WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2013; **13**:1082–1094
Published online 23 August 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.1163

RESEARCH ARTICLE

Design, implementation, and evaluation of a field-programmable gate array-based wireless local area network synchronizer

Christopher E. Kennedy[1,2], Dan J. Dechene[2] and Abdallah Shami[2*]

[1] Broadband Wizard Incorporated, Windsor, ON N8W 5J1, Canada
[2] Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON N6A 5B9, Canada

ABSTRACT

Synchronization is a critical operation required by majority of wireless receivers. This paper presents the design, implementation, and evaluation of an orthogonal frequency-division multiplexing baseband packet synchronizer deployed on a field-programmable gate array (FPGA). Packet detection, carrier frequency offset estimation/correction, and time synchronization are all performed in the time domain by processing samples before the fast Fourier transform computation on the receiver. We propose techniques to reduce the area complexity of the arithmetic computations while maintaining the performance of existing approaches. FPGA implementation results are reported, and the design is evaluated by simulation under additive white Gaussian noise channel conditions. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS
computer arithmetic; FPGA; OFDM; synchronization; VLSI; Xilinx

***Correspondence**
Abdallah Shami, Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON N6A 5B9, Canada.
E-mail: ashami@eng.uwo.ca

1. INTRODUCTION

Orthogonal frequency-division multiplexing (OFDM) [1] has become the *de facto* standard in modern wireless communication systems, including wireless local area networks (WLANs). The IEEE 802.11a/b/g/n [2] standards, for example, all employ OFDM as a physical layer (PHY) communication technology. OFDM technology benefits from its ability to compensate for frequency-selective fading without the need for complex channel equalization. Unfortunately, the performance of OFDM can be degraded if there is any loss of orthogonality between sub-carriers or inter-symbol interference (ISI) at the receiver. To combat these problems, OFDM receivers require strict synchronization in both time and frequency to maintain orthogonality between sub-carriers. For packet-based transmission systems, OFDM transmitters often employ a preamble prior to data transmission. This preamble (or training sequence) is formed of periodic symbols, and the receiver uses properties of the sequence for synchronization.

In an OFDM communication system, in the presence of channel noise, the receiver must be able to detect an incoming packet, estimate and correct any carrier frequency offset (CFO), and perform accurate time synchronization. For robust very-large-scale integration (VLSI) implementation of these three operations, field-programmable gate arrays (FPGAs) are an attractive platform. FPGA devices are reconfigurable, include built-in high-performance digital signal processing (DSP) slices in their fabrics, and offer quicker time-to-market compared with their application-specific integrated circuit (ASIC) counterparts.

There have been several advances in the research area of OFDM synchronization. In [3], a method for estimating the CFO in the frequency domain is presented. In [4], a method for detecting a packet and an improved CFO estimation technique in the time domain are proposed. In [5], the performance of different cross-correlation methods for time synchronization are evaluated. In [6], a multiplierless cross-correlation strategy is presented, where quantized versions of the training sequence samples are used, reducing the area complexity while maintaining acceptable performance. In [7], a different quantized cross-correlation approach is presented, which only requires bit shifts to perform the multiplications. In [8], some modifications to the computations in [4] are presented, as well as the correction of the estimated CFO in the top-level synchronizer design. In [9], a synchronizer architecture with an automatic gain control (AGC) circuit is presented. In [10], the author presents the overall design and evaluation of a WLAN synchronizer on FPGA.

In most of the literature described earlier, the OFDM synchronization architectures either are vague in terms of the specific hardware requirements of their implementations, provide little information concerning the precision of arithmetic operations, or have only considered a portion of the synchronizer. In this paper, we present the design, implementation, and evaluation of an IEEE 802.11a PHY receiver synchronizer VLSI architecture for deployment on a Xilinx Virtex-5 FPGA [11]. The architecture is presented in detail, including the precision of the arithmetic computations assuming two 14-bit, two's complement analog-to-digital converters (ADCs) are providing input samples to the FPGA and 16-bit precision for baseband processing operations, for example, the fast Fourier transform (FFT) computation. The architecture is described at the register transfer level (RTL) using VHDL [12], and implementation results (including the utilization of built-in ASIC resources) are reported. The performance of the architecture is evaluated under an additive white Gaussian noise (AWGN) channel with multipath conditions that model an indoor office environment. We propose novel and practical modifications to existing architectures, which result in efficient implementation on FPGA. The two most important contributions are the reduction of the CFO estimator area complexity and a new monotonic quantization scheme for the fine time synchronization cross-correlation computation. Even though the remainder of this paper focuses on synchronization using an IEEE 802.11a preamble sequence, the design methodology and complexity reduction techniques proposed can be easily extended to other OFDM synchronizers, including those in IEEE 802.11n [2] and IEEE 802.16 [13].

The remainder of this paper is organized as follows. In Section 2, the preliminaries are reviewed. In Section 3, our design is presented. In Section 4, the FPGA implementation results are reported. Finally, in Section 5, we summarize the work and propose future extensions.

2. PRELIMINARIES

In this section, we review the IEEE 802.11a OFDM packet format, explain the available Xilinx Virtex-5 FPGA resources that influence our design, and present the channel model used to evaluate the performance of the synchronizer blocks.

2.1. IEEE 802.11a orthogonal frequency-division multiplexing packet format

The format of an IEEE 802.11a OFDM packet is illustrated in Figure 1, and the preamble (training sequence) is used for synchronization in both frequency and time. In IEEE 802.11a [2], there are two training sequences, the short training sequence (STS) and the long training sequence (LTS). The STS consists of 10 copies of a 16-sample complex sequence, whereas the LTS consists of a 32-sample cyclic prefix followed by two copies of a 64-sample complex sequence. The received analog complex baseband signal is sampled by a pair of ADCs operating at 20 MHz; therefore, the duration of a sample is $T_s = 50$ ns, which translates to both the STS and LTS being 8 μs in length. Typically, the STS is used for AGC, packet detection, coarse CFO estimation, and coarse timing synchronization. On the other hand, the LTS is used for fine CFO estimation and fine timing synchronization [2]. Lastly, to reduce the probability of ISI, increase the robustness to multipath fading, and increase the tolerance range for time synchronization, a 16-sample cyclic prefix is added to each OFDM data symbol.

2.2. Xilinx Virtex-5 field-programmable gate array device

We have selected the Xilinx Virtex-5 xc5vlx110t-1ff1136 device [11] as the target FPGA for the synchronizer architecture as the research team is in possession of development boards with that FPGA mounted. In terms of available resources, xc5vlx110t-1ff1136 contains 17 280 slices, 64 DSP48E slices, 148 random access memory (RAM) blocks, and 680 input/output (I/O) pins. Each slice contains four flip-flops (FFs) and four look-up tables (LUTs). The DSP48E slices have an embedded, signed, 18-bit by 25-bit multiplier, adder, and accumulator. The block RAMs are dual port, are 36 kilobits in size, are highly configurable in terms of addressing modes, and have a maximum data bus width of 36 bits per port. The numerous I/O pins contain FFs that can clock the input and output signals.

2.3. Channel model

The channel model used to evaluate the performance of the forthcoming designs incorporates the effects of multipath fading, CFO, and AWGN. In Figure 2, we provide

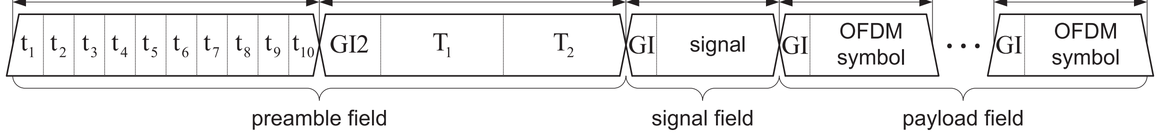


Figure 1. Packet format of the IEEE 802.11a PHY [2]. OFDM, orthogonal frequency-division multiplexing; GI, guard interval.

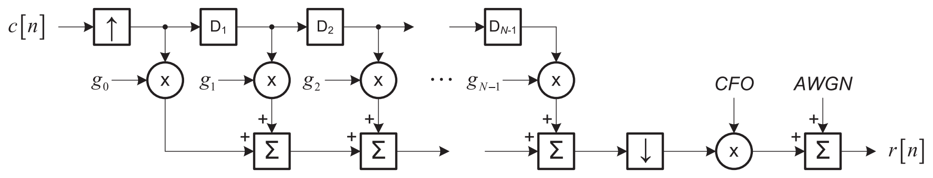


Figure 2. Illustration of the channel model for design evaluation. CFO, carrier frequency offset; AWGN, additive white Gaussian noise.

an illustration of the structure that generates the received samples which are used to evaluate the designs. To model multipath fading, a tapped-delay filter (TDF) is used, where the delay and gain coefficients are taken from the European Telecommunications Standards Institute (ETSI) delay profile for an indoor office environment with non-line-of-sight propagation (ETSI A Channel Model [14]). Note that first-order interpolation is used for up-sampling, the CFO is applied to the multipath preamble sequence as a complex multiplication, and MATLAB (MathWorks, Natick, MA, USA) is used to generate the AWGN values. In terms of notation, we let $c[n] = c_{\mathrm{re}}[n] + \mathrm{i}c_{\mathrm{im}}[n]$ denote the nth, real complex-pair of transmitted preamble samples constructed from the equations in [2]; and $r[n] = r_{\mathrm{re}}[n] + \mathrm{i}r_{\mathrm{im}}[n]$ denote the nth, quantized, 14-bit, two's complement, complex-pair of received samples output from the ADCs. Unless otherwise stated, the received samples $r[n]$ used during the simulations in this paper are generated under the following conditions:

- delay spread of 50 ns,
- CFO of +100 kHz, and
- signal-to-noise (SNR) ratio of +12 dB

Note that the SNR is referenced at the down-sampled output of the TDF.

Recall that the baseband processing specification of the FFT computation is 16-bit and 14-bit ADCs will be used to sample the received analog signal. Through Monte Carlo experiments, we found that in order to avoid arithmetic overflow, a safe fixed-point, two's complement representation for 16-bit IEEE 802.11a baseband processing uses 6 integer (including the sign bit) and 10 fractional bits. We will assume that AGC has been successfully completed; therefore, we quantize the received samples to have 6 integer and 8 fractional bits. However, we will design the architecture to be robust against arithmetic overflow under ADC saturation conditions.

3. METHODOLOGY

In this section, we present the design of the packet detector, CFO estimator and corrector, as well as the time synchronizer. For each block, we present the related mathematics, propose their architecture for implementation on the selected FPGA, and simulate the architecture under the prescribed channel model. We have completed the design of the synchronizer in the time domain, that is, before the FFT operation on the OFDM receiver, thereby simplifying the receiver control logic. Let $N_{\mathrm{s}} = 16$ and $N_{\mathrm{l}} = 64$ denote the number of samples in STS and LTS symbols, respectively.

3.1. Packet detector

The primary role of the packet detector is to alert the receiver that a packet is being transmitted. The packet detector architecture presented in this paper uses the well known STS delayed autocorrelation/average power thresholding technique proposed in [4].

3.1.1. Mathematics

We opted to implement the thresholding technique across one STS symbol averaged over a period of $L_{\mathrm{s}} = 16$ samples; then, the delayed autocorrelation computation is defined as

$$R[n] = \sum_{m=0}^{L_{\mathrm{s}}-1} r^*[n+m] \cdot r[n+m+N_{\mathrm{s}}] \tag{1}$$

where $r^*[\cdot]$ denotes complex conjugate of $r[\cdot]$. The average power computation is defined as

$$P[n] = \sum_{m=0}^{L_{\mathrm{s}}-1} |r[n+m]|^2 \tag{2}$$

Then, the STS of a packet can be detected when the ratio of

$$M[n] = \frac{|R[n]|^2}{(P[n])^2} \tag{3}$$

crosses a predetermined threshold, that is, $M[n] > th_{\mathrm{pd}}$, for a specified number of samples [4].

It is known that Equations (1) and (2) can be realized using a sliding window to reduce the area complexity [4]. For the sliding window implementation of Equation (1), its form is

$$\begin{aligned} R[n+1] = {} & R[n] - r^*[n-N_{\mathrm{s}}] \cdot r[n] \\ & + r^*[n] \cdot r[n+N_{\mathrm{s}}] \end{aligned} \tag{4}$$

and the approach is similar for Equation (2), that is,

$$P[n+1] = P[n] - |r[n-N_{\mathrm{s}}]|^2 + |r[n]|^2 \tag{5}$$

As division is generally an expensive operation to implement in hardware, the detection metric in Equation (3) can be alternatively realized as

$$|R[n]|^2 > th_{\text{pd}} \cdot (P[n])^2, \quad n > 0 \tag{6}$$

When selecting a value for th_{pd}, two factors must be considered:

- miss detection (MD) probability and
- false alarm (FA) probability.

A larger th_{pd} decreases the FA probability but increases the MD probability. Conversely, a smaller th_{pd} decreases the MD probability but increases the FA probability. Additionally, the number of consecutive true evaluations of Equation (6) can be varied before declaring a packet detected; increasing it beyond one sample decreases the FA probability but increases the MD probability and delays the detection time. In the literature, different values are suggested for th_{pd} (over different periods), for example, 0.5 (1 comparison), 0.5 (8 of the last 32 comparisons), 0.75 (1 comparison), and 0.81 (5 consecutive comparisons) in [15], [10], [16], and [17], respectively. In this architecture, we select $th_{\text{pd}} = 0.75$ because it is determined to be a good choice by simulation and the multiplication operation in Equation (6) can be implemented as an addition with the addends right-shifted [16].

3.1.2. Implementation

An illustration of the implementation of the packet detector architecture is shown in Figure 3. The top and bottom branches realize the delayed autocorrelation computation in Equation (4) and the average power computation in Equation (5), respectively. The 16-cycle delay blocks (denoted by 16D) are each implemented using a block RAM, a counter, and a simple finite state machine, with the input wired to block RAM port A and the delayed output on port B. The complex multiplication (with conjugation) is implemented using three DSP48E multipliers and five adder-subtractors as

$$(a_{\text{re}} + \mathrm{i}a_{\text{im}}) \cdot (b_{\text{re}} - \mathrm{i}b_{\text{im}}) = p_{\text{re}} + \mathrm{i}\,p_{\text{im}}$$

where

$$p_{\text{re}} = a_{\text{re}} \cdot b_{\text{re}} + a_{\text{im}} \cdot b_{\text{im}}$$

and

$$p_{\text{im}} = (a_{\text{re}} + a_{\text{im}}) \cdot (b_{\text{re}} - b_{\text{im}}) - (a_{\text{re}} \cdot b_{\text{re}} - a_{\text{im}} \cdot b_{\text{im}})$$

The power of a complex sample is computed as its squared magnitude, that is,

$$|r[n]|^2 = r_{\text{re}}[n] \cdot r_{\text{re}}[n] + r_{\text{im}}[n] \cdot r_{\text{im}}[n]$$

where this operation requires two multipliers and one adder. Because the ADC samples are 14-bit, the precision of the results from the multiplication operations for the delayed autocorrelation and average power computations are 29-bit. To prevent overflow in the three sliding window accumulators, the widths of the adder–subtractors and registers must be selected appropriately. Considering that the largest magnitude value that a 14-bit ADC can output is -2^{13}, the sliding window accumulators have to be capable of safely representing the sum of 16 copies of $2 \times (-2^{13}) \times (-2^{13})$, that is,

$$\sum_{0}^{15} \left(2 \cdot 2^{13} \cdot 2^{13}\right) = 2^{31}$$

Note that to represent $+2^{31}$ in two's complement requires 33-bits. However, if one clips the maximum magnitude ADC value from -2^{13} to $-2^{13} + 1$, all the sliding window accumulators can be implemented using 28-bit input and 32-bit output widths, resulting in hardware savings. This clipping technique will be applied to many other computations within this synchronizer architecture; however, the data forwarded to baseband processing will not be subjected to this operation.

With 32-bit two's complement outputs from the two delayed autocorrelation branch window accumulators and a 31-bit unsigned output from the power branch window accumulator in Figure 3, we discuss the precision of the

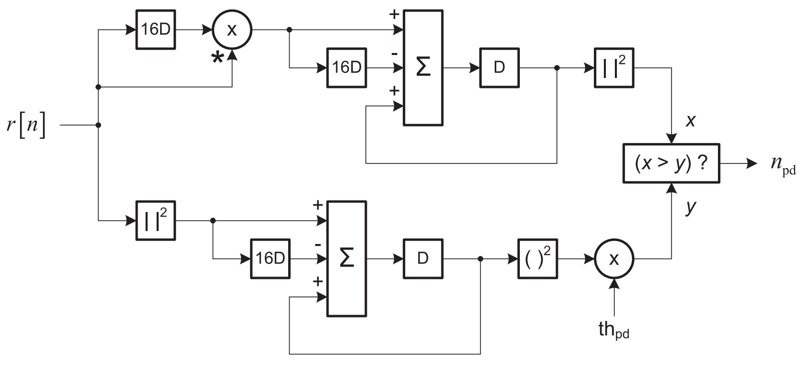


Figure 3. Illustration of the packet detector architecture.

computations in Equation (6). The output precision of the squared magnitude operation on the delayed autocorrelation branch is 64-bit unsigned, whereas the output precision of the squaring operation on the average power branch is 62-bit. The threshold multiplication is realized as a 62-bit adder; then, the sum is extended and requires a 64-bit unsigned comparator. Note that four DSP48E slices are required to realize a multiplication of two 31-bit or 32-bit numbers on a Xilinx Virtex-5 device, consequently a total of 12 DSP48E slices are utilized in the implementation of Equation (6).

3.1.3. Simulations

An illustration of a typical response for the terms in Equation (6) over the duration of the preamble is shown in Figure 4. We note that the data for this plot is obtained from functional simulations of the RTL model. One observes the rapid rise of the two values during the transmission of the STS. Then, $|R[n]|^2$ quickly drops off once the LTS samples begin arriving. We note that the sharp drop in $|R[n]|^2$ will be used for coarse time synchronization. The value of $th_{\mathrm{pd}} \cdot (P[n])^2$ varies over the entire preamble, and it is less than $|R[n]|^2$ for some parts of the STS.

3.2. Carrier frequency offset estimator and corrector

Because of hardware limitations of the local oscillator, a CFO will exist in the received packet samples. Similar to the approach described in [4], the CFO estimator architecture presented in this paper uses the phase angle of the delayed autocorrelated STS and LTS symbols for the computation the coarse and fine CFO estimates, respectively. We adopt the strategy presented in [8], where after the coarse CFO estimate is computed, it is applied to correct the future input samples. Then, the fine CFO estimate is calculated over coarse CFO corrected samples, and it is subsequently applied to them before being passed on to the FFT for baseband processing.

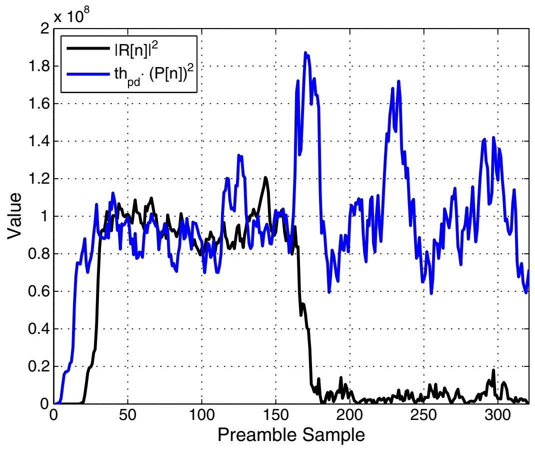


Figure 4. Response of the packet detection metrics over the duration of a received preamble sequence.

3.2.1. Mathematics

It is known that for two identical samples on the receiver, their phase difference is proportional to the CFO and the time difference between them [4]. Under noiseless conditions, the received samples can be expressed as

$$r[n] = e^{j2\pi n T_{\mathrm{s}} f + \phi_0} \cdot c[n] \tag{7}$$

If $c[n]$ represents a sample somewhere in the middle of the STS, then $c[n] = c[n + N_{\mathrm{s}}]$. Therefore, substituting Equation (7) and performing the following manipulations, Equation (1) can be rewritten as

$$\begin{aligned} R[n] &= \sum_{m=0}^{L_{\mathrm{s}}-1} e^{-j2\pi(n+m)T_{\mathrm{s}} f - \phi_0} \cdot e^{j2\pi(n+m+N_{\mathrm{s}})T_{\mathrm{s}} f + \phi_0} \\ &\quad \cdot c^{*}[n+m] \cdot c[n+m] \\ R[n] &= e^{j2\pi N_{\mathrm{s}} T_{\mathrm{s}} f} \cdot \sum_{m=0}^{L_{\mathrm{s}}-1} |c[n+m]|^2 \end{aligned} \tag{8}$$

From Equation (8), it is clear that the phase, $\Delta\phi$, of $R[n]$ can be computed from

$$\Delta\phi = \angle R[n] \tag{9}$$

where $\angle R[n]$ denotes the angle of $R[n]$. By substituting $\Delta\phi = 2\pi N_{\mathrm{s}} T_{\mathrm{s}} f$ into Equation (9), the coarse CFO estimate, $\hat{f_c}$, can be computed as

$$\hat{f_c} = \frac{\angle R[n_{\mathrm{cfo}}]}{2\pi \cdot N_{\mathrm{s}} \cdot T_{\mathrm{s}}}, \quad n_{\mathrm{cfo}} > 0 \tag{10}$$

where n_{cfo} denotes the STS sample index selected for the coarse CFO estimate. The coarse CFO estimate can be applied to the input samples as

$$r'[n] = r[n] \cdot e^{-j2\pi n T_{\mathrm{s}} \hat{f_c}} \tag{11}$$

and by substituting Equation (10) into Equation (11), one obtains

$$r'[n] = r[n] \cdot e^{-jn\angle R[n_{\mathrm{cfo}}]/N_{\mathrm{s}}} \tag{12}$$

Note that a similar formulation follows for fine CFO operations, when the estimation is computed over the coarse corrected the LTS samples with $N_{\mathrm{l}} = 64$ and $L_{\mathrm{l}} = 16$, that is,

$$R'[n] = \sum_{m=0}^{L_{\mathrm{l}}-1} r'^{*}[n+m] \cdot r'[n+m+N_{\mathrm{l}}]$$

Then, the fine estimate is applied to the $r'[n]$ sequence, with the goal of having the final corrected sequence $r''[n]$ closely approximate the transmitted sequence, that is, $r''[n] \approx c[n]$.

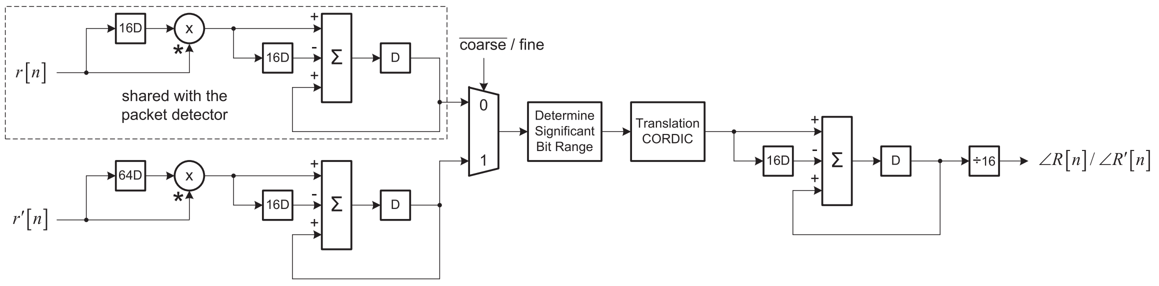

16D
x
16D
Σ
D
shared with the packet detector
$r[n]$
64D
16D
$r'[n]$
coarse / fine
0
1
Determine Significant Bit Range
Translation CORDIC
÷16
$\angle R[n] / \angle R'[n]$

Figure 5. Illustration of the carrier frequency offset estimator architecture. CORDIC, coordinate rotation digital computer.

3.2.2. Implementation

The implementation of the CFO estimator is illustrated in Figure 5. One observes that the delayed autocorrelation hardware for the STS can be shared with the packet detector [10]. The angle in Equation (10) is calculated by using a translation coordinate rotation digital computer (CORDIC) [18] and implemented by the CORDIC v4.0 LogiCORE (Xilinx, Inc., San Jose, CA, USA) [19]. Because the STS and LTS are not processed concurrently, one can time multiplex the translation CORDIC in the CFO estimator [10]. As the computed angle in Equation (9) will fluctuate over the durations of STS and LTS, we propose including an averaging circuit, which operates over a 16-sample duration to smooth the result.

The architectures of the coarse and fine CFO correctors are similar, and we have illustrated the coarse CFO corrector in Figure 6. The central block is a rotation CORDIC [18] that accepts the scaled angle, $\angle R[n_{\text{cfo}}]/N_{\text{s}}$, from the translation CORDIC in Figure 5 and applies multiples of it to the received samples as defined in Equation (12). As $N_{\text{s}} = 2^4$, the division operation in Equation (12) is implemented as a simple 4-bit right shift, whereas the multiplication operation is realized iteratively by addition as

$$\theta[n] = \theta[n-1] + \frac{\angle R[n_{\text{cfo}}]}{N_{\text{s}}} \tag{13}$$

where $\theta[n] = n\angle R[n_{\text{cfo}}]/N_{\text{s}}$.

The Cartesian inputs to the CORDIC must be within the interval $[-1.0, +1.0]$ and are formatted as si.fff⋯f, while the angle inputs must be within the interval $[-\pi, +\pi]$ and are formatted as sii.fff⋯f [19]. To prevent Cartesian overflow, we sign extend the inputs by 1 bit, that is, ssi.fff⋯f, which is interpreted by the CORDICs as ss.ifff⋯f, effectively dividing them by two and ensuring that they are always within the acceptable interval. As the value of $\theta[n]$ in Equation (13) will grow and eventually fall outside of the interval $-\pi \leq \theta[n] \leq +\pi$, depending on the sign of $\angle R[n_{\text{cfo}}]$, a circuit is required to add $\pm 2\pi$ when necessary to prevent angle overflow. We note that to perform the addition of $\theta[n] \pm 2\pi$, a 1-bit sign extension is applied to $\theta[n]$, which is subsequently discarded.

Recall that our baseband processing specification states that the FFT operation is to be completed with 16-bit precision. Consequently, the coarse CFO corrected samples $r'[n]$ in Equation (12) output from the rotation CORDIC can be 16-bit values. To accomplish this, we sign extend the 14-bit $r[n]$ values to 15-bit (to avoid Cartesian overflow) and pad three least significant zeros (two zeros to increase the precision of the outputs to 16 bits and one zero to reduce the rotation CORDIC output quantization error), that is, ssi.fff⋯f000. Afterward, we discard the most significant bit (MSB) and least significant bit (LSB) from the two Cartesian outputs and obtain a 16-bit $r'[n]$ result. The procedure is similar for the fine CFO correction, with $r''[n]$ being computed from $r'[n]$ requiring only a 1-bit sign extension and a 1-bit zero pad for the $r'[n]$ Cartesian inputs, and again, the MSB and LSB are discarded from the $r''[n]$ outputs. Therefore, we select the precision of both CFO correction rotation CORDICs to be $N_{\text{cr}} = 18$.

We note that the dominating design parameter in terms of area complexity in these CFO logic blocks is the size of the CORDICs (a function of their precision). In [16], the precision of the CFO estimation translation CORDIC is $N_{\text{ct}} = 40$, which we found through synthesis experiments, occupies approximately 7.9% and 8.0% of the

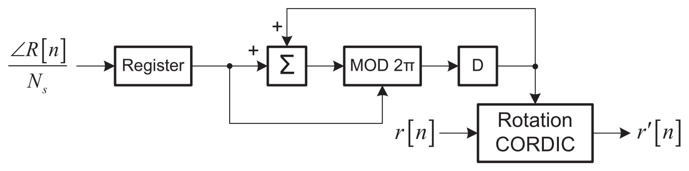


Figure 6. Illustration of the coarse carrier frequency offset corrector architecture. CORDIC, coordinate rotation digital computer.

xc5vlx110t-1ff1136 device slice FFs and LUTs, respectively. In fact, after applying the clipping technique to the 16-bit $r'[n]$ values, the output precision from the delayed autocorrelation accumulator on the bottom branch in Figure 5 is 36-bit. Therefore, a 1-bit sign extension and a 1-bit zero pad result in use of a 38-bit translation CORDIC, clearly at a large hardware penalty and latency cost.

Here, we propose a novel method to reduce the precision of the translation CORDIC without sacrificing significant performance. Recall that the angle input to the rotation CORDICs in the CFO correctors are computed by the translation CORDIC in the CFO estimator (see Figure 5). Because we previously selected $N_{\mathrm{cr}} = 18$ to meet the baseband processing specification, the number of bits accepted for the angle input in the two rotation CORDICs is 18. Therefore, the minimum translation CORDIC precision is found as

$$N_{\mathrm{ct}} = N_{\mathrm{cr}} - \min\left(\log_2 N_{\mathrm{s}}, \log_2 N_{\mathrm{l}}\right)$$

because of the division in Equation (13). In other words, using precision greater than $N_{\mathrm{ct}} = 14$ generates angle bits that will be discarded by the rotation CORDICs. However, we select $N_{\mathrm{ct}} = 16$ to prevent Cartesian overflow and reduce output quantization error [19]. Now, we make the following two observations:

- one is solely interested in $\angle R[n]$ and not $|R[n]|$ output from the translation CORDIC, and
- the greatest magnitude bits of $R[n]$ dominate the result of $\angle R[n]$.

Consequently, the lower order bits of $R[n]$ can be neglected in the computation of $\angle R[n]$. To accomplish this, one must include a logic circuit which ensures that the relevant portions of $R[n]$ are input to the translation CORDIC.

3.2.3. Simulations

To evaluate the CFO estimator architecture, we conduct functional simulations using the RTL model. The effect of the averaging circuit on the CFO estimate is shown in Figure 7. By including an averaging circuit, the output of the estimator is smoothed; consequently, the upper bound on the CFO estimate error is reduced. This is important, as there is no time synchronization at this point and any of the CFO values on the plot can be selected as the coarse CFO estimate (as the system has no *a priori* knowledge of the actual CFO).

Next, we investigate the effect of using a full sized 32-bit translation CORDIC versus the proposed 16-bit translation CORDIC technique. In Figure 7, we show a plot of the two responses compared with the actual CFO over a portion of the STS. One observes that there is virtually no difference between their estimates, but there is a significant area complexity reduction by using a smaller translation CORDIC and bit-range circuit.

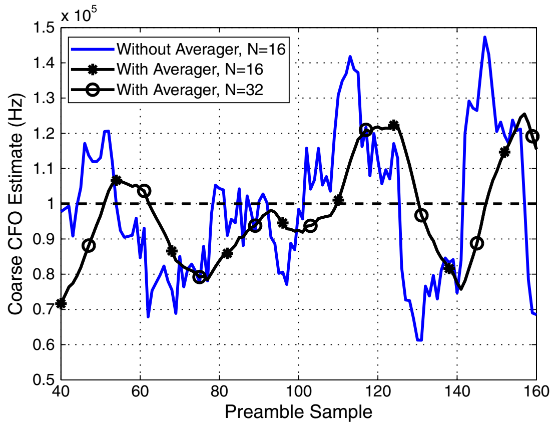


Figure 7. Response of the coarse carrier frequency offset estimator logic over a portion of the short training sequence. The dotted line denotes the actual carrier frequency offset (CFO).

3.3. Time synchronizer

The first six or seven samples in the 16-sample cyclic prefix of OFDM data symbols will be distorted because of multipath channel conditions and filtering. Therefore, there is a window of nine samples where the FFT computation can begin without introducing ISI [20]. By processing the preamble, the time synchronizer determines which sample is to be selected as the starting point for baseband processing on the receiver. In this architecture, the coarse time synchronizer locates a sample somewhere in the LTS cyclic prefix symbol. Afterward, the fine time synchronizer improves the coarse time estimate and locates a specific sample in the second LTS symbol. The coarse time synchronizer uses the STS delayed autocorrelation drop-off detector approach proposed in [8] and modified by [10]. For the fine time synchronizer, we propose some modifications to the existing quantized cross-correlation approaches.

3.3.1. Mathematics

After a packet has been detected by the metric in Equation (6), the coarse time synchronizer waits until the value of $|R[n]|^2$ falls below a threshold of its maximum, that is,

$$|R\,[n]|^2 < th_{\mathrm{ct}} \cdot \max |R\,[n]|^2, \quad n > n_{\mathrm{pd}} \tag{14}$$

and declares that sample as the coarse time estimate n_{ct}. At this point, the current $r[n]$ samples are expected to be from the LTS cyclic prefix [8]. We select $th_{\mathrm{ct}} = 0.25$ because of the ease of computing the multiplication in Equation (14) as a simple 2-bit right shift operation and to reduce the FA probability. Like most architectures, an FA will result in a time synchronization failure, and in the forthcoming simulations, we evaluate this selection.

For fine time synchronization, it has been shown that LTS cross-correlation approaches outperform autocorrelation ones but have larger area complexities [5]. In typical approaches, the ideal LTS samples are cross-correlated with the received samples, that is,

$$\Lambda[n] = \sum_{m=0}^{N_l-1} c^*[m] \cdot r[n+m] \tag{15}$$

where $\Lambda[n] = \Lambda_{\text{re}}[n] + \mathrm{i}\Lambda_{\text{im}}[n]$ and $c^*[m]$ denote the complex conjugate of the mth LTS sample. Then, one can arrive at a fine time estimate from

$$n_{\text{ft}} = \arg\max_n |\Lambda[n]|, \quad n > n_{\text{ct}} \tag{16}$$

and Equation (16) would ideally yield two n values when a sequence of N_l consecutive $r[n]$ samples exactly overlap the first and second LTS symbols in time.

To realize Equations (15) and (16) in hardware, we propose performing N_l cross-correlation computations in parallel using different cyclically shifted versions of the LTS, commencing after the coarse time estimate as

$$\Lambda_i = \sum_{m=0}^{N_l-1} c_i^*[m] \cdot r[n_{\text{ct}} + 48 + m], \quad 0 \le i \le N_l - 1 \tag{17}$$

where

$$c_i^*[m] = c^*[(i+m) \bmod N_l]$$

As n_{ct} represents the index of a sample located in the cyclic prefix of the LTS, we wait for 48 samples to begin the cross-correlation computations. This delay ensures that the first sample processed in Equation (17) is within the first LTS symbol. Then, the fine time estimate is obtained as

$$n_{\text{ft}} = \arg\max_i |\Lambda_i| + \Omega \tag{18}$$

where $\Omega = 192$ is the index of the first sample following the 32-sample cyclic prefix in the LTS.

Note that one complex multiply-accumulate (MAC) magnitude circuit is required per parallel Λ_i computation in Equation (18). To reduce the ASIC resource footprint of the fine time synchronizer, one can quantize the $c[m]$ samples such that DSP48E multipliers are not required to perform the multiplication computations in Equation (17). For example, see [6] and [7] for two different quantization approaches. Similar to [7], we map the LTS samples to zero and positive or negative powers of 2, that is,

$$c[n] = c_{\text{re}}[n] + \mathrm{i}c_{\text{im}}[n] \to q[n] = q_{\text{re}}[n] + \mathrm{i}q_{\text{im}}[n]$$

where

$$q_{\text{re}}[n],\ q_{\text{im}}[n] \in \left\{0, \pm 2^0, \pm 2^1, \cdots, \pm 2^i\right\}$$

and i denotes the quantization factor. However, in this paper, to ensure monotonicity, the quantization functions and cross-correlation multiplications are defined differently compared with [7]. Let us introduce two functions $Q_1(x)$ and $Q_2(x)$ that together will perform the quantization of the LTS, as

$$Q_1(x) \triangleq \frac{2^i \cdot x}{\max(|c_{\text{re}}[n]|, |c_{\text{im}}[n]|)}$$

and

$$Q_2(x) \triangleq \begin{cases} +2^{\text{int}(\log_2 x)}, & x \ge +1 \\ -2^{\text{int}(\log_2(-x))}, & x \le -1 \\ 0, & \text{otherwise} \end{cases}$$

where the function int(x) returns the closest integer to x. Then, define the set of quantized LTS symbols as

$$q[n] \triangleq Q_2(Q_1(c_{\text{re}}[n])) + \mathrm{i}Q_2(Q_1(c_{\text{im}}[n]))$$

Now, the cross-correlation in Equation (15) can be approximated as

$$\Lambda[n] \approx \sum_{m=0}^{N_l-1} q^*[n] \cdot r[n+m]$$

The estimate of the cross-correlated value can be calculated from Equations (19) and (20), that is,

$$\Lambda_{\text{re}}[n] \approx \sum_{m=0}^{N_l-1} (s_{\text{re}}[n] \times (r_{\text{re}}[n+m] \ll l_{\text{re}}[n]) - s_{\text{im}}[n] \times (r_{\text{im}}[n+m] \ll l_{\text{im}}[n])) \tag{19}$$

$$\Lambda_{\text{im}}[n] \approx \sum_{m=0}^{N_l-1} (s_{\text{re}}[n] \times (r_{\text{im}}[n+m] \ll l_{\text{re}}[n]) + s_{\text{im}}[n] \times (r_{\text{re}}[n+m] \ll l_{\text{im}}[n])) \tag{20}$$

where $\ll$ denotes a left shift operation, $\Re\{\cdot\}$ denotes the real component,

$$s_{\text{re}}[n] = \begin{cases} +1, & \Re\{q^*[n]\} > 0 \\ -1, & \Re\{q^*[n]\} < 0 \\ 0, & \Re\{q^*[n]\} = 0 \end{cases}$$

and

$$l_{\text{re}}[n] = \begin{cases} \log_2|\Re\{q^*[n]\}|, & \Re\{q^*[n]\} \ne 0 \\ 0, & \Re\{q^*[n]\} = 0 \end{cases}$$

Similar expressions can be obtained for $s_{\text{im}}[n]$ and $l_{\text{im}}[n]$ using the imaginary component of $q^*[n]$, that is, $\Im\{q^*[n]\}$.

Finally, to perform the complex magnitude operation in Equation (18), we adopt the strategy in [16] where the following approximation is used

$$|\Lambda[n]| \approx \max(|\Lambda_{\mathrm{re}}[n]|, |\Lambda_{\mathrm{im}}[n]|) + \frac{\min(|\Lambda_{\mathrm{re}}[n]|, |\Lambda_{\mathrm{im}}[n]|)}{2}$$

which avoids the use of DSP48E multipliers and the square root computation in hardware.

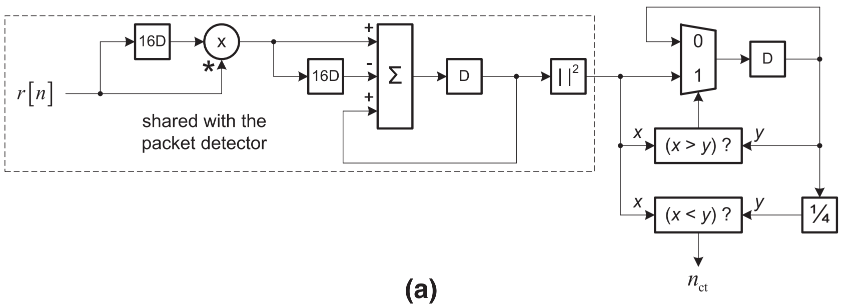
3.3.2. Implementation

The implementation of the coarse and fine time synchronizer architectures are illustrated in Figures 8a and 8b, respectively. Similar to the CFO estimator, the coarse time synchronizer shares a portion of its logic with the packet detector [10]. After a packet has been detected, the logic that implements Equation (14) is activated, and the coarse time estimate is signalled to the fine time synchronizer. Two notable advantages of this coarse time synchronizer design are its low area complexity and implementation simplicity.

The proposed fine time synchronizer architecture is a modification of the parallel cross-correlator presented in [5] with the incorporation of quantized multipliers. In this architecture, the correlated values for all the sequences are output at the same time, whereas in [5], each sequence is computed with additional latency. To represent the quantized LTS values, that is, $q^*[n]$, we used a signed magnitude format, requiring $\lceil \log_2(i+2) \rceil$ bits per numeric value. The circular shift register is implemented with FFs, has a width of $2 \cdot \lceil \log_2(i+2) \rceil + 2$ bits, and an initialization option where the nth register can be loaded with $\langle \Re\{q^*[n]\}, \Im\{q^*[n]\} \rangle$. The outputs from the positions in the circular shift register generate the $q_i^*[n]$ sequences. The quantized multiplications are implemented with a left bit shifter and a negation circuit.

Using clipped $r'[n]$ values, the precision of the products from the quantized complex multiplications in Equations (19) and (20) are $(i+16)$-bit. Therefore, the precision of the 64-sample complex accumulators are $(i+22)$-bit. The implementation of the magnitude operation as computed by the complex magnitude approximation in [16] is trivial in hardware. For a $(i+22)$-bit two's complement complex number, its magnitude requires $(i+22)$ bits in unsigned representation. Finally, the maximum circuit is implemented using a divide-and-conquer approach.

By performing quantized complex multiplication operations, one is effectively replacing the embedded ASIC multipliers with general purpose LUT logic. Through implementation experiments, we found that it is undesirable to implement 64 MAC branches as illustrated in Figure 8b, because the computation logic with $i = 3$ would occupy approximately $64 \times 0.78\% = 49.92\%$ of



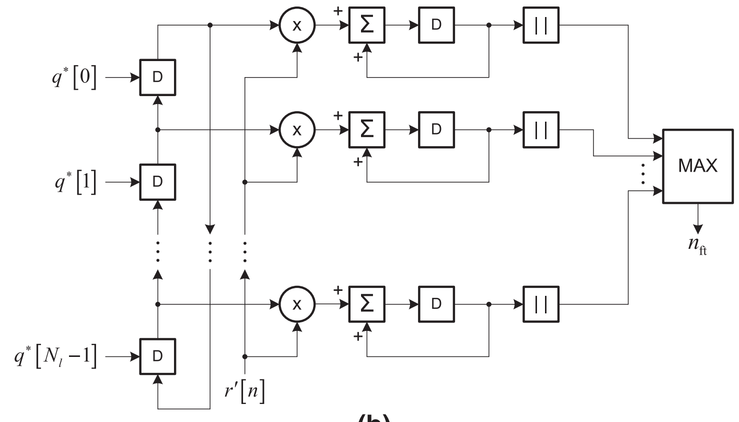


Figure 8. Illustrations of the time synchronization architectures: (a) coarse time synchronizer, (b) fine time synchronizer.

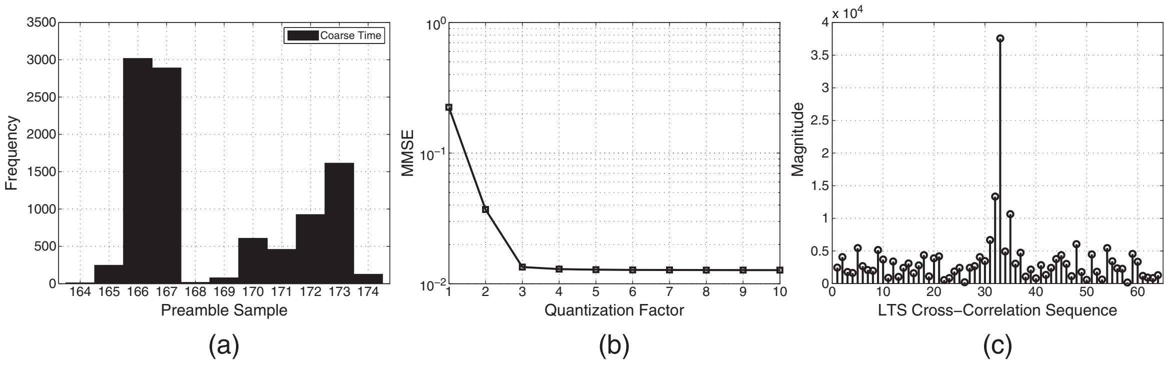


Figure 9. Time synchronization simulations: (a) histogram of the coarse time synchronization estimates over 10 000 iterations, (b) plot of the MMSE for LTS quantization, (c) response of a 64-branch quantized cross-correlation operation for fine time synchronization. LTS, long training sequence; MMSE, minimum mean square error.

the available FPGA slice LUT resources. In fact, the minimum number of required MAC branches depends on the confidence of the coarse time estimate. Through forthcoming evaluation, we will find that the coarse time estimate is accurate to within an 11-sample interval under the prescribed channel model. However, for increased reliability, the fine time synchronizer is constructed with 16 MAC branches. Recall that there is a nine-sample window for robust time synchronization [20], and we select sample 12 as the fine time synchronizer target for the OFDM data symbols.

3.3.3. Simulations

It was previously mentioned that we selected $th_{\mathrm{ct}} = 0.25$ in an attempt to reduce the FA probability. To find the confidence of the coarse time estimate, we constructed a bit-accurate model of the packet detector and coarse time synchronizer architectures using the MATLAB Fixed-Point Toolbox 3.0 [21]. We performed 10 000 iterations using different random seeds to generate the AWGN values. In Figure 9a, the results of the coarse time simulations are plotted as a histogram. The reader observes that the coarse time estimate is accurate to within an 11-sample interval $164 \leq n_{\mathrm{ct}} \leq 174$ and zero FAs occurred. The fact that we observe zero FAs justifies the selection of $th_{\mathrm{ct}} = 0.25$ under the prescribed channel model. For the fine time synchronizer, we assume the coarse time estimate to be reliable to the range $161 \leq n_{\mathrm{ct}} \leq 176$. Now, recall that it waits 48 samples after n_{ct} before commencing the cross-correlation operation. Thus, the 16 MAC branches compute cross-correlations with sequences $q^*_{16}[m]$ to $q^*_{31}[m]$.

To investigate the effect that the quantization factor has on accuracy, Figure 9b shows the minimum mean square error (MMSE) of the quantized LTS values for various quantization factors. Note that the MMSE is measured by normalizing both the floating-point and quantized LTS sequences. We observe significant performance improvements by increasing the quantization factor from $i = 1$ to $i = 3$, with marginal gains thereafter. Since area complexity increases with i, we select $i = 3$ for our implementation as it offers a low MMSE combined with reduced area complexity.

In Figure 9c, we show the response of a 64-branch quantized cross-correlator to the LTS under the prescribed channel model without any CFO. We note that no CFO has been applied to the input samples because the coarse CFO correction will be applied beforehand, and the values are obtained by functional simulations of the RTL model. The reader observes that only one clear peak exists, which is several orders of magnitude greater than the next greatest value.

3.4. Packet synchronizer

With each of the blocks presented, we proceed to discuss their interconnections to form the complete packet synchronizer. In Figure 10, we provide a high-level block diagram of the proposed synchronizer architecture. The reader observes that most of the internal connections are trivial. The 22-cycle delays on the n_{ct} and n_{ft} signals are included to account for the rotation CORDIC latency in the CFO correctors and are implemented with the SRLC32E primitive. The 20-cycle delay on the $r'[n]$ bus is to allow time to complete the fine CFO estimation computation. Similar to the other delays in the architecture, it is implemented with a block RAM. The reader observes that the clipping logic is only located on the data paths used by synchronizer logic and will not modify the data forwarded to baseband processing.

4. FIELD-PROGRAMMABLE GATE ARRAY IMPLEMENTATION RESULTS

For all the FPGA synthesis results in the paper, the target device is a Xilinx Virtex-5 xc5vlx110t-1ff1136 FPGA [11], and Xilinx ISE Design Suite 12.1 is used. The synthesis tool is XST (Xilinx, Inc., San Jose, CA, USA) with the optimization goal of area selected. The place and route tool is PAR (Xilinx, Inc., San Jose, CA, USA), with the

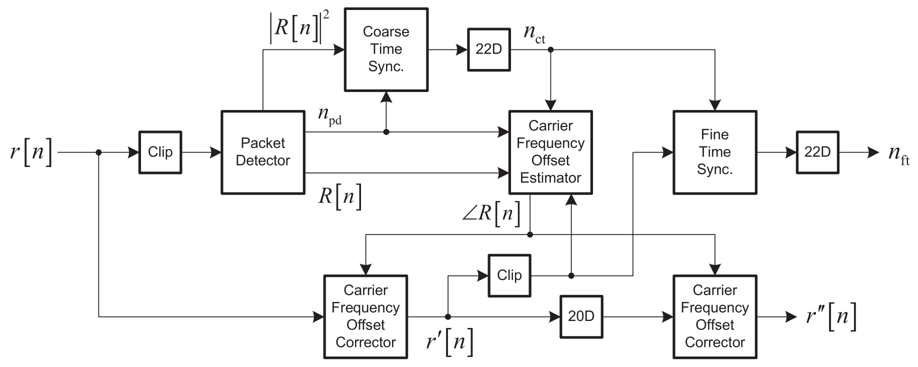


Figure 10. Illustration of the high-level synchronizer architecture.

Table I. Field-programmable gate array implementation results on xc5vlx110t-1ff1136.

	Slice FF	Slice LUT	Block RAM	DSP48E	Timing (ns)
Block					
Packet detector	116	514	4	17	16.582
Coarse time synchronizer	65	67	0	0	5.914
CFO estimator	1029	2027	4	3	15.441
Coarse CFO correlator	1423	1474	0	0	6.085
Fine CFO correlator	1429	1471	0	0	6.085
Fine time synchronizer	1340	8239	0	0	13.232
Complete synchronizer	5471	1403	9	20	15.562
	7.92%	20.31%	6.08%	31.25%	–

FF, flip-flop; LUT, look-up table; RAM, random access memory; CFO, carrier frequency offset.

effort level set to standard. All I/O pins are clocked by an FF, which is packed inside every IOB (input/output block) primative. The timing results are obtained under pessimistic operating conditions with a temperature of 85°C and voltage of 0.95 V.

The area and static timing analysis (STA) FPGA implementation results for the individual blocks and entire synchronizer are reported in Table I. The entire synchronizer achieves a timing of 64.26 MHz; consequently, it can be safely operated at 20 MHz without requiring any pipelining nor retiming techniques. We note that the critical path identified by STA consists of the $\max(|R[n]|^2)$ computation, which is implemented by logic in the packet detector and coarse time synchronizer. The largest block is the fine time synchronizer, which accounts for approximately 58.70% of the slice LUTs occupied by the synchronizer. The trade-off to use quantized complex multipliers instead of DSP48E blocks in the fine time synchronizer is to not limit the available built-in ASIC resources for the other baseband processing operations, such as the inverse and forward FFT computations.

5. SUMMARY AND FUTURE WORK

In this paper, we described our research and development process for the implementation of an OFDM synchronizer on a Xilinx Virtex-5 xc5vlx110t-1ff1136 FPGA [11]. Several improvements/extensions to the existing approaches were proposed. The important contributions being the following:

- the area complexity reduction of the accumulators by using the clipping technique,
- the reduction of the CFO translation CORDIC precision without performance penalty,
- the addition of an averaging circuit to stabilize the output from CFO translation CORDIC,
- the modification of the quantization approach in [7] to ensure that it is monotonic for the MAC operations in fine time synchronization, and
- a new cross-correlation architecture with lower latency and equal hardware complexity compared with the existing parallel architecture in [5] for fine time synchronization.

We evaluated the presented architecture under channel conditions that model an indoor office environment, assuming that AGC had been successfully performed on the incoming samples. FPGA implementation results verify that the presented architecture closes timing and requires approximately 7.92% and 20.55% of the available slice FF and LUT resources, respectively.

The proposed synchronizer architecture will be deployed as a component in our FPGA implementation of the IEEE

802.11a PHY, for use in part of a larger wireless communication platform. This work includes the development and integration of an AGC circuit.

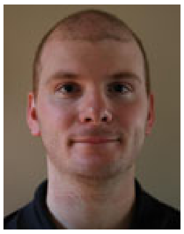
ACKNOWLEDGEMENTS

This work was supported in part by NSERC Discovery grant 262375-07 and Ontario Centres of Excellence grant WE-CR-10099-08.

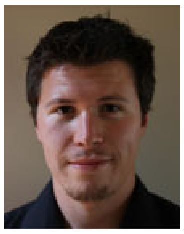
REFERENCES

1. Chang RW. Synthesis of band-limited orthogonal signals for multichannel data transmission. *The Bell System Technical Journal* 1966; **45**(10): 1775–1796.
2. IEEE Computer Society. *Wireless LAN MAC and PHY Specifications*, New York, NY, June 2007. IEEE Std 802.11-2007.
3. Moose PH. A technique for orthogonal frequency division multiplexing frequency offset calculation. *IEEE Transactions on Communications* 1994; **42**(10): 2908–2914.
4. Schmidl TM, Cox DC. Robust frequency and timing synchronization for OFDM. *IEEE Transactions on Communications* 1997; **45**(12): 1613–1621.
5. Fort A, Weijers JW, Derudder V, Eberle W, Bourdoux A. A performance and complexity comparison of auto-correlation and cross-correlation for OFDM burst synchronization. In *Proceedings of 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, 2003; II.341–II.344.
6. Yip K-W, Wu Y-C, Ng TS. Design of multiplierless correlators for timing synchronization in IEEE 802.11a wireless LANs. *IEEE Transactions on Consumer Electronics* 2003; **49**(1): 107–114.
7. Ha T, Lee S, Kim J. Low-complexity correlation system for timing synchronization in IEEE 802.11a wireless LANs. In *Proceedings of Radio and Wireless Conference (RAWCON 2003)*, 2003; 51–54.
8. Liu J, Li J. Parameter estimation and error reduction for OFDM-based WLANs. *IEEE Transactions on Mobile Computing* 2004; **3**(2): 152–163.
9. Jimenez VPG, Garcia MJF-G, Serrano FJG, Armada AG. Design and implementation of synchronization and AGC for OFDM-based WLAN receivers. *IEEE Transactions on Consumer Electronics* 2004; **50**(4): 1016–1025.
10. Pierri J. Design and implementation of an OFDM WLAN synchronizer, *Master's Thesis*, University of Waterloo, Waterloo, ON, 2007.
11. Xilinx Inc. *Virtex-5 Family Overview*, San Jose, CA, February 2009. Xilinx DS100 (v5.0).
12. IEEE Computer Society. *IEEE Standard for VHDL Register Transfer Level (RTL) Synthesis*, New York, NY, October 2004. IEEE Std 1076.6-2004.
13. IEEE Computer Society. *Air Interface for Broadband Wireless Access Systems*, New York, NY, May 2009. IEEE Std 802.16-2009.
14. Medbo J, Schramm P. *Channel models for HIPERLAN/2 in different indoor scenarios*, March 1998. ETSI/BRAN 3ERI085B.
15. Canet MJ, Vicedo F, Almenar V, Valls J, de Lima ER. A common FPGA based synchronizer architecture for Hiperlan/2 and IEEE 802.11a WLAN systems. In *Proceedings of the 15th IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC 2004)*, 2004; 531–535.
16. Liu L, Cheng T, Xiaoyu Q, Jiahui Q. Research on implementation of OFDM burst packet transmission on software radio platform of FPGA. In *Proceedings of the 11th International Conference on Advanced Communication Technology (ICACT 2009)*, 2009; 646–650.
17. Manavi F, Shayan YR. Implementation of OFDM modem for the physical layer of IEEE 802.11a standard based on Xilinx Virtex-II FPGA. In *Proceedings of IEEE Vehicular Technology Conference 2004-Spring (VTC 2004-Spring),* vol. 3, 2004; 1768–1772.
18. Volder J. The CORDIC trigonometic computing technique. *IRE Transactions on Electronic Computers* 1959; **EC-8**(3): 330–334.
19. Xilinx Inc. *LogiCORE IP CORDIC v4.0*, San Jose, CA April 2009. Xilinx DS249.
20. Canet MJ, Vicedo F, Almenar V, Valls J, de Lima ER. Hardware design of a FPGA-based synchronizer for Hiperlan/2. In *Proceedings of the 14th International Conference on Field Programmable Logic and Applications (FPL 2004),* vol. LNCS 3203, 2004; 494–504.
21. The MathWorks Inc. *MATLAB Fixed-Point Toolbox 3.0*, 2009.

AUTHORS' BIOGRAPHIES



Christopher E. Kennedy received the B.A.Sc. Degree in Computer Engineering from the University of Ottawa, Ottawa, ON, Canada, in 2006, and the M.E.Sc. Degree in Electrical and Computer Engineering from The University of Western Ontario, London, ON, Canada, in 2009. Since January 2009, he has been employed as a digital hardware designer with Broadband Wizard Incorporated in London, ON, Canada, working on the research and development of next-generation wireless communication products. His research interests include computer arithmetic, error control coding, and networking.



Dan J. Dechene received the B.Eng. Degree in Electrical Engineering from Lakehead University, Thunder Bay, ON, Canada, in 2004, and his M.E.Sc. Degree in Electrical and Computer Engineering from The University of Western Ontario, London, ON, Canada, in 2008. Currently, he is pursuing his Ph.D. Degree in Communication Systems Engineering at The University of Western Ontario. Dan's research interests include energy efficient resource allocation schemes, heterogeneous quality-of-service guarantees, and multiple antenna wireless systems.

Abdallah Shami received the B.E. Degree in Electrical and Computer Engineering from the Lebanese University, Beirut, Lebanon, in 1997, and the Ph.D. Degree in Electrical Engineering from the Graduate School and University Center, City University of New York, New York, NY, USA, in September 2002. In September 2002, he joined the Department of Electrical Engineering at Lakehead University, Thunder Bay, ON, Canada, as an assistant professor. Since July 2004, he has been with The University of Western Ontario, London, ON, Canada, where he is currently an associate professor in the Department of Electrical and Computer Engineering. His current research interests are in the area of wireless/optical networking.