A PERFORMANCE AND COMPLEXITY COMPARISON OF AUTO-CORRELATION AND CROSS-CORRELATION FOR OFDM BURST SYNCHRONIZATION

Andrew Fort, Jan-Willem Weijers, Veerle Derudder, Wolfgang Eberle, André Bourdoux *

IMEC, Kapeldreef 75, B-3001 Leuven, Belgium
tel.: +32.16.281.754 Fax.: +32.16.281.515

ABSTRACT

A symbol timing synchronization scheme is critical in the design of an OFDM receiver. Large timing errors can result in a loss of orthogonality between subcarriers, ISI and severe bit error degradation. To minimize this degradation, standards incorporate preambles suitable for two kinds of synchronization algorithms: auto-correlation and cross-correlation. Unfortunately, the performance and complexity tradeoffs between these algorithms have not been well explored. To address this problem, we have built an FPGA implementation of a synchronization system using both auto-correlation and cross-correlation. Based on our results, in this paper we propose a novel cross-correlation synchronizer and hardware architecture. We then compare its performance and complexity to auto-correlation algorithms for HiperLAN/2 and IEEE 802.11a preambles.

1. INTRODUCTION

OFDM (Orthogonal Frequency Division Multiplexing) is an attractive transmission scheme for wireless communication systems because it is robust in multipath environments. However, OFDM systems are known to be sensitive to frequency and timing errors which result in a loss of orthogonality between subcarriers, ISI, and severe bit error degradation. We will focus on timing synchronization in this paper.

IEEE 802.11a and ETSI HiperLAN/2 standards incorporate preambles suitable for two kinds of synchronization algorithms: auto-correlation and cross-correlation. The auto-correlation algorithms have been studied extensively in [1, 2, 3], and implementation aspects are considered in [4]. This class of algorithms performs an auto-correlation of the received signal to detect a repeated symbol pattern. An OFDM cross-correlation algorithm has been proposed in [5, 6]. This class of algorithms performs a cross-correlation of the received signal with a known training symbol. Unfortunately, a cross-correlation detector suitable for IEEE and HiperLAN/2 has not been investigated, and a low-complexity correlator hardware architecture has not been explored. Therefore, there does not exist a performance versus hardware cost comparison of the auto-correlation and cross-correlation schemes making it difficult for designers to select which strategy is appropriate for their application.

To address this problem, we have developed a novel cross-correlation algorithm and hardware architecture suitable for both IEEE and HiperLAN/2. We implemented this algorithm on an FPGA together with an auto-correlation algorithm, and tested our system using HiperLAN/2 simulation models. Based on these results, in this paper we compare the performance together with the hardware complexity of both auto-correlation and cross-correlation detection. Section 2 reviews the auto-correlation algorithm. Section 3 presents our novel cross-correlation algorithm. Sections 4 and 5 compare their performance and hardware complexity. Finally, section 6 summarizes our results.

*This work is partly funded by the IWT / Medea+ A105 UNILAN project

2. THE AUTO-CORRELATION ALGORITHM

In IEEE and HiperLAN/2 systems, each transmitted burst consists of a preamble followed by regular OFDM symbols as shown in figure 1:

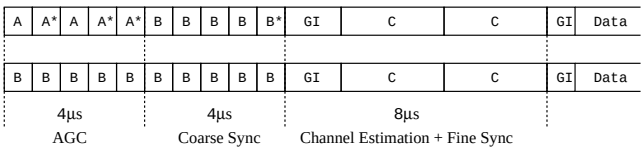

Fig. 1. HiperLAN/2 Broadcast preamble (top) and IEEE preamble (bottom).

For both standards, the first 4 μs are used for automatic gain control (AGC). The remaining B sequences can be used for packet detection, and coarse timing/frequency synchronization. The C sequences are used for channel estimation and synchronization refinement. The algorithm considered in this section uses the B sequences after the AGC settles.

All auto-correlation algorithms for IEEE and Hiper-

LAN/2 are based on the following function:

$$S(n) = \sum_{m=0}^{M-1} r^*(n+m-16)r(n+m) \qquad (1)$$

Where $r(n)$ is the received preamble sequence corrupted by the multipath channel and AWGN, 16 is the length of a B sequence, and M is matched to the length of the non-conjugate B sequences. If the channel length is less than 16 samples, then the magnitude of the noiseless part of the received preamble is periodic and $|S(n)|$ increases. A packet is detected when $|S(n)|$ exceeds a threshold.

For HiperLAN/2, the final B sequence is inverted so that $|S(n)|$ decreases for the last 16 samples creating a clear peak. Several detectors have been proposed to locate this peak, but we will use a max function since [4] suggests it is a good approximation of the Maximum Likelihood (ML) solution. Thus, the timing offset is estimated as follows:

$$\hat{n}_{acmax} = argmax_n(|S(n)|) \qquad (2)$$

The frequency offset can also be estimated for a sampling period T_s as follows:

$$\hat{f}_o = \frac{arg(S(\hat{n}_{acmax}))}{2\pi(16T_s)} \qquad (3)$$

A hardware structure for equations 1 and 2 is shown below. We have added a power normalization (the bottom path) as this makes the packet detection more robust to errors in the initial AGC stage.

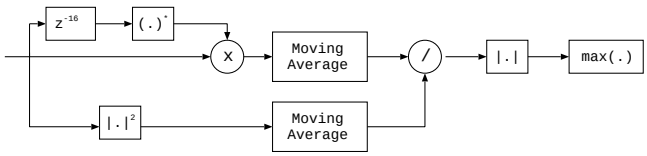

Fig. 2. Max Auto-correlation timing synchronizer structure.

For IEEE, the preamble does not have an inverted B sequence so the peak is less clear. Due to this drawback, we have found that some modifications to the algorithm in figure 2 are required in practice when front-end effects and the AGC are considered. A practical IEEE auto-correlation algorithm is the subject of a separate paper. For this analysis, we only consider the HiperLAN/2 preamble and the max function detector as this is sufficient for comparison with the cross-correlation algorithm.

3. THE CROSS-CORRELATION ALGORITHM

All cross-correlation algorithms are based on the correlation of the received signal with a known training sequence:

$$R(n) = \sum_{m=0}^{N-1} t^*(m)r(n+m) \qquad (4)$$

Where $t(m)$ is the transmitted training sequence of length N. The magnitude of the cross-correlator output consists of a large peak for each multipath component, and several small peaks due to AWGN and imperfect auto-correlation properties of $t(m)$ (see figure 3).

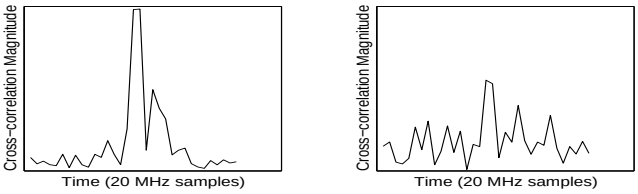

Fig. 3. Cross-correlation function for no frequency offset (left) and 200 kHz frequency offset (right).

While the training symbols can be chosen from any part of the preamble, the C sequence is preferred so that the B sequences can be used for an initial frequency correction. The left plot of figure 3 shows the cross-correlation output for the C sequence corrupted by a multipath channel at an SNR of 3 dB. The right plot shows the same conditions except we also apply a 200 kHz frequency offset. Clearly, the frequency offset reduces the size of the channel peaks complicating their detection. To minimize this problem, our simulations indicate that the auto-correlation algorithm can be used to correct the frequency offset to within 50 kHz by applying equation (3) to the B sequences. This is sufficient accuracy for robust cross-correlation with the C sequences.

Two cross-correlation timing detectors have been presented in literature, and we propose a third:

- XC SUM
  $\hat{n}_{xcsum} = argmax_n(\sum_{p=0}^{L} |R(n+p)|)$ [5]
- XC MAX
  $\hat{n}_{xcmax} = argmax_n(|R(n)|)$ [6]
- XC PROPOSED
  $\hat{n}_{xcp} = argmin_n(n) \mid |R(n)| \geq th \times |R(\hat{n}_{xcmax})|$

The XC Sum algorithm sums consecutive correlator outputs to locate a window of length L where the channel has the most energy. L is ideally the channel length. The XC Max algorithm selects the peak with the largest magnitude. Our proposal selects the earliest peak with a magnitude greater than some percentage of the largest peak. This improvement tends to select the first multipath component rather than later reflections thus reducing the variance of the timing estimate. The threshold must be chosen large enough to avoid selecting small noise peaks, but small enough to avoid selecting late multipath peaks. The performance of each algorithm is compared together with the auto-correlation algorithm in section 4.

The hardware complexity is dominated by the computation of equation 4 rather than the peak detectors. Figure 4 demonstrates two possible correlator structures:

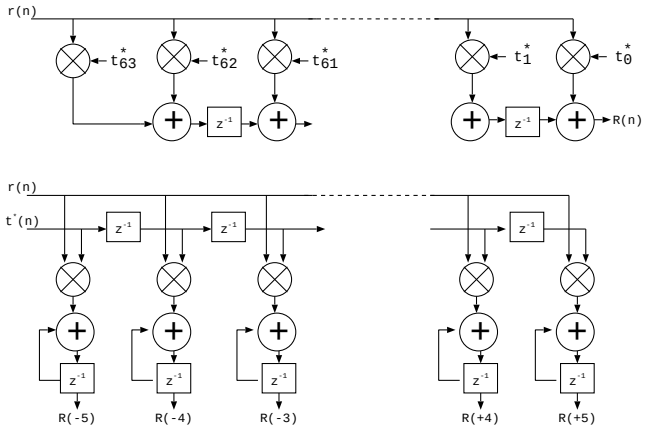

Fig. 4. Serial correlator (top) Parallel correlator (bottom).

The top structure shows the classic implementation with N complex taps. We propose the bottom structure having one complex multiplier and accumulator for each correlation lag. Since the cross-correlation must be preceded by an auto-correlation detector to correct the frequency offset, the same detector can also make a coarse timing estimate to reduce the number of lags that need to be computed making the second option more attractive. The complexity of both structures are compared in section 5 together with the auto-correlation structure from figure 2.

4. PERFORMANCE COMPARISON

We compared the auto-correlation and cross-correlation algorithm performance using the following simulation chain:

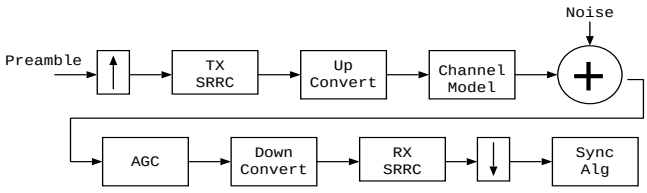

Fig. 5. Simulation model.

Each preamble was up-sampled, up-converted, corrupted by HiperLAN/2 Channel Model A [7] and AWGN, quantized with an AGC model, down-converted, and down-sampled before being applied to the detector. We used channel model A (delay spread = 50 ns) since it represents a typical indoor office environment, and 41 tap SRRC filters to include the impact of the transmit and receive filters. Figure 6 shows simulation results averaged over 5000 channels comparing the estimation variance versus SNR performance for each algorithm.

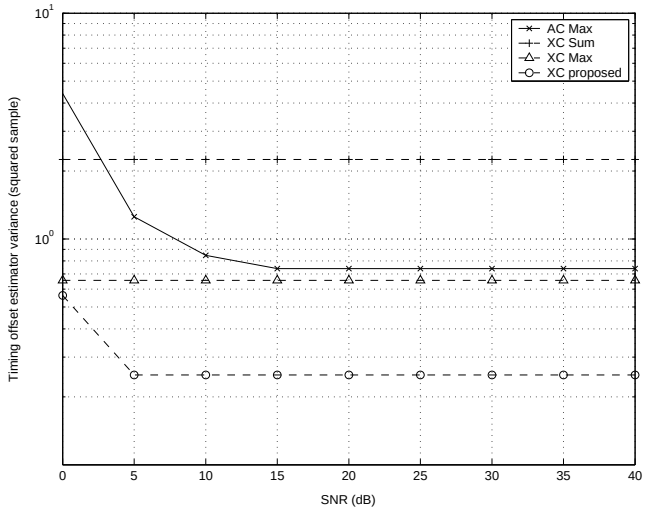

Fig. 6. Timing offset estimator performance comparison.

The auto-correlation algorithm (solid line) yields large timing errors at low SNRs as the noise complicates peak detection. However, this may not be a problem since IEEE and HiperLAN/2 systems typically operate above 5 dB SNR. At higher SNRs, a variance floor is reached due to the multipath channel. This floor corresponds to a symbol timing error between -2 and +2 samples 99% of the time.

The cross-correlation algorithms (dotted lines) provide better performance at low SNRs due to the averaging process of the correlator. The XC sum algorithm (L=5) did not work because many small channel reflections were indistinguishable from the noise making the energy estimate unreliable. The maximum peak detector (XC max) provides better performance similar to the auto-correlation. For our XC proposal, a 50% threshold was used for SNRs greater than 5 dB, but an 80% threshold was needed to avoid selecting noise peaks at very low SNRs. Only a single threshold is needed in practice since SNRs lower than 5 dB are not encountered for IEEE and HiperLAN/2 systems. This algorithm yielded the best performance since it is more robust to channel reflections. The variance of our proposal at high SNRs corresponds to a timing offset error between 0 and +2 samples 99% of the time.

In summary, our cross-correlation algorithm outperforms the auto-correlation algorithm. When we repeated these simulations for 200 kHz frequency offsets and channel model C (delay spread = 150 ns), we reached the same conclusion if the cross-correlator is preceded by a frequency correction. The auto-correlator was accurate to between -4 and +4 samples, while our cross-correlator was accurate to between 0 and +4 samples 99% of the time.

5. HARDWARE COMPLEXITY COMPARISON

Table 1 summarizes the hardware complexity of the auto-correlation algorithm from figure 2, and both correlator structures from figure 4. We added figures for a Fast Fourier Transform (FFT) and ML channel estimator for comparison:

hardware structure	*multi-pliers*	*add/ subs*	*regi-sters*	*word length*
AC	9	8	224	4
Serial XC	192	190	94	6
Parallel XC	44	44	44	6
Radix-2 FFT	12	24	128	10+
Channel est.	98	294	294	10+

Table 1. Hardware complexity

Figure 2 was used to estimate the complexity of the auto-correlation algorithm. It consists of 1 complex multiplication, 16 complex registers for the delay line, 2 real multipliers for each absolute value, and 48 complex registers and 2 complex adders for both moving average blocks. The normalization can be accomplished with a single multiplier since the result is only needed for a comparison. Note that in table 1 we assume that low delay complex multipliers consist of 4 real multipliers and 2 real adders, and complex registers consist of 2 real registers. Our simulations indicate that the input signal can be quantized to 4 bits without significant performance loss.

Figure 4 was used to estimate the complexity of both cross-correlators. For the serial correlator, our simulations indicate that only 48 of the 64 C sequence samples are required for robust performance above 5 dB SNR. Thus, this architecture requires 48 complex multipliers, and 47 complex adders and registers. For the parallel implementation, our performance results from section 3 show that above 5 dB SNR, the auto-correlation algorithm can provide a coarse timing estimate accurate to within ± 5 samples. Thus, 11 correlation lags need to be computed corresponding to only 11 complex multipliers and adders, and 22 registers for the accumulators and delay line. Our simulations indicate that all the complex multipliers can be quantized with 4x6 bit inputs. We chose the parallel structure for our FPGA implementation since it required less dedicated multiplier units. Note, however, that the serial structure allows efficient implementation of the constant multiplications and therefore may be more suitable for an ASIC.

While our proposed parallel architecture uses less operators than the serial architecture, table 1 indicates that a cross-correlation algorithm still requires at least 5 times more computation than the auto-correlation algorithm. On the other hand, a cross-correlator implementation is still feasible because of the short word lengths. For example, the size of a multiplier is proportional to the square of the input word lengths, so our correlator complexity is comparable to the 64-point FFT from table 1. Similarly, high performance OFDM systems are still be dominated by the channel estimator rather than the synchronizer.

6. CONCLUSIONS

In conclusion, our proposed cross-correlation algorithm out-performs the auto-correlation algorithm for IEEE and HiperLAN/2 system specifications. Our simulations show that for indoor environments, our cross-correlator detector is accurate to within 0 to 4 samples 99% of the time, while the auto-correlator detector is only accurate to within -4 to +4 samples 99% of the time. This performance improvement comes at a significant hardware cost. While our proposed correlator implementation requires less hardware operators than the classic correlator, it is still at least 5 times more complex than the auto-correlation algorithm.

7. REFERENCES

[1] Timothy M. Schmidl and Donald C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, 1997.

[2] J.J. van de Beek, M. Sandell, and P. Ola Böjesson, "ML estimation of timing and frequency offset in OFDM systems," *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1800–1805, 1997.

[3] H. Minn and V. K. Bhargava, "A simple and efficient timing offset estimation for OFDM," in *Vehicular Technology Conference Proceedings*, Tokyo, Japan, 2000, vol. 1, pp. 51–55.

[4] Stefan Johansson, Martin Nilsson, and Peter Nilsson, "An OFDM timing synchronization ASIC," in *IEEE Electronics, Circuits, and Systems Conference Proceedings*, 2000, vol. 1, pp. 324–327.

[5] L. Házy and M. El-Tanany, "Synchronization of OFDM systems over frequency selective fading channels," in *Vehicular Technology Conference Proceedings*, Pheonix, Arizona, 1997, vol. 3, pp. 2094–2098.

[6] Fredrik Tufvesson, Ove Edfors, and Mike Faulkner, "Time and frequency synchronization for OFDM using PN-sequence preambles," in *Vehicular Technology Conference Proceedings*, 1999, vol. 4, pp. 2203–2207.

[7] J. Medbo and P. Schramm, "Channel models for HiperLAN/2 in different indoor scenarios," Technical Report 3ERI085B, ETSI BRAN, 1998.