Canet Subiela, MJ.; Valls Coquillat, J.; Almenar Terre, V.; Marín-Roig Ramón, J. (2012). FPGA implementation of an OFDM-based WLAN receiver. Microprocessors and Microsystems. 36(3):232-244. doi:10.1016/j.micpro.2011.11.004.

FPGA implementation of an OFDM-based WLAN receiver

Authors:

María José Canet, Javier Valls, Vicenç Almenar and José Marín-Roig

Affiliation:

Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universidad Politécnica de Valencia, Universidad Politécnica de Valencia. Ctra. Nazaret-Oliva s/n, 46730 Gandia, Spain

macasu@iteam.upv.es

Abstract:

This paper deals with the design and implementation on FPGA of a receiver for OFDM-based WLAN. The circuit is particularized for IEEE 802.11.a,g standards. The system includes frame detection, time and frequency synchronization, demodulation, equalization and phase tracking. The algorithms to be implemented for each task are selected taking into account performance, hardware cost and latency. Also, a fixed point analysis is made for each algorithm. Our objective is to maintain the PER loss below 0.5 dB for a PER = 10^{-2}, 64-QAM and error correction. The whole system is composed of two main blocks (correlator and CORDIC) that are reused in different time intervals to perform all the necessary operations, so the required hardware resources are minimized. To verify it, the receiver is physically implemented and tested.

Keywords:

WLAN receiver, OFDM, synchronization, FPGA.

I. INTRODUCTION

IEEE 802.11a/g are WLAN standards from IEEE [1], [2], which work in the 5 GHz and 2.4 GHz bands and achieve data rates up to 54 Mbps. In the physical layer, Orthogonal Frequency Division Multiplexing (OFDM) was selected as the modulation scheme due to its good performance on highly dispersive channels, like the indoor scenarios where these standards are used. Data are transmitted in bursts, always preceded by a preamble (Fig. 1). This preamble consists of ten identical short symbols (SS) of 16 samples and two identical long symbols (LS) of 64 samples with a guard interval (GI) of 32 samples. It is used for time and frequency synchronization and channel estimation. At the receiver, once signal detection and automatic gain control (AGC) are completed (at sample n_i in Fig. 1), time synchronization begins: its purpose is to find the starting point

of GI (sample n_{GI}). So, a time reference is obtained and channel estimation can be correctly carried out by using the LS, once carrier frequency offset (CFO) is corrected.

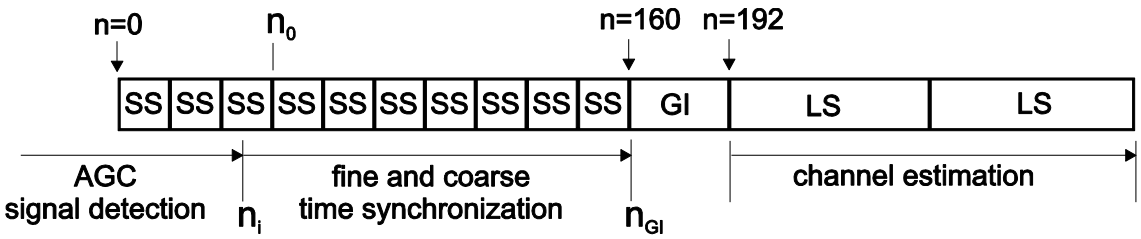

Fig. 1. IEEE 802.11a preamble.

Fig. 2 shows the block diagram of the WLAN base-band transceiver. In WLAN standards the base band OFDM signal is built using a 64-point Inverse Fast Fourier Transform (IFFT). Then a guard interval (also called cyclic prefix) of 16 samples is added to make the system robust to multipath and to prevent Inter-Symbol Interference (ISI) from happening, this prefix can also be employed to have some tolerance of symbol timing errors. Since the frequency sampling is 20 MHz, each symbol is 4 μs length (80 samples), including a guard interval of 800 ns. To facilitate implementation of filters and to achieve sufficient adjacent channel suppression, only 52 subcarriers are used: 48 are data carriers (with modulations types from BPSK to 64-QAM) and 4 are pilots for phase tracking. This makes that the subcarrier spacing is 312.5 kHz, and the spacing between two outmost subcarriers is 16.25 MHz.

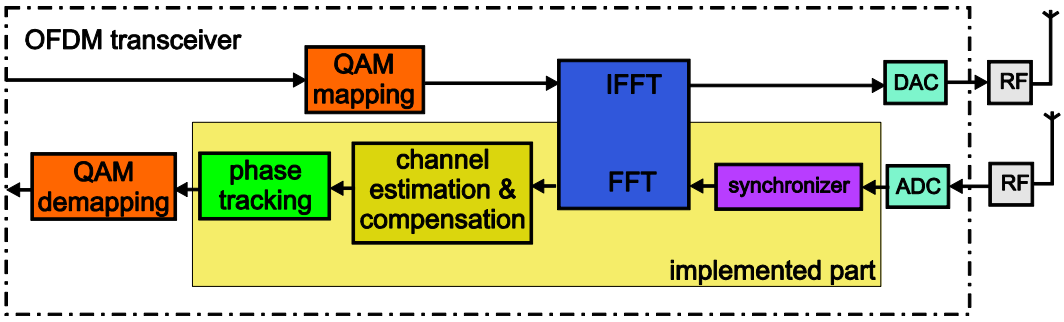

Fig. 2. WLAN transceiver.

The following stages are applied to the base-band received signal (see Fig. 3): frame detection, time synchronization, coarse and fine CFO estimation and correction, OFDM demodulation based on Fast Fourier Transform (FFT), channel estimation and compensation, and phase tracking.

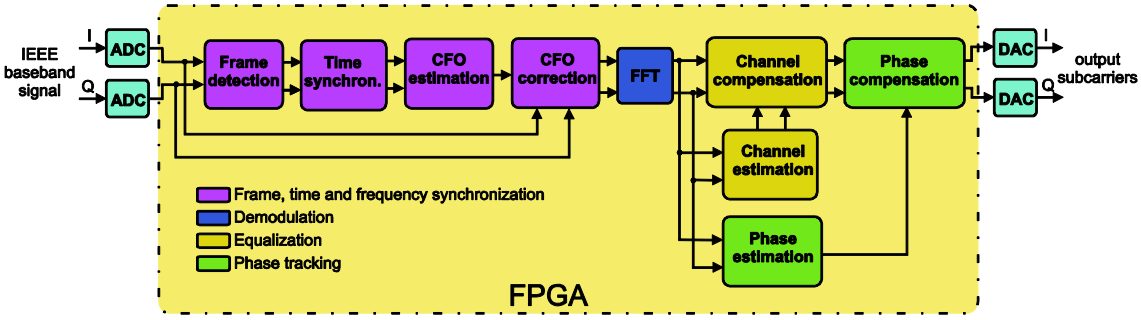

Fig. 3. Receiver structure.

An important fact to take into account in the receiver design is that there is a latency limitation given by the Short Inter-Frame Space (SIFS) in IEEE 802.11a/g [1]. Therefore, the main parameters to be considered in the design or selection of synchronization, channel estimation and phase tracking algorithms are not only performance and hardware cost, but also latency. We placed more emphasis on the synchronizer design because the performance of the receiver strongly depends on it. Our proposal is compared with other synchronization algorithms found in the literature ([3], [4], [5], [6]). The algorithms selected for channel estimation and phase tracking are simple in order to minimize the hardware cost and also the receiver latency. In any case, the implemented receiver achieves the performance required by the standard (signal to noise ratio – SNR at a 10% Packet Error Rate (PER) defined in [1] for different transmission modes) and also outperforms previous solutions found in the literature ([7], [8], [9]), as will be shown in Section VI.E. This performance was achieved thanks to the design flow followed, which minimizes the implementation losses.

The implementation of the receiver was done according to the following design flow: first, a floating-point simulator of the IEEE 802.11a/g physical layer without the synchronization stage (ideal synchronization is supposed) was used to obtain the PER performance of the ideal receiver. Then, floating-point models of the proposed algorithms for synchronization, demodulation, equalization and phase tracking were added to this simulator and the PER plot was checked. After that, a finite precision model of each algorithm was designed and a fixed-point analysis of the complete receiver was made: the data path of the proposed receiver (Fig. 3) was quantified, starting at the receiver input. At each stage we selected the minimum number of bits that guarantees a PER loss lower than 0.5 dB with respect to the ideal receiver, for a PER = 10^{-2}, modulation of 64-QAM including the error correction stage (Viterbi with Channel State Information (CSI) [10]). Next, the fixed-point receiver was designed using HDL (Hardware Description Language) and its output was compared to the fixed-point simulator output. Finally, the receiver was implemented on a prototype board connected to a logic analyzer, and its output was also compared to the fixed-point simulator output.

An important contribution of this work is how the hardware is reused in order to minimize hardware cost. In fact, the proposed receiver architecture is based on the reuse of the next blocks in different time intervals: the COordinate Rotation DIgital Computer (CORDIC), the FFT and the correlator.

This paper is organized as follows. Section II presents the proposed synchronization algorithm and it is compared to other algorithms found in the literature. Sections III, IV and V describe the FFT processor, the channel estimation algorithm and the phase tracking algorithm, respectively. Section VI is devoted to the architecture proposed for the complete receiver, including a finite-precision analysis, a diagram of the temporal use of the hardware resources, the description of the receiver test, the performance results and a comparison with other receivers found in the literature. Finally, in Section VII, conclusions are derived.

II. FRAME, TIME AND FREQUENCY SYNCHRONIZATION

In this section the structure selected for the synchronizer is described. First, the IEEE preamble must be detected. This is done by means of an auto-correlation of the SSs and, as a result, a coarse time reference is obtained. This coarse time estimation is not precise enough to achieve the desired performance in multipath channels and at low SNR, so a fine time synchronization algorithm was applied: a cross-correlation between the received signal and the known training preamble LS. This cross-correlation does not work properly in presence of CFO, so before calculating it, CFO is estimated and compensated from the LS. Next, the algorithm proposed for each task (frame detection and coarse time synchronization, frequency synchronization and fine time synchronization) will be described, 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.

A. Frame detection, coarse time and frequency synchronization

Frame detection is the first step in the synchronization process. After that, it is necessary to estimate a time reference: $\hat{n}_{GI}$. This estimation must be as accurate as possible to avoid ISI with previous or later symbols when the FFT window is taken. Positive time offsets cause ISI because the FFT window takes samples of the next symbol. Moreover, some samples of the CP are affected by multipath channel, analog filtering required in transmission and reception, and interpolate and decimate filters [11]. A common design rule [12] is to accept that the synchronization is correct if deviation from the ideal initial sample n_{GI} is between 0 and -4 samples, negative offsets higher than 4 samples can cause ISI in channels with moderate to high delay spread.

As stated above, the proposed algorithm is divided into two parts: coarse and fine time synchronization. Coarse synchronization is an adaptation of the auto-correlation method proposed by Schmidl and Cox [13]. The received signal is auto-correlated with a delay of 16 samples and averaged during 144 samples: $R_n = \mathbf{r}_n^{\mathrm{H}} \mathbf{r}_{n+16}$ where $\mathbf{r}_n = [r_n \quad r_{n+1} \quad \cdots \quad r_{n+143}]^T$ is a vector with 144 samples from the received signal. As a result, a single peak is obtained at sample n = 160 where the transition between the last SS and the GI is placed (see Fig. 4).

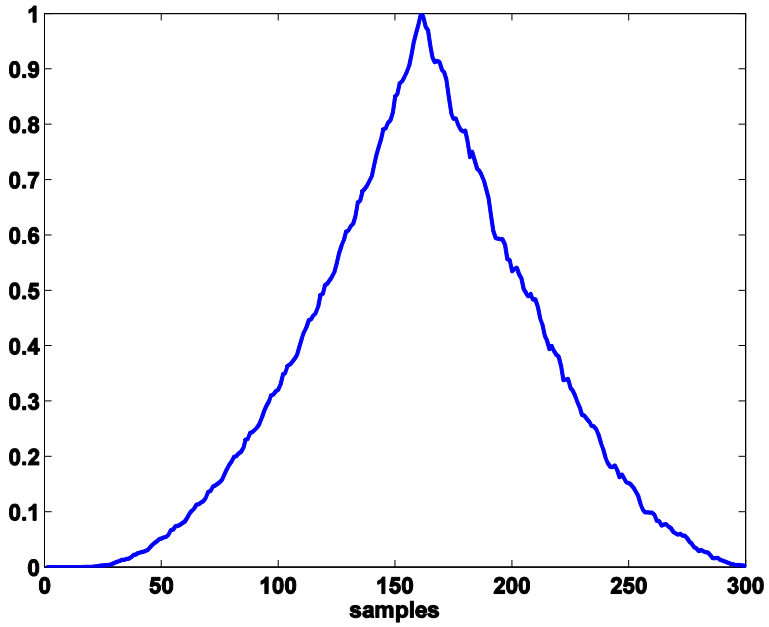

Fig. 4. Output of the auto-correlator $|R_n|^2$.

Next, the modulus of the auto-correlation output is normalized by the local mean power, $P_n = \|\mathbf{r}_n\|^2$, of the received signal. The timing metric is defined as [13]:

$$\overline{R}_n = |R_n|^2 / P_n^2 \quad . \tag{1}$$

Also, $|R_n|^2$ is averaged during 5 samples to improve the estimation of the maximum. To show this we calculated the probability of detecting the maximum outside a range of ±2 samples. The SNR required to maintain this probability below 10^{-2} is reduced from 13.25dB to 10.5dB thanks to the average. After the average, a threshold *Thr* is set in order to find the position of the peak ($\hat{n}_{GI}$), which is obtained by searching the maximum of the averaged $|R_n|^2$ between those samples that fulfill the condition $\overline{R}_n > Thr$. In order to circumvent the costly division operation, the next condition was used in the implemented design:

$$|R_n|^2 > \left(P_n^2 \cdot Thr^2\right). \tag{2}$$

Fig. 5 shows the block diagram of the proposed coarse time synchronization algorithm, which can be directly mapped in a VLSI implementation. Also, a maximum detection algorithm with low cost and low latency was implemented: present and previous samples are compared during two clock cycles. If the present sample is lower than the previous sample during two clock cycles, the maximum is considered to have been found.

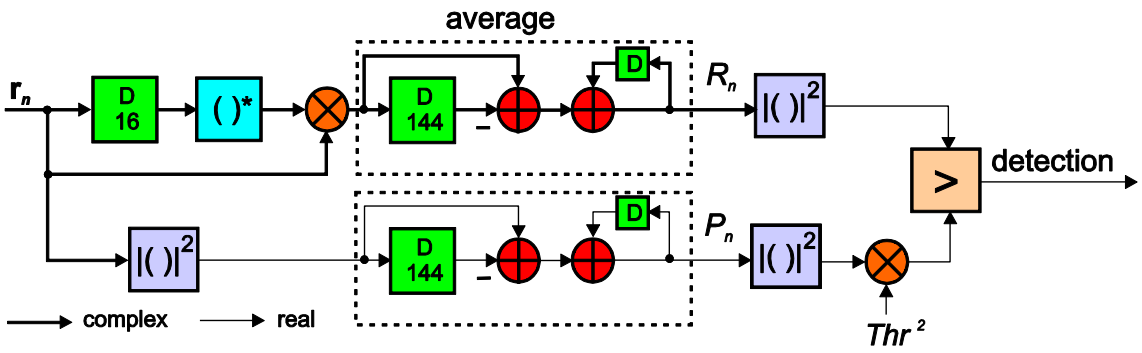

Fig. 5. Block diagram of the proposed coarse time synchronization algorithm.

The performance of this algorithm strongly depends on the threshold selection, so a theoretical analysis of the algorithm was made: the probability distributions of $\overline{R}_n$ were used to set a threshold which minimizes the probability of not detecting the beginning of the GI [14]. We fixed the threshold to 0.4375 at SNR equal to or higher than 6 dB, which guarantees that the probability of not detecting the beginning of the GI is lower than $1x10^{-3}$. Additionally, the multiplier needed in (2) can be efficiently implemented as shown in Fig. 6.

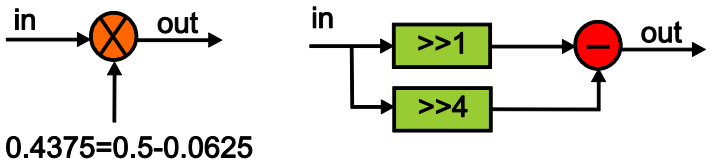

Fig. 6. Multiplier *Thr*^{2}.

Fig. 7 shows the deviation error of the estimated $\hat{n}_{GI}$ with respect to the ideal point n_{GI} at a SNR of 6 dB in a multipath channel. Three BRAN channel models were used [15]: A, B and C with an RMS (Root Mean Square) delay spread of 50 ns, 100 ns and 150 ns, respectively. This figure plots the relative frequency of deviation for 10^4 test frames (each one transmitted through a different channel realization for each channel model). The minimum and maximum deviation hardly changes for the tested channel models; in fact, more than 99.9 % of the frames were detected within a deviation from n_{GI} between −4 to +15 samples. This deviation range necessitates the use of a fine time synchronization algorithm.

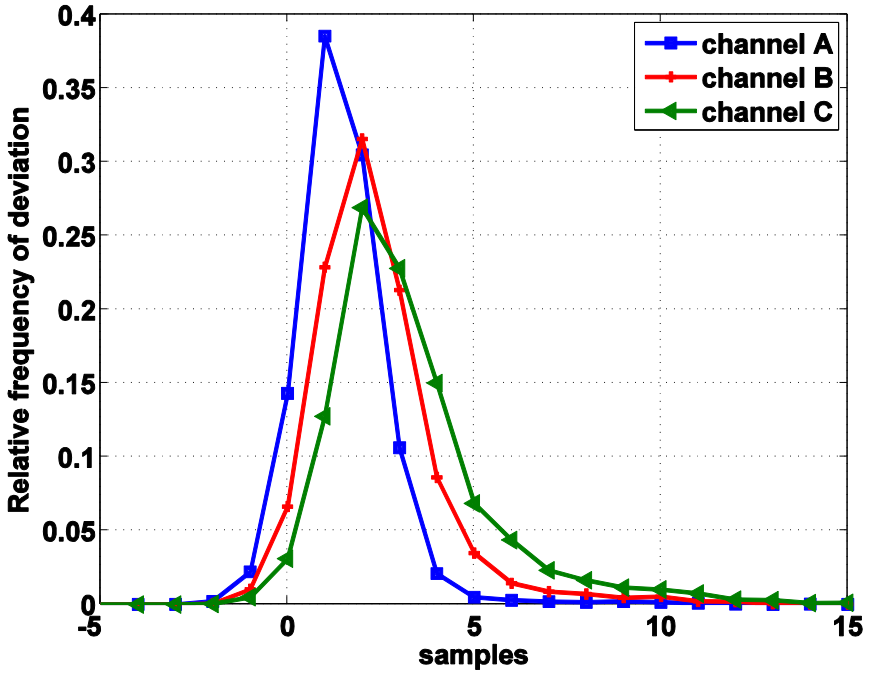

Fig. 7. Probability of the detection error for channel A (continuous line), B (dashed) and C (dotted).

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

$$\hat{f} = \frac{\angle R_n}{2\pi 16 T_s} , \tag{3}$$

where T_s is the sampling period. We confirmed that this coarse synchronization algorithm works without any degradation 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 [12], is not necessary and, as a result, the final latency of the synchronizer is considerably reduced.

The angle of the autocorrelator output ($\angle R_n$) is obtained by using the CORDIC in [16]. In [16], the CORDIC is optimized to be used in OFDM-based WLAN designs, so it can be reused in different time intervals with different operation modes. Also, the CORDIC output is normalized to [-1,1[, so division by 2π in (3) is avoided. The estimated CFO ($\hat{f}$) is used for the correction of the CFO from LS samples (previous to fine time synchronization) and later, for the correction of the CFO from the received OFDM symbols; this is carried out reusing the same CORDIC.

As the correlator and the CORDIC are reused for several tasks, the finite precision analysis results for both blocks are discussed in Section VII, where the complete system design is described.

After frequency synchronization, there is a residual CFO that continuously rotates the phase of the received OFDM signal and causes a constellation rotation [12]. After only a few symbols, the constellation points have just rotated over the decision boundaries, thus correct demodulation is no longer possible. This effect forces the receiver to track the carrier phase each time a new OFDM symbol arrives. This will be discussed in Section V.

B. Fine time synchronization

The proposed algorithm for fine time synchronization is based on a cross-correlation between the received signal and the known long training symbol (LS). The LS is quantized to values {-1, 1} for the real and the imaginary parts to avoid the use of multipliers, so the cross-correlation is efficiently implemented as a wired complex filter similar to [17]. We evaluated the minimum length of this cross-correlation, which gives approximately the same performance as the 64-sample length cross-correlation with no-quantized LS. This length is 32 samples. The performance is measured in terms of timing error probability, defined as the probability of detection outside the five-sample window for those frames correctly detected. Our simulations show that the timing error probability of the cross-correlator of 32 quantified coefficients is only 0.6% lower than the probability of the cross-correlator of 64 floating-point coefficients at 2dB SNR. For SNR higher than 6 dB, both cross-correlations have a timing error probability lower than 0.1%.

The 32-sample length cross-correlation with quantized LS only needs 63 real adders, whereas the 64-sample length cross-correlation with no-quantized LS requires 64 complex multipliers and 63 complex adders. Thus, a cross-correlation of 32 samples is calculated as: $C_n = \mathbf{g}_{32}^H \mathbf{r'}_n$, where $\mathbf{r'}_n = [r_n \quad r_{n+1} \quad \cdots \quad r_{n+31}]^T$ is a vector with 32 samples from the received signal after CFO compensation and $\mathbf{g}_{32} = [LS_0 \quad LS_1 \quad \cdots \quad LS_{31}]^T$ is a vector with the first 32 quantized samples from the LS. The objective of this length reduction is to minimize not only the hardware cost, but also the latency (a reduction of 1.6 μs is obtained). In ideal conditions, the cross-correlation gives a large peak at sample n = 224, that is, in the middle of the long training symbol (n_{GI+64}). Additionally, to reduce the computational complexity, this cross-correlation is only calculated for an interval window of 20 samples, which is the deviation given by our coarse time synchronization algorithm (between samples -4 and +15, see Fig. 7). Therefore, the position of the peak is estimated as:

$$\hat{n}_{GI+64} = \arg \max_{n_1<n<n_2} \left\{ |C_n|^2 \right\}, \tag{4}$$

being $n_1 \le n \le n_2$ the interval of 20 samples where the cross-correlation is computed ($n_1 = \hat{n}_{GI} + 64 - 15$ and $n_2 = \hat{n}_{GI} + 64 + 4$).

Fig. 8 shows the implemented cross-correlator as well as the maximum detection circuit. After 20 clock cycles, time offset estimation can be read from register D2. Additionally, the results of the fixed-point analysis are also included. The time reference given by this fine time estimation will be used to load correctly the average of both LS's in the FFT and to save correctly the OFDM symbols in the input buffer (input DPRAM in Fig. 16). As can be seen in Fig. 16, this maximum search does not increase the receiver latency.

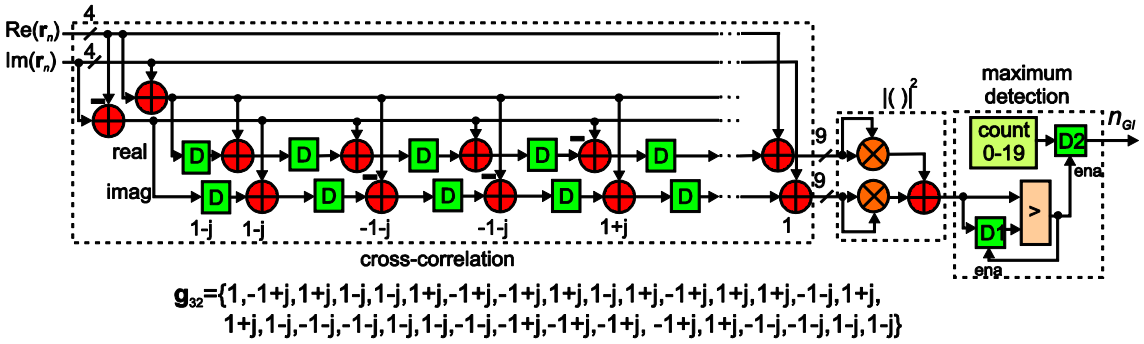

Fig. 8. Implementation of the proposed fine time synchronization algorithm.

C. Performance and algorithm comparison

In this section the performance of the proposed algorithm is compared to several synchronization algorithms which are specifically designed for the IEEE 802.11a/g standard: Troya et al. [5], Chang and Kelly [3] and the ML algorithm proposed in [4]. Algorithms proposed in [3] and [4] only include time synchronization, so they need some additional algorithm for frame detection. In these cases, we assume that AGC and signal detection are completed during the third SS (n_i in Fig. 1 is randomly set following a uniform distribution in the range 32 to 47 as in [3]). Both algorithms, [3] and [4], work without CFO correction, anyway an algorithm for CFO estimation is required to perform the channel estimation correctly.

Troya et al. [5] and the proposed algorithm are similar, so now we will explain the main differences between them. Like our proposal, Troya's solution [5] is divided in two parts: coarse and fine time synchronization based on auto-correlation and cross-correlation, respectively. Moreover, in [5] some simplifications to reduce hardware cost for VLSI implementation are introduced, but at the expense of performance degradation.

First, in [5] the received signal is auto-correlated with a delay of 64 samples and then averaged during 64 samples. As a result, a plateau of 32-sample length is obtained. Then, the difference between de auto-correlator output and a delayed version of itself (delay of 32 samples) is calculated, so a peak is obtained at sample n = 160. To detect this peak ($\hat{n}_{GI}$) a group peak detector and an instantaneous peak detector are used as described in [5]. In contrast, our proposed coarse time estimation directly obtains a large peak at sample n = 160.

On the other hand, the maximum CFO that can be estimated by using the auto-correlation proposed in [5] is 156.25 kHz, which is lower than the maximum CFO allowed by IEEE 802.11a/g standard (232 kHz), so another auto-correlation (with a delay of 16 samples and averaging during 16 samples) is required as explained in [5]. Also, a combination of both estimations is calculated. In contrast, our proposal only requires one CFO estimation, which reduces the hardware cost and the receiver latency.

Like the algorithm we propose in this paper, [5] makes use of a cross-correlation with the first 32 complex samples of the LS for fine time synchronization. Nevertheless, in [5] the cross-correlation is calculated by using only the sign bits of the complex input values and the sign bits of the reference (positive samples are considered '1' and negative samples '0'). This simplification allows a drastic reduction of the hardware cost: complex multipliers are replaced by XNOR 1-bit complex multipliers [6]. Unfortunately, not only is the hardware cost reduced, but also the performance. Another difference between Troya's work [5] and the proposed fine time synchronization algorithm is where the cross-correlation is applied. In our proposal the cross-correlation is calculated only during 20

samples, where we know that its first peak is located thanks to the coarse time synchronization. So, the search of the position of the peak is limited to 20 samples, which improves the performance of the fine time synchronization algorithm. In contrast, Troya's cross-correlation [5] begins after $\hat{n}_{GI}$ is detected, once CFO is removed from the preamble, so it is calculated approximately during 64 samples. Moreover, the position of the first peak is found by setting a threshold: the maximum is searched between those samples that exceed the threshold.

For performance comparison, we tested the different synchronization algorithms using the following conditions: BRAN channel models A, B and C [15]; with additive white Gaussian noise in a range of SNR values from 2 to 18 dB; and a CFO of 73% of the subcarrier spacing. For each synchronization algorithm, the success rate of the time estimation was measured using 10^4 test frames (each one with a different channel realization). For those algorithms that make use of thresholds, the performance results provided in this paper were obtained considering the optimal threshold (that is, the one that gives the highest success rate) at each SNR. In this way, the given results are the best ones that can be obtained by these algorithms under the test conditions.

Fig. 9 shows the probability of detection error (probability of detecting the starting point of GI outside a range of ±15 samples from the ideal sample). On the one hand, the poor results given by the algorithm proposed by Chang and Kelley [3] are due to the symbol synchronization stage. We observed that when the algorithm begins to work (instant n_i) at samples in the middle of a SS, the results are close to the 100% of success rate (see [3]), but when it begins to work at samples near the transition between two consecutive SS's the performance is drastically reduced (n_{GI} is often detected with an offset of ±16 samples). On the other hand, the frame synchronization stage of the ML method [4] does not give a good estimation of the channel order $\hat{L}$ at low SNR. As this estimation is used by the symbol synchronization stage, the algorithm performance is penalized causing a high probability of detection error, especially at low SNR. In conclusion, the best algorithms in terms of probability of detection error are the proposed one and Troya's [5], although the latter [5] has a higher probability of detection error than the former: around 1 % at 6 dB SNR. This is due to the fact that our auto-correlation obtains a larger peak than the auto-correlation and differentiation proposed in [5].

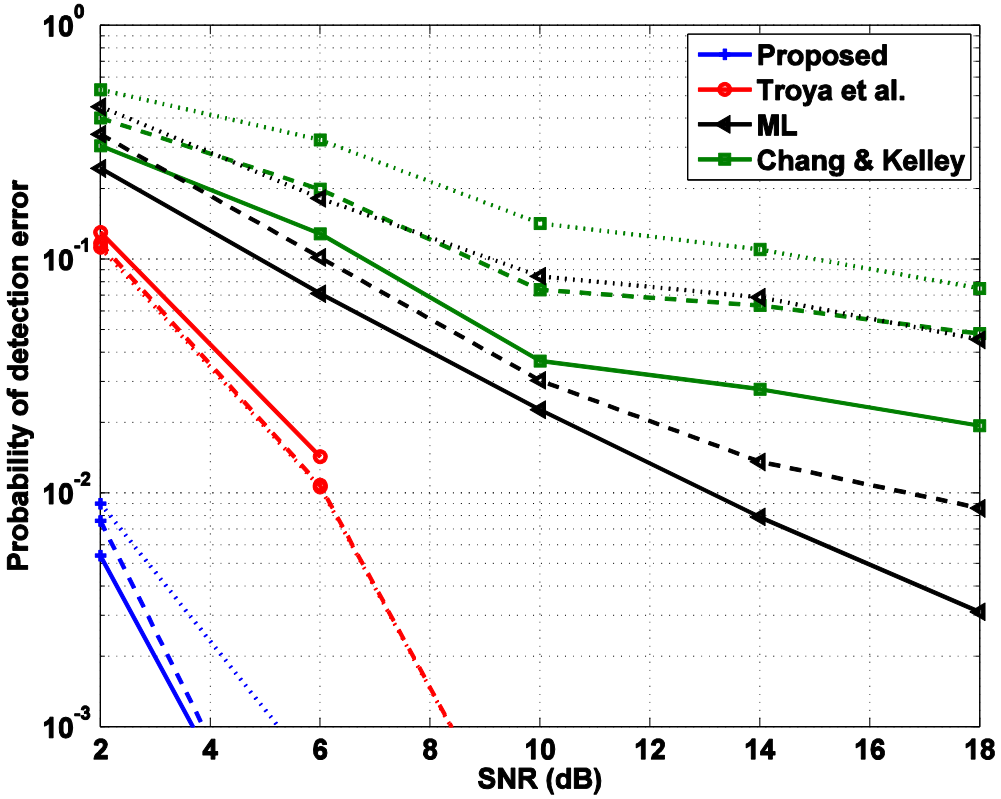

Fig. 9. Probability of detection error for channel A (continuous line), B (dashed) and C (dotted).

Additionally, we tested the proposed algorithm in more pessimistic conditions for channel model A: the received preamble presents clipping during its 48 first samples (3 SS). For a signal clipping equal to the RMS value of the preamble, we achieve approximately the same probability of detection error than without clipping. However, this probability increases for a signal clipping 5dB below the RMS value of the preamble and low SNR: $9.5 \cdot 10^{-2}$ for 2dB and $1.1 \cdot 10^{-3}$ for 6dB. Moreover, we checked how the input clipping affects to the coarse frequency estimation. Our simulations show that the minimum SNR needed to maintain the Bit Error Rate (BER) below 10^{-5} for 64-QAM is 28dB for channel model A. On the other hand, the standard deviation of the estimation error must be below 0.36% for an SNR loss lower than 0.4dB [19]. For all the studied clipping conditions we obtain a standard deviation below 0.36%, for channel model A and 28dB SNR. In conclusion, no fine frequency estimation is required

Fig. 10 shows the timing error probability (probability of detection outside the five-sample window for those frames correctly detected). The proposed algorithm presents the best results: a timing error probability lower than 0.1 % for channel A and 6 dB SNR. Troya's algorithm [5] presents a very poor performance: its error probability is higher than 10 % at 10 dB SNR and channel A, due to the 32 length cross-correlation based on XNOR 1-bit complex multipliers. This performance can be improved by extending the length of the reference to 64 samples as mentioned in [5], but this would increase the latency and the hardware cost.

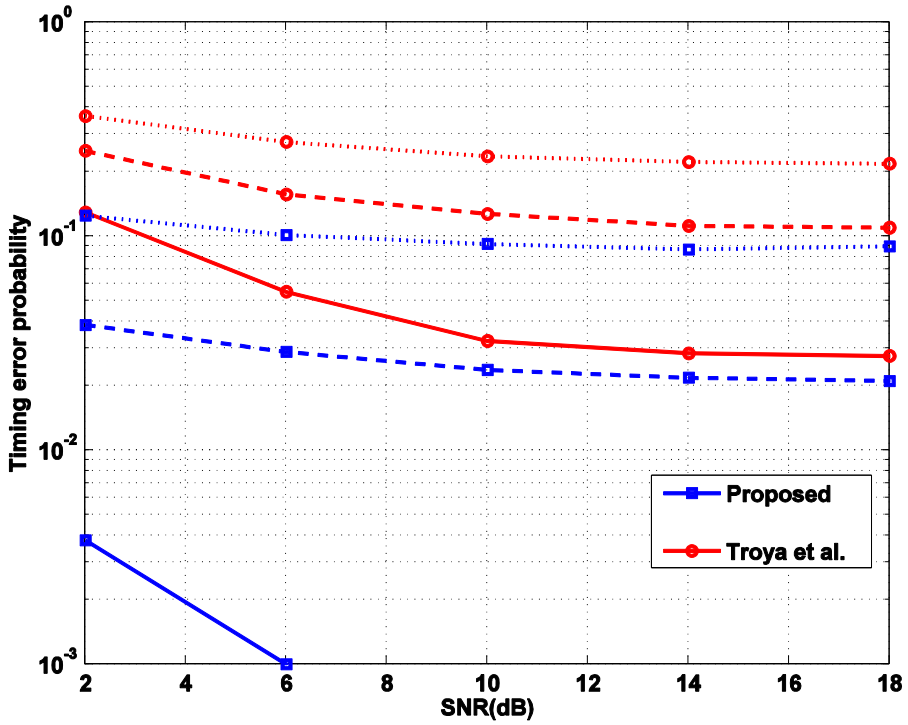

Fig. 10. Timing error probability for channel A (continuous line), B (dashed) and C (dotted).

In conclusion, the proposed algorithm has better performance (lower timing error probability and probability of detection error) than the rest of studied algorithms.

III. FFT/IFFT

The designed FFT/IFFT processor is basically composed of 2 dual-port memories and a radix-2 decimation-in-frequency butterfly (BF), as shown in Fig. 11. The dual-port memories are used to store the FFT/IFFT input, intermediate and output data. Data can be saved or read from the memories at the same time that FFT is calculated. For each OFDM symbol, its 64 data samples are saved in the DPRAMs and, after 384 clock cycles, the FFT result is obtained. An FFT must be calculated every 4 μs, the duration of an OFDM symbol sampled at 20MHz, to achieve a continuous data flow. So, the minimum clock is 96 MHz, but we selected 100 MHz because it is an entire multiple of the input signal frequency. For this clock, a FFT result is obtained each 3.84 μs. The FFT is disabled during the other 0.16 μs.

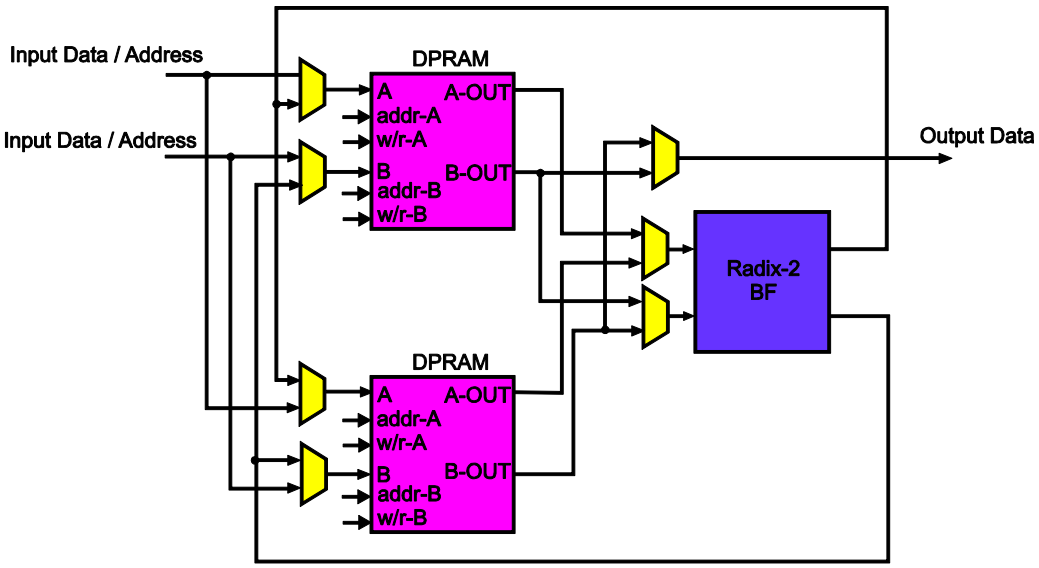

Fig. 11. FFT/IFFT implementation.

The fixed-point simulation shows that the FFT output precision must be 10 bits for a PER loss of our receiver lower than 0.5 dB (PER = 10^{-2}).

IV. CHANNEL ESTIMATION

After time and frequency synchronization, the frequency response of the radio channel must be estimated. The long symbols (LS) of the preamble, whose initial sample is given by the fine time synchronization algorithm, are used to do this estimation once the CFO is corrected.

The selected channel estimation algorithm is performed in the frequency domain. The contents of the two long symbols are identical, so they can be averaged to improve the quality of the channel estimation [12]. This average can be calculated before the FFT, because the FFT is a linear operation. The frequency response of the channel estimation ($\hat{H}$) is obtained as follows:

$$C_X = FFT\left(\frac{LS_1 + LS_2}{2}\right);\ \hat{H} = \frac{C_X}{C_L}, \tag{5}$$

where LS_1 and LS_2 are the first and the second received LS, respectively, and C_L is the transmitted LS in the frequency domain.

For simplicity, channel distortion is compensated by applying the zero forcing (ZF) solution: the received OFDM symbols are multiplied by the inverse of the channel estimation. To avoid the division by a complex number (C_X) the inverse is calculated as:

$$\frac{1}{\hat{H}} = \frac{C_L}{C_X} = \frac{1}{C_X \cdot C_X^*} \cdot C_X^* \cdot C_L = \frac{1}{|C_X|^2} \cdot C_X^* \cdot C_L. \tag{6}$$

In Fig. 12 the implementation of the channel estimation and compensation algorithms can be seen. $C_{L,}$ which is -1, 0 or 1, is saved in the ROM 64x2. The pre-calculated division

1/x is saved in the ROM 1024x16. First, the FFT of the averaged LS's $((LS_1 + LS_2)/2)$ is calculated. Next, $|C_x|^2$ is performed using the branch that calculates the input energy in the autocorrelator scheme. After that, $1/|C_x|^2$ is read from the ROM 1024x16. At the same time, C_L is read from the ROM 64x2 and $C_x^* \cdot C_L$ is obtained with 2 multiplexers and 2 complementers. Finally, $C_x^* \, C_L \cdot / |C_x|^2$ is calculated by using two real multipliers and saved in two DPRAM (64 values). These memories will be read for each received OFDM symbol after the FFT operation to equalize the received subcarriers (a complex multiplier is needed).

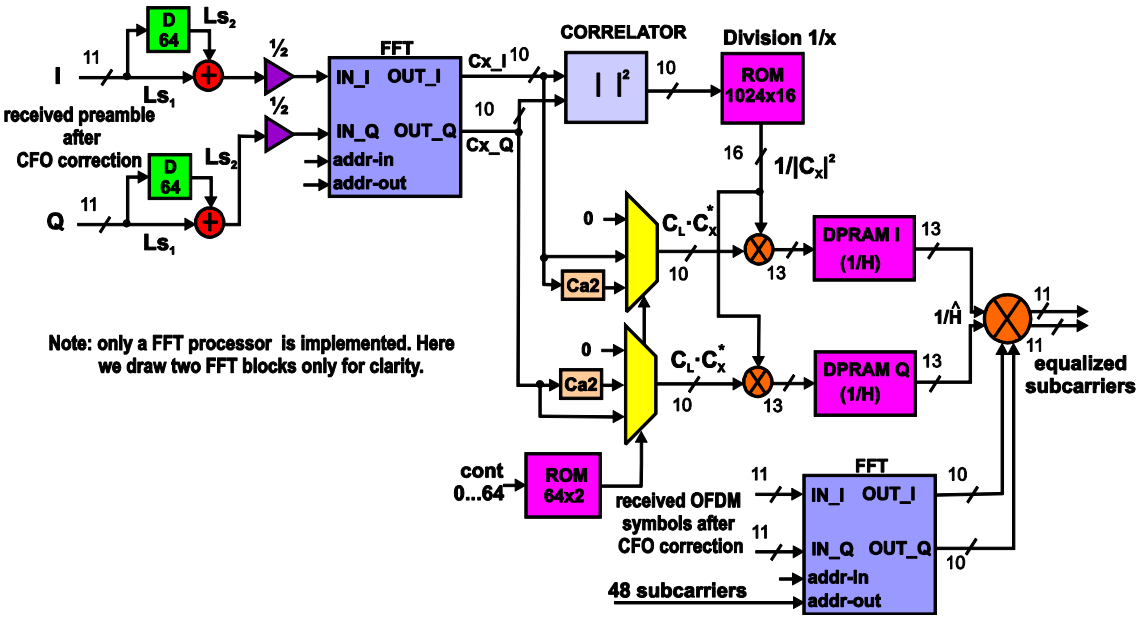

Fig. 12. Implementation of channel estimation and compensation.

Fig. 12 also shows the results of the fixed-point analysis. The precision detailed in each stage guarantees that the PER loss of our receiver is lower than 0.5 dB for a PER = 10^{-2}.

V. PHASE TRACKING

As commented above, the residual CFO progressively rotates the phase of the received signal. This rotation is constant for the subcarriers of an OFDM symbol and it is incremented from one OFDM symbol to another. This rotation causes a rotation in the constellation, which makes it impossible to perform a correct demodulation after receiving a few OFDM symbols. To avoid this, the rotation must be estimated and compensated for each OFDM symbol, which is known as phase tracking.

The phase tracking scheme makes use of the four pilot subcarriers embedded in the OFDM symbols. The phase rotation is detected by comparing the received pilot subcarriers (R_{nk}) against the known pilot subcarriers (P_{nk}) in the frequency domain. The phase estimate $\hat{\phi}_n$ is obtained by using the estimated channel frequency response for the pilot subcarriers ($\hat{H}_k$) [12]:

$$\hat{\phi}_n = \angle\left[\sum_{k=1}^{4} R_{nk}\left(\hat{H}_k \cdot P_{nk}\right)^*\right] \tag{7}$$

The transmitter generates the pilot subcarriers of each OFDM symbol using the vector $S_{0...126}$ = {1, 1, 1, 1, -1, -1, -1, 1, -1, -1, -1, -1, 1, 1, -1, 1, -1, -1, 1, 1, -1, 1, 1, -1, 1, 1, 1, 1, 1, -1, 1, 1, 1, -1, 1, 1, -1, -1, 1, 1, 1, -1, 1, -1, -1, -1, 1, -1, 1, -1, -1, 1, -1, -1, 1, 1, 1, 1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1, -1, -1, -1, 1, 1,-1, -1, -1, -1, 1, -1, -1, 1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, 1, -1, -1, -1, -1, -1, 1, -1, 1, 1, -1, 1, -1, 1, 1, 1, -1, -1, 1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1}. Pilot subcarriers for the *n*th transmitted OFDM symbol are [1 -1 1 1] if S_n = 1 and [-1 1 -1 -1] if S_n = -1.

In Fig. 13 the implementation of the phase tracking algorithm can be seen. First, $\hat{H}_k$ is obtained, being k = 7, 21, 43 and 57: the four pilots are read from the FFT of $\left((LS_1 + LS_2)/2\right)$ (positions 7, 21, 43 and 57 of the output memory). As the transmitted pilots of the LS are $C_{L7} = C_{L21} = C_{L57} = 1$ and $C_{L43} = -1$, the channel estimation for the four pilots is obtained as: $\hat{H}_7 = C_{X7}$, $\hat{H}_{21} = C_{X21}$, $\hat{H}_{43} = -C_{X43}$, $\hat{H}_{57} = C_{X57}$, and they are saved in the registers D. The vector $S_{0...126}$ is saved in the ROM 127x1 (the value 0 is saved if S_n = 1 and the value 1 is saved if S_n = -1) and the state machine generates [0 1 0 0] when S_n = 1 and [1 0 1 1] when S_n = -1. The output of the state machine controls both multiplexers, whose output is $\left(\hat{H}_k \cdot P_{nk}\right)^*$. After that, $\sum_{k=1}^{4} R_{nk}\left(\hat{H}_k \cdot P_{nk}\right)^*$ is calculated by reusing the correlator (its complex multiplier and its moving average). Finally, the estimated phase $\hat{\phi}_n$ is obtained with the CORDIC. For each OFDM symbol, the phase is estimated and the equalized data subcarriers are compensated by this phase reusing the CORDIC.

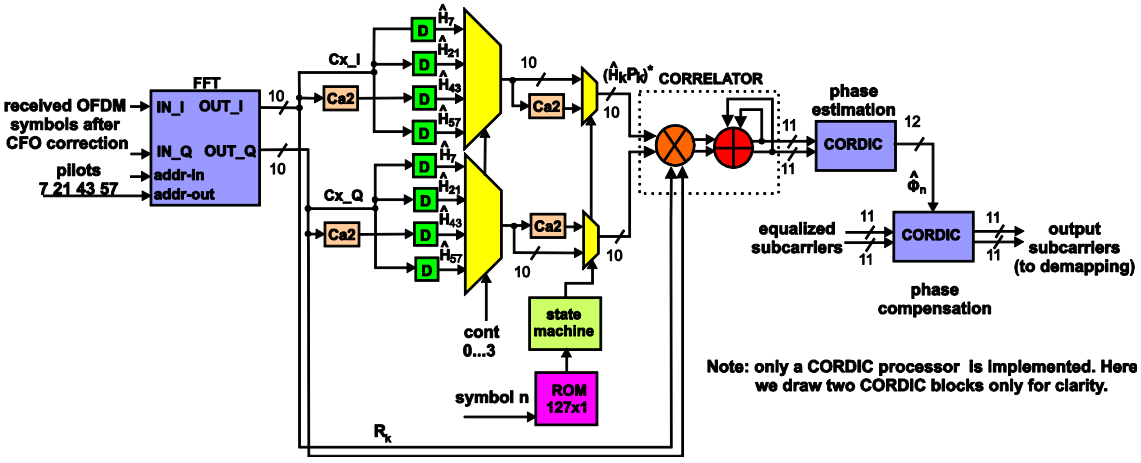

Fig. 13. Implementation of phase estimation and compensation.

Fig. 13 also shows the results of the fixed-point analysis. The precision detailed in each stage guarantees that the PER loss of our receiver is lower than 0.5 dB for a PER = 10^{-2}.

VI. COMPLETE SYSTEM

In this section, we describe the architecture selected for the complete receiver. The proposed architecture minimizes the hardware cost because the main blocks (the correlator and the CORDIC) are reused for several tasks in different time intervals. This reuse is controlled by a state machine. Each task needs a different quantification in order to achieve the desired performance (PER loss below 0.5 dB for a PER = 10^{-2}). Next, we study the required precision of the correlator and the CORDIC when they are employed in each task and, finally, the precision of both blocks is selected (it will be the most restrictive one).

A. Correlator precision

The main purpose of the correlator is frame detection and time synchronization, but it is also used in channel estimation and phase tracking. This block must carry out the following operations:

1) Frame detection: it correlates the SS's of the preambles and detects the peak at the end of the last SS. It is enough with 5 input bits.
2) Frequency offset estimation: the output of the correlator (R) at the peak is used to obtain this estimation. It needs almost 6 input bits.
3) Channel estimation: the squared modulus is calculated with the branch of the correlator that obtains P^2. The input and the output must have 10 bits.
4) Phase tracking: the complex multiplier and the moving average of the correlator are used to accumulate the 4 complex products for phase tracking. The input must also have 10 bits, and the output 11 bits.

The fixed point correlator can be seen in Fig. 14, as well as the extra hardware added to make the reuse possible.

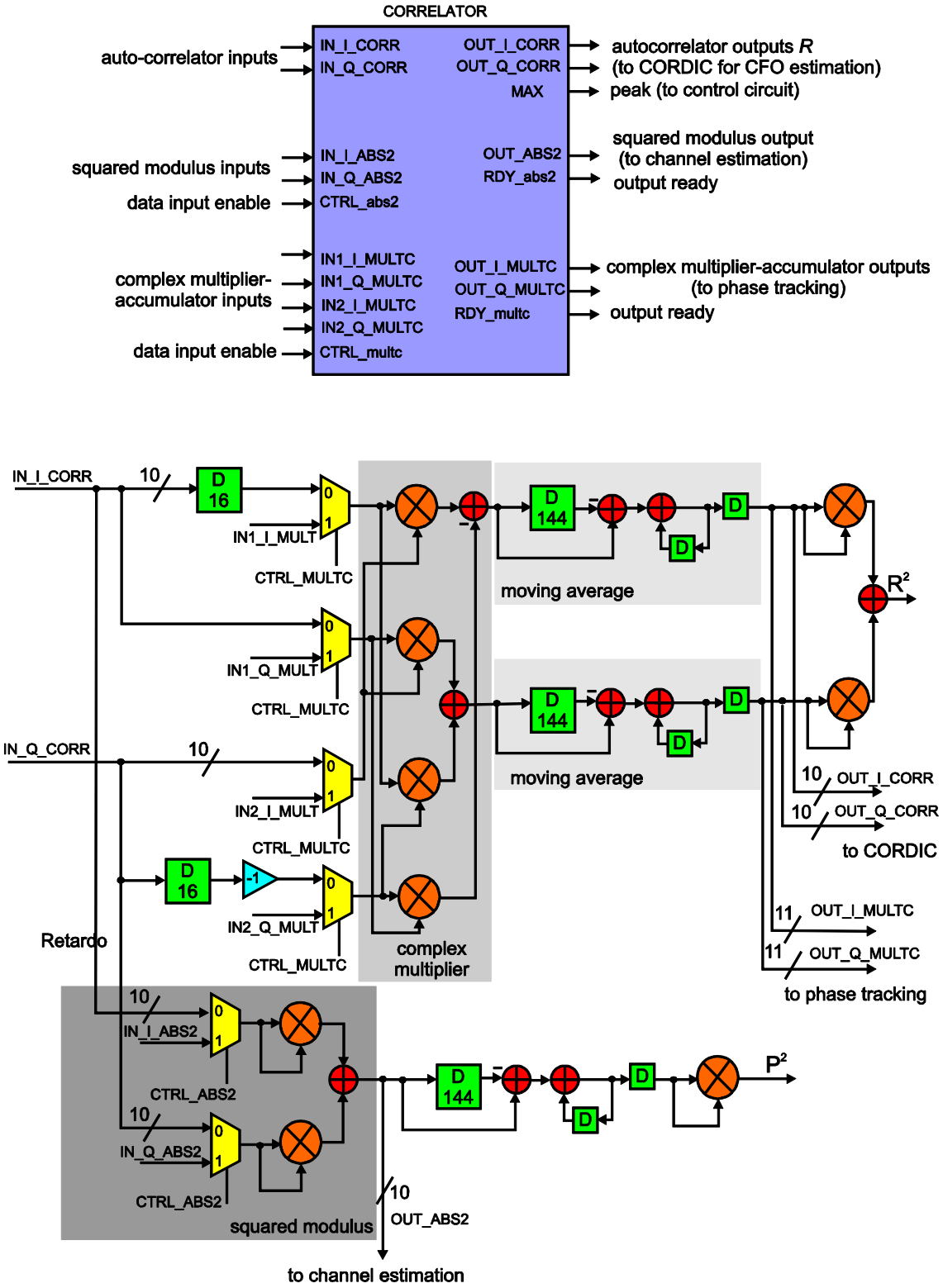

Fig. 14. Correlator block.

B. CORDIC precision

The CORDIC in [16] is employed in two basic operations: the calculation of the angle of a complex number (configured as circular vectoring mode), and the correction of a complex number by an angle (configured as circular rotation mode). In the synchronization process the CORDIC realize the following tasks:

1) CFO estimation: it needs 10-bit input data (IN_I, IN_Q) and 10 bits at the output angle (O_phase). The input angle (I_phase) must be 0.
2) CFO compensation from LS: the PDU train is 10-bit length, so input data (IN_I, IN_Q) must be 10-bit length and the output data (OUT_I, OUT_Q) must be 11-bit length. The input angle (I_phase) is the CFO estimation divided by 16, so it must be 14-bit length.
3) CFO compensation from the OFDM symbols: it requires the same precision as 2), the only difference is that, in this case, data are read from the DPRAM where the OFDM symbols are saved.
4) Phase tracking estimation: it requires 11-bit inputs and 12 bits at the output angle.
5) Phase tracking compensation: the 11-bit length equalized subcarriers are connected to the input data, and the 12-bit length phase estimation is connected to the input angle. Output data must be 11-bit length.

To sum up, the input and output data are 11-bit length, the input angle is 14-bit length, and the output angle is 12-bit length. To achieve this output precision a CORDIC with 11 iterations is used. Fig. 15 shows how the CORDIC is reused, as well as, the precision of its inputs/outputs.

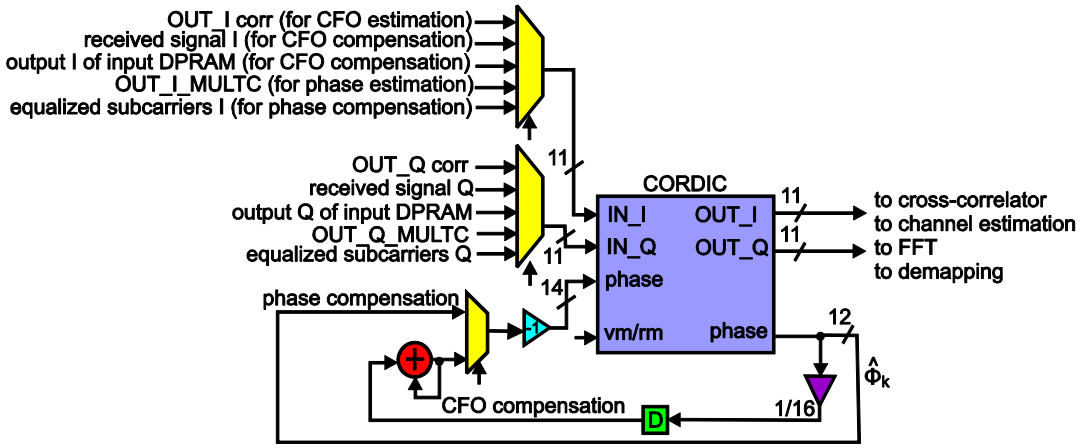

Fig. 15. CORDIC block.

C. Temporal diagram

Fig. 16a shows the temporal diagram for frame detection, time and frequency synchronization and channel estimation. Fig. 16b shows the temporal diagram for channel estimation and compensation, demodulation and phase tracking. As can be seen, there are two basic blocks, which are reused several times: the correlator and the CORDIC processor. First, PDU train is correlated and a large peak is detected when the IEEE preamble is present. After that, a coarse time reference is obtained and the CFO is calculated using the CORDIC. Next, the estimated CFO is compensated from the LS's and then, the corrected LS's are cross-correlated to achieve fine tine synchronization. By using this fine time reference, the OFDM symbols of the PDU train are correctly saved in the input buffer (input DPRAM). Then, channel estimation begins: first, the average of the two LS's is obtained and loaded in the FFT; after 3.2µs the result is obtained (C_x in Fig. 16b) and the inverse of the channel estimate is calculated and saved in a DPRAM as explained in Section IV.

An OFDM symbol is received each 4µs and the next operations are performed during this time: first, the 64 data samples of the OFDM symbol are read from the input DPRAM; next, the estimated CFO is removed and the corrected data samples are loaded in the FFT; after 4µs the result is read: first, the four pilots are read and the phase is estimated by using the correlator and the CORDIC; next, the 48 subcarriers are read from the FFT and equalized; finally, the equalized subcarriers are rotated by the phase estimation using the CORDIC and, then the signal is ready to be demapped.

As can be seen in Fig. 16, the first part of the process (frame detection and CFO estimation) works at a sample rate of 20MHz (input data rate), whereas the rest works at 100MHz. This is necessary to allow the FFT to obtain a result every 4µs and the CORDIC to do all the required operations, and also to minimize the total latency of the system. The first output subcarriers are obtained 24.6µs after the first sample of the preamble is received. Also, with this schedule, only two OFDM symbols must be saved in the input DPRAM. All blocks are disabled when they are not used in order to reduce the power consumption.

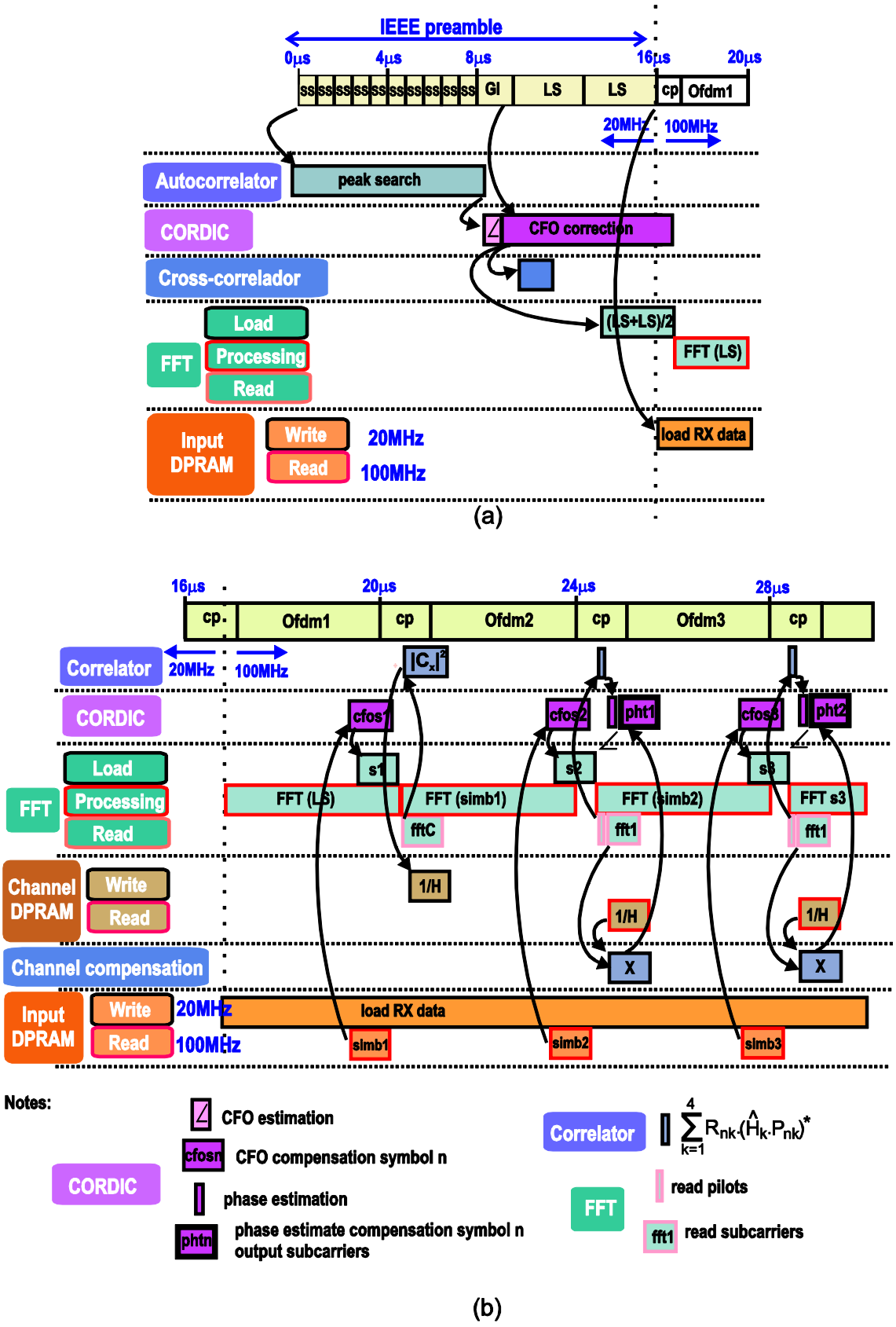

Fig. 16. Temporal use diagram.

D. Hardware implementation

Several tests were done to verify the designed receiver, with three different input signals: 1) QPSK (18 Mbits/s), SNR = 20 dB and CFO between transmitter and receiver of 150 kHz; 2) 16-QAM (36 Mbits/s), SNR = 25 dB and CFO = 150 kHz; and 3) 64-QAM (54 Mbits/s), SNR = 30 dB and CFO = 150 kHz. Each test frame was composed of 320 noise samples, the IEEE preamble and 100 OFDM symbols. For each operation mode, we selected a SNR that guarantees a BER lower than 10^{-5} with error correction and the floating-point receiver (ideal synchronization) in our simulations.

Firstly, the designed receiver was implemented on a Virtex-II FG676 Proto Board, which was connected to a logic analyzer (see Fig. 17). We checked that the output subcarriers of the implemented receiver were identical to the output subcarriers of the fixed-point model of the receiver, for the same digital test input. Secondly, we implemented our receiver on the XtremeDSP Development Kit of Nallatech, in order to simulate a more realistic scenario and observe the received constellation. This development kit includes a clock FPGA, a user FPGA (Xilinx Virtex II XC2V6000), two high-speed ADC (Analogue-to-Digital Converter) and two high-speed DAC (Digital-to-Analogue Converter). The analogue test inputs were generated by using a ROHDE&SCHWARZ I/Q Modulation Generation AMIC and a Signal Generator SMIQ04B (see Fig. 18). The output was connected to a TDS210 digital oscilloscope in XY mode to view the output constellation. As can be seen in Fig. 19, we obtained the desired constellation for each input signal.

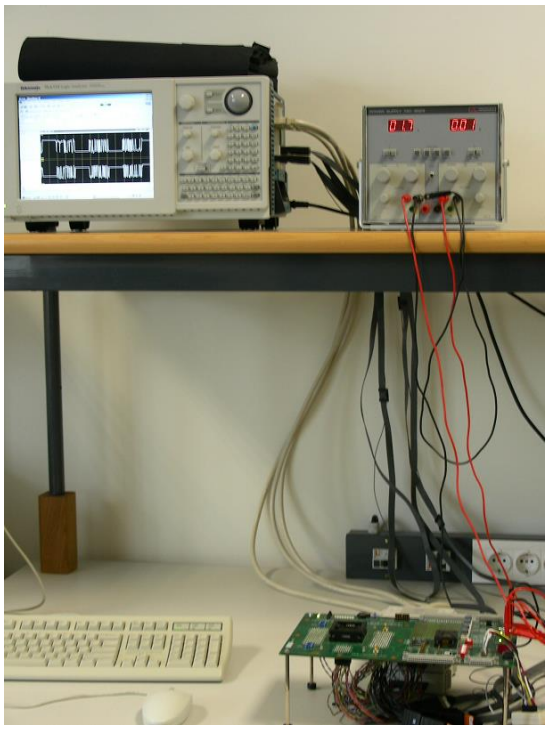

Fig.17. Virtex-II FG676 Proto Board test.

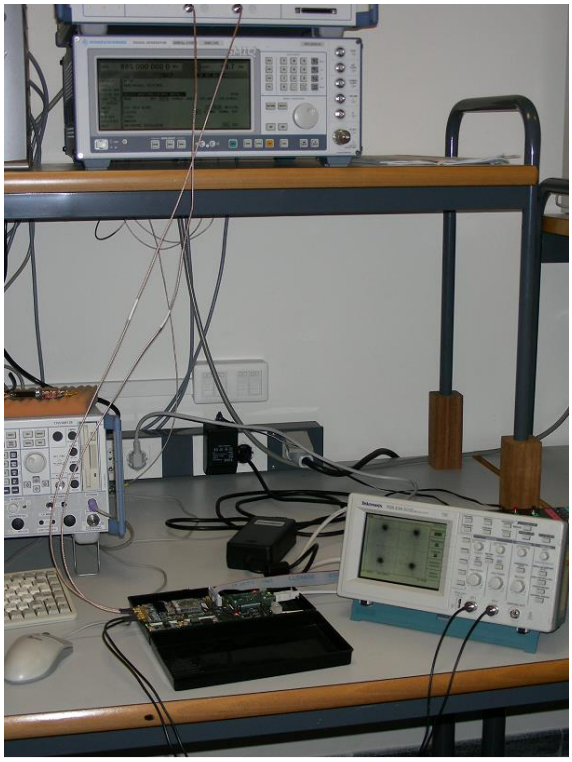

Fig.18. XtremeDSP Development Kit test.

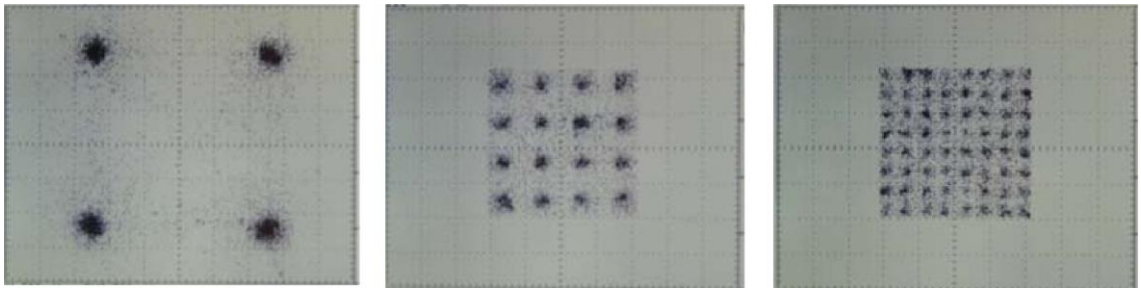

Fig.19. Output constellations capture for (a) QPSK, (b) 16-QAM and (c) 64-QAM.

The implemented receiver only requires 2986 slices, 20 embedded multipliers and 8 DPRAM.

E. Performance results and receiver structure comparison

The performance results of the proposed receiver were obtained by using a simulator of the IEEE 802.11a/g physical layer and including a fixed-point model of the designed receiver (frame, time and frequency synchronization, channel estimation, equalization and phase tracking). Simulation conditions were: channel model A [15][14], 64-QAM (54 Mbits/s), error correction (Viterbi with CSI [10]), 1000 frames (each frame was composed by 50 data packets of 1000 bytes and had a different channel realization).

Fig. 20 compares the receiver PER plot for ideal and real synchronization. The ideal synchronization plot was obtained by using a floating-point model of the receiver supposing that the synchronization is perfect (no time or frequency synchronization errors) and zero forcing channel compensation, whereas the real synchronization plot was

calculated by means of a fixed-point model of the designed receiver (each frame has time and frequency errors). As can be seen, the PER loss is only 0.5 dB at PER = 10^{-2}. Our simulations show that this loss is due to the quantification of the receiver (0.25 dB) and to the synchronization errors (for example, the floating-point phase tracking algorithm introduces a loss of 0.2 dB in the receiver).

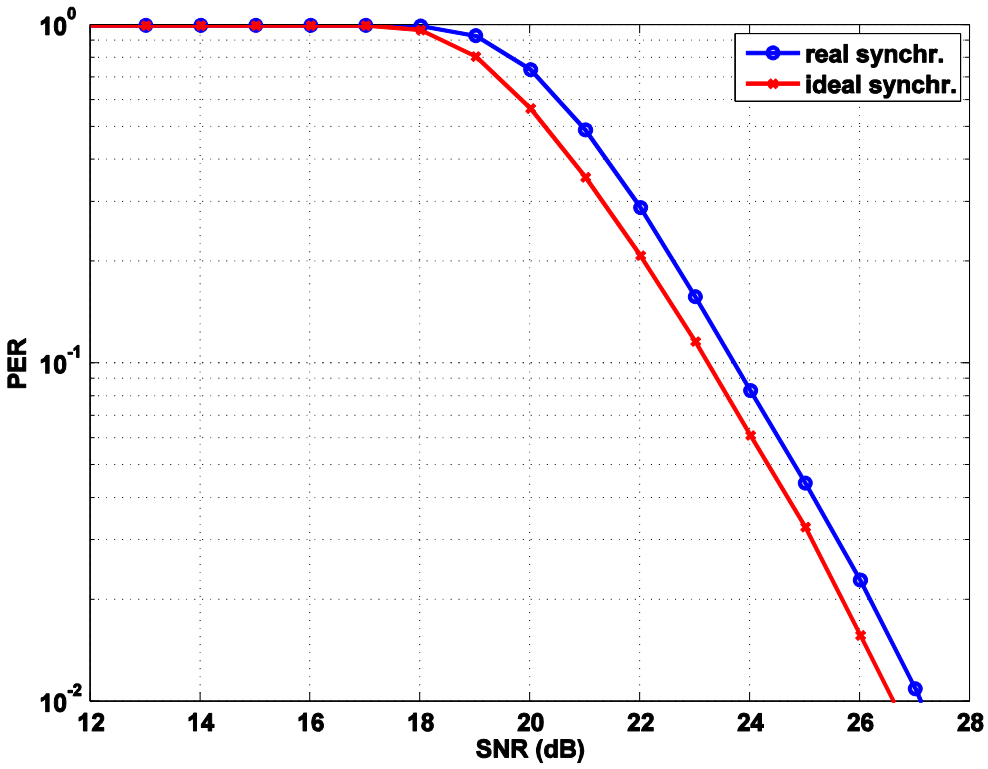

Fig. 20. PER performance.

Finally, we compare our receiver with other solutions proposed in the literature ([6], [7], [8] and [9]), in terms of algorithm implementation cost and performance. It is to be remarked that it is extremely difficult to make a fair comparison with these solutions due to the fact that different technologies and implementation strategies were utilized.

Table 1 summarizes the SNR at a 10% PER required by 802.11a standard and the studied receivers, for AWGN channel and different data rates. Troya's PER performance is not given in [6], so it cannot be included in this comparison. As can be seen, our proposal outperforms the receivers implemented previously and it needs smaller SNR at every data rate than the one required by the 802.11a standard.

Rate (Mb/s)	802.11a required SNR [1] (dB)	Proposed (dB)	Reference [7] (dB)	Reference [8] (dB)	Reference [9] (dB)
9 (BPSK)	10.7	3.9	5.8	5.8	9.7
18 (QPSK)	14.7	8.1	9.9	9.5	12.8
36 (16-QAM)	21.7	14.6	15.9	14.9	20.5
54 (64-QAM)	26.7	19.6	21.7	20.6	26.1

Table 1. Performance comparison at a 10% PER.

In [7] a transceiver architecture for OFDM-WLAN is designed. It includes synchronization, channel estimation and phase tracking. Few implementation details are given, so we only compare some aspects of the architecture in other to comment on our contributions. Coarse time synchronization in [7] is based on an auto-correlation with a delay of 16 samples (the moving average is not defined), and fine time synchronization is based on a matched filter of 64 coefficients. No performance results for the synchronization stage are given. This matched filter requires 64 complex multipliers and 63 complex adders, whereas the cross-correlator that we employ for fine time synchronization only needs 63 real adders. CFO estimation in [7] is done in three steps: coarse CFO estimation (using the auto-correlation with a delay of 16 samples), fine CFO estimation (using an auto-correlation with a delay of 64 samples) and CFO tracking using the pilot subcarriers. CFO is corrected with a phase-locked loop (PLL). Our design only needs one CFO estimation due to the large average used in the auto-correlation and, the CFO estimation and correction is done by reusing the same CORDIC, which reduces the hardware cost of our receiver. For phase tracking in [7], four phases are calculated, whereas only one phase is computed in our implemented algorithm.

In [8] and [9] the algorithms used for synchronization, channel estimation and phase tracking are not described, neither their implementation. So, we only compare the PER performance of the receiver in Table 1. Additionally, in [8] the implementation losses are also given: 1.6 dB for 54Mb/s at a 10% PER, whereas our implementation losses are only 0.45 dB under identical conditions.

In Section III.C, the main differences between the synchronization stage of [5], [6] and the proposed algorithm have been described, and it has been shown that our synchronization algorithm outperforms the one proposed in [5], [6]. Now, we will compare the implementation details of the complete receiver. For frequency synchronization, [6] uses a CORDIC for CFO estimation (16-iteration CORDIC, 16-bit inputs, 13 bit-output) and a numerically controlled oscillator NCO for CFO compensation (768 full adders and 533 registers). In contrast, we only use an 11-iteration CORDIC to do the same operations. For channel estimation, [6] describes the implementation of a simplified version of the channel estimation algorithm proposed in [5], but no performance comparison is included. First, a ZF estimation is obtained by using a unique LS (a 3 dB penalty occurs with respect to averaging both LS's as we do in our proposal). Afterwards, the ZF reference is updated by means of a decision-feedback mechanism using the decoded bits. This drastically increases the latency of the channel estimation algorithm, due to the delay of the Viterbi decoder, and also the hardware cost. The ZF estimation in [6] requires a complex divider which is replaced by a complex multiplier

and some memories where the limited number of values that the decoded bits can take (depending of the modulation scheme) are saved. The channel compensation based on a complex division is absolutely necessary in the solution proposed in [6] due to the proposed residual phase correction algorithm. This complex division is based on another CORDIC implementation (16 iterations, inputs data: 16 and 32 bits). The implementation of the channel estimator proposed in this paper is based on the ZF solution (averaging both LS's). It basically requires one ROM where the inverse of a constant is saved (the content of this memory do not depend on the modulation scheme) and a complex multiplier for channel correction. The latency and the hardware cost of our proposal are lower because we do not use any feedback and we do not need a complex division. For phase tracking, [5] proposes a phase estimation algorithm which does not need either an arctangent block or an NCO for phase correction. It requires 16 real adders and 1 complex multiplier [18]. In our proposal we basically need an accumulator of 4 complex products, an arctangent calculation and a phase rotator, but all these operations are implemented reusing the correlator and the CORDIC, so no extra hardware is required. In conclusion, our receiver requires less hardware resources than the receiver proposed in [6]. Finally, in contrast to [6], our design includes a complete fixed-point analysis, as well as, the PER performance of the implemented receiver and the implementation losses.

VII. CONCLUSIONS

In this work, a practical solution is given for the implementation of an OFDM-based WLAN receiver on a FPGA. The receiver is composed of frame, time and frequency synchronization, FFT-based OFDM demodulation, channel estimation and equalization and phase tracking.

Emphasis is given to the synchronizer design because the performance of the receiver strongly depends on it. The proposed synchronization algorithm outperforms the ones in [3], [4], [5] and [6]: it gives a probability of detection error and a probability of timing error below 0.1% at 6 dB SNR and channel model A [15]. Additionally, our synchronizer has low hardware cost and low latency due to the fact that only one CFO estimation is needed and thanks to the simplifications made on the cross-correlator.

The algorithms proposed for channel estimation and phase tracking are simple. In any case, the complete receiver achieves the desired performance: it outperforms the receivers implemented previously ([7], [8] and [9]) and needs smaller SNR at every data rate than the one required by the 802.11a standard. This is mainly due to the design flow which was used, making a complete fixed-point analysis which guarantees at each stage a PER loss below 0.5 dB for a PER = 10^{-2}, with a modulation scheme of 64-QAM and including the error correction stage. Once the precision of the different stages is determined, the designed receiver is implemented on a prototype board in order to verify it: first, we check that the output subcarriers are identical to those obtained by simulation and, then, we observe the output constellations for different modulation schemes.

The design of the complete receiver is based on the reuse of the main blocks (CORDIC, correlator and FFT) and so the final hardware cost is reduced (only 2986 slices, 20 embedded multipliers and 8 DPRAM are required on a Virtex-II Xilinx FPGA.). This reuse is possible thanks to the detailed schedule of the complete receiver that we present in this paper.

In conclusion, the proposed implementation of an OFDM-based WLAN receiver achieves excellent performance with low cost and low latency (the first output subcarrier is obtained 7.8μs after the first data sample of the first OFDM data symbol is received).

EFERENCES

[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, Dec. 1999.

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

[3] Sekchin Chang and B. Kelley, Time synchronization for OFDM-based WLAN systems, Electronics Letters, vol. 39, no. 13, pp. 1024-1026, June 2003.

[4] 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, November 2005.

[5] A. Troya, K. Maharatna, M. Krstic, E. Grass, U. Jagdhold and R. Kraemer, Efficient Inner Receiver Design for OFDM-based WLAN Systems: Algorithm and Architecture, IEEE Trans. on Wireless Communications, vol. 6, no. 4, pp. 1374-1385 April 2007.

[6] A. Troya, K. Maharatna, M. Krstic, E. Grass, U. Jagdhold and R. Kraemer, Low-power VLSI implementation of the inner receiver for OFDM-based WLAN systems, IEEE Trans. on Circuits and Systems - I, 55 (2). pp. 672-686. March 2008.

[7] Wei-Hsiang Tseng, Ching-Chi Chang, Chorng-Kuang Wang, Digital VLSI OFDM Transceiver Architecture for Wireless SoC Design, ISCAS 2005.

[8] J. Thomson et al., An integrated 802.11a baseband and MAC processor, Dig. Tech. Papers IEEE ISSCC 2002, Feb. 2002, vol. 1, pp.126-127.

[9] T. Fujisawa et al., A Single-Chip 802.11a MAC/PHY with a 32-b RISC processor, IEEE Journal of Solid State Circuits, vol. 38, no. 11, pp.2001-2009, November 2003.

[10] Weon-Cheol Lee, Hyung-Mo Park, Kyung-Jin Kang and Kuen-Bae Kim, Performance analysis of Viterbi decoder using channel state information in COFDM system, IEEE Trans. on Broadcasting, Vol. 44, No. 4, pp. 488-496, Dec. 1998.

[11] 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.

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

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

[14] M.J. Canet, V.Almenar, J. Marín-Roig and J. Valls, Low Complexity Time Synchronization for WLAN, submitted to Digital Signal Processing.

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

[16] F. Angarita, M.J. Canet, T. Sansaloni, A. Perez-Pascual, J.Valls, Efficient mapping of CORDIC Algorithm for OFDM-based WLAN, Journal of Signal Processing Systems, Vol. 52, No. 2, pp. 181-191, Aug. 2008.

[17] M.J.Canet, I. Wassel, V. Almenar, J. Valls, Performance evaluation of fine time synchronizer for WLANs, Proc. 13th european Conference on Signal Processing (EUSIPCO 2005), Sept. 2005.
[18] A. Troya, M.Krstic, K. Maharatna, Simplified residual phase correction mechanism for the IEEE 802.11a standard, Proc. IEEE VTC-Fall 2003, vol. II, Oct. 2003.
[19] T. Pollet, M. Van Bladel, M. Moeneclaey, "BER sensitivity of OFDM systems to carrier frequency offset and Wiener phase noise", IEEE Transaction on Communications, vol. 45, No 2/3/4, Febrero, Marzo, Abril 1995, pp. 191-193.

Figure captions