این گزارش فنی به زبان **فارسی** به صورت یک فایل PDF با نام **Report_Persian** به همراه فایل‌های پروژه موجود است.

Project Name: FPGA-Based Pulse Detection & Frequency Estimation (Project page on GitHub)

Designer Name: Milad Zadrafee

Project Creation Date and Version: December 2025 – Version 1.0

Design Software and Language: Xilinx ISE 14.7 – VHDL

Contact Details:

https://www.linkedin.com/in/zadrafee
https://www.github.com/zadrafee

This document is part of a series of FPGA projects published on the LogLab website. To view other related projects and technical articles, visit my official website:

http://www.LogLab.ir

Description of the main project files and folders

Frequency_Estimation_MATLAB	ifm.slx	Simulation in Simulink
Frequency_Estimation_VHDL	IF_FFT.xise	Project file in ISE
	IF_FFT_Top.vhd	Top module
	IQ_Demodulator.vhd	Demodulator module
	Noise_Mean_Calculator.vhd	Noise averaging module
	IF_IFF_Top_tb.vhd	Test bench module
Report_Persian.pdf		Technical report in Persian
Report_English.pdf		Technical report in English

Abstract

The aim of this project is to estimate the frequency of an IF signal using FFT in the digital domain. The input IF signal has a center frequency of 160Mhz and a bandwidth of 40Mhz. This signal is directly sampled at a frequency of 128Mhz and a resolution of 14 bits and sent to the FPGA for further processing.

It is assumed that the input signal is in the form of pulses and the pulse width is at least 200ns and the signal to noise ratio is at least 15dB. This system must be able to detect the received pulses and then calculate their frequency using FFT. If the received signal is a continuous wave, it must calculate and announce the frequency value at specific time intervals.

Introduction

Modulation is a process in which the characteristics of a carrier signal, such as amplitude, frequency, or phase, are changed based on a message signal to enable the transmission of information through a communication channel. This process causes the message signal, which is usually of low frequency, to be transferred to a higher frequency band to be more resistant to noise and to be able to be transmitted over long distances or through different environments.

In telecommunications systems, the RF signal is the modulated carrier signal on which the Baseband information is mounted. More precisely, the RF signal is an electromagnetic signal that is in the radio wave band, usually from a few kilohertz to several gigahertz. This signal is used for wireless transmission of information and is designed so that it can be propagated in space, generated and received by an antenna, and its characteristics such as power, bandwidth, and spectrum can be controlled and processed.

The IF signal or "intermediate frequency" is a signal that is converted to a fixed intermediate frequency in a receiver or transmitter, after initial modulation or demodulation, to make its processing, filtering or amplification simpler and more stable. The use of intermediate frequencies improves the accuracy of filters, increases receiver sensitivity and simplifies the design of telecommunications systems.

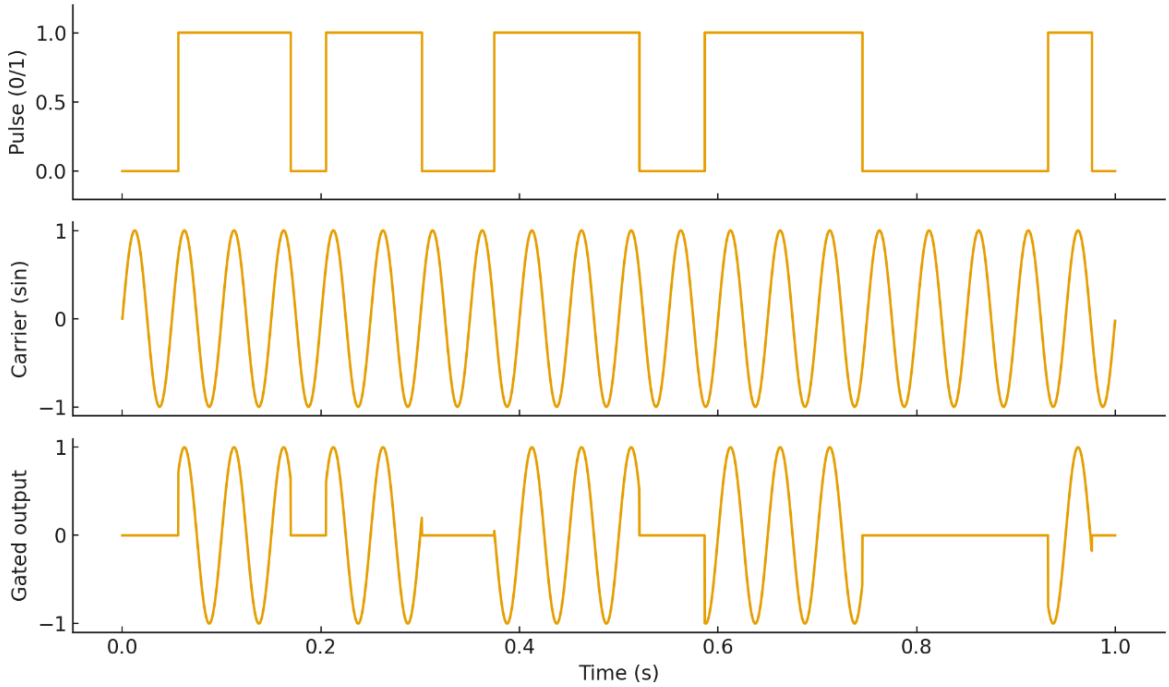
The baseband signal is the original information-carrying signal that is in the lowest possible frequency band and is produced before any modulation. This signal can include digital data, audio waves, or any other message signal. In receivers, after demodulation and carrier removal, the baseband signal is recovered for analysis, detection or final processing.

There are different levels of sampling in telecommunications systems. For example, in some systems, the RF signal is first analogically down-converted and converted to an intermediate IF signal, and then down-converted again to a baseband signal, and the baseband signal is sampled (converted to a digital sample using an A/D converter). However, in some telecommunication systems such as this project, the RF signal is

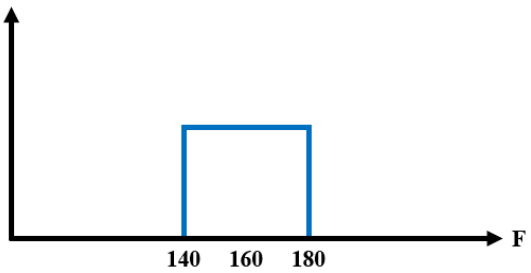
analogically down-converted to an IF signal and directly sampled from the IF signal. Such systems are called IF Sampling or IF Processing.

General System Description

In this project, the width of the received pulses is unknown. The only information we have about the width of the pulses is that it is more than 0.2µs. Note that the distance between the pulses is also unspecified.

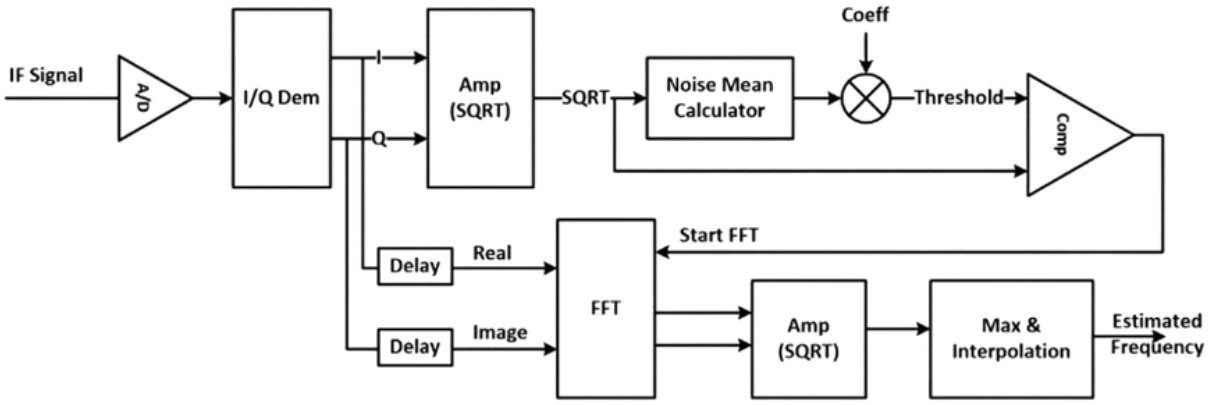


The IF signal that we receive for pulse detection is something like the third graph (Output), which is neither at the RF level nor at the Baseband level. In fact, our message signal is still riding on a carrier signal. Of course, this carrier signal does not have a high frequency at the RF level, but rather at the level of an intermediate signal or IF. In this project, our IF center frequency is 160Mhz, and our system bandwidth is 40Mhz. If we want to determine the spectrum of changes in the IF carrier signal, given that our IF center frequency is 160Mhz with a bandwidth of 40Mhz, the sinusoidal signal that we receive is something in the range of 140Mhz to 180Mhz.

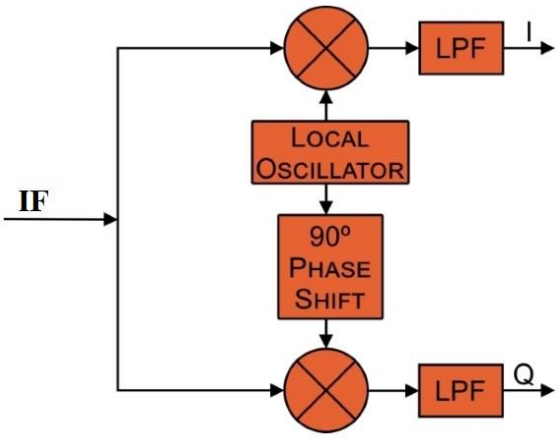


With the calculations performed at the system level, the sampling frequency and also the clock frequency of our circuit are 128Mhz. Since our IF signal is in the range of 140Mhz to 180Mhz and our sampling frequency is also 128Mhz, according to the Nyquist-Shannon sampling theorem, correct sampling is possible when the sampling frequency is greater than or equal to twice the largest frequency component of the original signal. Given that this condition does not exist, therefore, it is not possible to sample this signal. To solve this problem with the methods that will be explained below, we set the start of the IF signal frequency range, which is 160Mhz, to 0, of course, the bandwidth remains 40Mhz. With this trick, our IF frequency range is reduced to 0 to 40Mhz, now this range is compatible with the Nyquist-Shannon sampling theorem and correct sampling is possible. In fact, our sampling frequency is more than 3 times the largest frequency component (40Mhz), which is quite sufficient.

System Block Diagram



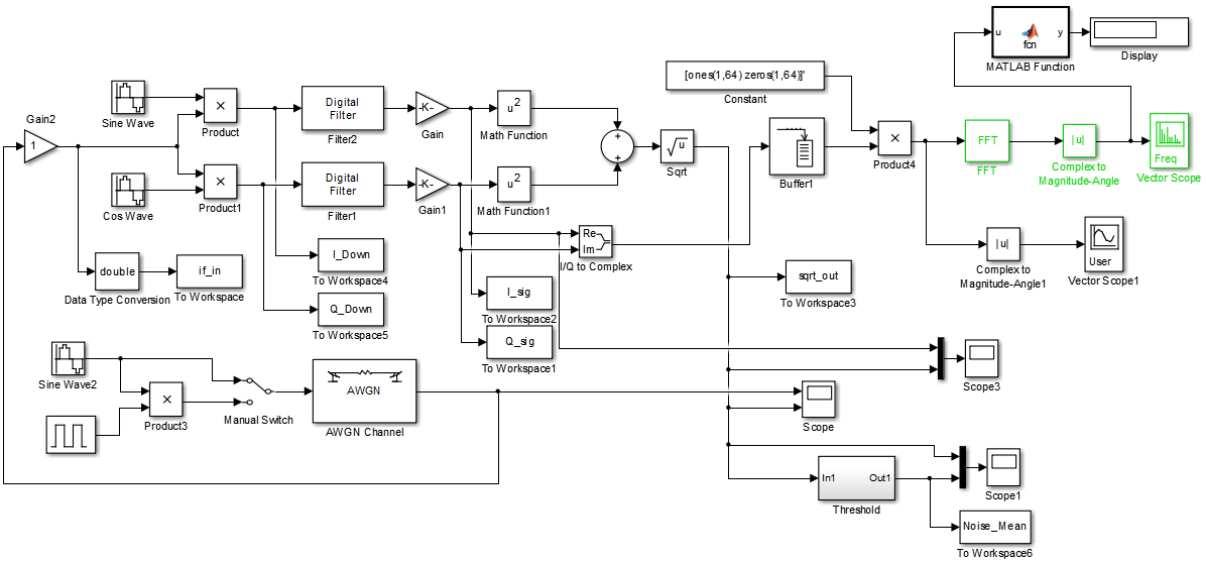
The upper part of the block diagram, which consists of the blocks A/D, I/Q Demodulator, SQRT, Noise Mean Calculator, Multiplier Coefficient, Threshold and Comparator, is responsible for pulse detection and recognition, and the lower part of the block is the frequency estimation section. The analog IF signal is converted into a discrete digital signal by passing through the A/D block. The I/Q Demodulator block takes the IF signal to the Baseband level.



In the I/Q Demodulator block diagram, there is an oscillator. This oscillator, whose frequency is 160Mhz, is built inside the FPGA and its frequency is the same as the IF signal frequency. The operation of this block is as follows: we multiply the input once by Sin 160Mhz and at the same time, shift (delay) the oscillator signal by 90 degrees, convert it to Cos 160Mhz and multiply it by the input. By doing this, our IF signal spectrum is shifted and is set to the value '0'.

According to the sampling theory, when we sample a signal, the spectrum of that signal is repeated in the frequency domain at certain intervals. Since we do not need more than one spectrum from these repetitions, we use 2 low-pass filter blocks at the output of the multipliers. The job of these low-pass filters is to keep only one signal spectrum from 0Hz to 40Mhz for us and filter (eliminate) the rest of the repetitions.

System simulation in Simulink, MATLAB software



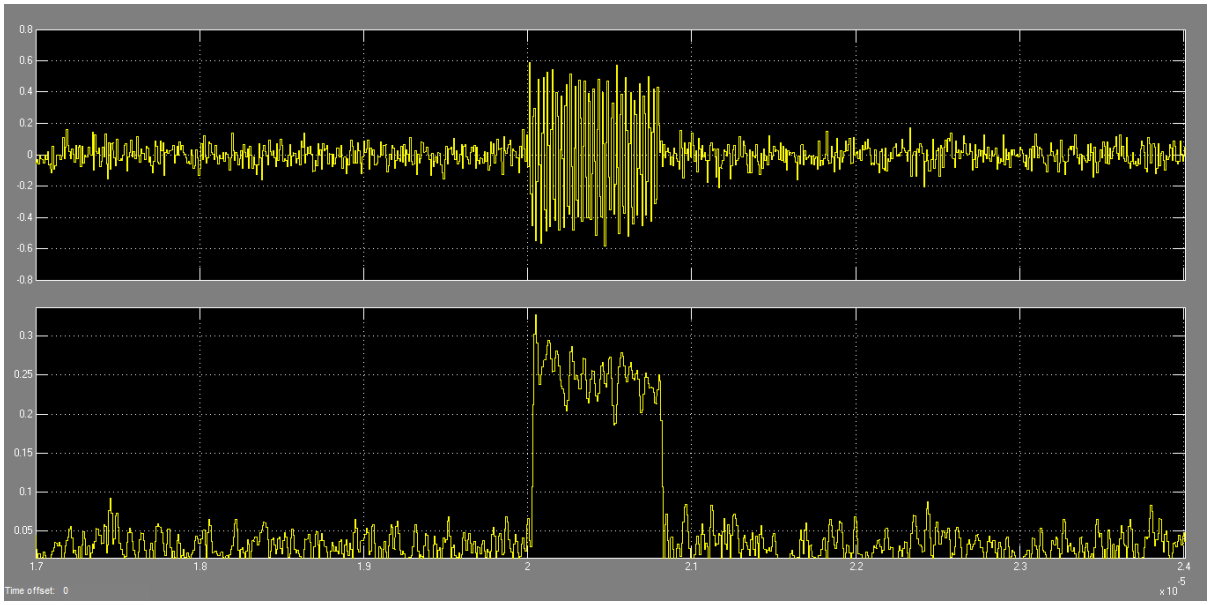
The Amp (SQRT) block calculates the signal amplitude for us through the following equation.

$$\text{Amplitude} = \sqrt{I^2 + Q^2}$$

In other words, the output of the SQRT block is the Baseband data or IF signal push that it extracts for us.

In the figure below you can see the IF signal along with the output of the I/Q Demodulator block:

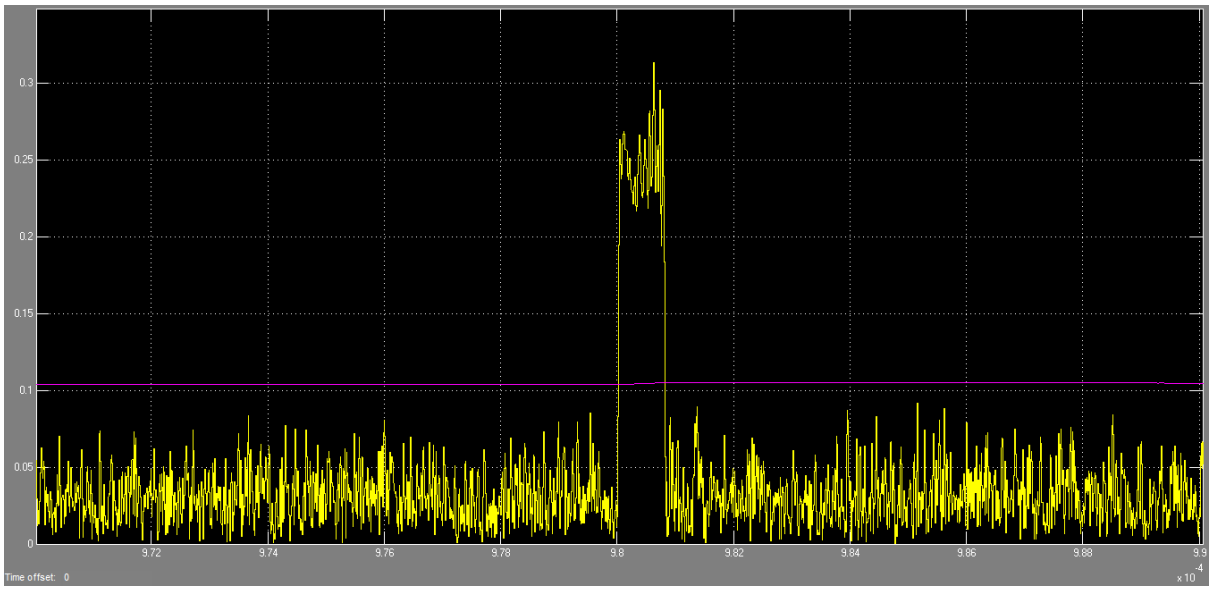
The top waveform is the IF signal and the bottom waveform is the Baseband signal.



To make our simulation closer to reality, we have used the AWGN block in Simulink after the main input that is the message signal. With the help of this block, we add some noise to our signal. (The AWGN block stands for Additive White Gaussian Noise, which means a white Gaussian noise that is equivalent to the noise that is added to the signal in the channel.)

So far, we have extracted the signal envelope. Now we want to identify the pulses from the envelope signal. For this purpose, we use a comparator. But the question that arises here is, what value should we compare the Baseband signal with? By carefully examining the Baseband signal waveform, the idea comes to our mind that by calculating the average noise and multiplying this value, we can define a reference called the Threshold signal to compare with the Baseband signal. If the Baseband signal value is greater than the Threshold value, it means that a pulse has entered.

But how to calculate the average noise? With the help of a low-pass filter, a low-pass filter can actually be used as an integrator or an averager in signal processing. The basis of the Noise Mean Calculator block is the low-pass filter mentioned here.



In the image above, the yellow waveform shows the Baseband signal and the purple waveform shows the Threshold signal. As you can see, the Threshold signal is above the noise and below the pulse.

Frequency Detection Section

So far, we have examined the pulse detection section. From this section onwards, we will discuss how to detect the pulse frequency. Note that our assumption in the frequency detection section is that our pulse has only one frequency. The main basis of frequency detection in this system is the FFT block.

The FFT block shows us the strength of the frequency components in the input signal. As we know, the detected pulse is mounted on an IF signal that has a frequency between 140Mhz and 180Mhz at the time of detection. By taking the FFT from the pulse signal, we should see a large amplitude at that specific frequency on which the Baseband signal is mounted and at other points, we should see an amplitude close to zero. In fact, the peak from the FFT output indicates the frequency of the input signal, or the pulse.

To obtain the peak of the input signal, we must first calculate the amplitude of the FFT outputs using the following formula. (R and I are the real and imaginary parts of the FFT output, respectively.)

$$\text{Amplitude} = \sqrt{R^2 + I^2}$$

Given that our sampling frequency is 128Mhz and we use a 128-point FFT, our sampling resolution and accuracy is 1Mhz.

$$Resolution = \frac{FS}{P} = \frac{128Mhz}{128} = 1Mhz$$

The fact that our resolution is 1Mhz means that if our pulse frequency is suppose to be 10.2Mhz, by maximizing it will give us either a value of 10Mhz or 11Mhz. To solve this problem and get the exact value of the pulse frequency, we can use a computational method called parabolic interpolation. In signal processing, parabolic interpolation is a standard technique for accurately estimating the position of a peak in the frequency domain. In this method, a parabola is constructed from the three maximum values and their two neighbors in the FFT, and its vertex is estimated as the actual frequency of the peak.

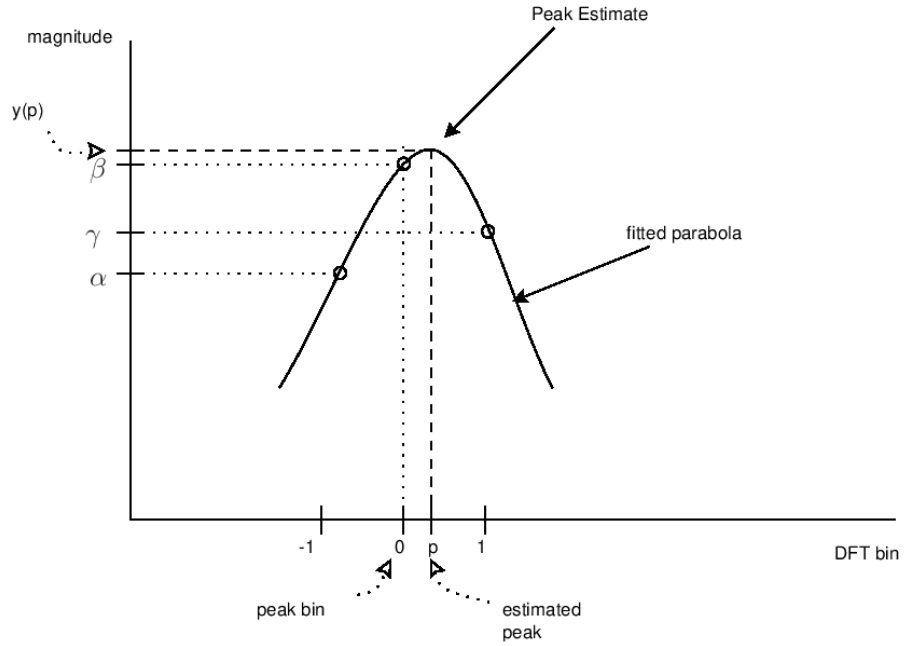


Illustration of parabolic peak interpolation using the three samples nearest the peak.

In the field of signal processing, the above method is called Quadratic Interpolation of Spectral Peaks.

We estimate the exact value of the amplitude through the following formula.

$$P = \frac{a - c}{2 * (a - 2 * b + c)}$$

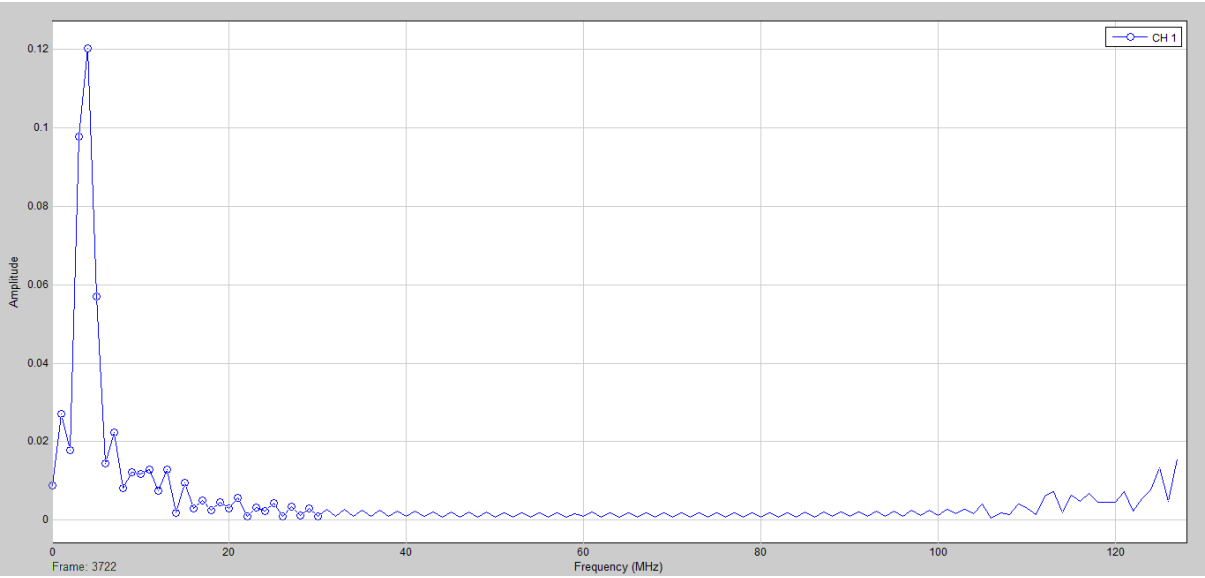
In this formula, the value of b is equal to the maximum amplitude of the FFT output, the value of a is equal to the amplitude of the left side of b, and the value of c is equal to the amplitude of the right side of b. The interpolated frequency value is equal to:

$$F_{Estimated} = (k + P) * \frac{f_s}{N}$$

In the above formula, k is the index corresponding to the maximum range, or in other words, the index b.

To learn more about this method, you can refer [to this page on the Stanford University website.]

The image below is the output domain of the FFT:



Review of VHDL code

In this project, the FPGA input is a 14-bit signal, which is the 14-bit output from the A/D (digitized IF signal) and has the main module IF_FFT_Top, the IQ_Demodulator module, the Noise_Mean_Calculator module, and the Low_Pass_Filter module.

The main module IF_FFT_Top

This module is the core of the system and is responsible for coordinating the steps of pulse detection, amplitude extraction, FFT execution, and accurate frequency estimation. Its most important functional parts are:

Demodulation and extraction of the Baseband signal

In this part, the input IF signal is first converted into I and Q components through the IQ_Demodulator module. Then, using IP SQRT, which is actually the same as IP Cordic, its instantaneous amplitude is calculated by the relationship √(I^2+Q^2), and thus the Baseband signal is created.

Calculate the noise mean and generate the pulse detection threshold

The output of the Baseband domain is fed into the Noise_Mean_Calculator module and the noise mean is extracted. Then the Threshold value is set to a multiple of the noise mean to detect the moment of pulse arrival.

Detect the start and end of the pulse

To compare the Baseband domain and the Threshold, two independent counters are used to detect the start of the pulse and the end of the pulse. Only after the signal is stable above the Threshold, a valid pulse is detected and FFT_Start is activated.

Delay buffer for I/Q alignment

Since the pulse detection process inevitably introduces a delay, two 31-stage arrays are used to create a delay on the I and Q signals so that the start of the FFT is exactly at the same time as the correct samples.

Calculate the FFT output amplitude

After performing the 128-point FFT, the real and imaginary parts of the output are converted to amplitude via the FFT_SQRT IP. The FFT_Data_Valid signal acts as a rhythmic sampling of the FFT output.

The FFT_128p IP is implemented as an Unscaled implementation with a length specification of 128 points, a Pipelined architecture, and inputs of 14 bits and a Phase Factor of 16 bits.

Spectrum Peak Search

When the FFT is ready (FFT_Amp_Ready), the amplitudes are checked to store three values A, B, C as left, peak, and right, respectively. These values are the input to the Quadratic Interpolation algorithm.

$$P = \frac{a - c}{2 * (a - 2 * b + c)}$$

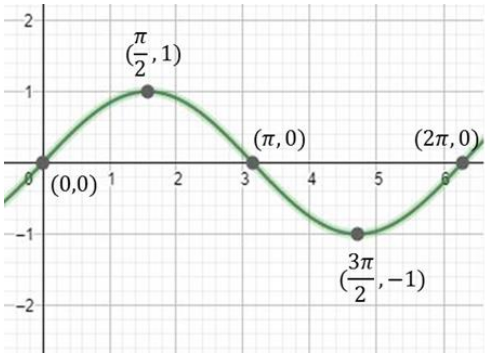
Output for ILA

Finally, the Peak Index and Bin Offset are placed together for viewing in the ILA so that the frequency estimation process can be debugged.

Note about the IQ_Demodulator module: If we want to create a 32Mhz sinusoidal signal with a sampling frequency of 128Mhz, we only need 4 points.

$$\frac{128\text{Mhz}}{32\text{Mhz}} = 4$$

The advantage of this method is that we no longer need to use IP DDS to generate a sinusoidal signal, and more importantly, we don't even need a multiplier because the generated sinusoidal signal only has the values 0, 1, 0, and -1. More clearly, where the sine value is '0', the input signal also becomes '0', and the parts where it is '1' become the input signal itself, and where it is '-1' become the opposite of the input signal.



Low-pass filters in IQ_Demodulator

After multiplication, the raw I and Q components have additional frequency components at 160MHz ± Baseband. So each path has a 5-Tap hybrid IIR/FIR filter that only preserves the 0 to 40MHz band.

Note: A tap is each unit of a digital filter (here FIR) that consists of a delay sample and a corresponding coefficient. A 5-tap filter means a filter with five coefficients and five delay stages.

Noise_Mean_Calculator module

The basis of this module is a very slow-response low-pass filter used to estimate the noise mean.

This block is a large Accumulator with a "Remainder Feedback" feature that produces a useful moving average of length N^2. The length of this filter in the project is Averaging_Length = 15, i.e. 32768 samples.

To prevent the real pulse from affecting the average noise value, the Input_Signal_Int_Saturate signal checks the input value and saturates it if it is greater than 3× the noise value.

The presence of extra bits (Extra_Bits = 8) allows for maintaining accuracy in calculating the average and prevents quantization errors.

Note: The method mentioned above for calculating the average noise and creating a reference called the Threshold signal for comparison is a completely scientific and standard method. However, this method is less used in real projects. In practice, by measuring the noise with the help of advanced and very accurate equipment, the actual average noise value is measured, then it is multiplied according to the noise range (in this project, 3 times) and we consider it as the Threshold signal. Why don't we implement the theoretical method in books in practice?

Because in real systems noise is constantly changing:

Change with temperature

Change with ADC load

Change with RF signal level

Ambient noise, EMI, digital switching

Channel-based noise, not white noise

In reality:

Noise is not zero average

We have DC offset.

We have leakage from LO.

We have Spur, Harmonic, Quantization Noise ADC.

We have drift in LNA/IF Gain.

AC coupling creates its own Offset.

As a result, "average noise" is not exactly defined and leads to erroneous measurements.

Although in theory determining the threshold based on the average noise is a scientific and standard method, in practice, due to the non-stationary nature of noise, the presence of hardware offsets and spurious, environmental changes, and the instability of real noise, a fixed Threshold value is usually used. The threshold value is initially determined by experimental measurement and then kept constant in the system to ensure stable and reliable performance.

Low-Pass Filter – I and Q Path

This filter is a hybrid IIR/FIR structure that is both fast and stable.

Filter structure

The FIR section consists of 5 multipliers with symmetric coefficients b[n], the IIR section consists of feedbacks with coefficients a[n], the coefficients are implemented as 18-bit S.2.15, which is suitable for processing decimal numbers.

High accuracy and overflow prevention

All multiplications are performed with a width of 41 bits and then applied to the Saturation output to prevent exceeding the numerical range. The final output is 14 bits.

The task of this filter is to remove the components:

160MHz + Baseband

160MHz – Baseband

and other harmonics and only pass the range from 0 to 40MHz.

Bin_Offset_Calculator Module

This module is a hardware Divider that calculates the "Offset" part of Quadratic Interpolation.

Inputs:
Dividend = (a – c)
Divisor = (a – 2b + c)
Output:
Fractional = P value (offset between two FFT bins)

This value together with FFT_Max_Index forms the final frequency value.