Technical University of Cluj–Napoca

D i g i l e n t C o n t e s t • FALL 2 0 19

PREAMBLE SYNCHRONIZATION

D i g i ta l D e s i g n R e p ort

Authors: Nazlıcan GÜNEŞ & Yıldız BİLGİN

Çankaya UNIVERSITY–Electronics and Communication Department

Advisor: Dr. Orhan GAZİ

Çankaya UNIVERSITY–Electronics and Communication Department

Contents

1. Introduction

1.1 Overview

The objective of this project is to investigate signal detection and frame synchronization for wireless network signals. Specifically, a receiver capable of differentiating between and synchronizing to three commercial wireless networking standards - IEEE 802.11a, IEEE 802.11b, and IEEE 802.16 - will be developed. These three waveforms were chosen because of their dominance in the commercial wireless networking market and because they offer an excellent example for military wireless networking applications. Since the receiver does not know when to expect the next frame, each frame must carry extra information to allow for detection and frame synchronization. In the IEEE 802.11a, IEEE 802.11b, and IEEE 802.16 standards, a header section is added at the front of each frame to accomplish this task. The header includes a signal, called the preamble, which is the same for each frame. The preamble is intended to identify the frame as belonging to a certain standard. The receiver, then, must have a way of recognizing the preamble. This project describes the properties of each of the preambles of the IEEE 802.11a, IEEE 802.11b, and IEEE 802.16 standards. Following this, an effective preamble recognition technique in which the receiver maintains a copy of each preamble of interest is presented. The receiver compares every received signal with the stored preambles to see if there is a match. This method of comparison is accomplished through the cross-correlation of the received signal with the stored preambles. The preambles were designed so that when two versions of the same preamble are aligned perfectly, the result is approximately an order of magnitude larger than when they are not aligned perfectly or when the preamble is not present. Signal detection occurs when the cross-correlation exceeds a predetermined threshold.

For a digital receiver to detect each of these signals without misidentifying any of them, each of the preambles must have low values when cross-correlated with the others. This thesis investigates the cross-correlation properties of the preambles. Following this, the project defines a decision threshold appropriate for joint detection of all three of the above-mentioned wireless networking signals.

The thesis presents a frame synchronization technique that uses the crosscorrelation result to find the first data symbol in the frame. The technique takes advantage of the fact that the place of the preamble within the signal header is known a priori. The technique also takes advantage of the fact that when the output of the crosscorrelation rises above the threshold, the last received sample in the filter should be the last sample of the received preamble. In other words, the cross-correlation peak can be used to find which sample corresponds to the end of the preamble. Once this sample is located, the first data sample is always known to be a fixed number of samples away.

Models were created in MATLAB and VHDL for the generation of IEEE 802.11a, IEEE 802.11b, and IEEE 802.16 frame headers. The components of the receiver for detection and synchronization were also developed in MATLAB and VHDL.

1.2 Summary of the design project

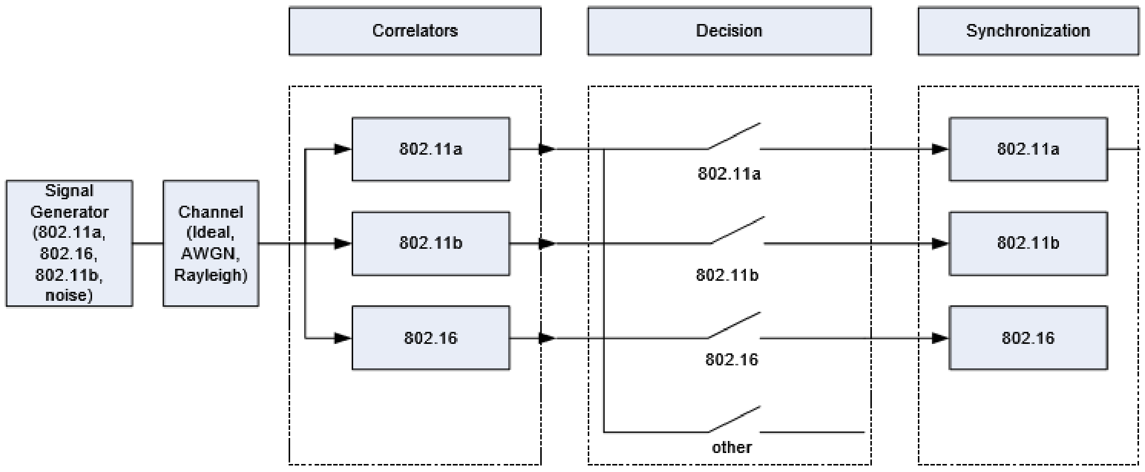


Figure 1. Single Receiver Block Diagram for Multiple Wireless Networking Waveform Detection and Synchronization.

The results of the simulations showed that, over a wide range of signal to noise ratios, the IEEE 802.11a, IEEE 802.11b, and IEEE 802.16 signals could be distinguished from each other. The implication of this is that a single digital receiver with a wideband air interface capable of sampling each signal can be used to receive multiple wireless networking signals. As more waveforms are to be added to the receiver's capabilities, the first consideration has to be the cross-correlation prosperities of the new signal's preamble with the preambles of the other signals. The only addition to the receiver is another software module, assuming that the new signal's frequency band is within the current operating range of the air interface. The results also showed that the cross-correlation result can be used very effectively to achieve signal detection for each of the signals tested.

1.3 Hardware requirements

- Digilent Zybo Z7 FPGA Board

-Digilent Analog Discovery 2

-Bread Board

-USB Cablo X2

-10k Resistor X9

-20k Resistor X7

-Vivado 2018.3

-Xilix ISE Design Suite 14.2

1.4 Software requirements

-MatLab 2016.b

-Microsoft Windows 10

-Digilent Adept ExPort

-WaveForms

2. Background

With the popularity of IEEE 802.11a/b/g wireless networks, the prevalence of cellular networks, and the introduction of the IEEE 802.16 standards, a constantly changing selection of wireless networking choices will be available to the mobile user. As the number of wireless networks increases, a radio device capable of connecting to multiple wireless networks will be necessary. The device will need to be able to detect and classify different wireless networking signals and to distinguish the signals from each other. The device must also decide which network to connect to depending upon select criteria, such as data rate or cost. One implementation towards this approach is to include a separate receiver for each communication standard desired. This requires an ever growing footprint as the number of communication options increases. Another solution is a receiver with a radio frequency (RF) front end that operates in the frequency ranges of the desired signals. This wideband air interface would sample the signals for digital processing. Once the RF signal had been sampled, the samples would be digitally filtered to determine if any of the signals of interest are present. The receiver would then decide which network connection is most suitable. For this solution to be practicable, the digital receiver must be able to detect and classify each of the signals of interest with a high probability of detection, while at the same time not falsely classifying one signal as another. The research interest in this project is to investigate the detection of IEEE 802.11a, IEEE 802.16, and IEEE 802.11b wireless networking signals by a single digital receiver. This project assumes that the receiver has a wideband air interface capable of receiving each of these signals and sampling them at baseband. The project then explores the ability of signal processing techniques to detect and classify the signals.

The Department of Defense (DoD) interest in these technological developments is two fold. First, as the DoD seeks high data-rate, battlefield communications, the underlying physical layer and medium access (MAC) layer technologies implemented by these standards offer significant opportunity for leveraging commercial technology into military communication systems. Second, as these signals become ever more commonplace, the interest will arise to be able to detect, classify and demodulate these signals for signal intelligence purposes.

3. Signal detection, Classification & Decision Statistics

3.1 Signal detection

IEEE 802.11a and IEEE 802.16 Preambles

High data rate, packet-based wireless local area networks (WLANs) require fast detection and classification times because of the bursty nature of packet-based communications. Without a continuous signal to maintain synchronization, WLANs must synchronize quickly to avoid suffering a lower data rate due to the overhead needed for synchronization. Detection of IEEE 802.11a and IEEE 802.16 signals is performed by searching for the preamble in a received signal by correlating the received signal against one symbol of the short or first preamble, respectively.
In discrete time, the cross-correlation between two signals, [ ] and [ ] x n y n , is defined as

$$C[k]=\sum_{n=0}^{L-1}x^{*}[n+k]y[n+k-L+1].$$

where L is the size of the sliding window of the correlator and * denotes complex conjugation. The power in one of the signals .

$$P[k]=\sum_{n=o}^{L-1}\left|x[n]\right|^{2}$$

can be used to normalize the cross-correlation such that

$$D[k]=\frac{C[k]}{P[k]}.$$

The cross-correlation result is normalized so that a single decision threshold can be used regardless of the received signal power. For the preambles to be useful in achieving signal detection in a joint detection receiver, they must have low normalized cross-correlation values. The following sections will detail the structure of the IEEE 802.11a and IEEE 802.16 preambles and their correlation properties.

a. IEEE 802.11a Preambles

Both the IEEE 802.11a and IEEE 802.16 standards begin a frame transmission with a preamble. The preamble is intended to be used for signal detection, classification, and time and frequency synchronization. In both standards, the preamble, or training sequence as it is also called, is broken into two parts. In IEEE 802.11a, the two parts are called the short preamble and the long preamble and are depicted in Figure 1. The short preamble consists of 10 short symbols and the long preamble consists of two long symbols. The 10 short symbols of the short preamble are comprised of 12 subcarriers of an OFDM symbol [1]. The transmission time of each of the short symbols is 0.8 µs.

After sampling at 20 MHz, there are 16 time domain samples per short symbol. The short preamble then consists of a total of 160 time domain samples lasting 8.0 µs.

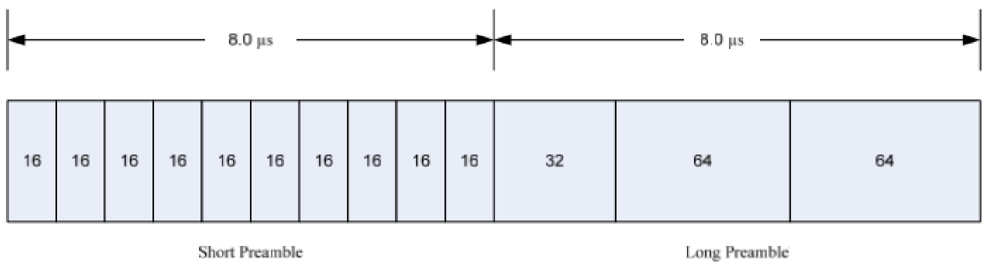


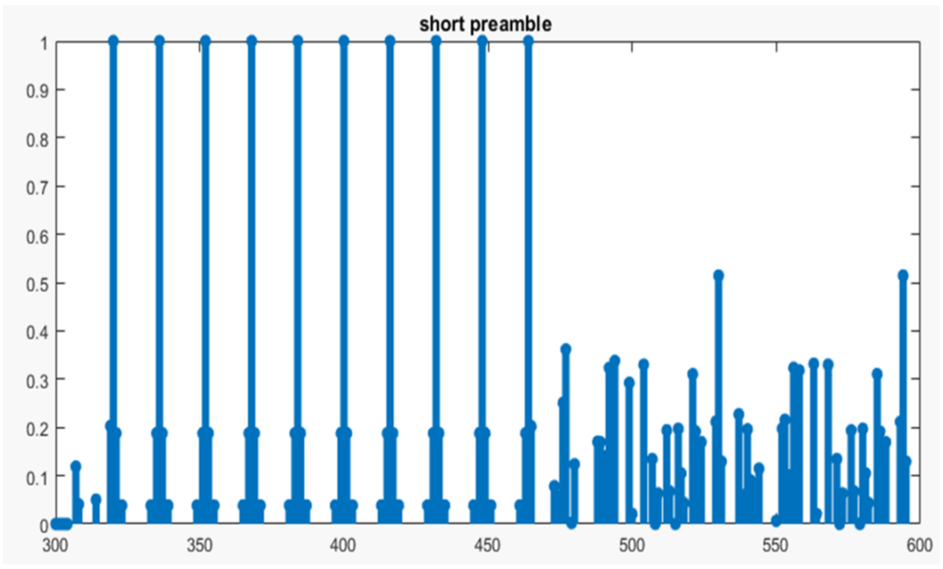
Figure 2. IEEE 802.11a Short and Long Preambles

Table 1 gives the standard time domain sample values of one of these short symbols. Both the real and the imaginary samples sum to zero and each has seven positive samples, eight negative samples and one zero term. Thus, when these samples are multiplied by a random input, the resulting magnitude will be at least an order of magnitude smaller than the original input. Further, since half the terms are positive and the other half negative, when the results of the multiplications are added together, the sum will again be smaller. Thus, when the input samples directly match these values, the magnitude of the correlated output is expected to be significantly higher than otherwise.

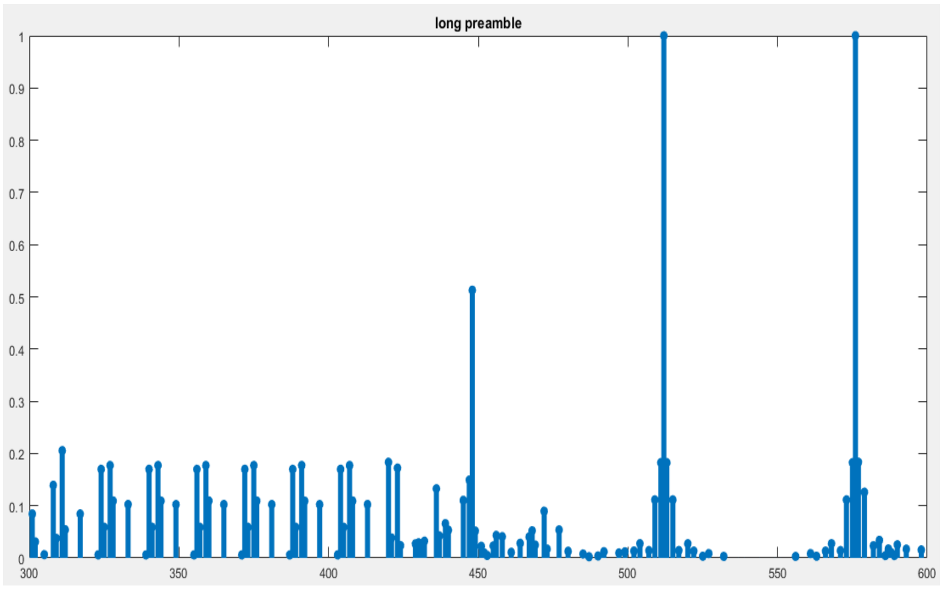
IEEE 802.11a Short Preamble Time Domain Samples	
Real Time Domain Samples	Imaginary Time Domain Samples
0.046	0.046
-0.1324	0.0023
-0.0135	-0.0785
0.1428	-0.0127
0.092	0
0.1428	-0.0127
-0.0135	-0.0785
-0.1324	0.0023
0.046	0.046
0.0023	-0.1324
-0.0785	-0.0135
-0.0127	0.1428
0	0.092
-0.0127	0.1428
-0.0785	-0.0135
0.0023	-0.1324

Table 1. Short Preamble Time Domain Sequence Samples.

The first six symbols of the short preamble are intended to be used for signal detection, automatic gain control, and diversity selection. The last four short symbols are intended for coarse frequency offset estimation and timing synchronization [1]. The long preamble consists of a cyclic prefix and two long symbols. In the time domain, each long symbol consists of 64 complex time domain samples when sampled at 20 MHz. The cyclic prefix is the last 32 samples of the symbol prepended to the first long symbol. The cyclic prefix is transmitted for 1.6 µsand each long symbol is transmitted for 3.2 µs. The total length of the long preamble is then 160 samples with a transmission time of 8.0 µs. The long preamble is intended for channel estimation and fine frequency offset estimation. Figure 2a shows one symbol of the short preamble correlated with the full IEEE 802.11a preamble in an ideal channel. After an initial sequence of twenty zeros, the ten symbols of the short preamble each create peaks that are 80% above the cross-correlation between short and long preamble. In Figure 2b, the cross-correlation of one symbol of the long preamble with the full IEEE 802.11a preamble results in one smaller peak that marks the extended cyclic prefix and then two taller peaks that are 3.8 times greater than the cross-correlation between the long and short preamble symbols.



(a)



(b)

Figure 3. Cross-correlation of the IEEE 802.11a Preamble with (a) Short Preamble Correlator and (b) Long Preamble Correlator.

b. IEEE 802.16 Preambles

In the IEEE 802.16 standard, the preambles are different on the downlink and the uplink. The uplink preamble is a shortened version of the downlink preamble. Here, the downlink preamble as shown in Figure 9 will be described. The first symbol consists of a cyclic prefix followed by four repetitions of a 64 sample sequence. The second symbol is comprised of a cyclic prefix followed by two repetitions of a 128 sample sequence. Both cyclic prefixes are the same length and each is also the same length as the cyclic prefixes within the OFDM data symbols. In IEEE 802.16, the cyclic prefix size is variable, depending upon the channel rms delay spread.

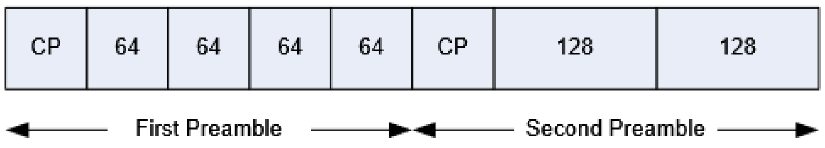
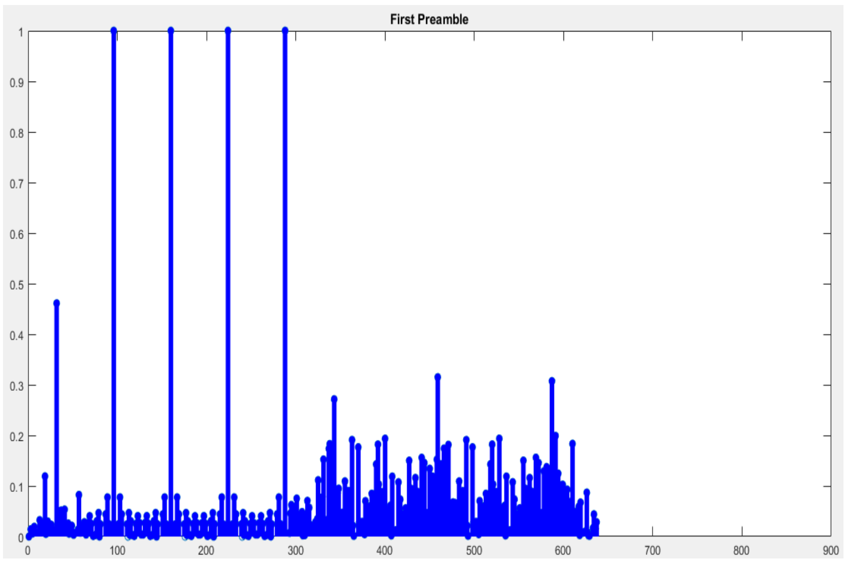
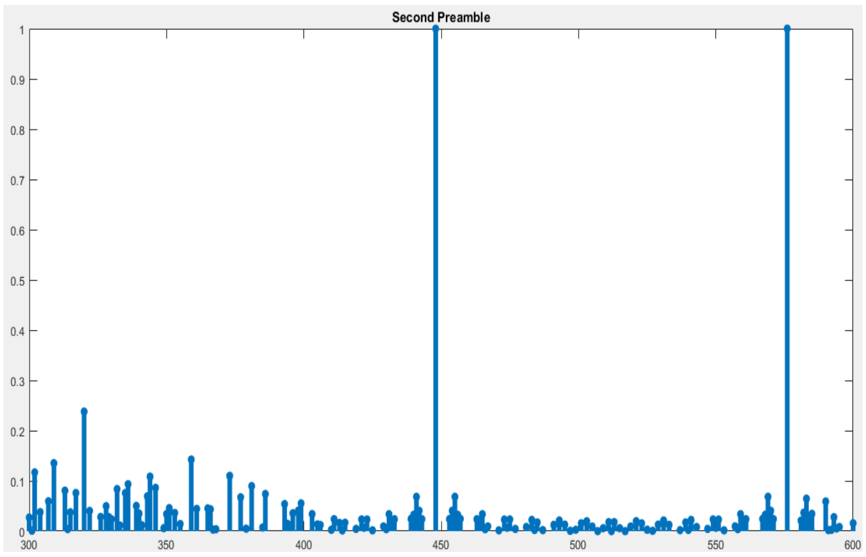


Figure 4. IEEE 802.16 Downlink Preamble

Using a cyclic prefix of 64 time domain samples, Figure 5(a) shows the crosscorrelation of the complete downlink preamble with one symbol of the first preamble. There are five peaks because the cyclic prefix matches the length of a symbol of the first preamble. The cross-correlation between one symbol of the first preamble and one symbol of the second preamble is 8.8 times less than the maximum auto-correlation of one symbol of the first preamble. Figure 5(b) shows the cross-correlation of one symbol of the second preamble with the complete downlink preamble. The smaller correlation peak results from the cyclic prefix. The maximum cross-correlation value between one symbol of second preamble and one symbol of the first preamble is more than seventeen times less than the max autocorrelation value of one symbol of the second preamble.



(a)



(b)

Figure 5. Cross-correlation of the IEEE 802.16 Preamble with (a) First Preamble Correlator and (b) Second Preamble Correlator.

Cross-correlation of IEEE 802.11a and IEEE 802.16 Preambles

Figure 6 shows the results of running the correlation between IEEE 802.11a and IEEE 802.16 Preambles

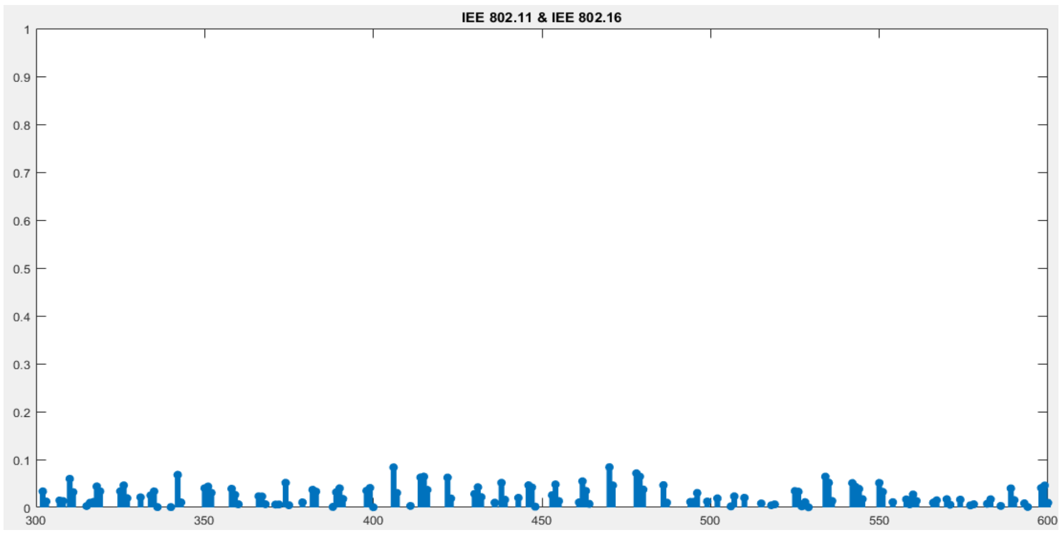


Figure 6. Correlation between IEEE 802.11a and IEEE 802.16 Preambles

3.2 Decision Statistics

This section will compare the decision statistics from two correlation methods used for signal detection. The first method is the autocorrelation method and is detailed. The second method is the cross-correlation method and will be explained below. For each method, the mean and the variance of the decision statistic will be determined when the signal is present and when only noise is present. Following this, the statistical distribution for the noise only case will be discussed in order to apply the NeymanPearson test to determine detection thresholds.

A PDF for the noise-only hypothesis was derived for an OFDM signal using autocorrelation to detect a repeated sequence. This paper, written before the IEEE 802.11a standard was adopted, called for a two symbol training sequence. The first symbol was to contain two identical sequences preceded by a cyclic prefix and was to be used for frame synchronization. The second symbol was to contain two separate pseudonoise sequences on the even and odd frequencies to enable frequency offset calculations. The simulations used an OFDM signal with 1000 subcarriers oversampled at 1024 samples per symbol. The guard interval was equal to 10% of the data symbol. The normalized decision statistic was defined as:

$$D_A[k] = \frac{|R[k]|^2}{(P[k])^2}$$

where the autocorrelation is defined as

$$R[k] = \sum_{n=0}^{L-1} (x[n+k]^* x[n+k-L+1]) .$$

Since the real and imaginary parts of the noise are assumed to be Gaussian, the autocorrelation of the noise is a chi-square random variable defined as

$$|R[k]|^2 = 2L\sigma_n^4 \chi_2^2$$

where χ^2_2 is a chi-square random variable with a mean of two and a variance of four. The square of the received noise power is Gaussian with a mean of $4L^2\ \sigma_n^4$ and a variance of $16L^4\ \sigma^8_n$. If each term in is divided by the mean of the denominator, the distribution of $D_A[k]$ is

$$D_A[k] \sim \frac{\frac{1}{2L}\chi_2^2}{n\left(1, \frac{4}{L}\right)}$$

where ~ signifies “is distributed” and ( ) , nab signifies a Gaussian random variable with mean a and variance b. Since the mean of the denominator is one and the variance is smaller than the mean, the PDF of the decision statistic for the noise-only case can be approximated as

$$D_A[k] \sim \frac{1}{2L}\chi_2^{\,2}.$$

The mean and variance of the decision statistic are

$$E\left[D_A[k]\right] = E\left[\frac{1}{2L}\chi_2^{\,2}\right] = \frac{1}{2L}E\left[\chi_2^{\,2}\right] = \frac{1}{2L}(2) = \frac{1}{L}$$

$$\mathrm{var}[D_A[k]] = \frac{1}{4L^2}E\left[(\chi_2^{\,2})^2\right] - \frac{1}{L^2} = \frac{\mathrm{var}(\chi_2^{\,2}) + E\left[\chi_2^{\,2}\right]}{4L^2} - \frac{1}{L^2} = \frac{4+4}{4L^2} - \frac{1}{L^2} = \frac{1}{L^2}$$

where [ ] E is the expectation operator.

This approach can be applied in an IEEE 802.11a receiver to determine the mean and variance of the decision statistic when a signal is present and when only noise is present.

In Figure 7, the cross-correlation output is shown at a SNR of 0 dB. The ten spikes in the figure are the first indices of the ten symbols of the short preamble. Even at 0 dB, the correlation with the preceding 300 noise samples is on an order of magnitude lower than the correlation peaks. Notice that the points between the spikes also remain almost an order of magnitude lower than the correlation peaks. A detector can then count the peaks and the spacing between them to detect and classify the signal.

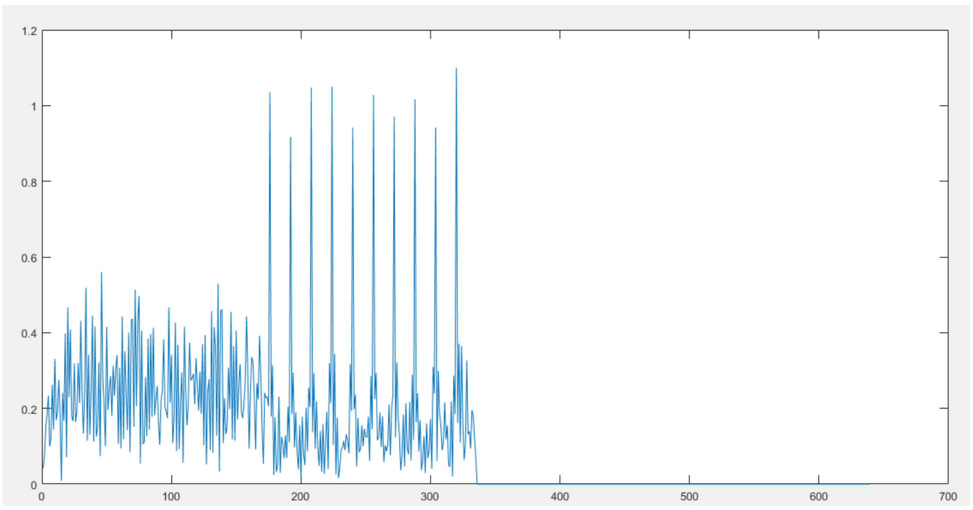


Figure 7. Cross-Correlation with IEEE 802.11a Short Preamble at 0 dB SNR.

4 Simulation Model and Results

4.1 Simulation model

We firstly tried to simulate 2 little signals correlation to see the result and correct if there is a mistake in our VHDL code. Here, we can see the correlation of two integer vector, where one of them is length of 4 and the other is 2.

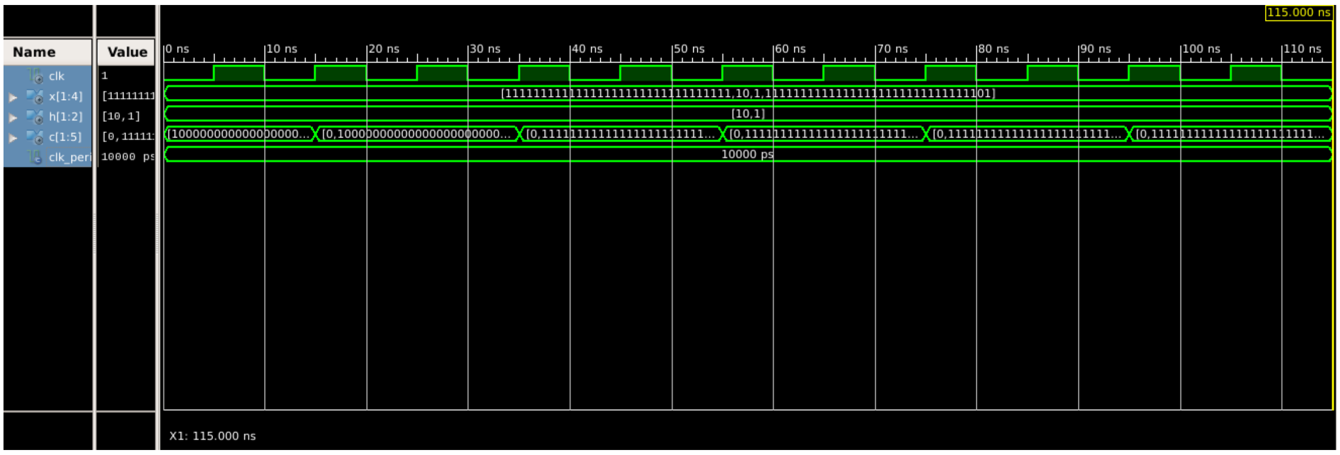


Figure 8. Correlation of two integer vectors

Where x is equal to (-1,2,1,-3) and h is equal to (2,1).

5 Conclusions

The objective of this project was to investigate if a single, digital, baseband receiver could detect, classify, and synchronize to IEEE 802.11a, IEEE 802.16, and IEEE 802.11b signals with MATLAB and Vivado with VHDL codes. The packet information available when an IEEE 802.11a signal was detected, and it was also investigated.

5.1 Summary of work done

First, a simulation model for the generation of IEEE 802.11a OFDM signals was implemented in MATLAB and Vivado with VHDL codes. Second, a partial mode l for the generation of IEEE 802.116 OFDM signals was developed in MATLAB and Vivado with VHDL codes. Third, a signal detection and synchronization model was developed to test the cross-correlation results of each signal by the other detector. Additionally, an attempt was made to define the probability density functions of the decision statistic at the receiver when a signal of interest was present and when only noise was present.

5.2 Significant Results

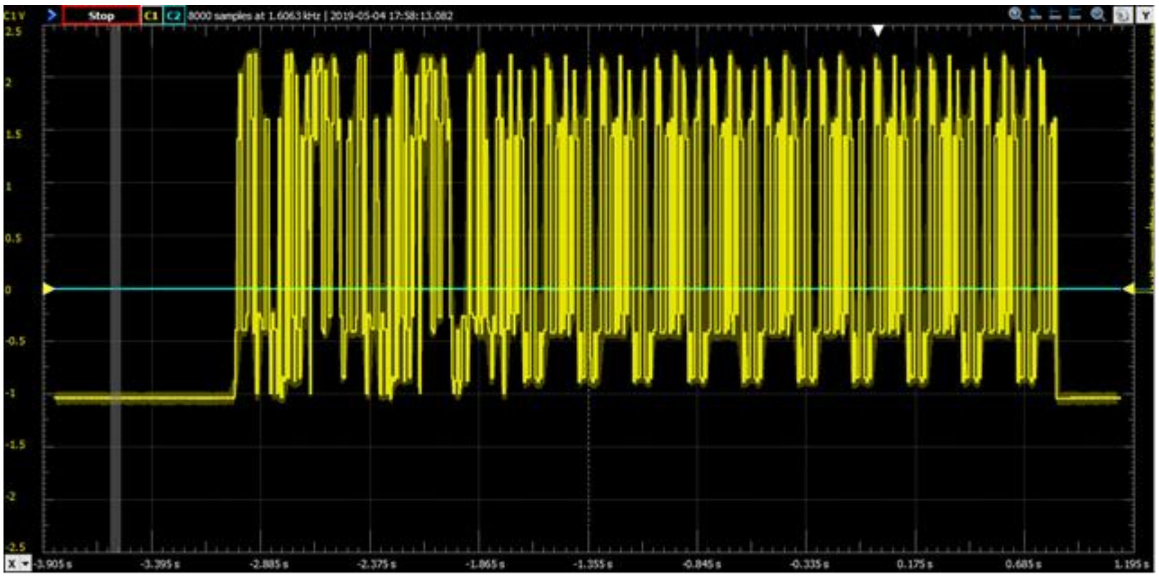
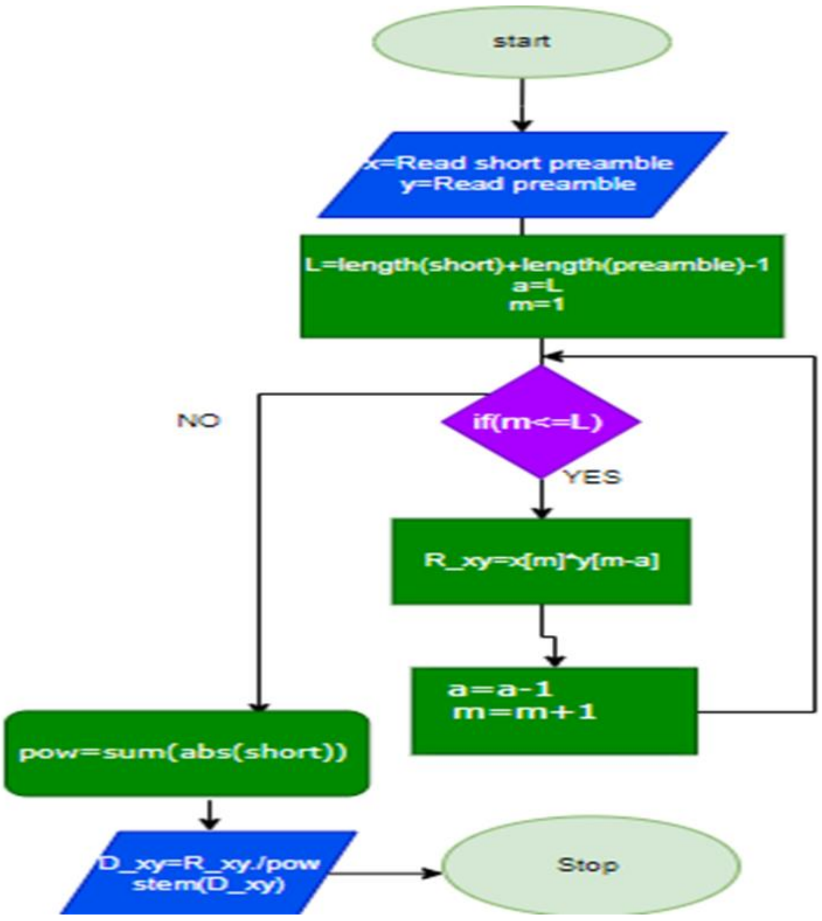


Figure 9. Cross-Correlation with IEEE 802.11a Short Preamble at 0 dB SNR

A single, digital, baseband receiver can separately detect and classify IEEE 802.11a, IEEE 802.11b, and IEEE 802.16 signals from each other. A bank of correlators matched to a symbol of each signal's preamble was shown to be a successful implementation for distinguishing IEEE 802.11a, IEEE 802.11b, and IEEE 802.16 signals from each other.

6. Appendix



MATLAB CODES

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear all;
clc;



onalti=[(0.023+0.023j)  (-0.132 +0.002j)  (-0.013-0.079j)  (0.143-
0.013j)...%short preamble imaginer and real values
 (0.092+0.000j)  (0.143-0.013j)  (-0.013-0.079j)      (-
0.132+0.002j)...
(0.046+0.046j) (0.002-0.132j)  (-0.079-0.013j)        (-
0.013+0.143j)...
 (0.000 +0.092j) (-0.013+0.143j)  (-0.079-0.013j)     (0.002 -
0.132j)];
otuziki=[(-0.078+0.000j)  (0.012-0.098j)  (0.092-0.106j)     (-
0.092-0.115j)... %cyclic prefix
 (-0.003-0.054j) (0.075+0.074j)  (-0.127+0.021j)  (-0.122+0.017j)...
 (-0.035+0.151j)  (-0.056+0.022j ) (-0.060-0.081j)  (0.070-
0.014j)...
```

```
 (0.082-0.092j)  (-0.131-0.065j)  (-0.057-0.039j)  (0.037-0.098j)...
 (0.062+0.062j)  (0.119+0.004j)  (-0.022-0.161j)  (0.059+0.015j)...
 (0.024+0.059)  (-0.137+0.047j)  (0.001+0.115j)  (0.053-0.004j)...
 (0.098+0.026j)  (-0.038+0.106j) (-0.115+0.055j)  (0.060+0.088j)...
 (0.021-0.028j)  (0.097-0.083j)  (0.040+0.111j)  (-0.005+0.120j) ];
altmisdort=[(0.156+0.000j) (-0.005+(-0.120j))  (0.040+(-0.111j))
(0.097+0.083j)...%long preamble imaginer and real values
  (0.021+0.028j)  (0.060+(-0.088j))  (-0.115+(-0.055j))  (-0.038+(-
0.106j))...
  (0.098+(-0.026j))  (0.053+0.004j) (0.001-0.115j) (-0.137-
0.047j)...
  (0.024+(-0.059j))  (0.059-0.015j)  (-0.022+0.161j)  (0.119-
0.004j)...
  (0.062+(-0.062j))  (0.037+0.098j)  (-0.057+0.039j)  (-
0.131+0.065j)...
  (0.082+0.092j)  (0.070+0.014j)  (-0.060+0.081j)  (-0.056-
0.022j)...
  (-0.035+(-0.151j))  (-0.122+(-0.017j))  (-0.127+(-0.021j))
(0.075+(-0.074j))...
  (-0.003+(0.054j))  (-0.092+0.115j)  (0.092+0.106j)
(0.012+0.098j)...
  (-0.156+0.000j)  (0.012+(-0.098j))  (0.092+(-0.106j))  (-0.092+(-
0.115j))...
  (-0.003+(-0.054j))  (0.075+0.074j)  (-0.127+0.021j)  (-
0.122+0.017j)...
  (-0.035+0.151j)  (-0.056+0.022j)  (-0.060-0.081j)  (0.070+(-
0.014j))...
  (0.082+(-0.092j))  (-0.131+(-0.065j))  (-0.057+(-0.039j))
(0.037+(-0.098j))...
  (0.062+0.062j)  (0.119+0.004j)  (-0.022+(-0.161j))
(0.059+0.015j)...
  (0.024+0.059j)  (-0.137+0.047j)  (0.001+0.115j)  (0.053+(-
0.004j))...
  (0.098+0.026j)  (-0.038+0.106j)  (-0.115+0.055j)
(0.060+0.088j)...
  (0.021+(-0.028j))  (0.097+(-0.083j))  (0.040+0.111j)  (-
0.005+0.120j)];
x=[onalti onalti onalti onalti onalti onalti onalti onalti onalti
onalti otuziki altmisdort altmisdort ];

y=[onalti zeros(1,304)];%we add zeros to equal our correlation terms



y1=[zeros(1,length(y)-1),y,zeros(1,length(x)-1)];

w=length(x)+length(y)-1;
a=w;
r_xy=0;
r1=0

rkatsayisi1=0
 for L=1:w%for loop calculates correlation coefficients
 for m=1:length(x)
    r_xy=conj(x(m)).*y1(m+a-1);
    r1=r_xy+r1;
 end
 a=a-1;
 rkatsayisi1(L)=r1;



 r1=0;
 end
figure(1);
 stem(rkatsayisi1./max(rkatsayisi1));...%We plot our corrrelation
between IEEE 802.11a preamble
                                          %and short preamble


 title('short preamble correlation ');
 axis([300 600 0 1])



 r2=0;

r_xy=0;

 yy=[altmisdort zeros(1,256)];
 y11=[zeros(1,length(yy)-1),yy,zeros(1,length(x)-1)];
ww=length(x)+length(yy)-1;
 a=ww;
 for L=1:ww%for loop calculates correlation coefficients
 for m=1:length(x)
    r_xy=conj(x(m)).*y11(m+a-1);
    r2=r_xy+r2;

 end
 a=a-1;
 rkatsayisi2(L)=r2;



 r2=0;
 end
figure(2);


stem(rkatsayisi2./max(rkatsayisi2));%We plot our corrrelation
between IEEE 802.11a preamble
                                          %and long preamble


 title ('long preamble correlation');
axis([300 600 0 1])



abs(real(b))
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Correlation Between noise with Preamble IEEE 802.11a & short
preamble
clear all;close all;clc;
onalti=[0.046+0.046i -0.1324+0.0023i -0.0135-0.0785i 0.1428-0.0127i
0.092 0.1428-0.0127i -0.0135-0.0785i -0.1324+0.0023i...
    0.046+0.046i 0.0023-0.1324i -0.0785-0.0135i -0.0127+0.1428i
0.092i -0.0127+0.1428i -0.0785-0.0135i 0.0023-0.1324i];%short
preamble IEEE 802.11a


otuziki=[(-0.078+0.000j)  (0.012-0.098j)  (0.092-0.106j)     (-
0.092-0.115j)...%cyclix prefix  IEEE 802.11a preamble
 (-0.003-0.054j) (0.075+0.074j)  (-0.127+0.021j)  (-0.122+0.017j)...
 (-0.035+0.151j)  (-0.056+0.022j ) (-0.060-0.081j)  (0.070-
0.014j)...
 (0.082-0.092j)  (-0.131-0.065j)  (-0.057-0.039j)  (0.037-0.098j)...
 (0.062+0.062j)  (0.119+0.004j)  (-0.022-0.161j)  (0.059+0.015j)...
 (0.024+0.059)  (-0.137+0.047j)  (0.001+0.115j)  (0.053-0.004j)...
 (0.098+0.026j)  (-0.038+0.106j) (-0.115+0.055j)  (0.060+0.088j)...
 (0.021-0.028j)  (0.097-0.083j)  (0.040+0.111j)  (-0.005+0.120j) ];
altmisdort=[(0.156+0.000j) (-0.005+(-0.120j))  (0.040+(-0.111j))
(0.097+0.083j)...%long preamble  IEEE 802.11a preamble
  (0.021+0.028j)  (0.060+(-0.088j))  (-0.115+(-0.055j))  (-0.038+(-
0.106j))...
  (0.098+(-0.026j))  (0.053+0.004j) (0.001-0.115j) (-0.137-
0.047j)...
  (0.024+(-0.059j))  (0.059-0.015j)  (-0.022+0.161j)  (0.119-
0.004j)...
  (0.062+(-0.062j))  (0.037+0.098j)  (-0.057+0.039j)  (-
0.131+0.065j)...
  (0.082+0.092j)  (0.070+0.014j)  (-0.060+0.081j)  (-0.056-
0.022j)...
  (-0.035+(-0.151j))  (-0.122+(-0.017j))  (-0.127+(-0.021j))
(0.075+(-0.074j))...
  (-0.003+(0.054j))  (-0.092+0.115j)  (0.092+0.106j)
(0.012+0.098j)...
  (-0.156+0.000j)  (0.012+(-0.098j))  (0.092+(-0.106j))  (-0.092+(-
0.115j))...
  (-0.003+(-0.054j))  (0.075+0.074j)  (-0.127+0.021j)  (-
0.122+0.017j)...
  (-0.035+0.151j)  (-0.056+0.022j)  (-0.060-0.081j)  (0.070+(-
0.014j))...
  (0.082+(-0.092j))  (-0.131+(-0.065j))  (-0.057+(-0.039j))
(0.037+(-0.098j))...
  (0.062+0.062j)  (0.119+0.004j)  (-0.022+(-0.161j))
(0.059+0.015j)...
  (0.024+0.059j)  (-0.137+0.047j)  (0.001+0.115j)  (0.053+(-
0.004j))...
  (0.098+0.026j)  (-0.038+0.106j)  (-0.115+0.055j)
(0.060+0.088j)...
  (0.021+(-0.028j))  (0.097+(-0.083j))  (0.040+0.111j)  (-
0.005+0.120j)];
preamble=[onalti onalti onalti onalti onalti onalti onalti onalti
onalti onalti otuziki altmisdort altmisdort ];%IEE 802.11a preamble
```

```
sample=100;
st=0.11;%our noise standart deviation constant
a1=0;
std_signal=std(onalti);
for c=1:48%calculation part of correlation

   for t=1:sample

       nois = st/(sqrt(2)).*((wgn(1,16,0))+(i*wgn(1,16,0)));%noise
generation with chageable standart deviation

      x1=onalti;
       y1=nois+onalti;%WE add noise for short preamble
       payda1=0.0413;
              payda2=sum((abs(y1)).^2)^2;

             % std_signal=std((y1));


      a1(t)=abs((y1)*(x1')).^2;

     a2(t)=abs((y1)*(y1')).^2;




   end
   var1(c)=var(a1./payda1);
     var2(c)=var(a2./payda2);
  k=sum(a1)/payda1;
  m=sum(a2)/payda2;
  n=sum(var1)/payda1;
  p=sum(var2)/payda2;
corrMatlab1(c)=k/sample;
varcorrMatlab1(c)=var(a1)/(sample*payda1);

corrMatlab2(c)=m/sample;
varcorrMatlab2(c)=var(a2)/(sample*payda2);
 crossteori(c)=1;
  ototeori(c)=(std_signal^4)/(((std_signal^2)+(st^2))^2);

 st=st-0.0025;%st=st+0.005 st-0.0001

end


 st2=0.030;
  hs=0;


  nois=st2/sqrt(2).*((wgn(1,320,0))+(i*wgn(1,320,0)));
 y2=preamble+nois;
 hs=abs(xcorr(onalti,y2))./sqrt(0.0413);%we calculate correlation
between short preamble and  noise with preamble


  plot(hs);
```

VHDL CODES

```
--------------------------------
  --  SHORT PREAMBLE CORRELATOR
--------------------------------
library IEEE;
use ieee.fixed_pkg.all;


package fp_vector IS
	type sfixed_vector is array (natural range<>) of sfixed(3 downto -4); --defining a vector
consists of sfixed numbers
	type sfixed_array is array (natural range <>) of sfixed_vector(1 to 2); --a vector consists of
complex numbers such a+bi in the form (a,b) where a and b are sfixed numbers
end;



library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use ieee.fixed_pkg.all;
use work.fp_vector.ALL;


entity crosscorr is
	port(clk:in std_logic;
		rst:in std_logic;
		Cns:out sfixed(3 downto -4));
end crosscorr;


architecture Behavioral of crosscorr is
```

```
--function to sum two complex numbers in the form such as
(a+bi)+(c+di)=(a,b)+(c,d)=(a+c,b+d)=(a+b)+(c+d)i

            function complex_sum (x,y:sfixed_vector(1 to 2)) return sfixed_vector is

                        variable s:sfixed_vector(1 to 2);

begin

                        s:=(resize(x(1)+y(1),3,-4),resize(x(2)+y(2),3,-4));

return s;

end function complex_sum;


--function to multiplex two complex number in the form such as (a+bi).(c+di)=ac+adi+bci-bd=(ac-
bd)+(ad+bc)i=(ac-bd,ad+bc)

            function complex_mux (x,y:sfixed_vector(1 to 2)) return sfixed is

                        variable m:sfixed(3 downto -4);

begin

                        m:=(resize(((x(1)*y(1)+x(2)*y(2))*(x(1)*y(1)+x(2)*y(2)))+((x(1)*y(2)-
x(2)*y(1))*(x(1)*y(2)-x(2)*y(1))),3,-4));

return m;

end function complex_mux;


            signal count1:positive range 1 to 335;

            signal sp:sfixed_array(1 to 16);

            signal cp:sfixed_array(1 to 32);

            signal lp:sfixed_array(1 to 64);

            signal preamble:sfixed_array(1 to 320);

            signal zero:sfixed_vector(1 to 2);

            signal zeros1:sfixed_array(1 to 30);

            signal preamble1:sfixed_array(1 to 350);

            signal Cs:sfixed_vector(1 to 335);

   signal Css:sfixed_vector(1 to 335);

   signal count: natural range 1 to 500_000;

   signal temp_clk_out: std_logic;


begin
```

```
--process  of 100Hz clock divider
 process(clk)
begin
if (rising_edge(clk)) then
count <= count + 1;
if(count= 500_000) then
temp_clk_out<=not temp_clk_out;
count<=1;
end if;
end if;
end process;

zero<=(to_sfixed(0,3,-4),to_sfixed(0,3,-4));--zero vector to add to the preamble for correlation
zeros1<=(others=>zero);

--short preamble values
sp<=(
(to_sfixed(0.05,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(-0.01,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.14,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.14,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(-0.01,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(0.05,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.13,3,-4)),
(to_sfixed(-0.08,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.14,3,-4)),
(to_sfixed(0.00,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.14,3,-4)),
(to_sfixed(-0.08,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.13,3,-4)));

--cyclic prefix values
cp<=(
(to_sfixed(-0.08,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.01,3,-4),to_sfixed(-0.09,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(-0.11,3,-4)),(to_sfixed(-0.09,3,-4),to_sfixed(-0.12,3,-4)),
(to_sfixed(-0.00,3,-4),to_sfixed(-0.05,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.07,3,-4)),
(to_sfixed(-0.13,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(-0.04,3,-4),to_sfixed(0.15,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.07,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(0.08,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(-0.07,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.04,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(-0.09,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(0.12,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(-0.02,3,-4),to_sfixed(-0.16,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(0.02,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(-0.14,3,-4),to_sfixed(0.05,3,-4)),
(to_sfixed(0.00,3,-4),to_sfixed(0.12,3,-4)),(to_sfixed(0.05,3,-4),to_sfixed(-0.00,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.03,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(0.11,3,-4)),
(to_sfixed(-0.12,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.09,3,-4)),
(to_sfixed(0.02,3,-4),to_sfixed(-0.03,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.08,3,-4)),
(to_sfixed(0.04,3,-4),to_sfixed(0.11,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.12,3,-4)));

--long preamble values
lp<=(
(to_sfixed(-0.01,3,-4),to_sfixed(-0.12,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(-0.11,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.08,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(0.03,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(-0.04,3,-4),to_sfixed(-0.11,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.03,3,-4)),
(to_sfixed(0.05,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.12,3,-4)),
(to_sfixed(-0.14,3,-4),to_sfixed(-0.05,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.02,3,-4),to_sfixed(0.16,3,-4)),
(to_sfixed(0.12,3,-4),to_sfixed(-0.00,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(0.04,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.04,3,-4)),
(to_sfixed(-0.13,3,-4),to_sfixed(0.07,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.09,3,-4)),
(to_sfixed(0.07,3,-4),to_sfixed(0.01,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.08,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(-0.15,3,-4)),
(to_sfixed(-0.12,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(-0.02,3,-4)),
(to_sfixed(0.08,3,-4),to_sfixed(-0.07,3,-4)),(to_sfixed(-0.00,3,-4),to_sfixed(0.05,3,-4)),
(to_sfixed(-0.09,3,-4),to_sfixed(0.115,3,-4)),(to_sfixed(0.092,3,-4),to_sfixed(0.106,3,-4)),
(to_sfixed(0.012,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.16,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(0.01,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.11,3,-4)),
(to_sfixed(-0.09,3,-4),to_sfixed(-0.12,3,-4)),(to_sfixed(-0.00,3,-4),to_sfixed(-0.05,3,-4)),
(to_sfixed(0.08,3,-4),to_sfixed(0.07,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(-0.12,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(0.15,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.08,3,-4)),
(to_sfixed(0.07,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(-0.09,3,-4)),
(to_sfixed(-0.13,3,-4),to_sfixed(-0.07,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(-0.04,3,-4)),
(to_sfixed(0.04,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.06,3,-4)),
(to_sfixed(0.12,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(-0.02,3,-4),to_sfixed(-0.16,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(0.06,3,-4)),
(to_sfixed(-0.14,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(0.12,3,-4)),
(to_sfixed(0.05,3,-4),to_sfixed(-0.00,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(0.03,3,-4)),
(to_sfixed(-0.04,3,-4),to_sfixed(0.11,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(0.06,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(-0.03,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(0.11,3,-4)),
(to_sfixed(-0.01,3,-4),to_sfixed(0.12,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.00,3,-4)));


preamble<=sp&sp&sp&sp&sp&sp&sp&sp&sp&sp&cp&lp&lp;--preamble structure is constructed by
concatenating

preamble1<=preamble&zeros1;--adding zero vector to preamble because we are going to shift the
signal during correlation


--process of short preamble and preamble correlation

process(clk)

        variable m:integer range 1 to 16:=1; --index of short preamble

        variable k:integer range 1 to 335:=1; --index of correlation vector = short
preamble+preamble-1=16+320-1=335

        variable corr:sfixed(3 downto -4):=to_sfixed(0,3,-4);--correlation value

        variable sum:sfixed_vector(1 to 335):=(others=>(to_sfixed(0,3,-4)));--summation of the
correlated values for one short preamble index

        variable a:integer range 0 to 334:=0;--shifting value 0 to (correlation vector length-1=335-
1=)334

begin

        if (rising_edge(clk)) then

                corr:=complex_mux(sp(m),preamble1(m+a));--calculated for every index of short
preamble

                sum(k):=sum(k)+corr;--summation of corr values for every index of short preamble
while a is not changing
```

```
                        if (m<16) then--check if we calculate every index of the short preamble
                                m:=m+1;
                        else
                                m:=1; --if we have calculated every index we are going to calculate it again
for another shifting value a
                                if(a<334) then--check if the shifting value reached its max
                                        a:=a+1;--if it isn't continue shifting
                                end if;
                                Cs(k)<=sum(k);--every summation value is assigned to the corresponding
element of correlation vector
                                if (k<335) then
                                        k:=k+1;
                                end if;
                        end if;
                end if;
end process;
--normalization is done by dividing every elements of correlation vector by 2^4=16
normalization1: for i in 1 to 335 generate
        Css(i)<=Cs(i) srl 4;
end generate;


--process to represent every elements of the correlation vector one by one in every clock signal
process(temp_clk_out,rst)
begin
        if (rst='1') then
                count1<=1;
        elsif (rising_edge(temp_clk_out)) then
                if (count1=335) then
                        count1<=1;
                else
                        count1<=count1+1;
                 end if;
        end if;
end process;

Cns<=Css(count1);


end Behavioral;





----------------------------

--LONG PREAMBLE CORRELATION

----------------------------




library IEEE;

use ieee.fixed_pkg.all;


package fp_vector IS

	type sfixed_vector is array (natural range<>) of sfixed(3 downto -4); --defining a vector
consists of sfixed numbers

	type sfixed_array is array (natural range <>) of sfixed_vector(1 to 2); --a vector consists of
complex numbers such a+bi in the form (a,b) where a and b are sfixed numbers

end;




library IEEE;

use IEEE.STD_LOGIC_1164.all;

use IEEE.NUMERIC_STD.all;

use ieee.fixed_pkg.all;

use work.fp_vector.ALL;


entity crosscorr is

	port(clk:in std_logic;

		rst:in std_logic;

		Cnl:out sfixed(3 downto -4));
```

```
end crosscorr;



architecture Behavioral of crosscorr_lp is


--function to sum two complex numbers in the form such as
(a+bi)+(c+di)=(a,b)+(c,d)=(a+c,b+d)=(a+b)+(c+d)i

            function complex_sum (x,y:sfixed_vector(1 to 2)) return sfixed_vector is

                        variable s:sfixed_vector(1 to 2);

begin

                        s:=(resize(x(1)+y(1),3,-4),resize(x(2)+y(2),3,-4));

return s;

end function complex_sum;


--function to multiplex two complex number in the form such as (a+bi).(c+di)=ac+adi+bci-bd=(ac-
bd)+(ad+bc)i=(ac-bd,ad+bc)

            function complex_mux (x,y:sfixed_vector(1 to 2)) return sfixed is

                        variable m:sfixed(3 downto -4);

begin

                        m:=(resize(((x(1)*y(1)+x(2)*y(2))*(x(1)*y(1)+x(2)*y(2)))+((x(1)*y(2)-
x(2)*y(1))*(x(1)*y(2)-x(2)*y(1))),3,-4));

return m;

end function complex_mux;


   signal count2:positive range 1 to 383;

            signal sp:sfixed_array(1 to 16);

            signal cp:sfixed_array(1 to 32);

            signal lp:sfixed_array(1 to 64);

            signal preamble:sfixed_array(1 to 320);

            signal zero:sfixed_vector(1 to 2);

            signal zeros2:sfixed_array(1 to 130);

            signal preamble2:sfixed_array(1 to 450);

            signal Cl:sfixed_vector(1 to 383);

   signal Cll:sfixed_vector(1 to 383);
```

```
    signal count: natural range 1 to 500_000;
    signal temp_clk_out: std_logic;

begin
zero<=(to_sfixed(0,16,-16),to_sfixed(0,16,-16));--zero vector to add to the preamble for correlation
zeros2<=(others=>zero);

--process  of 100Hz clock divider
 process(clk)
begin
if (rising_edge(clk)) then
count <= count + 1;
if(count= 500_000) then
temp_clk_out<=not temp_clk_out;
count<=1;
end if;
end if;
end process;

--short preamble values
sp<=(
(to_sfixed(0.05,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(-0.01,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.14,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.14,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(-0.01,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(0.05,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.13,3,-4)),
(to_sfixed(-0.08,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.14,3,-4)),
(to_sfixed(0.00,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.14,3,-4)),
(to_sfixed(-0.08,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.13,3,-4)));

--cyclic prefix values
cp<=(
(to_sfixed(-0.08,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.01,3,-4),to_sfixed(-0.09,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(-0.11,3,-4)),(to_sfixed(-0.09,3,-4),to_sfixed(-0.12,3,-4)),
(to_sfixed(-0.00,3,-4),to_sfixed(-0.05,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.07,3,-4)),
(to_sfixed(-0.13,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(-0.04,3,-4),to_sfixed(0.15,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.07,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(0.08,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(-0.07,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.04,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(-0.09,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(0.12,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(-0.02,3,-4),to_sfixed(-0.16,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(0.02,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(-0.14,3,-4),to_sfixed(0.05,3,-4)),
(to_sfixed(0.00,3,-4),to_sfixed(0.12,3,-4)),(to_sfixed(0.05,3,-4),to_sfixed(-0.00,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.03,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(0.11,3,-4)),
(to_sfixed(-0.12,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.09,3,-4)),
(to_sfixed(0.02,3,-4),to_sfixed(-0.03,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.08,3,-4)),
(to_sfixed(0.04,3,-4),to_sfixed(0.11,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.12,3,-4)));

--long preamble values
lp<=(
(to_sfixed(-0.01,3,-4),to_sfixed(-0.12,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(-0.11,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.08,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(0.03,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(-0.04,3,-4),to_sfixed(-0.11,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.03,3,-4)),
(to_sfixed(0.05,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.12,3,-4)),
(to_sfixed(-0.14,3,-4),to_sfixed(-0.05,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.02,3,-4),to_sfixed(0.16,3,-4)),
(to_sfixed(0.12,3,-4),to_sfixed(-0.00,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(0.04,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.04,3,-4)),
(to_sfixed(-0.13,3,-4),to_sfixed(0.07,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.09,3,-4)),
(to_sfixed(0.07,3,-4),to_sfixed(0.01,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.08,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(-0.15,3,-4)),
(to_sfixed(-0.12,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(-0.02,3,-4)),
(to_sfixed(0.08,3,-4),to_sfixed(-0.07,3,-4)),(to_sfixed(-0.00,3,-4),to_sfixed(0.05,3,-4)),
(to_sfixed(-0.09,3,-4),to_sfixed(0.115,3,-4)),(to_sfixed(0.092,3,-4),to_sfixed(0.106,3,-4)),
(to_sfixed(0.012,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.16,3,-4),to_sfixed(0.00,3,-4)),

(to_sfixed(0.01,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.11,3,-4)),

(to_sfixed(-0.09,3,-4),to_sfixed(-0.12,3,-4)),(to_sfixed(-0.00,3,-4),to_sfixed(-0.05,3,-4)),

(to_sfixed(0.08,3,-4),to_sfixed(0.07,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.02,3,-4)),

(to_sfixed(-0.12,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(0.15,3,-4)),

(to_sfixed(-0.06,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.08,3,-4)),

(to_sfixed(0.07,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(-0.09,3,-4)),

(to_sfixed(-0.13,3,-4),to_sfixed(-0.07,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(-0.04,3,-4)),

(to_sfixed(0.04,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.06,3,-4)),

(to_sfixed(0.12,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(-0.02,3,-4),to_sfixed(-0.16,3,-4)),

(to_sfixed(0.06,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(0.06,3,-4)),

(to_sfixed(-0.14,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(0.12,3,-4)),

(to_sfixed(0.05,3,-4),to_sfixed(-0.00,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(0.03,3,-4)),

(to_sfixed(-0.04,3,-4),to_sfixed(0.11,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(0.06,3,-4)),

(to_sfixed(0.06,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(-0.03,3,-4)),

(to_sfixed(0.09,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(0.11,3,-4)),

(to_sfixed(-0.01,3,-4),to_sfixed(0.12,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.00,3,-4)));


preamble<=sp&sp&sp&sp&sp&sp&sp&sp&sp&sp&cp&lp&lp;--preamble structure is constructed by
concatenating

preamble2<=preamble&zeros2;--adding zero vector to preamble because we are going to shift the
signal during correlation



--process of long preamble and preamble correlation

process(clk)

            variable m:integer range 1 to 64:=1;--index of long preamble

            variable k:integer range 1 to 383:=1; --index of correlation vector = long preamble+preamble-
1=64+320-1=383

            variable corr:sfixed(16 downto -16):=to_sfixed(0,16,-16);--correlation value

            variable sum:sfixed_vector(1 to 383):=(others=>(to_sfixed(0,16,-16)));--summation of the
correlated values for one short preamble index

            variable a:integer range 0 to 382:=0;--shifting value 0 to (correlation vector length-1=383-
1=)382
```

```
begin
        if (rising_edge(clk)) then
                corr:=complex_mux(lp(m),preamble2(m+a));
                sum(k):=sum(k)+corr;
                if (m<64) then
                        m:=m+1;
                else
                        m:=1;
                        if(a<382) then
                                a:=a+1;
                        end if;
                        Cl(k)<=sum(k);
                        if (k<383) then
                                k:=k+1;
                        end if;
                end if;
        end if;
end process;

--normalization is done by dividing every elements of correlation vector by 2^4=16
normalization2: for i in 1 to 383 generate
        Cll(i)<=Cl(i) srl 4;
end generate;

--process to represent every elements of the correlation vector one by one in every clock signal
process(temp_clk_out,rst)
begin
        if (rst='1') then
                count2<=1;
        elsif (rising_edge(temp_clk_out)) then
                if (count2=383) then
                        count2<=1;
                else
```

```
                                count2<=count2+1;
                        end if;
            end if;
end process;
Cnl<=Cll(count2);

end Behavioral;
--------------------------------------------------------------------------------
-- CORRELATION OF PREAMBLE WITH NOISE
--------------------------------------------------------------------------------


library IEEE;
use ieee.fixed_pkg.all;


package fp_vector IS
            type sfixed_vector is array (natural range<>) of sfixed(3 downto -4); --defining a vector
consists of sfixed numbers
            type sfixed_array is array (natural range <>) of sfixed_vector(1 to 2); --a vector consists of
complex numbers such a+bi in the form (a,b) where a and b are sfixed numbers
end;



library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
use ieee.fixed_pkg.all;
use work.fp_vector.ALL;


entity crosscorr_noise is
            port(clk:in std_logic;
                        rst:in std_logic;
                        C:out sfixed(3 downto -4));
end crosscorr_noise;
```

```
architecture Behavioral of crosscorr_noise is

--function to sum two complex numbers in the form such as
(a+bi)+(c+di)=(a,b)+(c,d)=(a+c,b+d)=(a+b)+(c+d)i

            function complex_sum (x,y:sfixed_vector(1 to 2)) return sfixed_vector is

                        variable s:sfixed_vector(1 to 2);

begin

                        s:=(resize(x(1)+y(1),3,-4),resize(x(2)+y(2),3,-4));

return s;

end function complex_sum;


--function to multiplex two complex number in the form such as (a+bi).(c+di)=ac+adi+bci-bd=(ac-
bd)+(ad+bc)i=(ac-bd,ad+bc)

            function complex_mux (x,y:sfixed_vector(1 to 2)) return sfixed is

                        variable m:sfixed(3 downto -4);

begin

                        m:=(resize(((x(1)*y(1)+x(2)*y(2))*(x(1)*y(1)+x(2)*y(2)))+((x(1)*y(2)-
x(2)*y(1))*(x(1)*y(2)-x(2)*y(1))),3,-4));

return m;

end function complex_mux;

            signal count3:positive range 1 to 335;

            signal sp:sfixed_array(1 to 16);

            signal cp:sfixed_array(1 to 32);

            signal lp:sfixed_array(1 to 64);

            signal preamble:sfixed_array(1 to 320);

            signal preamble_w_noise,noise:sfixed_array(1 to 320);

            signal zero:sfixed_vector(1 to 2);

            signal zeros1:sfixed_array(1 to 30);

            signal preamble3:sfixed_array(1 to 350);

            signal Cn:sfixed_vector(1 to 335);

            signal Cnn:sfixed_vector(1 to 335);

begin


--process  of 100Hz clock divider
```

```
 process(clk)
begin
if (rising_edge(clk)) then
count <= count + 1;
if(count= 500_000) then
temp_clk_out<=not temp_clk_out;
count<=1;
end if;
end if;
end process;

zero<=(to_sfixed(0,3,-4),to_sfixed(0,3,-4));--zero vector to add to the preamble for correlation
zeros1<=(others=>zero);

--short preamble values
sp<=(
(to_sfixed(0.05,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(-0.01,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.14,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.14,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(-0.01,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(0.05,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.13,3,-4)),
(to_sfixed(-0.08,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.14,3,-4)),
(to_sfixed(0.00,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.14,3,-4)),
(to_sfixed(-0.08,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.13,3,-4)));

--cyclic prefix values
cp<=(
(to_sfixed(-0.08,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.01,3,-4),to_sfixed(-0.09,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(-0.11,3,-4)),(to_sfixed(-0.09,3,-4),to_sfixed(-0.12,3,-4)),
(to_sfixed(-0.00,3,-4),to_sfixed(-0.05,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.07,3,-4)),
(to_sfixed(-0.13,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(-0.04,3,-4),to_sfixed(0.15,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.07,3,-4),to_sfixed(-0.01,3,-4)),
(to_sfixed(0.08,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(-0.07,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.04,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(-0.09,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(0.12,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(-0.02,3,-4),to_sfixed(-0.16,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(0.02,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(-0.14,3,-4),to_sfixed(0.05,3,-4)),
(to_sfixed(0.00,3,-4),to_sfixed(0.12,3,-4)),(to_sfixed(0.05,3,-4),to_sfixed(-0.00,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.03,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(0.11,3,-4)),
(to_sfixed(-0.12,3,-4),to_sfixed(0.06,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.09,3,-4)),
(to_sfixed(0.02,3,-4),to_sfixed(-0.03,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.08,3,-4)),
(to_sfixed(0.04,3,-4),to_sfixed(0.11,3,-4)),(to_sfixed(-0.01,3,-4),to_sfixed(0.12,3,-4)));

--long preamble values
lp<=(
(to_sfixed(-0.01,3,-4),to_sfixed(-0.12,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(-0.11,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(0.08,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(0.03,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(-0.04,3,-4),to_sfixed(-0.11,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.03,3,-4)),
(to_sfixed(0.05,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(-0.12,3,-4)),
(to_sfixed(-0.14,3,-4),to_sfixed(-0.05,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.02,3,-4),to_sfixed(0.16,3,-4)),
(to_sfixed(0.12,3,-4),to_sfixed(-0.00,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(-0.06,3,-4)),
(to_sfixed(0.04,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.04,3,-4)),
(to_sfixed(-0.13,3,-4),to_sfixed(0.07,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.09,3,-4)),
(to_sfixed(0.07,3,-4),to_sfixed(0.01,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.08,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(-0.15,3,-4)),
(to_sfixed(-0.12,3,-4),to_sfixed(-0.02,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(-0.02,3,-4)),
(to_sfixed(0.08,3,-4),to_sfixed(-0.07,3,-4)),(to_sfixed(-0.00,3,-4),to_sfixed(0.05,3,-4)),
(to_sfixed(-0.09,3,-4),to_sfixed(0.115,3,-4)),(to_sfixed(0.092,3,-4),to_sfixed(0.106,3,-4)),
(to_sfixed(0.012,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(-0.16,3,-4),to_sfixed(0.00,3,-4)),
(to_sfixed(0.01,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(-0.11,3,-4)),
(to_sfixed(-0.09,3,-4),to_sfixed(-0.12,3,-4)),(to_sfixed(-0.00,3,-4),to_sfixed(-0.05,3,-4)),
(to_sfixed(0.08,3,-4),to_sfixed(0.07,3,-4)),(to_sfixed(-0.13,3,-4),to_sfixed(0.02,3,-4)),
(to_sfixed(-0.12,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.04,3,-4),to_sfixed(0.15,3,-4)),
(to_sfixed(-0.06,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(0.08,3,-4)),
(to_sfixed(0.07,3,-4),to_sfixed(-0.01,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(-0.09,3,-4)),
(to_sfixed(-0.13,3,-4),to_sfixed(-0.07,3,-4)),(to_sfixed(-0.06,3,-4),to_sfixed(-0.04,3,-4)),
(to_sfixed(0.04,3,-4),to_sfixed(-0.09,3,-4)),(to_sfixed(0.06,3,-4),to_sfixed(0.06,3,-4)),
(to_sfixed(0.12,3,-4),to_sfixed(0.00,3,-4)),(to_sfixed(-0.02,3,-4),to_sfixed(-0.16,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(0.02,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(0.06,3,-4)),
(to_sfixed(-0.14,3,-4),to_sfixed(0.05,3,-4)),(to_sfixed(0.00,3,-4),to_sfixed(0.12,3,-4)),
(to_sfixed(0.05,3,-4),to_sfixed(-0.00,3,-4)),(to_sfixed(0.09,3,-4),to_sfixed(0.03,3,-4)),
(to_sfixed(-0.04,3,-4),to_sfixed(0.11,3,-4)),(to_sfixed(-0.12,3,-4),to_sfixed(0.06,3,-4)),
(to_sfixed(0.06,3,-4),to_sfixed(0.09,3,-4)),(to_sfixed(0.02,3,-4),to_sfixed(-0.03,3,-4)),
(to_sfixed(0.09,3,-4),to_sfixed(-0.08,3,-4)),(to_sfixed(0.04,3,-4),to_sfixed(0.11,3,-4)),
(to_sfixed(-0.01,3,-4),to_sfixed(0.12,3,-4)),(to_sfixed(0.08,3,-4),to_sfixed(0.00,3,-4)));

--additive white gaussian noise samples with 0 mean and variance of 1 which are created in Matlab.
noise<=(
(to_sfixed(0.014,16,-16),to_sfixed(-0.014,16,-16)),(to_sfixed(0.055,16,-16),to_sfixed(-0.052,16,-16)),
(to_sfixed(0.015,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(0.018,16,-16),to_sfixed(-0.001,16,-16)),
(to_sfixed(0.028,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.004,16,-16),to_sfixed(0.019,16,-16)),
(to_sfixed(-0.001,16,-16),to_sfixed(0.025,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(-0.001,16,-16)),
(to_sfixed(-0.025,16,-16),to_sfixed(0.016,16,-16)),(to_sfixed(0.014,16,-16),to_sfixed(-0.014,16,-16)),
(to_sfixed(0.036,16,-16),to_sfixed(0.035,16,-16)),(to_sfixed(0.020,16,-16),to_sfixed(-0.007,16,-16)),
(to_sfixed(0.029,16,-16),to_sfixed(0.014,16,-16)),(to_sfixed(-0.004,16,-16),to_sfixed(-0.012,16,-16)),
(to_sfixed(0.011,16,-16),to_sfixed(-0.013,16,-16)),(to_sfixed(0.006,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.015,16,-16),to_sfixed(0.002,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(0.049,16,-16)),
(to_sfixed(0.048,16,-16),to_sfixed(0.053,16,-16)),(to_sfixed(-0.030,16,-16),to_sfixed(-0.019,16,-16)),
(to_sfixed(-0.017,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.040,16,-16),to_sfixed(0.017,16,-16)),
(to_sfixed(0.038,16,-16),to_sfixed(-0.018,16,-16)),(to_sfixed(0.001,16,-16),to_sfixed(-0.011,16,-16)),
(to_sfixed(-0.022,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.034,16,-16),to_sfixed(0.011,16,-16)),
(to_sfixed(0.009,16,-16),to_sfixed(0.031,16,-16)),(to_sfixed(0.043,16,-16),to_sfixed(0.025,16,-16)),
(to_sfixed(0.001,16,-16),to_sfixed(0.031,16,-16)),(to_sfixed(0.003,16,-16),to_sfixed(0.023,16,-16)),
(to_sfixed(0.001,16,-16),to_sfixed(-0.017,16,-16)),(to_sfixed(-0.021,16,-16),to_sfixed(0.028,16,-16)),
(to_sfixed(-0.013,16,-16),to_sfixed(0.000,16,-16)),(to_sfixed(0.004,16,-16),to_sfixed(0.011,16,-16)),
(to_sfixed(0.011,16,-16),to_sfixed(0.002,16,-16)),(to_sfixed(0.015,16,-16),to_sfixed(0.035,16,-16)),
(to_sfixed(-0.001,16,-16),to_sfixed(0.011,16,-16)),(to_sfixed(-0.036,16,-16),to_sfixed(-0.029,16,-16)),
(to_sfixed(0.016,16,-16),to_sfixed(-0.037,16,-16)),(to_sfixed(0.007,16,-16),to_sfixed(0.010,16,-16)),
(to_sfixed(0.031,16,-16),to_sfixed(0.005,16,-16)),(to_sfixed(-0.020,16,-16),to_sfixed(0.032,16,-16)),
(to_sfixed(0.013,16,-16),to_sfixed(-0.008,16,-16)),(to_sfixed(0.022,16,-16),to_sfixed(0.026,16,-16)),
(to_sfixed(0.022,16,-16),to_sfixed(-0.012,16,-16)),(to_sfixed(-0.002,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.010,16,-16),to_sfixed(-0.017,16,-16)),(to_sfixed(-0.011,16,-16),to_sfixed(0.036,16,-16)),
(to_sfixed(-0.002,16,-16),to_sfixed(0.011,16,-16)),(to_sfixed(0.011,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.006,16,-16),to_sfixed(-0.033,16,-16)),(to_sfixed(-0.026,16,-16),to_sfixed(-0.029,16,-16)),
(to_sfixed(0.013,16,-16),to_sfixed(-0.009,16,-16)),(to_sfixed(0.028,16,-16),to_sfixed(-0.003,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.028,16,-16),to_sfixed(0.012,16,-16)),
(to_sfixed(0.044,16,-16),to_sfixed(-0.001,16,-16)),(to_sfixed(0.009,16,-16),to_sfixed(0.030,16,-16)),
(to_sfixed(0.029,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(0.006,16,-16)),
(to_sfixed(0.028,16,-16),to_sfixed(0.045,16,-16)),(to_sfixed(0.036,16,-16),to_sfixed(-0.023,16,-16)),
(to_sfixed(-0.003,16,-16),to_sfixed(-0.018,16,-16)),(to_sfixed(0.012,16,-16),to_sfixed(-0.019,16,-16)),
(to_sfixed(-0.035,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.014,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.021,16,-16),to_sfixed(-0.047,16,-16)),(to_sfixed(-0.042,16,-16),to_sfixed(0.007,16,-16)),
(to_sfixed(0.030,16,-16),to_sfixed(0.014,16,-16)),(to_sfixed(0.007,16,-16),to_sfixed(-0.007,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(-0.038,16,-16)),(to_sfixed(-0.012,16,-16),to_sfixed(0.049,16,-16)),
(to_sfixed(-0.007,16,-16),to_sfixed(0.005,16,-16)),(to_sfixed(-0.026,16,-16),to_sfixed(0.003,16,-16)),
(to_sfixed(-0.011,16,-16),to_sfixed(0.009,16,-16)),(to_sfixed(0.006,16,-16),to_sfixed(0.001,16,-16)),
(to_sfixed(0.007,16,-16),to_sfixed(0.007,16,-16)),(to_sfixed(0.034,16,-16),to_sfixed(0.020,16,-16)),
(to_sfixed(-0.017,16,-16),to_sfixed(-0.031,16,-16)),(to_sfixed(-0.013,16,-16),to_sfixed(-0.015,16,-
16)),
(to_sfixed(-0.026,16,-16),to_sfixed(0.026,16,-16)),(to_sfixed(-0.033,16,-16),to_sfixed(0.003,16,-16)),
(to_sfixed(0.050,16,-16),to_sfixed(-0.008,16,-16)),(to_sfixed(-0.017,16,-16),to_sfixed(0.008,16,-16)),
(to_sfixed(0.026,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(-0.011,16,-16),to_sfixed(0.032,16,-16)),
(to_sfixed(-0.063,16,-16),to_sfixed(-0.001,16,-16)),(to_sfixed(-0.012,16,-16),to_sfixed(0.000,16,-16)),
(to_sfixed(-0.045,16,-16),to_sfixed(0.018,16,-16)),(to_sfixed(-0.001,16,-16),to_sfixed(-0.011,16,-16)),
(to_sfixed(-0.002,16,-16),to_sfixed(0.0048,16,-16)),(to_sfixed(0.038,16,-16),to_sfixed(-0.005,16,-16)),
(to_sfixed(0.006,16,-16),to_sfixed(0.036,16,-16)),(to_sfixed(0.016,16,-16),to_sfixed(-0.028,16,-16)),
(to_sfixed(0.010,16,-16),to_sfixed(0.026,16,-16)),(to_sfixed(-0.020,16,-16),to_sfixed(0.029,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(0.013,16,-16)),(to_sfixed(-0.030,16,-16),to_sfixed(0.015,16,-16)),
(to_sfixed(0.021,16,-16),to_sfixed(-0.020,16,-16)),(to_sfixed(0.011,16,-16),to_sfixed(-0.023,16,-16)),
(to_sfixed(0.014,16,-16),to_sfixed(-0.014,16,-16)),(to_sfixed(0.055,16,-16),to_sfixed(-0.052,16,-16)),
(to_sfixed(0.015,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(0.018,16,-16),to_sfixed(-0.001,16,-16)),
(to_sfixed(0.028,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.004,16,-16),to_sfixed(0.019,16,-16)),
(to_sfixed(-0.001,16,-16),to_sfixed(0.025,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(-0.001,16,-16)),
(to_sfixed(-0.025,16,-16),to_sfixed(0.016,16,-16)),(to_sfixed(0.014,16,-16),to_sfixed(-0.014,16,-16)),
(to_sfixed(0.036,16,-16),to_sfixed(0.035,16,-16)),(to_sfixed(0.020,16,-16),to_sfixed(-0.007,16,-16)),
(to_sfixed(0.029,16,-16),to_sfixed(0.014,16,-16)),(to_sfixed(-0.004,16,-16),to_sfixed(-0.012,16,-16)),
(to_sfixed(0.011,16,-16),to_sfixed(-0.013,16,-16)),(to_sfixed(0.006,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.015,16,-16),to_sfixed(0.002,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(0.049,16,-16)),
(to_sfixed(0.048,16,-16),to_sfixed(0.053,16,-16)),(to_sfixed(-0.030,16,-16),to_sfixed(-0.019,16,-16)),
(to_sfixed(-0.017,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.040,16,-16),to_sfixed(0.017,16,-16)),
(to_sfixed(0.038,16,-16),to_sfixed(-0.018,16,-16)),(to_sfixed(0.001,16,-16),to_sfixed(-0.011,16,-16)),
(to_sfixed(-0.022,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.034,16,-16),to_sfixed(0.011,16,-16)),
(to_sfixed(0.009,16,-16),to_sfixed(0.031,16,-16)),(to_sfixed(0.043,16,-16),to_sfixed(0.025,16,-16)),
(to_sfixed(0.001,16,-16),to_sfixed(0.031,16,-16)),(to_sfixed(0.003,16,-16),to_sfixed(0.023,16,-16)),
(to_sfixed(0.001,16,-16),to_sfixed(-0.017,16,-16)),(to_sfixed(-0.021,16,-16),to_sfixed(0.028,16,-16)),
(to_sfixed(-0.013,16,-16),to_sfixed(0.000,16,-16)),(to_sfixed(0.004,16,-16),to_sfixed(0.011,16,-16)),
(to_sfixed(0.011,16,-16),to_sfixed(0.002,16,-16)),(to_sfixed(0.015,16,-16),to_sfixed(0.035,16,-16)),
(to_sfixed(-0.001,16,-16),to_sfixed(0.011,16,-16)),(to_sfixed(-0.036,16,-16),to_sfixed(-0.029,16,-16)),
(to_sfixed(0.016,16,-16),to_sfixed(-0.037,16,-16)),(to_sfixed(0.007,16,-16),to_sfixed(0.010,16,-16)),
(to_sfixed(0.031,16,-16),to_sfixed(0.005,16,-16)),(to_sfixed(-0.020,16,-16),to_sfixed(0.032,16,-16)),
(to_sfixed(0.013,16,-16),to_sfixed(-0.008,16,-16)),(to_sfixed(0.022,16,-16),to_sfixed(0.026,16,-16)),
(to_sfixed(0.022,16,-16),to_sfixed(-0.012,16,-16)),(to_sfixed(-0.002,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.010,16,-16),to_sfixed(-0.017,16,-16)),(to_sfixed(-0.011,16,-16),to_sfixed(0.036,16,-16)),
(to_sfixed(-0.002,16,-16),to_sfixed(0.011,16,-16)),(to_sfixed(0.011,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.006,16,-16),to_sfixed(-0.033,16,-16)),(to_sfixed(-0.026,16,-16),to_sfixed(-0.029,16,-16)),
(to_sfixed(0.013,16,-16),to_sfixed(-0.009,16,-16)),(to_sfixed(0.028,16,-16),to_sfixed(-0.003,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.028,16,-16),to_sfixed(0.012,16,-16)),
(to_sfixed(0.044,16,-16),to_sfixed(-0.001,16,-16)),(to_sfixed(0.009,16,-16),to_sfixed(0.030,16,-16)),
(to_sfixed(0.029,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(0.006,16,-16)),
(to_sfixed(0.028,16,-16),to_sfixed(0.045,16,-16)),(to_sfixed(0.036,16,-16),to_sfixed(-0.023,16,-16)),
(to_sfixed(-0.003,16,-16),to_sfixed(-0.018,16,-16)),(to_sfixed(0.012,16,-16),to_sfixed(-0.019,16,-16)),
(to_sfixed(-0.035,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.014,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.021,16,-16),to_sfixed(-0.047,16,-16)),(to_sfixed(-0.042,16,-16),to_sfixed(0.007,16,-16)),
(to_sfixed(0.030,16,-16),to_sfixed(0.014,16,-16)),(to_sfixed(0.007,16,-16),to_sfixed(-0.007,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(-0.038,16,-16)),(to_sfixed(-0.012,16,-16),to_sfixed(0.049,16,-16)),
(to_sfixed(-0.007,16,-16),to_sfixed(0.005,16,-16)),(to_sfixed(-0.026,16,-16),to_sfixed(0.003,16,-16)),
(to_sfixed(-0.011,16,-16),to_sfixed(0.009,16,-16)),(to_sfixed(0.006,16,-16),to_sfixed(0.001,16,-16)),
(to_sfixed(0.007,16,-16),to_sfixed(0.007,16,-16)),(to_sfixed(0.034,16,-16),to_sfixed(0.020,16,-16)),
(to_sfixed(-0.017,16,-16),to_sfixed(-0.031,16,-16)),(to_sfixed(-0.013,16,-16),to_sfixed(-0.015,16,-
16)),
(to_sfixed(-0.026,16,-16),to_sfixed(0.026,16,-16)),(to_sfixed(-0.033,16,-16),to_sfixed(0.003,16,-16)),
(to_sfixed(0.050,16,-16),to_sfixed(-0.008,16,-16)),(to_sfixed(-0.017,16,-16),to_sfixed(0.008,16,-16)),
(to_sfixed(0.026,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(-0.011,16,-16),to_sfixed(0.032,16,-16)),
(to_sfixed(-0.063,16,-16),to_sfixed(-0.001,16,-16)),(to_sfixed(-0.012,16,-16),to_sfixed(0.000,16,-16)),
(to_sfixed(-0.045,16,-16),to_sfixed(0.018,16,-16)),(to_sfixed(-0.001,16,-16),to_sfixed(-0.011,16,-16)),
(to_sfixed(-0.002,16,-16),to_sfixed(0.0048,16,-16)),(to_sfixed(0.038,16,-16),to_sfixed(-0.005,16,-16)),
(to_sfixed(0.006,16,-16),to_sfixed(0.036,16,-16)),(to_sfixed(0.016,16,-16),to_sfixed(-0.028,16,-16)),
(to_sfixed(0.010,16,-16),to_sfixed(0.026,16,-16)),(to_sfixed(-0.020,16,-16),to_sfixed(0.029,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(0.013,16,-16)),(to_sfixed(-0.030,16,-16),to_sfixed(0.015,16,-16)),
(to_sfixed(0.021,16,-16),to_sfixed(-0.020,16,-16)),(to_sfixed(0.011,16,-16),to_sfixed(-0.023,16,-16)),
(to_sfixed(0.014,16,-16),to_sfixed(-0.014,16,-16)),(to_sfixed(0.055,16,-16),to_sfixed(-0.052,16,-16)),
(to_sfixed(0.015,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(0.018,16,-16),to_sfixed(-0.001,16,-16)),
(to_sfixed(0.028,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.004,16,-16),to_sfixed(0.019,16,-16)),
(to_sfixed(-0.001,16,-16),to_sfixed(0.025,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(-0.001,16,-16)),
(to_sfixed(-0.025,16,-16),to_sfixed(0.016,16,-16)),(to_sfixed(0.014,16,-16),to_sfixed(-0.014,16,-16)),
(to_sfixed(0.036,16,-16),to_sfixed(0.035,16,-16)),(to_sfixed(0.020,16,-16),to_sfixed(-0.007,16,-16)),
(to_sfixed(0.029,16,-16),to_sfixed(0.014,16,-16)),(to_sfixed(-0.004,16,-16),to_sfixed(-0.012,16,-16)),
(to_sfixed(0.011,16,-16),to_sfixed(-0.013,16,-16)),(to_sfixed(0.006,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.015,16,-16),to_sfixed(0.002,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(0.049,16,-16)),
(to_sfixed(0.048,16,-16),to_sfixed(0.053,16,-16)),(to_sfixed(-0.030,16,-16),to_sfixed(-0.019,16,-16)),
(to_sfixed(-0.017,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.040,16,-16),to_sfixed(0.017,16,-16)),
(to_sfixed(0.038,16,-16),to_sfixed(-0.018,16,-16)),(to_sfixed(0.001,16,-16),to_sfixed(-0.011,16,-16)),
(to_sfixed(-0.022,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.034,16,-16),to_sfixed(0.011,16,-16)),
(to_sfixed(0.009,16,-16),to_sfixed(0.031,16,-16)),(to_sfixed(0.043,16,-16),to_sfixed(0.025,16,-16)),
(to_sfixed(0.001,16,-16),to_sfixed(0.031,16,-16)),(to_sfixed(0.003,16,-16),to_sfixed(0.023,16,-16)),
(to_sfixed(0.001,16,-16),to_sfixed(-0.017,16,-16)),(to_sfixed(-0.021,16,-16),to_sfixed(0.028,16,-16)),
(to_sfixed(-0.013,16,-16),to_sfixed(0.000,16,-16)),(to_sfixed(0.004,16,-16),to_sfixed(0.011,16,-16)),
(to_sfixed(0.011,16,-16),to_sfixed(0.002,16,-16)),(to_sfixed(0.015,16,-16),to_sfixed(0.035,16,-16)),
(to_sfixed(-0.001,16,-16),to_sfixed(0.011,16,-16)),(to_sfixed(-0.036,16,-16),to_sfixed(-0.029,16,-16)),
(to_sfixed(0.016,16,-16),to_sfixed(-0.037,16,-16)),(to_sfixed(0.007,16,-16),to_sfixed(0.010,16,-16)),
(to_sfixed(0.031,16,-16),to_sfixed(0.005,16,-16)),(to_sfixed(-0.020,16,-16),to_sfixed(0.032,16,-16)),
(to_sfixed(0.013,16,-16),to_sfixed(-0.008,16,-16)),(to_sfixed(0.022,16,-16),to_sfixed(0.026,16,-16)),
(to_sfixed(0.022,16,-16),to_sfixed(-0.012,16,-16)),(to_sfixed(-0.002,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.010,16,-16),to_sfixed(-0.017,16,-16)),(to_sfixed(-0.011,16,-16),to_sfixed(0.036,16,-16)),
(to_sfixed(-0.002,16,-16),to_sfixed(0.011,16,-16)),(to_sfixed(0.011,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.006,16,-16),to_sfixed(-0.033,16,-16)),(to_sfixed(-0.026,16,-16),to_sfixed(-0.029,16,-16)),
(to_sfixed(0.013,16,-16),to_sfixed(-0.009,16,-16)),(to_sfixed(0.028,16,-16),to_sfixed(-0.003,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.028,16,-16),to_sfixed(0.012,16,-16)),
(to_sfixed(0.044,16,-16),to_sfixed(-0.001,16,-16)),(to_sfixed(0.009,16,-16),to_sfixed(0.030,16,-16)),
(to_sfixed(0.029,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(0.006,16,-16)),
(to_sfixed(0.028,16,-16),to_sfixed(0.045,16,-16)),(to_sfixed(0.036,16,-16),to_sfixed(-0.023,16,-16)),
(to_sfixed(-0.003,16,-16),to_sfixed(-0.018,16,-16)),(to_sfixed(0.012,16,-16),to_sfixed(-0.019,16,-16)),
(to_sfixed(-0.035,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.014,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.021,16,-16),to_sfixed(-0.047,16,-16)),(to_sfixed(-0.042,16,-16),to_sfixed(0.007,16,-16)),
(to_sfixed(0.030,16,-16),to_sfixed(0.014,16,-16)),(to_sfixed(0.007,16,-16),to_sfixed(-0.007,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(-0.038,16,-16)),(to_sfixed(-0.012,16,-16),to_sfixed(0.049,16,-16)),
(to_sfixed(-0.007,16,-16),to_sfixed(0.005,16,-16)),(to_sfixed(-0.026,16,-16),to_sfixed(0.003,16,-16)),
(to_sfixed(-0.011,16,-16),to_sfixed(0.009,16,-16)),(to_sfixed(0.006,16,-16),to_sfixed(0.001,16,-16)),
(to_sfixed(0.007,16,-16),to_sfixed(0.007,16,-16)),(to_sfixed(0.034,16,-16),to_sfixed(0.020,16,-16)),
(to_sfixed(-0.017,16,-16),to_sfixed(-0.031,16,-16)),(to_sfixed(-0.013,16,-16),to_sfixed(-0.015,16,-
16)),
(to_sfixed(-0.026,16,-16),to_sfixed(0.026,16,-16)),(to_sfixed(-0.033,16,-16),to_sfixed(0.003,16,-16)),
(to_sfixed(0.050,16,-16),to_sfixed(-0.008,16,-16)),(to_sfixed(-0.017,16,-16),to_sfixed(0.008,16,-16)),
(to_sfixed(0.026,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(-0.011,16,-16),to_sfixed(0.032,16,-16)),
(to_sfixed(-0.063,16,-16),to_sfixed(-0.001,16,-16)),(to_sfixed(-0.012,16,-16),to_sfixed(0.000,16,-16)),
(to_sfixed(-0.045,16,-16),to_sfixed(0.018,16,-16)),(to_sfixed(-0.001,16,-16),to_sfixed(-0.011,16,-16)),
(to_sfixed(-0.002,16,-16),to_sfixed(0.0048,16,-16)),(to_sfixed(0.038,16,-16),to_sfixed(-0.005,16,-16)),
(to_sfixed(0.006,16,-16),to_sfixed(0.036,16,-16)),(to_sfixed(0.016,16,-16),to_sfixed(-0.028,16,-16)),
(to_sfixed(0.010,16,-16),to_sfixed(0.026,16,-16)),(to_sfixed(-0.020,16,-16),to_sfixed(0.029,16,-16)),
(to_sfixed(-0.014,16,-16),to_sfixed(0.013,16,-16)),(to_sfixed(-0.030,16,-16),to_sfixed(0.015,16,-16)),
(to_sfixed(0.021,16,-16),to_sfixed(-0.020,16,-16)),(to_sfixed(0.011,16,-16),to_sfixed(-0.023,16,-16)),
(to_sfixed(0.014,16,-16),to_sfixed(-0.014,16,-16)),(to_sfixed(0.055,16,-16),to_sfixed(-0.052,16,-16)),
(to_sfixed(0.015,16,-16),to_sfixed(0.012,16,-16)),(to_sfixed(0.018,16,-16),to_sfixed(-0.001,16,-16)),
(to_sfixed(0.028,16,-16),to_sfixed(-0.022,16,-16)),(to_sfixed(0.004,16,-16),to_sfixed(0.019,16,-16)),
(to_sfixed(-0.001,16,-16),to_sfixed(0.025,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(-0.001,16,-16)),
(to_sfixed(-0.025,16,-16),to_sfixed(0.016,16,-16)),(to_sfixed(0.014,16,-16),to_sfixed(-0.014,16,-16)),
(to_sfixed(0.036,16,-16),to_sfixed(0.035,16,-16)),(to_sfixed(0.020,16,-16),to_sfixed(-0.007,16,-16)),
(to_sfixed(0.029,16,-16),to_sfixed(0.014,16,-16)),(to_sfixed(-0.004,16,-16),to_sfixed(-0.012,16,-16)),
(to_sfixed(0.011,16,-16),to_sfixed(-0.013,16,-16)),(to_sfixed(0.006,16,-16),to_sfixed(0.005,16,-16)),
(to_sfixed(0.015,16,-16),to_sfixed(0.002,16,-16)),(to_sfixed(-0.008,16,-16),to_sfixed(0.049,16,-16)),
(to_sfixed(0.048,16,-16),to_sfixed(0.053,16,-16)),(to_sfixed(-0.030,16,-16),to_sfixed(-0.019,16,-16))
);


preamble<=sp&sp&sp&sp&sp&sp&sp&sp&sp&sp&cp&lp&lp;--preamble structure is constructed by
concatenating


--adding noise to the preamble
add_noise:for i in 1 to 320 generate
preamble_w_noise(i)<=complex_sum(preamble(i),noise(i));
end generate;
preamble3<=preamble_w_noise&zeros1;


--process of short preamble and preamble with noise correlation
process(clk)
        variable m:integer range 1 to 16:=1;
        variable k:integer range 1 to 335:=1;
        variable corr:sfixed(16 downto -16):=to_sfixed(0,16,-16);
        variable sum:sfixed_vector(1 to 335):=(others=>(to_sfixed(0,16,-16)));
        variable a:integer range 0 to 334:=0;
```

```
begin
        if (rising_edge(clk)) then
                corr:=complex_mux(sp(m),preamble3(m+a));
                sum(k):=sum(k)+corr;
                if (m<16) then
                        m:=m+1;
                else
                        m:=1;
                        if(a<334) then
                                a:=a+1;
                        end if;
                        Cn(k)<=sum(k);
                        if (k<335) then
                                k:=k+1;
                        end if;
                end if;
        end if;
end process;

--normalization is done by dividing every elements of correlation vector by 2^4=16
normalization3: for i in 1 to 639 generate
        Cnn(i)<=Cn(i) srl 4;
end generate;

--process to represent every elements of the correlation vector one by one in every clock signal
process(temp_clk_out,rst)
begin
        if (rst='1') then
                count3<=1;
        elsif (rising_edge(clk)) then
                if (count3=335) then
                        count3<=1;
                else
```

```
                    count3<=count3+1;
               end if;
          end if;
end process;
C<=Cnn(count3);
end Behavioral;
```

6 References

1) Thesis of Signal detection and frame synchronization of m)ultiple wireless networking waveforms By Howland, Keith C.

2)Thesis of IEEE Standard for Information Technology— Telecommunications and information exchange between systems— Local and metropolitan area networks— Specific requirements by Stuart J. Kerry, Al Petrick, Harry R. Worstell,
3) Thesis of UPLINK AND DOWNLINK SYNCHRONIZATION OF 802.16   Presented to the Faculty of San Diego State University by Qing Wang Summer 2011
4)http://host.uniroma3.it/laboratori/sp4te/teaching/sp4bme/documents/LectureCorrelation.pdf
5) A Tutorial Introduction to VHDL Programming by Orhan GAZİ