Analysis of Efficient Time Synchronization Algorithms in OFDM WLAN

Mohanraj.V,
P.G Scholar ,Bannari Amman Institute of Technology,Sathyamangalam

K.Shoukath Ali
Assistant Professor, Bannari Amman Institute of Technology,Sathyamangalam.

.

ABSTRACT

OFDM receiver starts its work with detecting the received signals from the transmitter, thus to detect the received signal synchronization algorithm is worked out especially time synchronization algorithm. This paper deals with the design of Coarse time synchronization module for OFDM-based WLAN. The circuit is particularized for IEEE 802.11a/g standards. The algorithms simulated are selected taking into account performance, hardware cost and latency. Our objective is to simulate Coarse time synchronization algorithm using modelsim 6.3 tool through floating point representation and analyse it to have minimum hardware resources for the efficient receiver design. Thereby, comparing with fine time synchronization.

Keywords:
Coarse time synchronization algorithm, floating point models, WLAN Receiver,fine time synchronization.

1. INTRODUCTION

An OFDM system carries payload data on orthogonal sub-carriers for parallel transmission, combating the distortion caused by the frequency-selective channel. However, the advantages of OFDM can only be realized when orthogonality is maintained. If not, its performance may be degraded due to inter-symbol interference (ISI) and inter-channel interference (ICI). In this junction, analyzing the effects of the symbol-time offset (STO) in OFDM systems. The inverse fast Fourier transform (IFFT) and fast Fourier transform (FFT) are the fundamental functions required for the modulation and demodulation at the transmitter and receiver, respectively. In order to determine the N-point FFT in the receiver, there is a need that the exact samples of the transmitted signal for the OFDM symbol duration. In other words, a symbol-timing synchronization must be performed to detect the starting point of each OFDM symbol.First, the IEEE preamble must be detected. This is done by means of an auto-correlation of the SS's and, as a result, a coarse time reference is obtained.

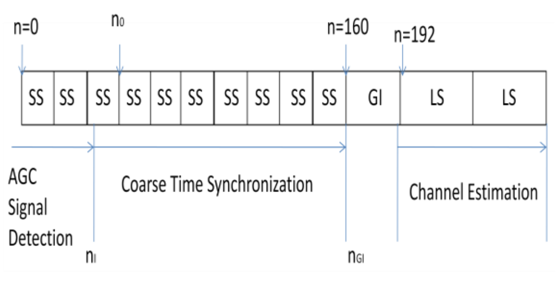


Fig.1 IEEE 802.11a Preamble Format

Next, the algorithm proposed for coarse time synchronization including the implementation details and the fixed point analysis. Finally, our synchronizer will be compared with other synchronization schemes in terms of performance and hardware resources.

2. COARSE TIME SYNCHRONIZATION

Synchronization process starts with detecting the data frame from the burst of data using AGC as shown in fig 1.1. Immediately, it is mandatory to estimate accurate time reference n^$_{GI}$ in order to avoid ISI with neighbouring symbols. Positive time offsets cause ISI because FFT window considers the next symbol. Despite of which, some samples of CP are affected due to multipath channels. In order to avoid it, analog filtering is required for both transmission and reception respectively. And interpolate and decimate filters[11] are also required.

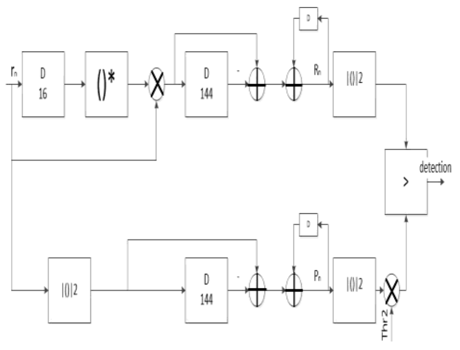


Fig.2Coarse time Synchronization Block diagram

A design rule[4] in common is to accept that the synchronization is exactly correct up to the deviation of 0 to 4 samples. In case of over 4 samples occur during synchronization, then it is defined as ISI in channels with moderate to high delay spread.

As mentioned above, the proposed algorithm is coarse time synchronization which is an adaptation of the auto-correlation method proposed by Schmidl and Cox[5]. The detected signal is auto-correlated with a delay of 16 samples and averaged during 144 samples which can be given as,

$$Rn=r_n^H r_{n+16}$$

where $r_n=[r_n .....r_{n+143}]$

This Rn is a vector with 144 samples from the received signal. Due to this delay and sampling, a peak result is obtained in n=160. At this peak, the transition from Short Symbols to Guard interval occurs.

Next, the modulus of the auto-correlation output is normalized by the local mean power, $P_n= \|r_n\|^2$, of the received signal. As defined in the timing metric [5]:

$$\overline{R_n} = \left|R_n\right|^2 / P_n^2 \qquad \text{----} \quad (1)$$

Estimation is improved by averaging the $|R_n|^2$ to 5 samples. Following which, a threshold Thr is set to find the position of peak $n\wedge_{GI,}$ which is obtained by searching the maximum of the averaged $|R_n|^2$ between those samples that full fill the condition R_n>Thr. To eliminate the costly division operation, again a condition is used. It is given by,

$$\left|R_n\right|^2 > (P_n^2 .\text{Thr}^2) \qquad \text{----} \quad (2)$$

The block diagram of proposed coarse time synchronization algorithm is shown in fig 2. This algorithm can be directly mapped in a VLSI implementation which implies that it is the maximum detection algorithm with low cost and low latency comparing present and previous samples during two clock cycles. The comparison results in obtaining the greater value amongst the samples.

The main part of the algorithm is choosing the threshold value to the system and theoretical analysis were made in order to have perfect threshold value, the probability distributions of R_n were used which minimizes the probability of not detecting the beginning of GI. In this paper, threshold value is taken as 0.4375 at SNR equal to or higher than 6dB. This guarentees that the probability of not detecting the GI is less. Additionally, the multiplier needed in (2) can be efficiently implemented.

By this algorithm, the deviation error obtained with respect to the ideal point n_{GI} at a SNR of 6dB in a multipath channel. Three BRAN channel models were used [6]: A, B and C with an RMS (Root Mean Square) delay spread of 50 ns, 100 ns and 150 ns, respectively. Through this relative models test frames are plotted in frequency of deviation such that each one transmitted through a different channel realization for each channel model. The minimum and maximum deviation rarely changes for test models. And the range is between 4 and 15 samples.

Moreover, the CFO $\hat{f}$ is estimated at $n\wedge_{GI}$ as [12]:

$$\hat{f} = \frac{\angle R_n}{2\prod 16 T_s} \qquad \text{----} \quad (3)$$

Where T_s is the sampling period. This coarse time synchronization works well when the maximum CFO allowed by IEEE 802.11a/g standard [1,2] occurs: 232 kHz (73% of the subcarrier spacing); and that, thanks to the large average length selected for the auto-correlation, the achieved CFO estimation is precise enough for the highest modulation order used in the standard (64-QAM): the estimation error has a standard deviation of 0.35% of the subcarrier spacing for SNR higher than 20 dB, which gives a Bit Error Rate (BER) below 10^5 in the floating-point receiver.

Therefore, fine frequency synchronization, which is usually estimated using an autocorrelation of the LS [4], is not necessary and, as a result, the final latency of the synchronizer is considerably reduced.

3. FLOATING POINT REPRESENTATION

All the operations as mentioned in the block diagram is carried out using floating point representation. Thus, each block such as floating point addition and multiplier is carried out through specific algorithm as mentioned below.

3.1 Floating Point Addition

Consider two floating point representations such as X and Y. Initially, convert the two representations to scientific notation. Thus, we explicitly represent the hidden. In order to add, we need the exponents of the two numbers to be the same. We do this by rewriting Y. This will result in Y being not normalized, but value is equivalent to the normalized Y. Add X-Y to Y's exponent. Shift the radix point of the mantissa (significant) Y left by X-Y to compensate for the change in exponent. Add the two mantissas of X and the adjusted Y together. If the sum in the previous step does not have a single bit of value 1, left of the radix point, then adjust the radix point and exponent until it does. Convert back to the one byte floating point representation. Thus, adder block in the block diagram shown in fig 2.1 works under the above mentioned algorithm.

3.2 Floating Point Multiplication

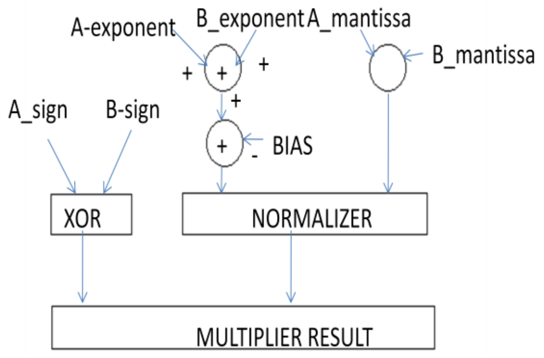


Fig.3 Floating point multiplier

Floating point multiplier in the coarse time synchronization algorithm works based on [11] . Thus, it considers single precision for the floating point representation but it can be extended to double precision with little more complexity in the algorithm.

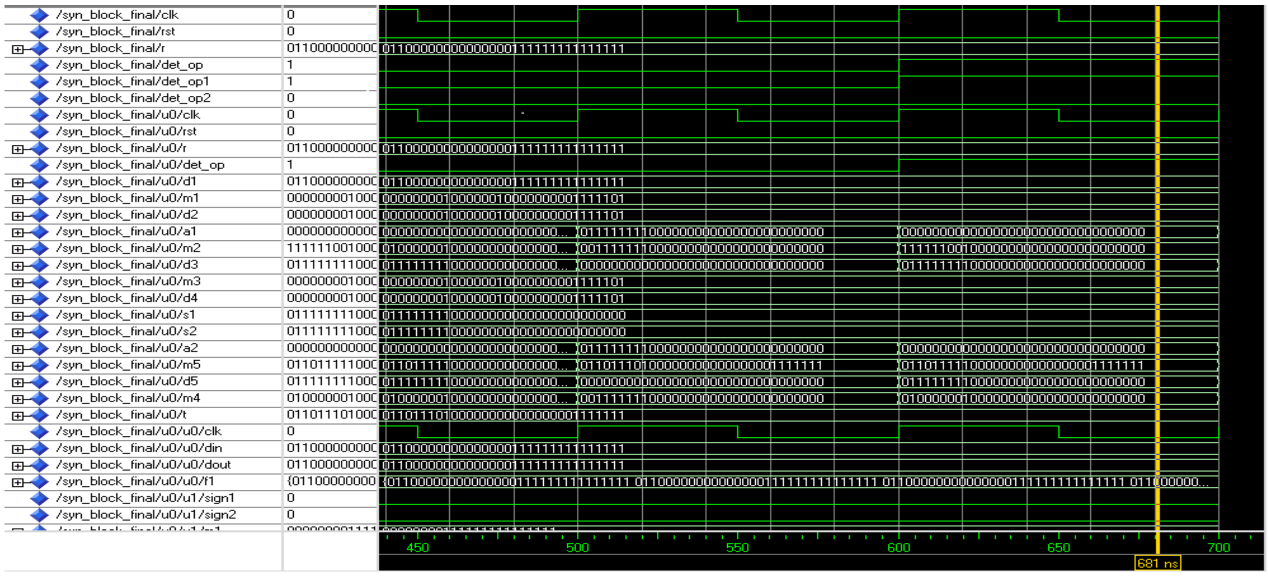
3.3 Other Blocks

Apart from these major blocks, there are some minute blocks such as D flip flop, Conjugate and comparator. These blocks are normally coded such as delay of 1,16 and 144 samples are just proceeded with counting the delays. And conjugate block is executed with respect to [7]. General, comparison of signals with decision as output is used in this time synchronization block.

4. RESULTS AND COMPARISON

4.1 Results

This coarse time synchronization algorithm can be analysed through the decision that the detected signal is synchronized or not. The proposed algorithm detects the signals that are synchronized and rejects the signal that is not synchronized. From the result shown in the fig (4.1), it can be inferred that the sample signal given is synchronized by showing the decision bit to 1.



```
/syn_block_final/clk            0
/syn_block_final/rst            0
/syn_block_final/r              011000000000
/syn_block_final/det_op         1
/syn_block_final/det_op1        1
/syn_block_final/det_op2        0
/syn_block_final/u0/clk         0
/syn_block_final/u0/rst         0
/syn_block_final/u0/r           011000000000
/syn_block_final/u0/det_op      1
/syn_block_final/u0/d1          011000000000
/syn_block_final/u0/m1          000000001000
/syn_block_final/u0/d2          000000001000
/syn_block_final/u0/a1          000000000000
/syn_block_final/u0/m2          111111001000
/syn_block_final/u0/d3          011111111000
/syn_block_final/u0/m3          000000001000
/syn_block_final/u0/d4          000000001000
/syn_block_final/u0/s1          011111111000
/syn_block_final/u0/s2          011111111000
/syn_block_final/u0/a2          000000000000
/syn_block_final/u0/m5          011011111000
/syn_block_final/u0/d5          011111111000
/syn_block_final/u0/m4          010000001000
/syn_block_final/u0/t           011011101000
/syn_block_final/u0/u0/clk      0
/syn_block_final/u0/u0/din      011000000000
/syn_block_final/u0/u0/dout     011000000000
/syn_block_final/u0/u0/f1       {011000000000
/syn_block_final/u0/u1/sign1    0
/syn_block_final/u0/u1/sign2    0
```

Fig.4 Coarse time synchronization result

4.2 Comparison

The coarse time synchronization algorithm is done using auto correlation and this is not well enough to compete with prevailing multipath channels in the wireless communication systems. Thus, comparing this coarse time synchronization with fine timing synchronization algorithm, we can come to a conclusion that it cross correlation concept is widely used with cross correlating the received signal with known training preamble. Thus, comparing coarse time with fine time synchronization algorithm, fine time is more advantageous and one more thing is the estimation of CFO which is mandatory in fine time synchronization adds as a disadvantage compared to coarse time synchronization

5. CONCLUSION

From the results and comparison, it can be concluded that an efficient time synchronization can be designed using the mentioned algorithm and it can be extended with comparing with other algorithms to improve its efficiency and usage in various applications. In near future, it can also be extended with number of bits, thus improving the method consistently.

6. REFERENCES

[1] IEEE Standard 802.11a, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band, December, 1999.

[2] IEEE 802.11g: Wireless LAN Specifications: Further Higher Data Rate Extension in the 2.4 GHz Band, June, 2003.

[3] M.J. Canet, F. Vicedo, J. Valls, V. Almenar, Design of a Digital Front–end Transmitter for OFDM-WLAN Systems Using FPGA, ISCCSP 2004, Hammamet,Tunisia, 2004.

[4] J. Heiskala, J. Terry, OFDM Wireless LANs: A Theoretical and Practical Guide,SAMS Publishing, 2001.

[5] T. Schmidl, D. Cox, Robust frequency and timing synchronization for OFDM,IEEE Transactions on Communications 45 (12) (1997).

[6] J. Melbo, P. Schramm, Channel Models for HIPERLAN/2 in Different Indoor Scenarios, 3ERI085B, HIPERLAN/2 ETSI/BRAN Contribution, 1998.

[7] Chu Yu, Mao-Hsu Yen, Pao-Ann Hsiung, Sao-Jie Chen. Low-Power 64-point Pipeline FFT/IFFT Processor for OFDM Applications. IEEE transactions on consumer electronics, Vol. 57, No. 1, 2011.

[8] Sekchin Chang, B. Kelley, Time synchronization for OFDM-based WLAN systems, Electronics Letters 39 (13) (2003) 1024–1026.

[9] Yik-Chung Wu, Kun-Wah Yip, Tung-Sang Ng, Erchin Serpedin, Maximum Likelihood symbol synchronization for IEEE 802.11a WLANs in unknown frequency-selective fading channels.