PERFORMANCE EVALUATION OF FINE TIME SYNCHRONIZERS FOR WLANS

M.J. Canet[1], *I.J. Wassell*[2], *J. Valls*[1], *V. Almenar*[1]

[1] Universidad Politècnica de València, Gandia, Spain, macasu@doctor.upv.es, www.gised.upv.es
[2] Engineering Department, University of Cambridge, Cambridge, U.K.

ABSTRACT

In this paper the performance and implementation cost of three different fine time synchronization algorithms have been evaluated for their use in WLANs. In order to evaluate the performance, the residual time offset after fine time synchronization have been calculated. A hardware structure has been proposed for each algorithm and some simplifications are added that reduce the hardware cost without performance reduction. A Virtex II FPGA device has been selected as a target technology for the implementation. The results indicate that a cross-correlation algorithm with only 28 coefficients achieves the lowest hardware cost with similar performance with respect to the others algorithms.

1. INTRODUCTION

New WLAN (WirelessLAN) standards in the 5 GHz band (HiperLAN2 and IEEE 802.11a) and in the 2.4 GHz band (IEEE 802.11g) are based on Orthogonal Frequency Division Multiplexing (OFDM) transmission [1, 2, 3]. In WLAN systems synchronization and channel compensation depend on using a preamble. In HiperLAN2 a broadcast preamble is used (Fig. 1). This preamble has 3 sections: A, B, and C. Section A is used for AGC (Automatic Gain Control) and frame detection, section B is intended for time synchronization and coarse CFO (Carrier Frequency Offset) estimation and section C can be used for fine CFO estimation and channel estimation.

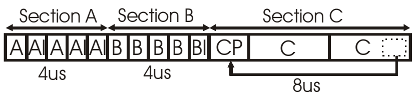

Fig. 1. Broadcast preamble

A synchronizer was designed and implemented on a FPGA (Field Programmable Gate Array) [4], which comprises frame detection, coarse time synchronization and coarse and fine CFO estimation and compensation. After coarse CFO compensation, the maximum residual CFO is 10kHz, while after fine CFO compensation, the maximum residual CFO is 1kHz. Finally, the corrected section C is used for channel estimation in the frequency domain. The sampling frequency is 20MHz, but the FFT used to do channel estimation needs to work at 120MHz [4].

With the implemented coarse time synchronizer, we get a deviation with respect to the ideal initial sample between -4 and 8 samples, for multipath channel model A [5] and 10dB SNR. Positive time offsets (1-8) causes ISI (InterSymbol Interference) because the FFT (Fast Fourier Transform) window takes samples of the next symbol. Negative time offsets do not create significant problems, but time offsets higher than (-5-0) are not recommended [6], because some samples from the cyclic prefix are affected by multipath channel. So, a fine time synchronization algorithm is needed.

The main idea of this paper is to add fine time synchronization to the previously implemented synchronizer. Three different algorithms have been reviewed. Before performing fine time estimation, coarse-timing estimation permits some timing correction to be performed, which enables the subsequent fine estimation to use the ISI-free part of the cyclic prefix. Each algorithm is mathematically analyzed and their performance is obtained. Then, a hardware structure is proposed, modifying the original algorithm in order to reduce hardware cost as much as possible. Once hardware structure is known, a fixed-point analysis is made in order to use the minimum number of bits in each stage and reduce hardware cost with a performance loss lower than 0.5%. Finally, performance of the implemented algorithm is checked for frequency offsets of 200kHz (maximum allowed in HiperLAN2), 10kHz (maximum residual frequency offset after our coarse CFO compensation) and 1kHz (maximum residual frequency offset after our fine CFO compensation), at SNR of 10dB, 15dB and 20dB in multipath channel A.

The paper is organized as follows. Sections II, III, and IV describe three different fine time synchronization algorithms: channel impulse response estimation algorithm, ISOCA algorithm and a cross-correlation algorithm respectively. Also, a hardware structure is proposed for each method. In Section V performance and implementation cost is evaluated. Finally, in Section IV the conclusions are presented.

2. CHANNEL IMPULSE RESPONSE ESTIMATION ALGORITHM

A time domain algorithm is proposed in [7]. First, the ML (Maximum Likelihood) channel impulse response is estimated. After that, this estimate is used to detect the time offset, since the maximum channel tap indicates where it is located. In this method it is assumed that the channel response remains constant over at least one OFDM symbol interval. The received sample vector (**r**) can be expressed as:

$$\mathbf{r} = \mathbf{S}\cdot\mathbf{h} + \mathbf{n} \tag{1}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}(0) & \mathbf{s}(-1) & \ldots & \mathbf{s}(-K+1) \\ \mathbf{s}(1) & \mathbf{s}(0) & \ldots & \mathbf{s}(-K+2) \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{s}(N-1) & \mathbf{s}(N-2) & \ldots & \mathbf{s}(-K+N) \end{bmatrix} \tag{2}$$

where **n** is a noise vector of N samples, **h** is the channel impulse response of K samples length, and **s** is the transmitted Section C of the preamble including a cyclic prefix of 32 samples.

Equation (3) performs a ML channel response estimate, where $\mathbf{C}_{rx}$ is received Section C signal and $[\mathbf{S}^H\,\mathbf{S}]^{-1}\,\mathbf{S}^H$ is the pseudo inverse of **S** which is an NxK matrix that can be previously calculated, because it only depends on samples from the transmitted Section C.

$$\hat{\mathbf{h}} = \left[\mathbf{S}^H \cdot \mathbf{S}\right]^{-1} \cdot \mathbf{S}^H \cdot \mathbf{C}_{rx} \tag{3}$$

Once channel impulse response is estimated, it can be used for fine time synchronization. The position of the first sample that agrees with equation (4), where *THR* is the selected threshold, is considered as the estimated timing offset.

$$\left(\left|\hat{\mathbf{h}}(n)\right|^2 - \left|\hat{\mathbf{h}}(n-1)\right|^2\right) \Big/ \max\left(\left|\hat{\mathbf{h}}\right|^2\right) > THR \tag{4}$$

Now, hardware implementation is studied. To obtain $\hat{\mathbf{h}}$ it is necessary to multiply a Kx64 complex matrix by a 64x1 complex vector. So, 64 complex products and 64 complex additions are needed for each sample of $\hat{\mathbf{h}}$. Hardware resources needed to do these operations are: K complex multipliers, K complex accumulators and a ROM where the precomputed elements of the pseudo inverse matrix are stored. The input data rate is 20 MHz, and the maximum clock frequency of our design in the FPGA is 120 MHz. In order to reduce hardware cost, multipliers and accumulators can work at 120MHz, so K/6 multipliers and K/6 accumulators are needed. For multipath channel A, the required performance is achieved with an estimation of 12 samples of the channel impulse response. In Fig. 2 the hardware implementation of this method is shown. It requires two complex memories, ROM1 and ROM2, two complex multipliers, M1 and M2, two complex accumulators and a complex output multiplexer. Rows 1 to 6 of the pseudo inverse matrix are stored in ROM1, while rows 7 to 12 are saved in ROM2.

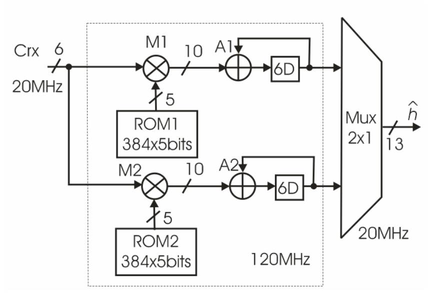

Fig. 2. Implemented channel impulse response estimator.

In Fig. 3 the implementation of equation (4) can be seen. On the left side, a maximum detection circuit is shown. First, the squared modulus is obtained and then, instant and previous samples are compared. If the latest sample is greater, it is saved in the register. After 12 cycles, the maximum value can be read from the register. On the right side of Fig. 3, time offset estimation implementation can be seen. The threshold (*THR*) is set to 0.125, so a multiplier is saved. After 12 cycles, the time offset estimation can be read from the register.

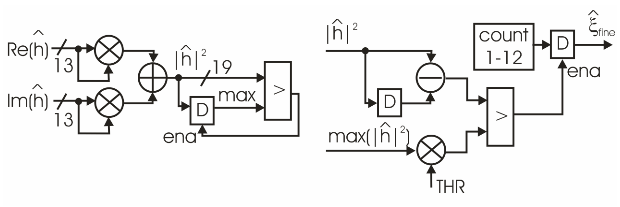

Fig. 3. Implementation of equation (4).

3. ISOCA ALGORITHM

The frequency domain based Iterative Symbol Offset Correction Algorithm (ISOCA) is proposed in [8]. ISOCA was designed for Broadband Fixed Wireless Access Systems and is tested in SUI-II channels. In this algorithm, symbol offset is iteratively estimated and corrected until the value is zero. The algorithm is based on the phase gradient caused by the symbol offset. Channel models for HiperLAN2 are not as benign as SUI-II channel and distort the phase gradient. For this reason, in a WLAN scenario, performance is better if only one iteration is used.

First, the phase difference between the received and the transmitted section C is obtained:

$$\boldsymbol{\theta}_i = \angle\left(\mathbf{Crx}_i / \mathbf{Ctrx}_i\right),\ 1 \le i \le 52 \tag{5}$$

where **Crx** is received Section C (in the frequency domain), **Ctrx** is the transmitted Section C and i includes data sub carriers and pilots. Then, this phase difference is unwrapped and, finally, the ML gradient of the unwrapped phase in the LS (Least Square) sense is found (equation 6). $\hat{\boldsymbol{\theta}}$ is the unwrapped phase vector.

$$\hat{\xi}_{fine} = round\left(grad(\hat{\boldsymbol{\theta}}_i) \cdot 52 / 2\pi\right),\ 1 \le i \le 52 \tag{6}$$

Fig. 4 shows the implementation scheme of equation (5) and the unwrap algorithm. First, $\mathbf{Crx}_i/\mathbf{Ctrx}_i$ is performed with 2 multiplexers and 2 two's complementers, since the original Section C has been previously stored in Look-Up Table (LUT) based RAM and it is only composed of values 1 and -1. After that, phase difference $\boldsymbol{\theta}$ is obtained using a CORDIC (COordinate Rotation DIgital Computer) in vectoring mode [9]. The output range of the CORDIC is set to [-1, 1] in order to normalize equation (6) by π and so avoid operations involving π. Then, a very simple unwrap algorithm is implemented. Since initial time offset is always negative, we only have to check if difference between actual and previous phases is greater than 1 (equivalent to π). If this condition is met, a counter is increased and phase is unwrapped by 2.

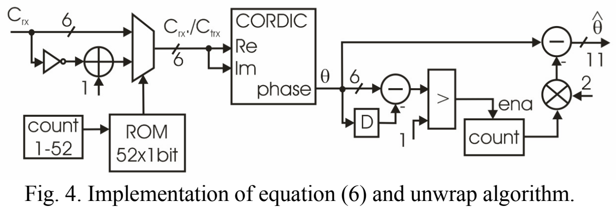

Fig. 4. Implementation of equation (6) and unwrap algorithm.

Finally, in Fig. 5 the implementation of equation (6) can be seen. To find the LS solution, the Vandermonde matrix **V** [10] is used. Equation (7) must be solved, where $a = a_0 + a_1 \cdot x$ is the straight line where $\hat{\boldsymbol{\theta}}_i$ gets the best fit to.

$$\mathbf{V}^T = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & 52 \end{bmatrix}, \mathbf{V}^T\mathbf{V}\cdot a = \mathbf{V}^T\cdot\hat{\boldsymbol{\theta}}_i,\ 1 \le i \le 52 \qquad (7)$$

$\mathbf{V}^T\,\mathbf{V}$ is a 2x2 constant matrix, and $\mathbf{V}^T\cdot\hat{\boldsymbol{\theta}}_i$ is a 1x2 vector which can be easily obtained (equation (8)). The term k_1 is obtained by adding the 52 unwrapped phases (an accumulator is needed), and the term k_2 can be obtained using a multiplier-accumulator. After 52 clock cycles, k_1 and k_2 are obtained.

$$\mathbf{V}^T\cdot\hat{\boldsymbol{\theta}}_i = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & 52 \end{bmatrix} \cdot \begin{bmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \dots \\ \hat{\theta}_{52} \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \qquad (8)$$

Equation (7) can be expressed as equation (9). The gradient a_1 can be obtained with two constant coefficient multipliers (M1 and M2) and one adder. The coefficients of the multipliers are λ_1= 0.85375e-4 and λ_2=0.22624e-2. Finally, the gradient is multiplied by 52/2 and the result is rounded.

$$\begin{bmatrix} 52 & 1378 \\ 1378 & 48230 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \rightarrow a_1 = k_2\cdot\lambda_1 + k_1\cdot\lambda_2 \qquad (9)$$

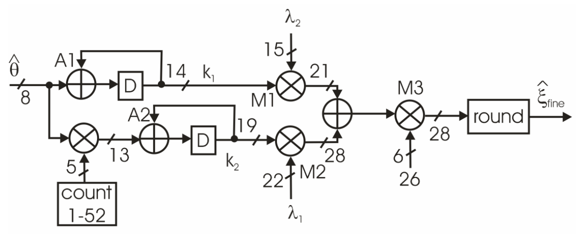

Fig. 5. Implementation of equation (6) .

The multipliers M1, M2 and M3 in Fig. 5 are efficiently implemented as wired multipliers by using only adders and hard-wired shifters. Fig. 6 shows implementation of multiplier M1, which only requires 2 adders. Multiplier M2 needs 4 adders and multiplier M3, 2 adders.

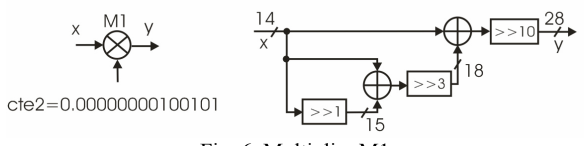

Fig. 6. Multiplier M1

As can be noticed, the hardware cost needed to get the ML gradient of the unwrapped phase in LS sense is quite high. For this reason, a simplification of the algorithm is proposed in equation (10). A straight line between the unwrapped phase of the first sub carrier and the unwrapped phase of the last sub carrier is considered. Fine time offset is obtained from the gradient of this line.

$$\hat{\xi}_{fine} = round\left(\left|\hat{\theta}_{52} - \hat{\theta}_1\right| / 2\pi\right) \qquad (10)$$

Fig. 7 shows the implementation of the modified algorithm. First, the difference between the first and the last sub carrier unwrapped phase is obtained. Then, the two's complement of this gradient is obtained because it is always negative. After that, the result is multiplied by ½ and rounded.

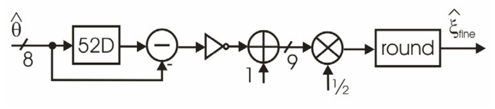

Fig. 7. Implementation of equation (10).

The main disadvantage of ISOCA based algorithm is its high latency. It uses the same information as the channel estimate, the received Section C in frequency domain. So, before channel estimation, it is necessary to correct the received Section C by the time correction given by the estimated fine time offset. This correction on the received Section C can be done using the same CORDIC which obtains the phase difference, $\boldsymbol{\theta}$.

4. CROSS-CORRELATION ALGORITHM

A time domain algorithm based on cross-correlation is presented in [11]. A correlation between the received Section C and the transmitted Section C is made in the time domain (equation (11)). $\mathbf{C}_{trx}$ is the transmitted Section C and $\mathbf{C}_{rx}$ is the received Section C. A magnitude peak is obtained, and its position indicates the time offset.

$$R(n) = \sum_{m=1}^{63} C^*_{trx}(m)\cdot C_{rx}(n+m) \qquad (11)$$

The cross-correlation can be implemented as a complex filter with coefficients $[C^*_{trx}(63)\ C^*_{trx}(62)\ C^*_{trx}(61) \dots C^*_{trx}(1)]$. The hardware cost of this complex filter (Fig. 8) is very high: 63 complex multipliers (189 real multipliers and 315 adders) and 62 registered complex adders (124 real adders).

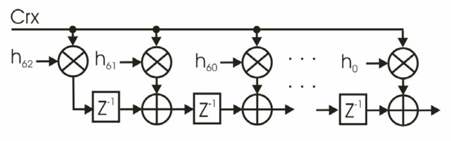

Fig. 8. Implementation of equation (11). Cross-correlation.

In [11] it is proposed to quantize the filter coefficients to {-1, 0,1} values for real and imaginary parts. This eliminates the use of multipliers, yielding a relatively low hardware complexity. In our implementation coefficients lower than 0.02 are quantized to 0. Performance of the cross-correlation process with quantized coefficients is nearly equal to ideal cross-correlation. Moreover, the number of coefficients can be reduced from 64 to 28 without degradation of performance.

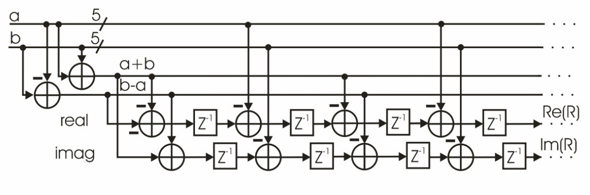

Fig. 9. Simplified cross-correlator.

Fig. 9 shows the cross-correlator implemented as a wired complex filter. The first five coefficients are 1+j, -1+j, -1, -1+j and -1+j. The input is C_{rx} = a+jb, so the first multiplier is replaced by the addition –(b-a)+j(a+b), the second multiplier is replaced by –(a+b)-j(b-a), and so on.

In Fig. 10 the implemented time offset estimator can be seen. The maximum deviation of coarse time synchronization is 12, so the search for fine time offset only occurs during the first 12 cycles. After 12 clock cycles, the time offset estimation can be read from the register D2.

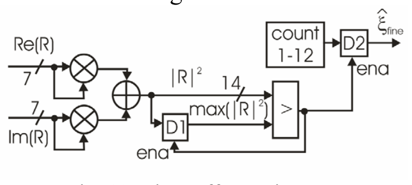

Fig. 10. Time offset estimator.

5. PERFORMANCE EVALUATION AND IMPLEMENTATION COST

The performance of all the algorithms is compared in Fig. 11, which shows deviation with respect to the ideal initial sample after applying the fine time synchronization algorithms (in fixed-point), for multipath channel A at SNR of 10dB and for 10000 test frames. The following algorithms have been evaluated: channel impulse response estimation method, cross-correlation with 63 coefficients, cross-correlation with 28 quantized coefficients {-1, 0, 1}, ISOCA algorithm and modified ISOCA algorithm. For clarity, the same results are shown in Table 1. For each algorithm, time offset estimation ξ must be pre-advanced by a different value in order to avoid ISI with the next OFDM symbol. For a multipath channel, a deviation between 4 and 6 samples can be assumed [6]. For all the algorithms, more than 99% of test frames are detected with a deviation of 0-3 samples.

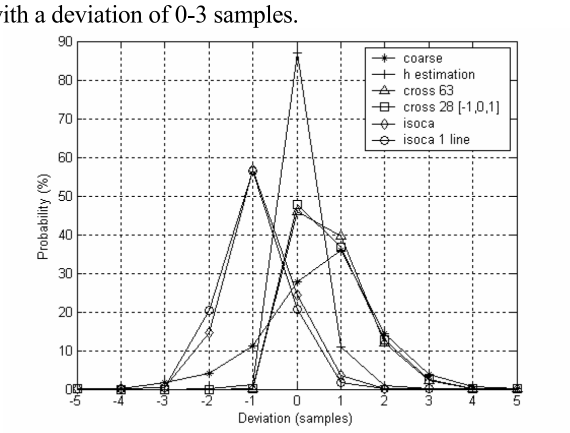

Fig. 11 Fine time algorithms performance.

Dev.	Coarse (%)	h (%)	Isoca (%)	Isoca2 (%)	Cr63 (%)	Cr28 (%)
<-3	1.77	0.03	0.58	0.59		0.01
-2	4.05	0.08	14.60	20.42		0
-1	11.28	1.09	56.68	56.46	0.01	0.05
0	27.76	87.07	24.47	20.60	45.91	47.74
1	35.85	10.85	3.51	1.82	39.6	36.84
2	14.43	0.84	0.16	0.1	11.93	12.8
3	3.82	0.03		0.01	2.3	2.42
>4	1.06	0.01			0.25	0.14
ξ		-2	-1	-1	-3	-3
(%)		99.85	99.26	99.3	99.74	99.8

Table 1. Fine time algorithms performance.

The fine time algorithms have been implemented on a Virtex-II Xilinx FPGA. These devices have slices (composed of LUTs to implement logic and arithmetic resources to propagate carries), embedded multipliers (MULTs) and Block Select RAMs (BSRAMs). In Table 2, the hardware resources needed by each algorithm can be seen. The proposed modifications to the ISOCA algorithm and the cross-correlation algorithm considerably reduce the hardware cost.

Algorithm	MULTs	BSRAMs	slices
h estimation	8	4	169
	8	0	339
ISOCA	1	0	413
ISOCA (straight line)	0	0	347
Cross-correaltion 63	65	0	1346
Cross-correlation 28	2	0	291

Table 2. Fine time algorithms implementation cost.

The three algorithms work perfectly for CFO equal to 1kHz and 10kHz, so fine time estimation can be made in parallel with fine CFO estimation in time domain algorithms, and no latency is added. On the other hand, ISOCA algorithm introduces a latency of 6.7μsec.

6. CONCLUSIONS

Three different algorithms for fine time synchronization have been studied, taking into account performance, hardware resources and latency. In terms of performance, all of them are quite similar, with perhaps the time impulse response algorithm performing the best. In terms of implementation cost, the best one is the cross-correlation algorithm using only 28 quantized coefficients. Finally, the ISOCA is the only one which introduces latency to the system.

REFERENCES

[1] ETSI TS 101475 v1.2.2, Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer, 2002
[2] IEEE 802.11a: Wireless LAN specifications: High-speed Physical Layer in the 5 GHz
[3] IEEE 802.11g: Wireless LAN specifications: Further Higher Data Rate Extension in the 2.4 GHz Band
[4] M.J. Canet, F.Vicedo, V.Almenar, J.Valls. "FPGA implementation of an IF transceiver for OFDM-based WLAN", IEEE Workshop on Signal Processing Systems, 2004
[5] BRAN WG3 PHY Subgroup. *Criteria for Comparison*. ETSI/BRAN document no. 30701F, 1998.
[6] J. Heiskala, J. Terry. OFDM Wireless LANs: A theoretical and practical guide. SAMS Publishing, 2001
[7] H. Minn, V.K. Bhargava, K.B. Letaief "A Robust Timing and Frequency Synchronization for OFDM Systems", IEEE Transactions on Wireless Communications. Vol. 2, No. 4, July 2003
[8] V. S. Abhayawardhana, Investigation of OFDM as a Modulation Technique for Broadband Fixed Wireless Access Systems. PhD thesis, May 2003.
[9] J.E. Volder, "The CORDIC Trigonometric Computing Technique", IRE Trans.Electronic Computers. Vol. EC-8, no 3,1959
[10] Biswa Nath Datta. Numerical Linear Algebra and applications. Brooks/Cole Publishing Company.
[11] R. Van Nee, R. Prasad. OFDM for Wireless Multimedia Communications. Artech House Publishers, 2000