional to the carrier frequency offset. Using $M_1(\theta)$ in (4), the carrier frequency offset estimate $\hat{\varepsilon}$ can be expressed as

$$\hat{\varepsilon} = \frac{angle(M_1(\hat{\theta}))}{\pi} \tag{8}$$

where $\hat{\theta}$ is the symbol timing estimate obtained in subsection A. The advantage of this coarse frequency offset estimation is a large estimation range up to two sub-carrier spacing which is sufficient for WLANs, whereas the disadvantage of this estimator is that it leads to a reduced estimate accuracy. However, by averaging the auto-correlation metric over four short training symbols, as described by (9), the estimation accuracy can be improved to a level that meets the requirements for WLAN.

$$M(\theta) = \sum_{m=0}^{4N_s-1} r(m+\theta)r^*(m+\theta+N_s) \tag{9}$$

The frequency offset estimate is

$$\hat{\varepsilon}_f = \frac{angle(M(\hat{\theta}))}{\pi} \tag{10}$$

Thus the feed-forward frequency offset compensation is performed as

$$r'(n) = r(n)e^{-\frac{2\pi\varepsilon_f n}{N}} \tag{11}$$

where $r(n)$ is the received sample and $r'(n)$ the frequency offset corrected sample.

C. *Computational procedure*

Based on the analysis in subsection A and subsection B, a computational procedure of timing and frequency synchronization is summarised as follows.

Step 1. Compute the auto-correlation metrics $M_1(\theta)$ and $M_2(\theta)$ in equations (4) and (5), respectively.

Step 2. Find symbol timing estimate $\hat{\theta}$ using equation (6).

Step 3. Obtain $M(\theta)$ from equation (9) and compute the frequency offset estimate $\hat{\varepsilon}_f$ from equation (10).

Step 4. The received samples are phase corrected using (11).

IV. ARCHITECTURE AND BUILDING BLOCKS

As is known, the trade-off between performance and computational complexity is evaluated at various stages during product development. At the algorithm level, it is shown in [7] that the proposed synchronization scheme has a lower implementation complexity without performance loss than the conventional schemes. Further refinement is discussed in this section.

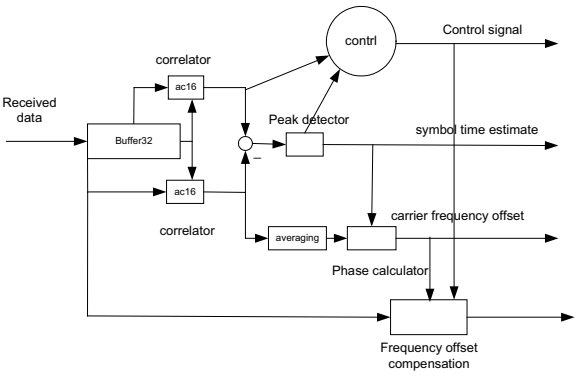

Figure 1 Block diagram of the synchronization algorithm

As depicted in Figure 1, the computational building blocks for the proposed algorithm include buffers, correlation, angle calculation, amplitude calculation, and maximum value search. The buffers are implemented using on-chip asynchronous FIFOs with reset.

The correlation metric $M_1(\theta)$ can be computed iteratively as shown in Figure 2. This simplifies the hardware implementation to one complex multiplier, one complex adder and one complex subtractor.

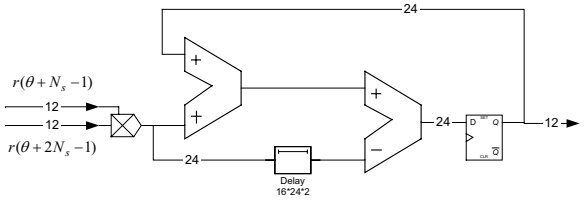

Figure 2 The correlator using iterative calculation

In Step 2 of the computational procedure, it is required to calculate the amplitude of the correlation. In general, producing amplitude of a complex number needs to use square root, which is a difficult function to implement in hardware. However, the correlation amplitude can be approximated [6] by $\left|\mathrm{Re}\{M_1(\theta)\}\right| + \left|\mathrm{Im}\{M_1(\theta)\}\right|$.

The carrier frequency offset estimation is to find the angle of the complex number $M(\theta)$, which is

$$\angle M_1(\theta) = \arctan\frac{\mathrm{Im}\{M_1(\theta)\}}{\mathrm{Re}\{M_1(\theta)\}} \qquad (12)$$

This angle is obtained using a Coordinate Rotation Digital Computer (CORDIC) circuit [9] in circular vector mode. The CORDIC is an iterative algorithm that needs one iteration per significant bit of output. The CORDIC function works by seeking to minimize the y_i component of the residual vector at each iteration. To compute $z = \arctan\frac{y_0}{x_0}$ from a complex vector $x_0 + jy_0$, the CORDIC equations are given as

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot \arctan 2^{-i}$$

where $d_i = \begin{cases} +1, & y_i < 0 \\ -1, & y_i \geq 0 \end{cases}$ and i is the iteration number which determines the accuracy of the result. After n iterations, the angle $z = z_n$. It only involves shifts, additions, subtractions and a look up table for $\arctan 2^{-i}$, which results in a low gate count. In order to generate one angle computed every clock cycle, each iteration of the CORDIC circuit is pipelined. To compensate for the frequency offset, each received sample should be back rotated by that amount. This is carried out by a CORDIC circuit in circular rotation mode.

The peak detector is performed by a finite state machine (FSM), as shown in Figure 3. The FSM detects the symbol time by comparing the amplitude difference of the correlation, as expressed in (6), with a predefined threshold and validates the symbol timing by comparing the amplitude difference with another predefined threshold. Heuristics is applied by using two thresholds.

As illustrated Figure 3, in the idle state, the amplitude of the correlation is compared with threshold1. If it is great than or equal to the threshold1, the state machine enters the search state. The current amplitude is saved as peak and the current index is saved as peak_index. In the search state, the incoming amplitude of the correlation is compared with the peak. It replaces the peak when it is greater than the peak. The peak_index is updated with the current index. When the amplitude is less than threshold2, the state machine enters the validation state. If the peak and peak_index are valid, the state machine transits to the idle state and the peak index is the symbol timing offset, otherwise the state machine returns to the search state.

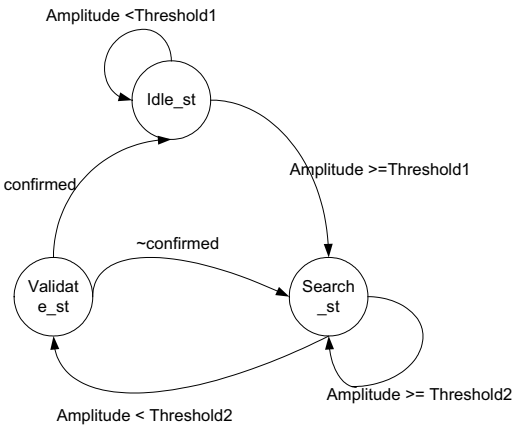

Figure 3 State diagram for peak detection

V. Implementation and Results

The algorithm and block models are first developed and evaluated in floating point format using Matlab. The fixed point C model is then developed, evaluated and optimised. This C model is used to design hardware architecture and generate test vectors. Simulations show that signal to quantization noise ratio maintains the required level of accuracy if the bit-width is chosen to be 12 bits. A comparison between floating point and fixed point simulation of the amplitude difference of the correlation, as expressed in (6), is shown in Figure 4. The design of register transfer level (RTL) model is described in Verilog HDL. The RTL model is verified against the C model by simulation. The RTL design is synthesised using Synplify and simulated using Modelsim.

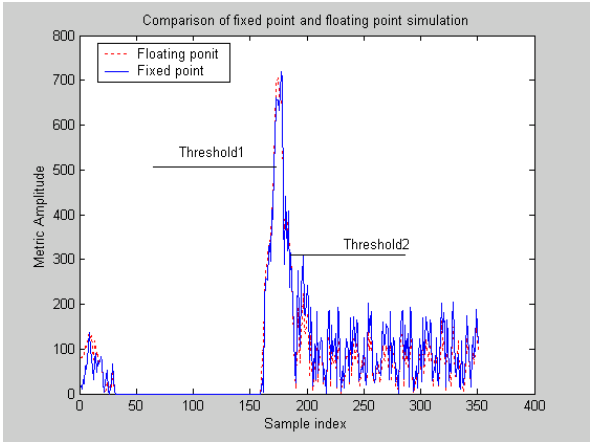

Figure 4 Comparison of floaing point and fixed point simulation

The RTL simulation waveform is shown in Figure 5. The main cursor indicates the maximum value of the amplitude difference of the correlation. The IEEE802.11a specifies that every OFDM symbol is output in $4\mu s$, which corresponds to 20MHz chip rate. In order to complete FFT within $3.2\mu s$, the system clock rate is selected to be 40MHz. The implementation of the synchronizer utilizes 1778 logic elements, 4kbits RAM and 2 DSP blocks of Altera Stratix device.

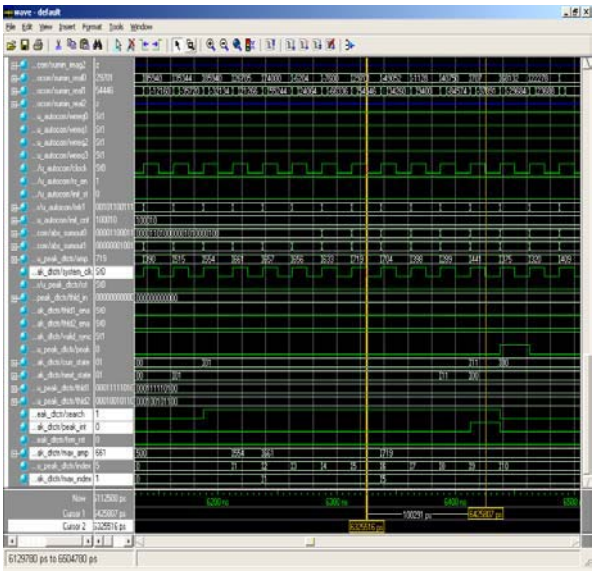

Figure 5 The wave window of RTL simulation

VI. Conclusion

In the paper, we have presented a new timing and frequency synchronization scheme and its FPGA implementation for IEEE 802.11a WLAN. The hardware architecture consists of the following building blocks: correlator, angle calculator, peak detector and buffers. The correlator is implemented by an iterative process, the angle calculator is realized using a pipelined CORDIC circuit, and peak detector is performed by FSM. The HDL synthesis and simulation results are presented, which show a low implementation complexity and latency.

VI. References

[1] R. V. Nee and R. Prasad, *OFDM for Wireless Multimedia Communications*: Artech House, 2000.

[2] J.-J. v. d. Beek, M. Sandell, and P. O. Börjesson, "ML estimation of time and frequency offsets in OFDM systems," *IEEE Trans. Signal Processing*, vol. 45, pp. 1800-1805, 1997.

[3] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Trans. Commun.*, vol. 45, pp. 1613-1621, 1997.

[4] I. Greenberg, "Retraining WLAN Receivers for OFDM Operation," *Communication Systems Design*, vol. 8, 2002.

[5] A. J. Coulson, "Maximum Likelihood Synchronization for OFDM Using a Pilot Symbol: Algorithms," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 2484-2494, 2001.

[6] S. Johansson, M. Nilsson, and P. Nilsson, "An OFDM Timing Synchronisation ASIC," presented at IEEE-ICECS 2000, Lebanon, 2000.

[7] K. Wang, M. Faulkner, J. Singh, and I. Tolochko, "Timing Synchronization for 802.11a under Multipath Channels," to appear in the proceedings of Australian Telecommunications, Networks and Applications Conference, Melbourne, 2003.

[8] IEEE, "IEEE802.11a-1999 part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer(PHY) specifications," IEEE 1999.

[9] H. Zhang, Z. Wang, and S. S. Chandra, "Implementation of Frequency Offset Correction Using CORDIC Algorithm for 5GHz WLAN Applications," presented at IEEE-ICC2002, 2002.