ECE4305: Software-Defined Radio Systems and Analysis

Laboratory 3: Frame Synchronization

Objective

This laboratory will introduce the concept of frame synchronization between transmitting and receiving nodes. Specifically, a simplified error model will be discussed along with an efficient method with various implementations. Moreover, MATLAB® will also be introduced as development tools for digital communication systems, especially the construction of a prototype software-defined radio (SDR) based simulation.

Contents

1 Theoretical Preparation

The fundamental concepts of digital communication systems and related theoretical background material covered in this section will serve as a basis for the implementation and design of prototype systems throughout the rest of this course.

1.1 O Frame, Where Art Thou?

In previous labs we have discussed frequency correction, timing compensation, and matched filtering. The final aspect of synchronization is frame synchronization. At this point it is assumed that the available samples represent single symbols and are corrected for timing, frequency, and phase offsets. However, since in a realistic system the start of a frame will still be unknown, we need to perform an additional correction. We demonstrate this issue visually in Figure 1.1, which contains a sample synchronized frame with an unknown offset of p samples. Once we have an estimate $\hat{p}$ we can extract

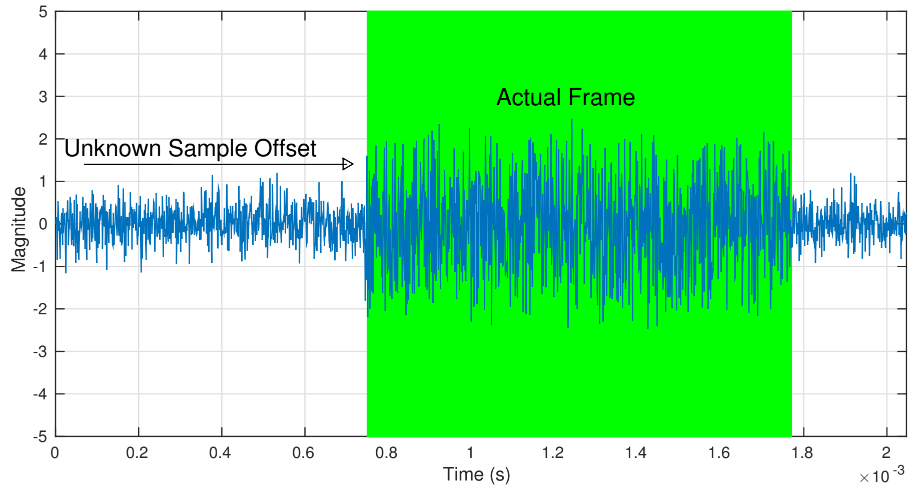

Figure 1: Example received frame in AWGN with an unknown sample offset.

data from the desired frame, demodulated to bits, and perform any additional channel decoding or source decode originally applied to the signal. There are various way to accomplish this estimation but the implemented outlined in this lab is based around cross-correlation.

2 Frame Synchronization

The common method of determining the start of a given frame is with the use of markers, even in wired networking. However, in the case of wireless signals, this problems becomes more difficult as made visible in Figure 1.1 which actually uses a marker. Due to the high degree of noise content in the signal, specifically designed preamble sequences are appended to frames before modulation. Such sequences are typically known exactly at the receiver and have certain qualities that make frame estimation accurate. In Figure 2 we outline a typical frame ordering containing: preamble, header, and payload data. Header and payloads are unknown are the receiver, but will maintain some structure so they can be decoded correctly.

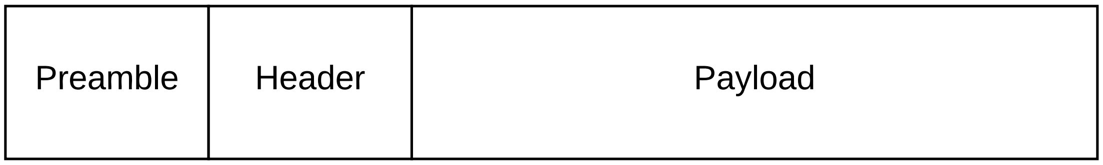

Figure 2: Common frame structure of wireless packet with preamble followed by header and payload data.

Before we discuss the typical sequences utilized we will introduce a technique for estimation of the start of a known sequence starting at an unknown sample in time. Let us consider a set of N different binary sequences y_n, where $n \in \left[1, ..., N\right]$, each of length L. Given an additional binary sequence x, we want to determine how similar x is to the existing N sequences. The use of a cross-correlation would provide us the appropriate estimate, which we perform as:

$$C_{xy}(k) = \sum_m x^*(m) y_n(m+k), \tag{1}$$

which is identical to a convolution without a time reversal on the second term. When $x = y_n$ for a given n, C_{xy} will be maximized compared with the other $n-1$ sequences, and produce a peak at L^{th} index at least. We can use this concept to help build our frame start estimator, which we have discussed will contain a known sequence called the preamble.

Common sequences utilized in preambles for narrowband communications are Barker Codes [2]. Barker Codes are utilized since they have unique auto-correlation properties that have minimal or ideal off-peak correlation. Specifically, such codes or sequences $a(i)$ have autocorrelation functions defined as:

$$c(k) = \sum_{i=1}^{N-k} a(i) a(i+k), \tag{2}$$

such that:

$$|c(v)| \leq 1, \quad 1 \leq v < N. \tag{3}$$

However, only nine sequences are known $N \in \left[1, 2, 3, 4, 5, 7, 11, 13\right]$, provided in Table 1. We provide a visualization of these auto-correlations in Figure 2, for a select set of lengths. As the sequence becomes longer the central peak becomes more pronounced. For communication systems we typically append multiple such codes together to produce longer sequences for better performance, as well as for other identifications.

Using these codes we have implemented a small example to show how a barker sequence $a(k)$ can be used to locate sequences in a larger set of data $r(k)$, which we have provided in `lab3part1.m`. In this example we insert a barker code within a larger random sequence at an unknown position p, similar to our original error model in Section 1.1, show as the top plot in Figure 2. A cross-correlation is performed using MATLAB®'s `xcorr` function, providing the lower plot in Figure 2. The cross-correlation will be of length $2L_r - 1$, where L_r is the length of r. Since `xcorr` will pad zeros to a so its length is equal to L_r [3], this will result in at least $L_r - L_a$ zeros to appear in the correlation where L_a is the original length of a. From Figure 2, we know that the peak will appear at L_a samples from the start of the sequence. Taking this into account we can directly determine at what offset position of our desired sequence:

$$\hat{p} = \underset{k}{\operatorname{argmax}}\ C_{ra}(k) - L_r, \tag{4}$$

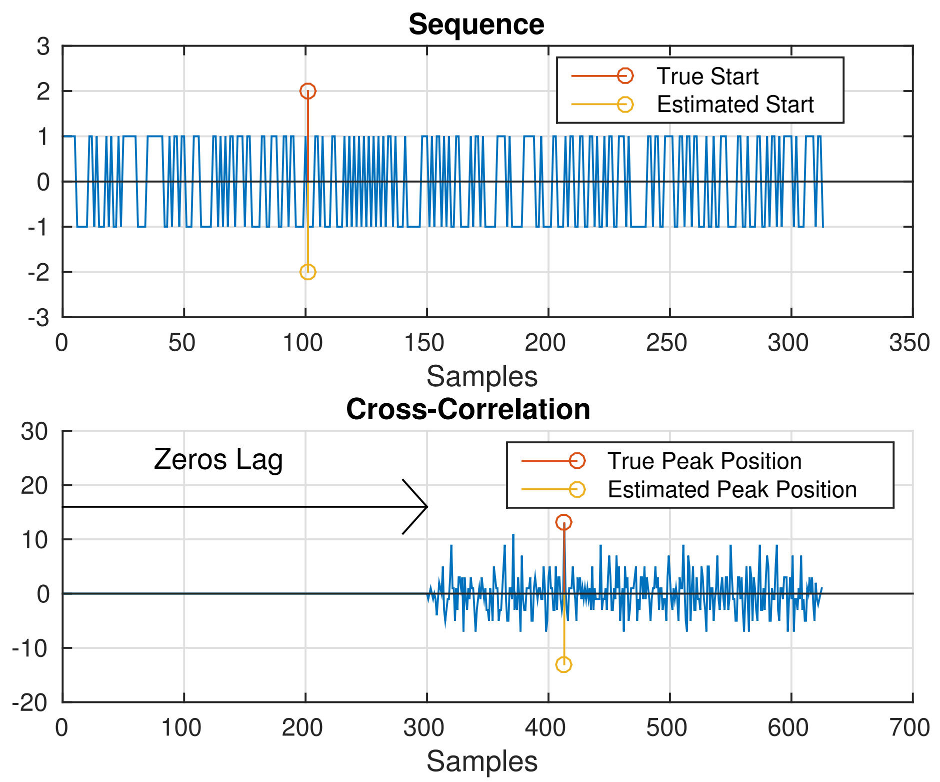

Figure 3: Sample Barker Code implementation for sequence offset determination.

which is what we observe from our estimates in the top plot of Figure 2.

Now that we have a method for estimating the start of a frame, let us consider a slightly simplier problem. Can we determine that a frame exist in the correlation? This can be useful if wish to handle data in smaller pieces rather than working with complete frames, or possibly a mechanism for determining if a channel is occupied. When we consider signal detection, we typically define this feature as a power sensitivity, or the minimum received power at the receiver to be detected. However, this sensitivity will be based on some source waveform and cannot be generalized. Therefore, such a value will should never be given on its own, unless given with respect to some standard transmission. Even when considering formal methods of detection theory, such as Neyman-Pearson or even Baysian, you must have some knowledge or reference to the source signal [4]. The receiver sensitivity requirement for 802.11ac specifically is defined as the minimum received signal power to maintain a packet error rate of 10%, for a give modulation and coding scheme [1].

When we consider the implementation consequences of detecting a signal, our design become more complicated than for example `lab3part1.m`. In the most basic sense detection becomes a thresholding problem for our correlator. Therefore, the objective becomes in determining a reference or criteria for validating a peak, which can be radically different over time depending on channel noise and the automatic gain control of the Pluto. However, even in simulations appropriate thresholding become non-trivial, which we can demonstrate with Figures 4 and 5. In these Figures the peak appears larger relative to the rest of the correlation in the case where no frame exists in the receive signal, compared to the condition when a frame exists. Therefore, for an implementation that performs well it should handle such conditions, and operate no matter the input scaling.

In `lab3part2.m` we have enhanced the baseline bit generation to include support for ASCII char-

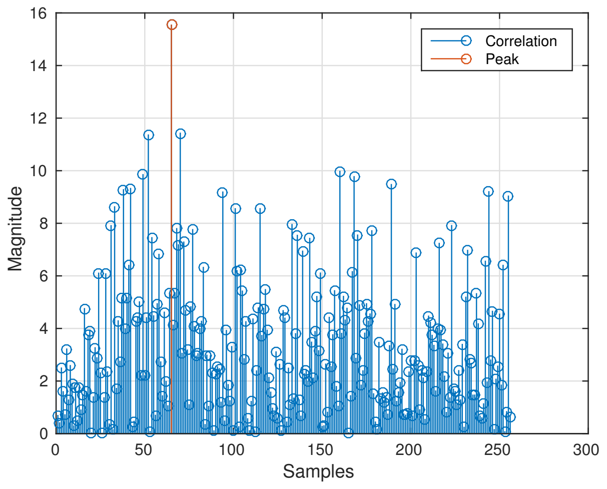

Figure 4: Correlation without signal present.

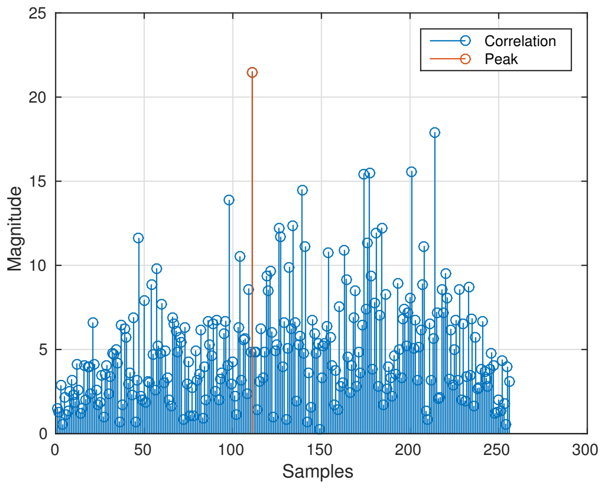

Figure 5: Correlation with signal present.

acters to bits and back to ASCII. Included as well is an example demodulation and packet error rate measurement. These will be used to determine overall performance of your receiver design, and frame synchronizer.

- Question 0: Based on `lab3part1.m` implement an alternative scheme that utilized the `filter` function for estimation rather than `xcorr`. (`filter` is generally much faster than `xcorr` in interpreted MATLAB.) Provide a benchmark using `tic` and `toc` for their relative performance for different sequence lengths.
- Question 1: Based on template provided in `lab3part2.m` and the example in `lab3part1.m` implement a frame synchronization scheme.
- Question 2: Test your frame synchronization implementation over the SNR range $[0, 10]$ dB and provide the resulting plot for detection probability and packet error rate.
- Question 3: For a coherent modulation scheme (such as QPSK) what modification would be require to correctly decode or demodulate the received symbols after the frame has been recovered? Describe provide two techniques for solving this problem.

?

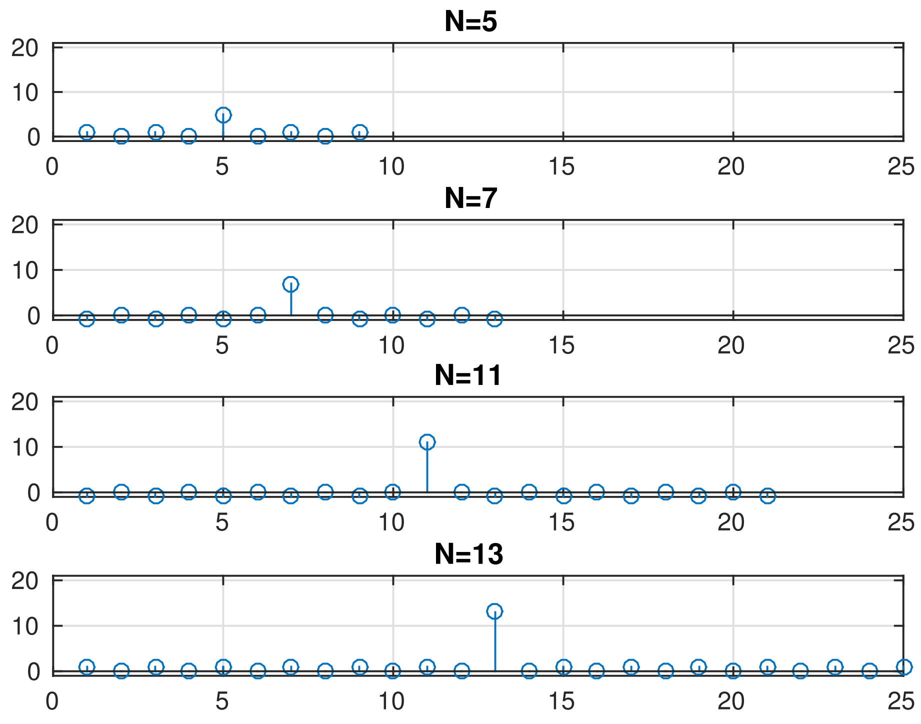

Figure 6: Barker code autocorrelations for different lengths.

Table 1: Barker Codes From `comm.BarkerCode`

N	Code
2	$-1, +1$
3	$-1, -1, +1$
4	$-1, -1, +1, -1$
5	$-1, -1, -1, +1, -1$
7	$-1, -1, -1, +1, +1, -1, +1$
11	$-1, -1, -1, +1, +1, +1, -1, +1, +1, -1, +1$
13	$-1, -1, -1, -1, -1, +1, +1, -1, -1, +1, -1, +1, -1$

3 Open-ended Design Problem: Full Frame Recovery

3.1 Objective

The objective of this problem is to design and implement a software-defined radio (SDR) communication system capable of automatically transmitting and recovering full frames between two Pluto SDRs.

In Sections 2 you have implemented and simulated frame estimation. Now we will introduce the Pluto to deal with realistic sample offset behavior. To simplify this process a set of required tasks are staged to gradually increase the difficulty of the overall implementation. We have presented a method for sample offset estimation, but you are free to use these in any arrangement you want, or use your own algorithms. However, you must provide an evaluation of the overall system performance. Your final implementation should be tuned to handle any set of Pluto radios, not just the ones used by your team.

With your implementation perform the following tasks:

1. First, using a single Pluto transmit your DBPSK or QPSK reference signal across to the same radio. Ignore timing correction and frequency correction for now, since they should be minimal. Provide packet error rate results for this implementation.

2. Repeat this experiment but include your frequency correction implementation and manually introduce an offset at the transmitter. Provide packet error rate results. You should observe initial errors during the lock period of the PLL.

3. Next, reset your transmitter and receivers to have identical frequencies again. Now insert your timing estimation design from lab 2, and provide packet error rate results here.

4. Finally, if you have had reasonable packet error rate result so far using a pair of Plutos repeat this estimation again as previously performed with a single Pluto. Provide packet error rates again.

4 Lab Report Preparation & Submission Instructions

Include all your answers, results, and source code in a laboratory report formatted as follows:

- Cover page: includes course number, laboratory title, names and student numbers of team, submission date.
- Table of contents, list of tables, list of figures.
- Commentary on designed implementations, responses to laboratory questions, and explanation of observations.
- Responses to open-ended design problem.
- Source code (as an appendix).

Remember to write your laboratory report in a narrative approach, explaining your experience and observations in such a way that it provides the reader with some insight as to what you have accomplished. Furthermore, please include images and outputs wherever possible in your laboratory report document.

Each group is required to submit a single report electronically (in PDF format) to `alexw@ece.wpi.edu` by the scheduled due date and time. Reports that do not meet these specifications will be returned without evaluation and will receive a grade of "0" for the report segment of the laboratory experiment.

References

[1] IEEE Standard for Information technology– Telecommunications and information exchange between systemsLocal and metropolitan area networks– Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz. *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)*, pages 1–425, Dec 2013.

[2] RH Barke. Group synchronizing of binary digital sequences. *Communication theory. Butterworth, London*, pages 273–287, 1953.

[3] The MathWorks Inc. xcorr. [Online]: https://www.mathworks.com/help/signal/ref/xcorr.html, 2017.

[4] Steven M Kay. Fundamentals of statistical signal processing, vol. ii: Detection theory. *Signal Processing. Upper Saddle River, NJ: Prentice Hall*, 1998.