



XeThru X4 Radar User Guide

UWB Basic Principles and X4 Operation

XeThru Application Note **by Novelda AS**

Rev. A - September 25. 2018

Summary

The XeThru X4 SoC is a complete UWB radar system. This application note takes the reader through basic principles and gives practical examples on configuration and use of the system.





1 Introduction

The XeThru X4 is a compact, Impulse-Radio Ultra-Wideband (IR-UWB) radar system on a chip. It is configurable and gives the developer a high degree of freedom to develop new applications from basic presence detection to advanced imaging-solutions.

This application notes goes through basic radar principles and explains how to configure important chip settings. All chip communication is handed by X4Driver, a portable open-source module written in C. It's a part of the XeThru Embedded Platform (XEP) [1] and it's highly recommended to use it for any application that will communicate directly with the X4 chip to simplify development. The curious developer can check the source code to see exactly what happens in each method (`\xtXEP_source\xtXEP\src\XDriver\x4driver.c`). Module Connector [2] is a host computer library that provides access to XeThru modules and X4Driver, including functionality for streaming and storing radar data with MATLAB, Python and C++ interfaces.



2 The Basics of X4 Impulse Radar

XeThru X4 is a complete IR-UWB radar system on chip. To configure it correctly, it's important to have a good understanding of how an impulse radar system works and how the received data is sampled and presented. The fundamentals of an impulse radar system are shown in Fig. 1. The radar sends out an electromagnetic impulse through the Tx antenna which is reflected from any object in front of it. The reflections travel back and are received and sampled through the Rx antenna.

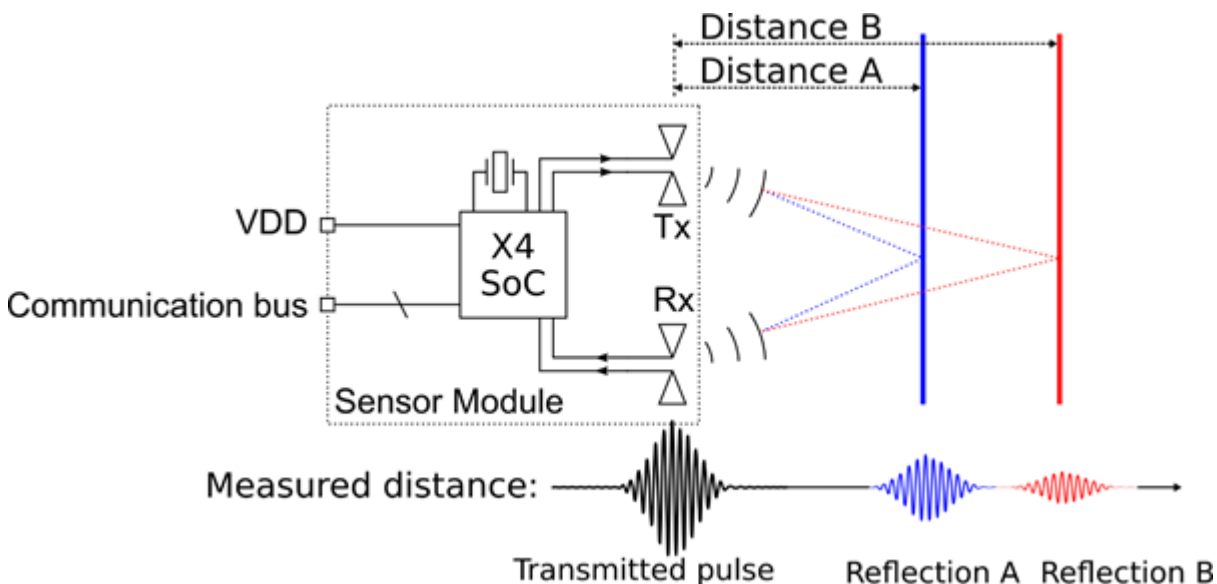


Fig. 1: Basic UWB radar concept.

The pulse that X4 transmits is configurable within two different bands, supporting world-wide regulations. The lower pulse generator setting enables transmission within the band 6-8.5 GHz, shown in Fig. 2, and the high settings within the band 7.25-10.2 GHz. To receive the reflected energy, it uses a high-speed sampler with a sampling rate of 23.328 GS/s that can sample up to 1536 samples. Since electromagnetic waves travel by the speed of light, this corresponds to sampling of reflected pulses in a window of about 9.9 meters long. Each sampling point is referred to as a range bin.



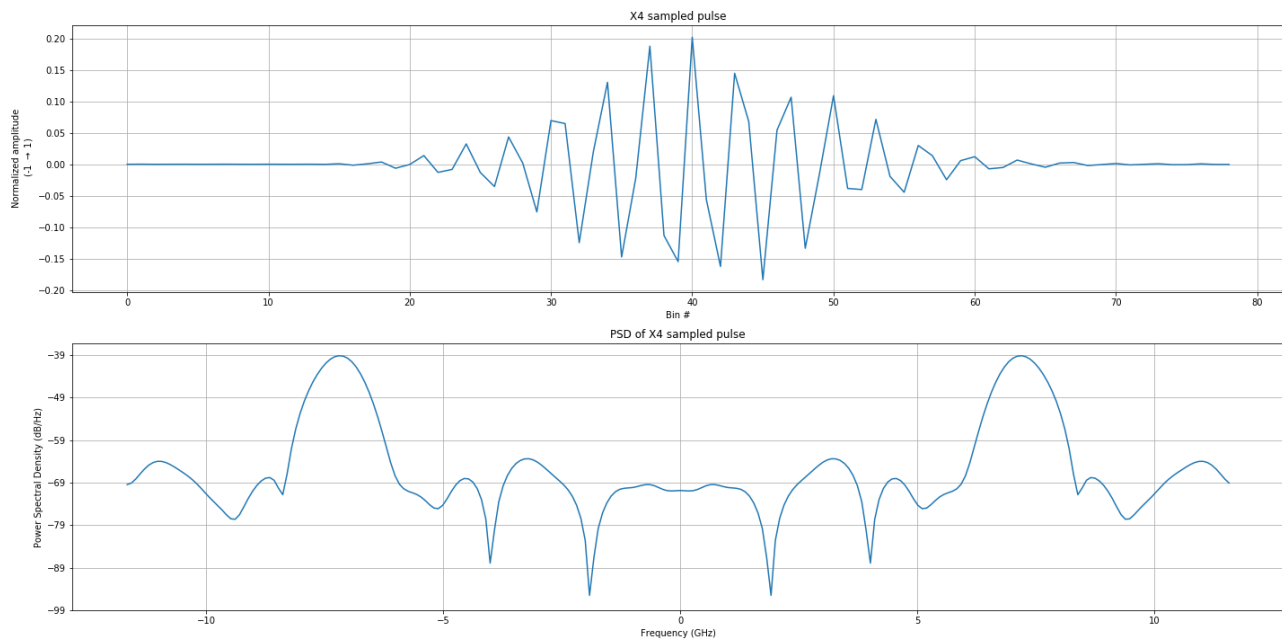


Fig. 2: The pulse as sent and sampled with X4.

2.1 Radar Frames

Fig. 3 shows the result from sampling data over the 1536 bins, referred to as a radar frame. By starting the sampler right after transmitting a pulse, an object 2 meters away shows up on the frame as a pulse around range bin 280. The energy seen from bin 0 is caused by the energy transmitted directly from Tx to Rx antenna, namely the **direct path**.

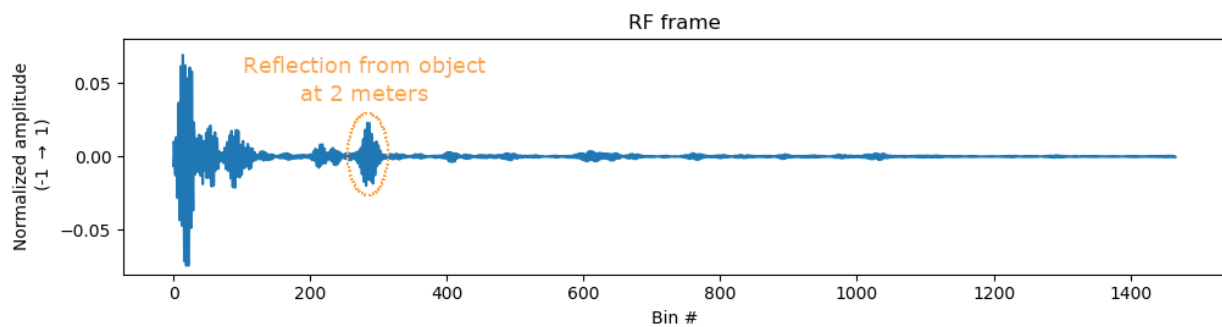



Fig. 3: A radar frame.

The range from the radar to the object is known as slant range and is calculated by

$$(1) \quad \text{Range} = \frac{c \times \tau}{2}$$

where c is the speed of light, τ is the measured delay to the reflection and the divisor of 2 is caused by the radar signal has to travel to the target and back, twice the distance.



With X4, you can choose to read out RF data or  enable on-chip digital downconversion to read baseband data. Downconversion shifts the frequency content down, filters out of band energy and decimates the frame with a factor of 8. The result is a frame with less noise represented by less data, see section [Downconverting to Baseband Data](#). Fig. 4 shows the amplitude of the downconverted data and the reflection from the object at 2 meters can now be seen around bin 35.

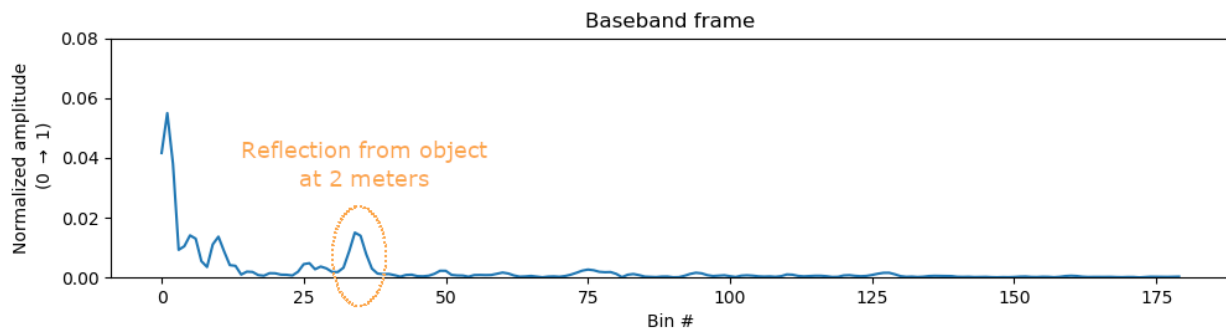


Fig. 4: The amplitude of a downconverted radar frame.

- ✔ The best way to get a good understanding of the radar frame is to experience it firsthand. You can use any XeThru module on a computer with XeThru Explorer [3] to look at the raw data and see for yourself how the pulses of a radar frame move as you move in front of the radar. If you are using X4M03, you can choose between raw (RF) or downconverted (baseband) data. If you are using X4M200 or X4M300, the data you see will always be downconverted.

2.1.1 Object Reflections

The radar captures the reflections of every object in its field of view, and the distance to the object corresponds to the position of the reflection in the radar frame. Larger objects give reflections with higher amplitudes and the amplitude of the reflection is reduced the further away the object is. The object material also plays a role as different materials give different reflections, for example a metal object gives a larger reflection than a plastic object. The amount of energy reflected to the radar is referred to as the object's radar cross section (RCS).

2.1.2 Multipath

In addition to the original reflection from an object, there are also multipath reflections for an object at distances further away than the object. This is because the reflection from the object might also bounce, for example, in the roof and back down to the radar. Since this reflection travels a longer path, it will appear at a distance further away than the original object reflection and with a smaller amplitude.

2.1.3 Direct Path

The strong pulse starting at bin 0 in Fig. 3 and 4 is caused by energy going directly from the transmitting (Tx) antenna to the receiving (Rx) antenna, referred to as the direct path. The amount of energy in the direct path is determined by the antenna isolation and will differ from different modules and antenna setups.



3 Configuring X4

Configuring the transceiver system is about finding the best trade-offs between SNR, frame size and speed for the desired application. In an application such as respiration detection, good signal-to-noise ratio might be more important than high detection speed. In an application such as tracking a person walking in a room, it might be more important to have a high frame rate to ensure good tracking.

3.1 Initializing X4

Initializing X4 includes uploading firmware, configuring system clocks, enabling the sampler and more. The X4Driver has a method for doing everything necessary and configuring the system with default values, the method is called `x4driver_init()`. This method should be called as the first step before configuring the system.

3.2 Configuring the Sampler System

The X4 uses a *swept-threshold* sampling method. The input signal is compared to a threshold, and the resulting 1-bit value is summed at each range bin to incrementally build the multi-bit frame. The threshold is kept constant while receiving echoes for a given pulse and then stepped in-between radar pulse transmissions, sweeping over the range of the DAC steps. This means that one radar frame is built from many transmitted pulses. For an in-depth look at the XeThru X4 radar system, see this paper [4].

3.2.1 Pulse Repetition Frequency

The Pulse Repetition Frequency (PRF) configures how often a pulse is transmitted. The PRF will be a division of the Common PLL output. In normal operation, the Common PLL should be configured to output a 243 MHz clock signal. The method `x4driver_set_prf_div(...)` sets the PRF by dividing the Common PLL output by the given argument.

The default value is 16 which gives a PRF of $143 \text{ MHz} / 16 = 15.1875 \text{ MHz}$. Maximum PRF is 40.5 MHz.

Warning: Regulatory Compliance

By increasing the PRF, the emitted RF signal power level will be increased and could break regulatory compliance. See X4M200/X4M300/X4M03 datasheets for more information.

Considerations when changing PRF

1. The frame length must be shorter than $1/\text{PRF}$.
2. Avoid sampling the previous pulse while transmitting the next.

The direct path generates noise and transmitting the next pulse while sampling the previous is not recommended. Substituting τ with a Pulse Repetition Interval (PRI) of $1/15.1875 \text{ MHz}$ in eq. (1), shows that each new pulse is then transmitted at an equivalent distance of close to 9.9 meters from the radar. This PRF has to be changed to avoid noise when sampling data at ranges above 9.9 meters, the obvious way of doing this is to reduce the PRF.



3.2.2 DAC Range

An 11-bit DAC controls the sampler quantization threshold, giving a DAC range from 0 to 2047. The DAC sweep range is controlled with the methods `x4driver_set_dac_max(...)` and `x4driver_set_dac_min(...)`. Setting a small sweep range results in faster sweeps, but might also result in losing part of the signal if set too narrow. In practical applications, the required dynamic range can vary significantly depending on the sensor location, the distance to the target, and the presence of large unwanted reflectors close to the targets.

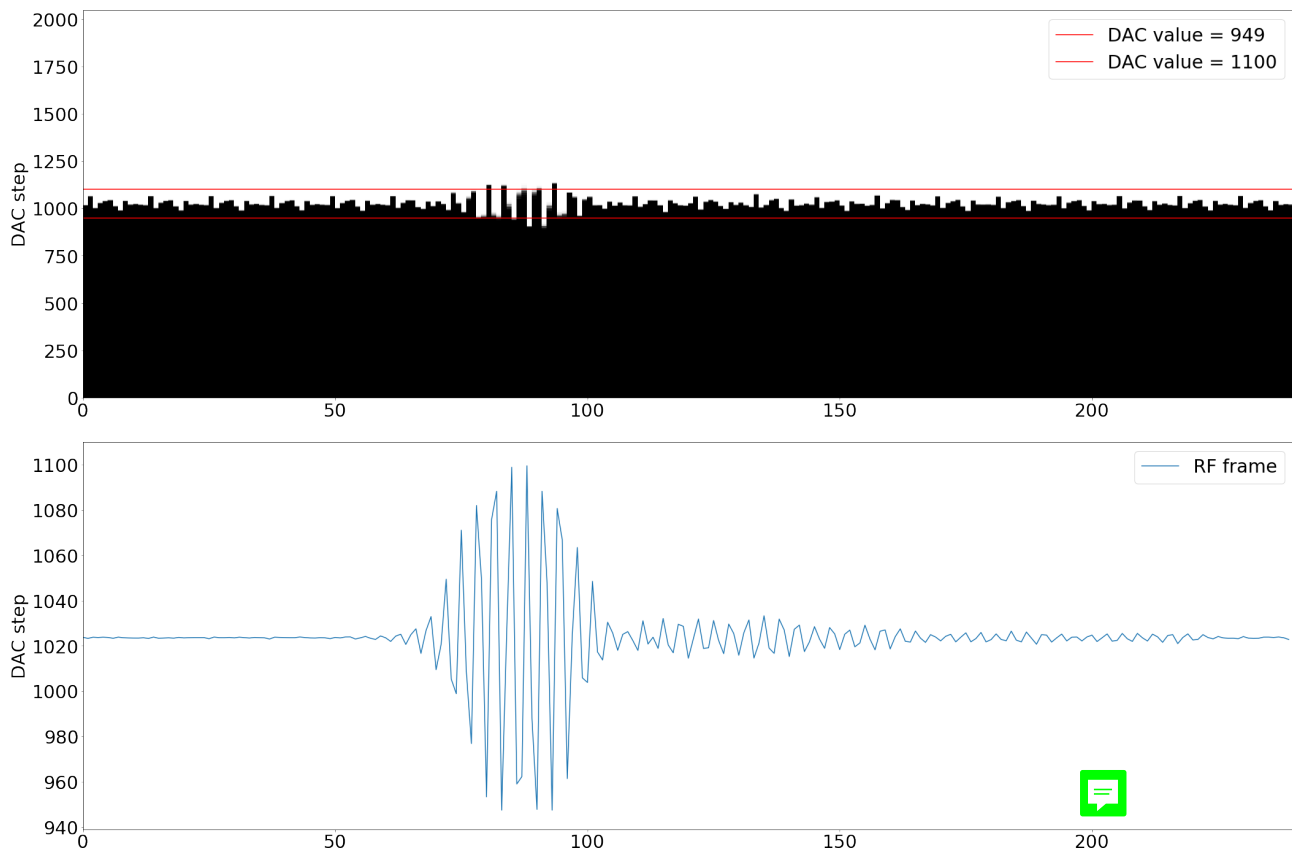


Fig 5. DAC sweep range.

The top plot of Fig. 5 shows the result from reading out the sampler values for each of the DAC steps from 0 to 2047 from an X4M02 module. The red lines mark the narrowest recommended DAC range, `dac_min = 949` and `dac_max = 1100`. The pulse seen is the direct path and some of the amplitude information is lost by using the narrow DAC range. A wider DAC range is needed if objects are expected close to the sensor, within the length of the direct path and amplitude information is needed. A narrow DAC range allows for maximum on-chip integration. The lower plot shows the result of summing each of the bins in the upper plot, this is how X4 accumulates each sample in normal operation.

3.2.3 Chip Integration

Chip integration is done by repeating pulse transmission and the DAC sweep. Every doubling of integration reduces white noise by 3 dB, improving the SNR. Integration can be repeated under the assumption that the reflecting objects does not move during the sweep. In addition to the DAC sweep itself, there are two settings controlling chip integration, `pulses_per_step` and `iterations`.



Pulses Per Step

For every DAC step the threshold is kept constant while receiving echoes for a given pulse. `pulses_per_steps` controls how many pulses to send per DAC step. Increasing this number will improve SNR, but also reduce the time it takes to create a frame and thereby reduce the maximum Frames Per Second (FPS). Valid values are 1-65535.

Iterations

It is also possible to configure how many times the radar should repeat the DAC sweep and is controlled by the setting `iterations`. Valid values are 1-255 and should be divisible by 4. It is recommended to set iterations to 64 and adjust `pulses_per_step` according to the desired Frames Per Second (FPS).

3.3 Calculating Maximum Frames Per Second

The maximum number of frames per second (FPS) is estimated by the formula:

$$(2) \quad \text{FPS} = \frac{\text{PRF}}{\text{iterations} * \text{pulses_per_step} * (\text{dac_max} - \text{dac_min} + 1)} * D$$

where D denotes the X4 duty cycle. See section *Processing Gain and Sweep Time* in the X4 datasheet [5] for more information. Also note that there might be some overhead in the system reading out data from the chip that reduces the maximum FPS, a maximum duty cycle of 95% is usually sufficient. Testing on the target platform is recommended.

When setting a desired FPS, the actual FPS can be read with the function `x4driver_get_fps()`:

Python example:

```
In [19]: xep.x4driver_set_fps(17)
In [20]: xep.x4driver_get_fps()
Out[20]: 17.049379348754883
```

3.3.1 Example 1: Configure X4 with FPS of 17

- PRF: 15.1875 MHz
- X4_duty_cycle: 95%
- dac_max: 1100
- dac_min: 949
- iterations: 64
- FPS: 17

By solving the FPS equations above for `pulses_per_step`, it becomes:

$$(3) \quad \text{pulses_per_step} = \frac{\text{PRF}}{\text{iterations} * \text{FPS} * (\text{dac_max} - \text{dac_min} + 1)} * D = \left\lfloor \frac{15.1875 \text{ MHz}}{64 * 17 * 150} * 0.95 \right\rfloor = 87$$

3.3.2 Example 2: Configure X4 with FPS of 255

- PRF: 15.1875 MHz
- X4_duty_cycle: 95%



- dac_max: 1100
- dac_min: 949
- iterations: 64
- FPS: 255

By solving the FPS equations above for pulses_per_step, it becomes:

$$(4) \quad \text{pulses_per_step} = \lfloor \frac{15.1875\text{MHz}}{64 * 255 * 150} * 0.95 \rfloor = 5$$

3.4 Setting the Frame Area

In many applications it is not necessary, or even not desired, to look at the whole radar frame of 9.9 meters. In these cases, it's possible to reduce the amount of data to handle by setting a smaller frame area. The method `x4driver_set_frame_area(...)` takes two arguments to set the start and the end of the frame in meters. Use `x4driver_get_frame_area(...)` to read actual settings.

Before setting the frame area, a 0-meter reference must be set. This reference is hardware dependent. X4M02 and X4M03 has an offset of 0.18 meters. Use the method `x4driver_set_frame_area_offset(...)` to set the offset reference.

4 Sampling a Radar Frame

Sampling of radar frames is started by setting the desired FPS using the function `x4driver_set_fps(...)` method. The maximum allowed FPS is given by the configurations explained in section [Configuring the X4](#). If FPS is set too high, the behavior is undefined. The actual FPS is read back with `x4driver_get_fps()`. By using Module Connector [2], the method `peek_message_data_float()` is used to check if any frames have been received and the method `read_message_data_float()` reads out a frame from the frame FIFO.



5 Downconverting to Baseband Data

X4 has a built-in downconversion and decimation feature that shifts the sampled signal to DC, filters out of band energy and decimates the result by a factor of 8 in order to reduce the data rate without losing any information. 1536 bins become 188 complex bins after downconversion. Downconversion can be enabled with the method `x4driver_set_downconversion(...)`.



Quadrature signals

A good and comprehensive introduction to quadrature signals can be found here: http://www.ieee.li/pdf/essay/quadrature_signals.pdf. It covers continuous signals, but the principles are the same.



5.1 Downconversion

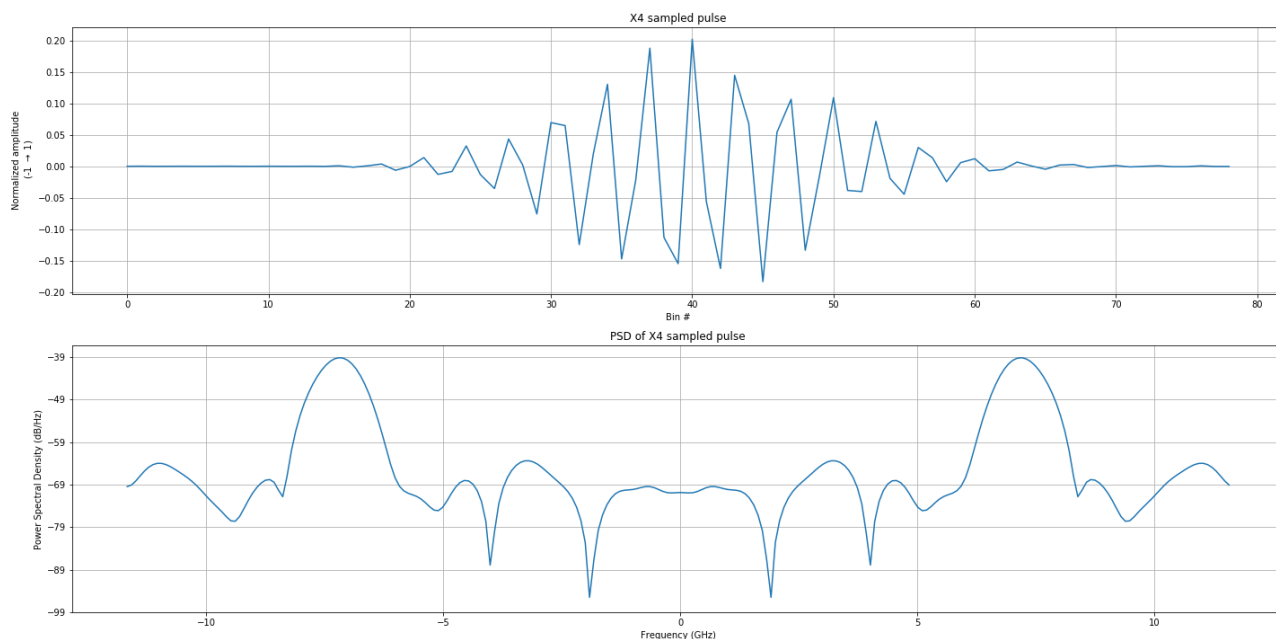


Fig. 6: An RF pulse with its frequency spectrum.

In Fig. 6 the raw RF pulse is shown with its corresponding frequency spectrum. The downconversion process can be done off-chip with the following steps:

1. Sample and read RF frame from X4
2. Multiply frame with a complex sine with frequency equal to the transmitted pulse center frequency
3. Low pass the result
4. Decimate

Given a radar frame, `rf_frame`, the downconversion is done by:

Python example:

```
import numpy as np

fc = 7.29e9 # Lower pulse generator setting
fs = 23.328e9 # X4 sampling rate
csine = np.exp(-1j*fc/fs*2*np.pi*np.arange(len(rf_frame)))
cframe = rf_frame * csine
```

The result is a complex frame where the pulse frequencies are centered at the sum and difference frequencies. Then a lowpass filter pass the difference and rejects the sum frequencies. Fig. 7 shows the spectrum of the different intermediate steps in the downconversion process. A 26 taps low pass hamming filter was used for the low pass filtering.

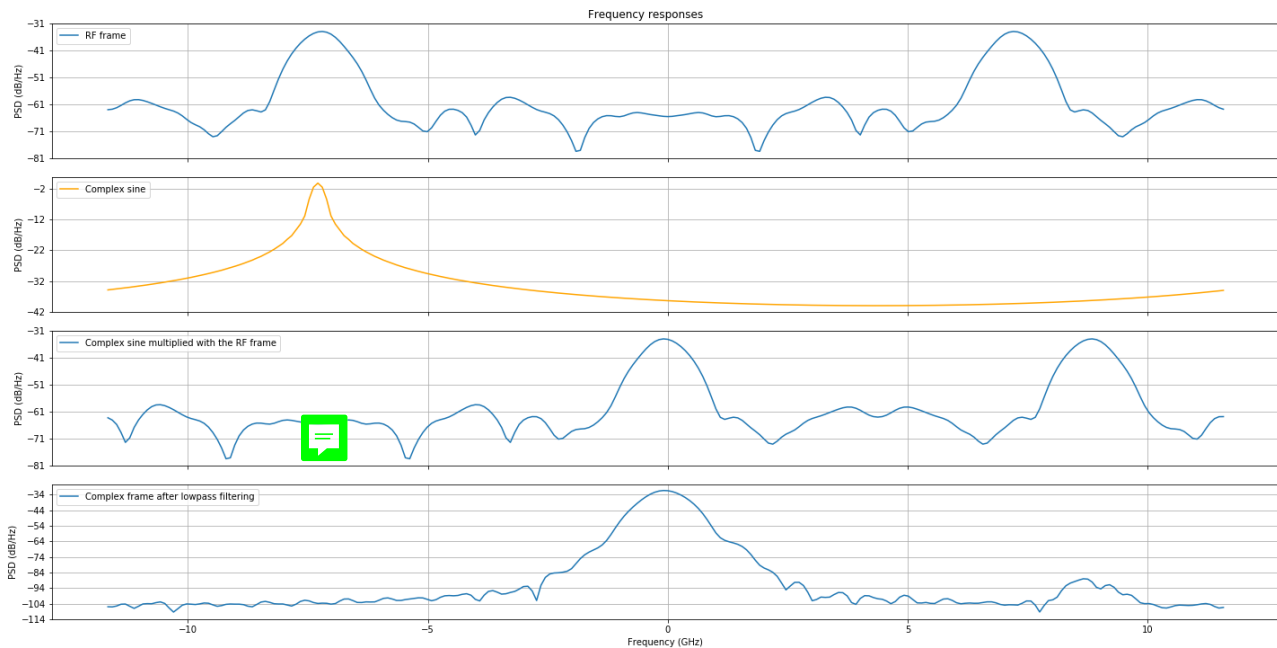


Fig. 7: Overview of the downconversion process.

The baseband frame amplitude is calculated from the absolute value of the complex baseband frame and is shown in Fig. 8 together with the raw RF frame.

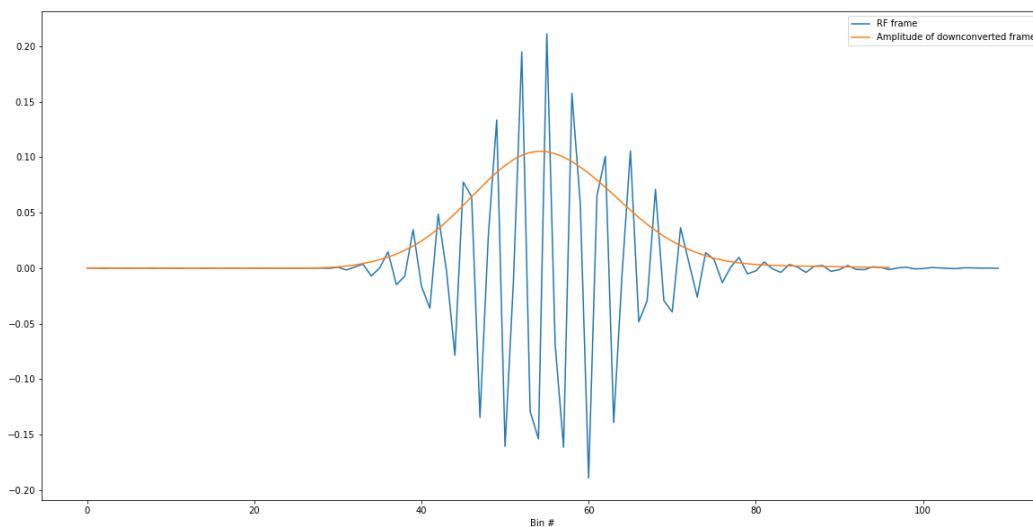


Fig. 8: The RF pulse and the amplitude of the baseband pulse.

The downconverted baseband signal can be represented at a lower sampling rate and the X4 chip uses a fixed factor of 8. The baseband frame is mathematically equivalent to the original frame.



6 References

[1]	Novelda, "XeThru Embedded Platform (XEP)", https://www.xethru.com/community/resources/xep-source.90/
[2]	Novelda, "Module Connector", https://www.xethru.com/community/resources/categories/software.31/
[3]	Novelda, "XeThru Explorer", https://www.xethru.com/community/resources/categories/xethru-explorer.3/
[4]	N. Andersen <i>et al.</i> , "A 118-mW Pulse-Based Radar SoC in 55-nm CMOS for Non-Contact Human Vital Signs Detection," in <i>IEEE Journal of Solid-State Circuits</i> , vol. 52, no. 12, pp. 3421-3433, Dec. 2017, https://ieeexplore.ieee.org/document/8106658/
[5]	Novelda, "X4 Datasheet", https://www.xethru.com/community/resources/x4-datasheet.106/

7 Document History

Rev.	Release date	Change description
A	2018-Sept-25	Initial release.



8 Disclaimer

The information in this document is provided in connection with Novelda products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Novelda products. EXCEPT AS SET FORTH IN THE NOVELDA TERMS AND CONDITIONS OF SALES LOCATED ON THE NOVELDA WEBSITE, NOVELDA ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL NOVELDA BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF NOVELDA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Novelda makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Novelda does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Novelda products are not suitable for, and shall not be used in, automotive applications. Novelda products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.