IMPROVEMENT OF A TIME SYNCHRONIZATION ALGORITHM FOR IEEE 802.11a/g WLAN STANDARD

Mª José Canet, Vicenç Almenar, Santiago Flores, and Javier Valls

Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universidad Politécnica de Valencia
Ctra. Nazaret-Oliva s/n, 46730, Gandia, Spain
phone: + (34) 962849346, fax: + (34) 962849313, email: valmenar@dcom.upv.es
web: www.gised.upv.es

ABSTRACT

In this paper a time synchronization algorithm for IEEE 802.11a/g OFDM-WLAN standard is evaluated and some modifications are proposed to improve its performance. The original synchronization algorithm utilizes coarse and fine estimation. In this paper fine time estimation is done using a cross-correlation as the original algorithm does, but different solutions are evaluated to cope with the problems of the coarse estimation in the original algorithm. The performances of these alternatives are tested by simulation in multipath channels, at low signal to noise ratio and with carrier frequency offset. Also, the computational cost is evaluated.

1. INTRODUCTION

IEEE 802.11a is a WLAN standard from IEEE [1] that works in the 5 GHz and 2.4 GHz bands and achieves data rates up to 54 Mbps. For the physical layer, Orthogonal Frequency Division Multiplexing (OFDM) is used, since it allows getting high bit rates in highly dispersive fading environments. Data are transmitted in bursts, always preceded by a preamble (Figure 1). This preamble consists of ten identical short symbols (SS) of 16 samples and two identical long symbols (LS) of 64 samples with a cyclic prefix (GI) of 32 samples. As it is shown in Figure 1, once signal detection and automatic gain control (AGC) are completed (at sample n_i), fine and coarse time synchronization begin. The purpose of the time synchronization is to find the starting-point of GI (sample n_{GI}), so channel estimation can be correctly done using the long symbols, and a reference for the first OFDM symbol is obtained.

Several time synchronization methods for OFDM signals can be found in the literature, some of them make use of the preamble structure of the IEEE 802.11a/g, such as [2, 3, 4]. Among them, the algorithm proposed in [2] can achieve a good performance with a moderate computational cost. In this paper some modifications are introduced that improve its performance and reduce its computational cost.

This paper is organized as follows. Section 2 describes the time synchronization algorithm proposed in [2] and comments on its main drawback. Section 3 presents three modifications of the original algorithm to cope with the problem commented in Section 2. In Section 4 the performance of the different algorithms is analyzed and, in Section 5 the computational cost of every algorithm is given. Finally, in Section 6 conclusions are presented.

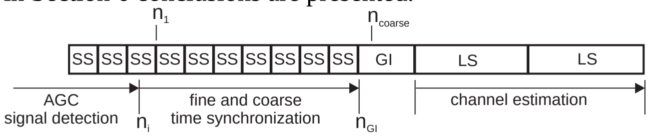
n1
ncoarse
SS SS SS SS SS SS SS SS SS SS
GI
LS
LS
AGC signal detection
fine and coarse time synchronization
channel estimation
ni
nGI

Figure 1 – IEEE 802.11a/g preamble

2. TIME SYNCHRONIZATION ALGORITHM

In this section the time synchronization algorithm for the IEEE 802.11a/g WLAN standard proposed in [2] is described. This is divided in two parts: one for coarse time estimation and another for fine time estimation. The performance of the coarse estimation is evaluated and its main drawback stated.

2.1 Fine Time Synchronization

Fine time synchronization tries finding the transition between two SS's (sample n_1 in Figure 1), once AGC and signal detection are accomplished. This task is achieved using a cross-correlation between the received signal $x(n)$ beginning at sample n_i and the pilot data $SS(k)$:

$$n_1 = \arg\max_{0\le n\le 15}\left(\sum_{m=0}^{Nreg}\left|\sum_{k=0}^{15} x(n+16m+k)\cdot SS^*(k)\right|\right). \quad (1)$$

This gives a peak value every 16 samples (the length of a SS), whose position indicates the starting point of each SS. An average of N_{reg} blocks of 16 samples is made to reduce distortion of peak positions due to channel noise. Parameter N_{reg} is calculated by the coarse estimation stage.

2.2 Coarse Time Synchronization

Using a maximum likelihood (ML) algorithm similar to [5], coarse time synchronization tries finding where the first sample of GI (n_{GI}) is. First, this metric is obtained:

$$f_{ML}(n) = \left|\sum_{k=0}^{15} x^*(n+k)\cdot x(n+k+16)\right| - \frac{\sigma_s^2}{2\left(\sigma_s^2+\sigma_n^2\right)}\sum_{k=0}^{15}\left[\left|x(n+k)\right|^2 + \left|x(n+k+16)\right|^2\right] \quad (2)$$

where σ_s^2 and σ_n^2 are signal and noise power respectively. As a result, a plateau, which falls abruptly when the guard interval (GI) of the long training sequence begins, is obtained. Next, the number of SS's between signal detection (sample n_i) and the first sample of GI (n_{GI}) is calculated, this value is the parameter N_{reg}:

$$N_{reg} = \left\lfloor \min_n \left\{ \arg\left[f_{ML,norm}(n) < Thr \right] \right\} \Big/ 16 \right\rfloor$$
$$f_{ML,norm}(n) = \frac{f_{ML}(n)}{\dfrac{\sigma_s^2}{2\left(\sigma_s^2+\sigma_n^2\right)} \sum_{k=0}^{15} \left[\left| x(n+k) \right|^2 + \left| x(n+k+16) \right|^2 \right]} \tag{3}$$

where Thr is a threshold value.

Once N_{reg} is known, the position of the first sample of GI is calculated as:

$$n_{GI} = N_{reg} \cdot 16 + n_1 \tag{4}$$

2.3 Performance evaluation

This synchronization method has been tested and the optimal threshold has been found. As a result, it is concluded that the fine time algorithm works correctly, even if the number of SS's used in (1) is as low as 4.

However, we have observed that the main problem of this synchronization method is the coarse time estimation. Although this has a good performance when it begins to work (instant n_i) at samples in the middle of a SS (in [2] it is assumed that AGC and signal detection are completed during the third SS), some problems arise when it begins to work at samples around the boundary of two consecutive SS's. In that case, the GI starting point (4) is often detected with an offset of ±16 samples (see Figure 2), giving a synchronization failure as a result. In this paper, to improve the synchronization success rate some modifications are proposed in the coarse estimation algorithm, meanwhile the original fine time estimation algorithm from [2] is maintained.

3. COARSE TIME ESTIMATION ALTERNATIVES

The problem commented in the previous section is due to the rounding to the lowest integer operation that is performed in (3), this is the cause of the ±16 samples time offset. The objective of (3) is to find where the transition between the last SS and the GI occurs. In this section three alternatives to (3) are evaluated, the first one is a modification of the original algorithm, the second one is based on a new metric, and the third one is obtained from the literature.

3.1 Alternative 1

Next we propose a way of circumvent the need of doing a rounding to the lowest integer. First, a coarse estimation (called n_{coarse} in Figure 1) of the beginning of GI is obtained:

$$n_{coarse} = \min_n \left\{ \arg\left[f_{ML,norm}(n) < Thr \right] \right\} \tag{5}$$

Then the last transition point (n_1+16, n_1+32, n_1+48,…) before n_{coarse} is found, since this will be the first sample of GI (n_{GI}).

Once the OFDM signal has been detected, the synchronization algorithm begins at sample n_i the next procedure:

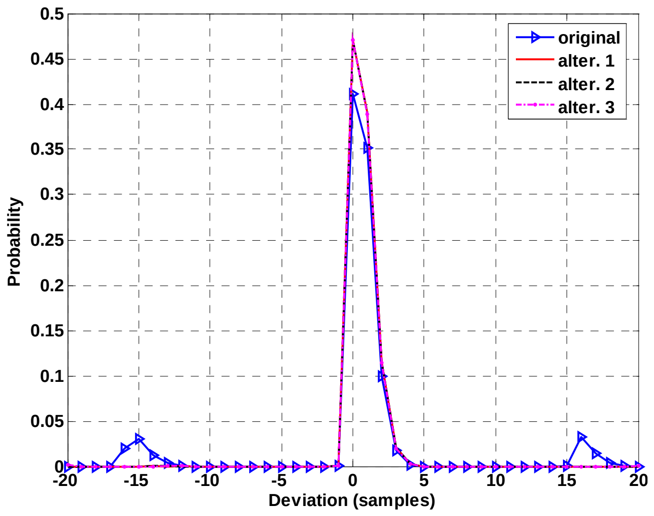

Figure 2 – Deviation with respect to the ideal initial sample

1) Metric (3) is calculated from n_i and during 128 samples to include the abrupt change that occurs at n_{GI}. Then n_{coarse} is estimated using (5). And q is set to 1.
2) The received signal is cross-correlated to obtain n_1 using (1), with $N_{reg} = q - 1$.
3) If $n_1 + q \cdot 16 > n_{coarse}$, then $n_{GI} = n_1 + (q-1) \cdot 16$; if not, then $q = q + 1$ and step 2 is repeated.

At every iteration, the number of terms in the average (N_{reg}) of (1) is incremented before n_1 is re-calculated, so the estimation of n_1 becomes more and more accurate.

3.2 Alternative 2

Next we present a new method for coarse time estimation, this makes use of a new metric and the iterative procedure introduced in Alternative 1. The new metric is calculated as follows:

$$f_2(n) = \frac{1}{4} \sum_{m=1}^{4} \left[\left| x(n) - x(n-16 \cdot m) \right|^2 \right]. \tag{6}$$

This calculates the squared difference between the actual SS and the four previous SS's, and finally these differences are added. Therefore, $f_2(n)$ will be around zero while samples from SS's are processed and it will grow abruptly when samples from GI start. To set a threshold to check if the abrupt change happens, it is necessary to normalize the new metric by the average input signal power:

$$f_{2\_norm}(n) = \sum_{k=0}^{15} f_2(n+k) \Big/ \sum_{k=0}^{15} \left| x(n+k) \right|^2 . \tag{7}$$

Next, n_{coarse} can be obtained as:

$$n_{coarse} = \min_n \left\{ \arg\left[f_{2\_norm}(n) < Thr \right] \right\}, \tag{8}$$

and, n_{GI} can be calculated by means of the same procedure as in Alternative 1.

3.3 Alternative 3

This algorithm is proposed in [3] for coarse time synchronization. It is based on the Neyman-Pearson detection approach [6] and this compares the probability that the received signal belongs to the short training symbol (P_{SS}) with the probability that it belongs to the cyclic prefix of the long symbol GI (P_{GI}). These probabilities are only evaluated at transitions points: $n_q = n_1 + 16 \cdot q$, with q = 1, 2, 3… The first time that the next condition:

$$\left(\mathbf{r}_{n_q}^{\mathrm{H}} \mathbf{G}_0 \left(\mathbf{G}_0^{\mathrm{H}} \mathbf{G}_0\right)^{-1} \mathbf{G}_0^{\mathrm{H}} \mathbf{r}_{n_q}\right) > \left(\mathbf{r}_{n_q}^{\mathrm{H}} \mathbf{B}_0 \left(\mathbf{B}_0^{\mathrm{H}} \mathbf{B}_0\right)^{-1} \mathbf{B}_0^{\mathrm{H}} \mathbf{r}_{n_q}\right) \quad (9)$$

is met means that n_{GI} has been found. In [3] it is shown that (9) is equivalent to $P_{GI}(n_q) > P_{SS}(n_q)$, where L is the estimated number of paths of the multipath channel, $\mathbf{r}_n = [r_n \;\; r_{n+1} \;\; \cdots \;\; r_{n+15}]^T$ is a signal vector with 16 samples from the received signal, and $\mathbf{G}_0$ and $\mathbf{B}_0$ are matrices of dimension 16 x L:

$$\mathbf{G}_0 = \begin{bmatrix} GI_0 & SS_{15} & SS_{14} & \cdots & SS_{16-L+1} \\ GI_1 & GI_0 & SS_{15} & \cdots & SS_{16-L+2} \\ GI_2 & GI_1 & GI_0 & \cdots & SS_{16-L+3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ GI_{15} & GI_{14} & GI_{13} & \cdots & GI_{16-L} \end{bmatrix} \quad (10)$$

$$\mathbf{B}_0 = \begin{bmatrix} SS_{\mathrm{mod}(n,16)} & SS_{\mathrm{mod}(n-1,16)} & \cdots & SS_{\mathrm{mod}(n-L+1,16)} \\ SS_{\mathrm{mod}(n+1,16)} & SS_{\mathrm{mod}(n,16)} & \cdots & SS_{\mathrm{mod}(n-L+2,16)} \\ \vdots & \vdots & \ddots & \vdots \\ SS_{\mathrm{mod}(n+15,16)} & SS_{\mathrm{mod}(n+14,16)} & \cdots & SS_{\mathrm{mod}(n-L+16,16)} \end{bmatrix}$$

Once the OFDM signal has been detected, the synchronization algorithm begins at sample n_i the next procedure:

1) Set q = 1.
2) The received signal is cross-correlated to obtain n_1 using (1) with $N_{reg} = q - 1$.
3) If $P_{GI}(n_q) > P_{SS}(n_q)$ (with $n_q = n_1 + 16 \cdot q$), then $n_{GI} = n_q$; if not, then $q = q + 1$ and step 2 is repeated.

4. SIMULATION RESULTS

The original algorithm and the proposed alternatives have been tested for BRAN channel model A, which represents a typical office environment with an rms delay spread of 50 ns, and channel model B which represents an open office environment with an rms delay spread of 100 ns [7]. The performance results consider the optimal threshold for each algorithm, and the optimal length channel (L) for Alternative 3. It is assumed that AGC and signal detection are completed at any sample of the third SS, so n_i is randomly set following a uniform distribution in the range 32-47. This assumption is the same as the one used in [2], as it was commented above we have checked that this algorithm works even if n_i happens during the sixth SS.

The success rate of the time estimation has been measured under the following conditions: 10^4 test frames, each one with a different realization of the multipath channel; carrier frequency offset (CFO) from 0 to 73% of the sub-carrier spacing (this is the maximum value allowed in [1]); and additive white Gaussian noise with a SNR from 2dB to 18dB.

Figure 2 shows the deviation from the ideal initial sample (n_{GI}) achieved with each algorithm. This has been measured with multipath channel A, 18dB of SNR and no CFO. As it has been mentioned above, the original algorithm [2] occasionally detects n_{GI} with an offset of ±16 samples. In contrast, the proposed alternatives do not have this problem, and all of them achieve more or less the same deviation. All the algorithms have the same MSE value in the deviation due to the fact that all of them use the same fine time algorithm. For multipath channel A the MSE value is around 0.65 samples, whereas the MSE value is around 1.55 for multipath channel B.

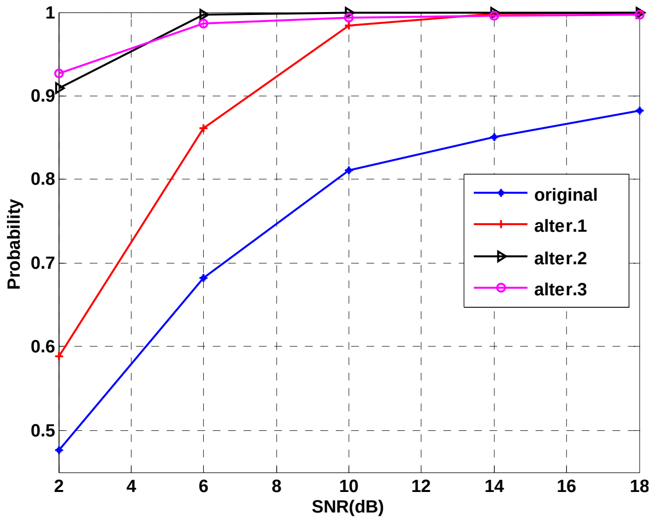

Figure 3 – Correct final synchronization. Channel A. CFO 0%

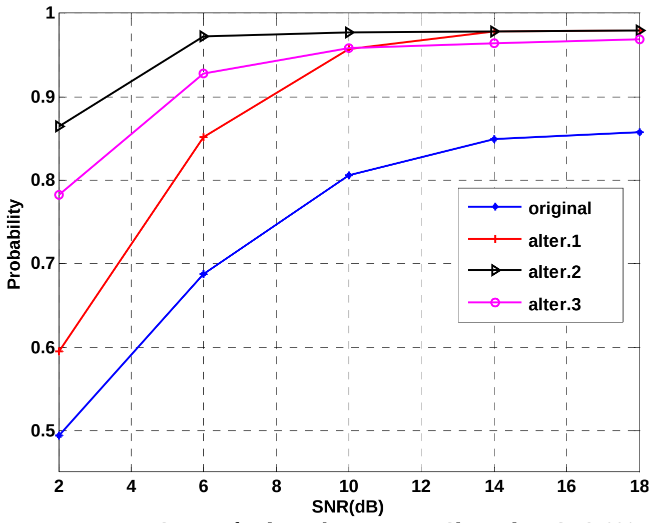

Figure 4 – Correct final synchronization. Channel B. CFO 0%

Figure 3 and Figure 4 show the probability of correct time synchronization for multipath channel A and B, respectively. No CFO is considered. In this paper it is considered that a frame is correctly synchronized when deviation from the ideal initial sample is between 0 and 4 samples, because a deviation lower than 4 samples does not introduce any performance loss [8].

Additionally, the effect of the CFO has been studied. The fine time synchronization algorithm works perfectly with CFO. Figure 5 and Figure 6 show the obtained results for a

CFO of 73% of sub-carrier spacing for channel A and channel B, respectively.

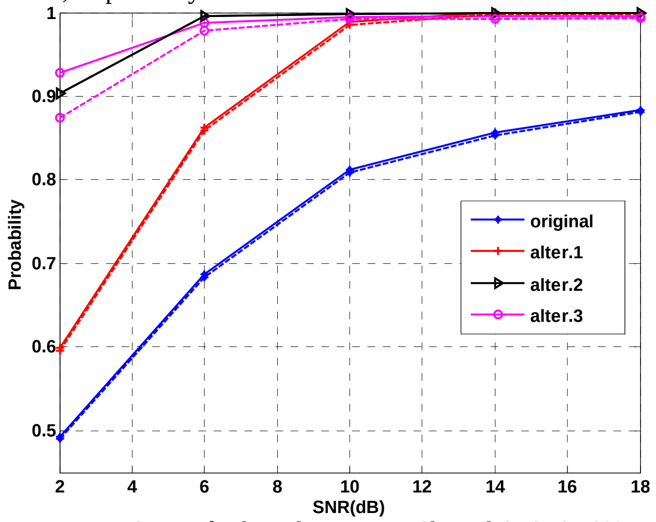

Figure 5 – Correct final synchronization. Channel A. CFO 73%, solid line CFO compensated, dashed line CFO uncompensated

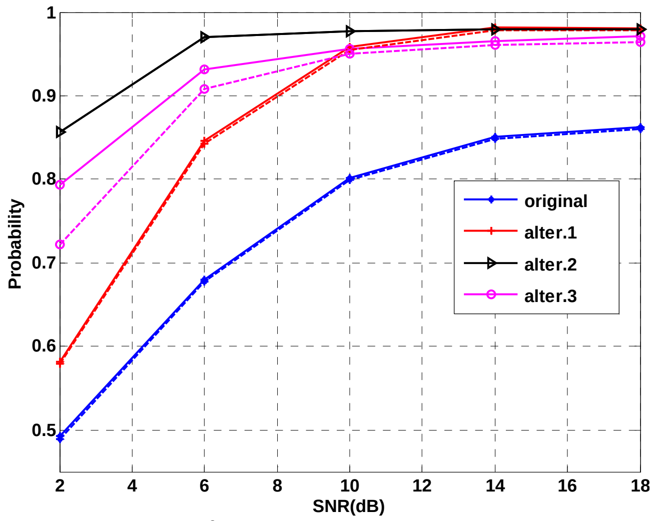

Figure 6 – Correct final synchronization. Channel B. CFO 73%, solid line CFO compensated, dashed line CFO uncompensated

On the one hand, the original algorithm and Alternative 1 have the same success rate with and without CFO. On the other hand, the success rate at 2dB SNR is reduced by 4% (channel A) or 8% (channel B) for Alternative 3. Finally, Alternative 2 does not work with CFO higher than 30 kHz, so CFO must be estimated and compensated before applying this synchronization method as it is shown in the figures.

The CFO estimation can be obtained by doing an autocorrelation as it is proposed in [9]:

$$z(n)=\sum_{k=0}^{K-1} x^{*}(n+k)\cdot x(n+k+16) \tag{11}$$

After that, the CFO ($\hat{f}$) can be estimated as:

$$\hat{f}=\frac{\angle z(n_{cfo})}{2\pi 16 T_s}, \tag{12}$$

where T_s is the sampling period and n_{cfo} is the sample where the estimated CFO is obtained, this instant must be greater than n_{i+K}. The average length (K) must be at least of 64 samples to have a standard deviation of the estimated CFO lower than 30 kHz. Apart from being necessary in Alternative 2, the success rate of Alternative 3 can be improved by compensating the CFO before applying the algorithm.

In summary, if CFO is compensated, the best algorithms are Alternative 2 and Alternative 3, although this last one has a lower success rate for channels with higher delay spread like channel model B.

5. COMPUTATIONAL COST

In this section the computational cost of every alternative has been evaluated. Alternative 3 makes use of two 16 x 16 matrices $\mathbf{G}_0\left(\mathbf{G}_0^{\mathrm{H}}\mathbf{G}_0\right)^{-1}\mathbf{G}_0^{\mathrm{H}}$ and $\mathbf{B}_0\left(\mathbf{B}_0^{\mathrm{H}}\mathbf{B}_0\right)^{-1}\mathbf{B}_0^{\mathrm{H}}$ that can be pre-calculated, but 544 complex products (i.e. 2176 real products and 1088 real additions) and 510 complex additions (1020 real additions) are needed to compute (9). If the algorithm for coarse estimation begins during the third SS, condition in (9) is evaluated 7 times; this gives a total complexity of 15232 real products and 14756 real additions. Moreover, this algorithm has an important disadvantage: the channel length L must be known a priori, that is, in a practical system this must be estimated.

Alternative 2 makes use of a division in (7), this can be avoided if condition in (8) is modified as follows:

$$\sum_{k=0}^{15} f_2(n+k) < Thr\cdot\sum_{k=0}^{15}\left|x(n+k)\right|^2 . \tag{11}$$

In that case the computational cost per input sample of this alternative is: 1 moving average (MA), 1 complex addition and 1 squared modulus in (6), and 2 MA, 1 product by a constant and 1 squared modulus in (11). Since each MA can be done with 2 real additions, and each squared modulus costs 2 real products and 1 real addition, this gives a complexity of 5 real products and 10 real additions. Moreover, as this algorithm does not work with CFO, it is necessary to estimate and remove it from the SS’s before the coarse time synchronization is carried out. The CFO compensation needs 1 complex product per sample, and then Alternative 2 has a computational cost per input sample of 9 real products and 12 real additions.

Again, if the algorithm begins during the third SS this means that (6) and (11) must be evaluated during 128 samples as it was commented in section 3.1. Therefore, the total computational cost is 1152 real products and 1536 real additions. It must be stated that after time synchronization the receiver must carry out channel estimation using the two long symbols from the preamble. Before this, the CFO must be removed; so, CFO always must be estimated, and for this reason it has not been taken into account in the previous computational complexity study.

Original algorithm and Alternative 1 make use of the same metric (2), the difference between them is how the coarse estimation is done. This metric has a higher computational cost than Alternative 2 because in (2) is computed 1 complex valued MA (4 real additions) and 1 real valued MA

(2 real additions), 2 squared modulus (4 real products and 2 real additions), 2 real scaling products one in (2) and another to avoid the division, a complex valued autocorrelation (1 complex product), 2 real additions and a modulus operation (it can be calculated with a CORDIC in vectoring mode, this can be implemented with $(N_{bits} + 2) \cdot 3$ additions [10], where N_{bits} is the number of bit precision needed at the output of the CORDIC, for a fixed point implementation without performance degradation a precision of 10 bits is necessary, then the CORDIC would employ 36 additions). Then, original algorithm and Alternative 1 have a computational cost per input sample of 10 real products and 12 real additions (and 36 real additions for a CORDIC of 10 bit precision), moreover it is necessary to estimate the signal and noise power. Again if this algorithm works during 128 samples, the total complexity is: 1280 real products and 1536 real additions (and 4608 real additions for the CORDIC).

Table 1 shows the computational complexity per input sample, and the total cost in brackets: Alternative 1 and 2 are calculated during 128 samples, Alternative 3 is calculated at 7 samples.

	Alt. 1	Alt. 2	Alt. 3
Real products	10 (1280)	9 (1152)	2176 (15232)
Real addition	12 +36 (1536 + 4608)	12 (1536)	2108 (14756)
Other	estimate σ_s^2, σ_n^2		estimate L

Table 1 – Comparison of computational cost

6. CONCLUSIONS

In this work several algorithms have been studied to solve the problem found in the coarse time estimation proposed in [2]: the starting-point of GI can be detected with an offset of ±16 samples if the synchronization process begins (once AGC and signal detection are accomplished) in samples around the boundary of two SS's. All the studied alternatives make use of the same fine time estimation algorithm as in [2].

Simulation results show that Alternative 2 and Alternative 3 have the best probability of correct detection. As Alternative 3 has higher computational cost and its performance decreases when the delay spread increases (like in channel model B), it is concluded that the best solution is Alternative 2 (compared to Alternative 3 this solution reduces the number of products in 92.4% and the number of additions in 89.6%). This solution needs the CFO to be estimated and removed before applying the synchronization algorithm.

ACKNOWLEDGEMENT

This work was supported by FEDER, the Spanish Ministerio de Educación y Ciencia under grants TEC2005-08406-C03-01 and TEC2006-14204-C02-01.

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] Sekchin Chang and B. Kelley, "Time synchronization for OFDM-based WLAN systems" *Electronics Letters* Vol.39, No.13, June 2003

[3] Yik-Chung, Kun-Wah Yip, Tung-Sang Ng, and Erchin Serpedin, "Maximum-Likelihood symbol synchronization for IEEE 802.11a WLANs in unknown frequency-selective fading channels", *IEEE Trans. on wireless communications*, vol.4, no.6, 2005.

[4] E.G Larsson, G. Liu, J. Li, and G.B. Giannakis, "Joint Symbol Timing and Channel Estimation for OFDM Based WLANs", *IEEE Commun. Letters*, 2005, 5, (8), pp. 325-327

[5] Beek, J.J., Sandell,M., and Börjesson, P.O.: "ML estimation of time and frequency offset in OFDM system", *IEEE Trans. Signal Process.*, 1997, 45, (7), pp. 1800-1805.

[6] S. M. Kay, *Fundamentals of Statistical Signal Processing, Vol.2: Detection.Theory*, New York: Prentice-Hall, 1998.

[7] J. Melbo and P. Schramm, "*Channel models for HIPERLAN/2 in different indoor scenarios*', 3ERI085B, HIPERLAN/2 ETSI/BRAN contribution, 1998

[8] J. Heiskala, J. Terry. *OFDM Wireless LANs: A theoretical and practical guide.* SAMS Publishing, 2001.

[9] T.Schmidl, and D.Cox. "Robust Frequency and Timing Synchronization for OFDM*". IEEE Trans. On Comm.* Vol 45, No. 12, December 1997.

[10] F. Angarita, T. Sansaloni, A. Pérez-Pascual, and J. Valls, "Efficient FPGA implementation of CORDIC algorithm for circular and linear coordinates", *International Conference in Field Programmable Logic and Applications* (FPL2005), Tampere, Finland, Ago. 2005