Master's Thesis

Software Defined Radio based MIMO channel sounding

Jens Elofsson
Peter Seimar

Department of Electrical and Information Technology,
Faculty of Engineering, LTH, Lund University, 2016.

Software Defined Radio based MIMO channel sounding

Jens Elofsson
adi09jel@student.lu.se
Peter Seimar
adi09pse@student.lu.se

Department of Electrical and Information Technology
Lund University

Advisor: Fredrik Tufvesson
Joao Vieira

May 4, 2016

Printed in Sweden
E-huset, Lund, 2016

Abstract

With todays increased use of smartphones and other wireless devices, the demands on the wireless networks have increased dramatically. This has prompted the research and development of new types of wireless systems using multiple antennas at both the transmitter and the receiver, known as MIMO. Since wireless systems are very susceptible to interference and to changes in the environment, it is important to know how the wireless channel behaves in the environment that the wireless system is intended to operate in. This is done by sounding the channel, e.g. where a signal is transmitted from the transmitter and processed on the receiver in order to obtain the impulse response of the channel. When performing channel sounding on a MIMO system it is important that the signals transmitted from the antennas interfere with each other as little as possible so as to get a good estimation for how the channel behaves. In this thesis we implement a 2 by 2 MIMO channel sounder using m-sequences and Zadoff Chu-sequences, both with good autocorrelation properties. The major part of the implementation is done on an FPGA in order to be able to perform the sounding in real time.

Acknowledgements

We would like to express our our gratitude to our supervisors Fredrik Tufvesson and Joao Vieira for their support and guidance during our thesis work and for making this thesis possible. We would also like to show our appreciation to Bertil Lindvall and Josef Wajnblom for their help with installing software and hardware needed for our work.

Table of Contents

Abbreviations

Abbreviations

ACF	Auto-Correlation Function
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
FPGA	Field-Programmable Gate Array
GF	Galois field
HDL	Hardware Description Language
IC	Integrated Circuit
IO	Interacting Object
LFSR	Linear Feedback Shift Register
LOS	Line Of Sight
LUT	LookUp Table
MIMO	Multiple Input Multiple Output
MPC	Multi Path Component
NI	National Instruments
SCTL	Single Cycle Timed Loop
SISO	Single Input Single Output
SNR	Signal-to-Noise Ratio
USRP	Universal Software Radio Peripheral
VI	Virtual Instrument

Chapter 1

Introduction

1.1 Background

In the past 10 years, technology has taken huge steps forward in the fields of mobile communication. With the introduction of smart phones, tablets and wearable devices there are more wireless signals than ever before. The current development indicates that this trend will continue in the future with an increasing number of wireless devices. This trend of smart equipment and communication is now starting to affect other parts of society, for example the car industry, where a lot of development is already being put into the development of self-driving cars and smart car systems [1]. Future cars could communicate with each other and be able to warn each other of lane shifting, sudden brakes and fast emergency vehicles behind low sight corners, to lower the risk for accidents or to avoid them entirely [2]. When advanced communication systems as this one becomes standard equipment for vehicles, a robust communication system will be needed between the vehicles to ensure the safety and reliability of the systems.

The performance of a wireless system relies heavily on the environment it operates in, i.e. the wireless channel. Objects in the surrounding area usually interact with the wireless signal between the transmitter and the receiver. These interactions all contributes to the version of the signal that the receiver sees. When designing a wireless system, these and other factors need to be considered, so an understanding for how the channel behaves in different scenarios is crucial. This is where channel sounding proves useful. Channel sounding is the process of measuring properties of the channel (e.g. the impulse response), which can be done in a variety of ways [3, p. 150]. One possible technique is to transmit a signal from the transmitter to the receiver and letting the original signal be known to the receiver. This way the receiver is able to estimate how the channel affected the signal, which in turn could be used to model the channels behaviour.

In order to perform channel sounding and other signal processing operations, software defined radios, or SDRs, are becoming increasingly popular as the performance of the devices increases. SDRs offer a flexible and relatively fast alternative to hardware implementations, since much of the functionality is implemented in software.

1.2 Thesis purpose and aim

The purpose of this thesis is to implement and evaluate a 2x2 MIMO channel sounder on software defined radio devices with onboard FPGA:s. The design should support the use of different sounding sequences. The performance of the channel sounder framework will be evaluated when sounding with either Zadoff-Chu sequences or m-sequences. As for future applications, the sounder could act as foundation for larger and more advanced channel sounders. The evaluation of the sounding sequences could also give an indication of which of the sequences could be the better one to use for different sounding scenarios. The possibilities to further improve and scale the sounder will be discussed, as well as future applications of the sounder.

1.3 Thesis outline

This report is structured as follows. In chapter 2, the theoretical background. The concept of a wireless channel is explained, different channel models are presented, and a theoretical model for MIMO systems is described. A more thorough explanation of channel sounding is provided, and different methods of channel sounding are presented. Two sequences, namely m-sequences and Zadoff-Chu sequences, suitable for channel sounding are also described. In chapter 3 the sounding equipment used is presented, including many aspects of the SDRs and their programming environment. In chapter 4, we propose an FPGA implementation of the cross-correlation algorithm that processes the received signals for the channel sounding purposes. We also address the details of our particular implementaion, which was performed using the programming language of the SDRs.

Chapter 2

Wireless channels and channel sounding

2.1 Wireless channels

In wireless communications systems there are many factors that influence the performance. Some examples of these are path loss between the transmitter and the receiver, interference from other wireless systems, or objects in the terrain between the transmitter and the receiver. The transmitted signal might travel through buildings and terrain, be reflected by obstacles in the environment etc. All of these things contribute to the distortion of the transmitted signal, and they collectively make up the wireless channel.

2.1.1 Channel properties and parameters

One property that affects a wireless signal is path loss, that is, the median loss of power the signal experiences when travelling between the transmitter and the receiver. In the most basic scenario, with no obstacles in the way, the power received at the receiver is given by Friis law [3, p. 48]

$$P_{RX}(d) = P_{TX} G_{TX} G_{RX} \left(\frac{\lambda}{4\pi d} \right)^2, \tag{2.1}$$

where G_{RX} and G_{TX} are the antenna gains for the receiving and transmitting antennas, respectively, λ is the wavelength and d is the distance between the transmitter and the receiver.

Generally, wireless systems are not deployed in areas free from obstacles, but rather in densely populated areas, where there are many buildings, cars etc. Thus, the simple model for the received power is not enough to fully describe the wireless channel. As the signal travels it will interact with the obstacles in its paths (so called interacting objects, or IOs). Two of the ways a signal can interact with IOs are reflection and transmission. Reflection occurs when a electromagnetic signal hits a smooth surface, and is reflected off it [3, p. 69]. When the surface is rough the signal a will scatter against the surface, and the power of the reflected signal will be dispersed in many directions.

Electromagnetic waves can also be subject to transmission, which means that the wave passes through an object. How much power the electromagnetic signal retains depends on the material of the object and the wavelength of the signal. It is this phenomenon that allows cellphones and other wireless devices to be used inside buildings, where there is no direct path to the base station.

If either the transmitter or the receiver is moving, the signal is subjected to the Doppler effect, which means that the frequency increases or decreases depending on if the transmitter and the receiver moves towards or away from each other. This results in a frequency shift of the received signal, given by [3, p. 73]

$$\nu = -f_c \frac{v}{c} \cos\alpha, \tag{2.2}$$

where f_c is the carrier frequency, v is the relative speed of the receiver in relation to the transmitter, c is the propagation speed of the signal and α is the incident angle.

2.1.2 Multipath propagation

Due to the interactions with IOs, the transmitted signal can take multiple paths as it travels between the transmitter and the receiver. This causes multiple copies of the transmitted signal to arrive at the receiver, and these copies are called multipath components (MPCs). The MPCs will have different attenuation and phase depending on their individual paths, and they will also arrive at slightly different times since the length of each propagation path typically is different. This is known as a time-dispersive channel, and transferred into the frequency domain, it makes the channel frequency selective, that is, different frequencies will have different amplitudes [4, p. 11].

Since the receiver may not able to differentiate between different MPCs, they will interfere with each other. The interference can be constructive, ie. the crests of multiple MPCs align, which will increase the amplitude of the received signal at a specific time and frequency. They can also be misaligned, in that case they will cancel each other out, ie. interfere destructively [3, p. 28].

Due to different MPCs having different delays, the channel impulse response will not be a single Delta function [3, p. 27]. Instead it will consist of several peaks, each corresponding to one MPC. This phenomena is referred to as time dispersion. A way of modelling time dispersion is the tapped delay line model [3, pp. 31,128]

$$h(t,\tau) = a_0\delta(\tau-\tau_0) + \sum_{i=1}^{N} c_i(t)\delta(\tau-\tau_i), \tag{2.3}$$

where a_0 is the amplitude of the first component, c_i typically is the zero-mean complex Gaussian distributed random process with a mean power following the so called power delay profile and τ_i is the delay for each MPC.

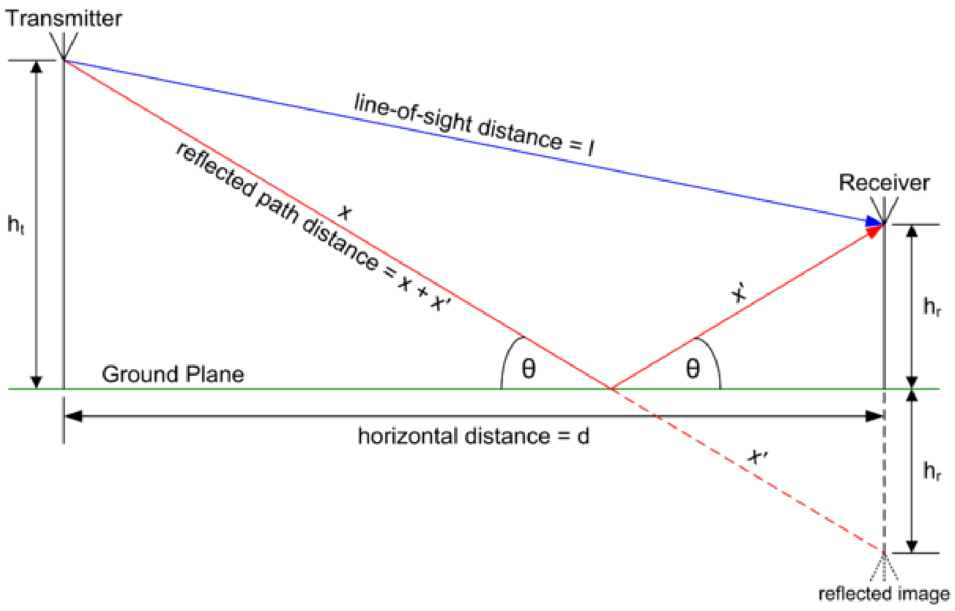


Figure 2.1: Example of a two-path model [5].

A simple case of the tapped delay line model is the two-path channel, visualized in Figure 2.1. The channel consists of two components, the LOS component, and the component reflected off the ground. Figure 2.2 shows an example of a possible impulse response for a two-path channel where the amplitude of the first MPC is normalized.

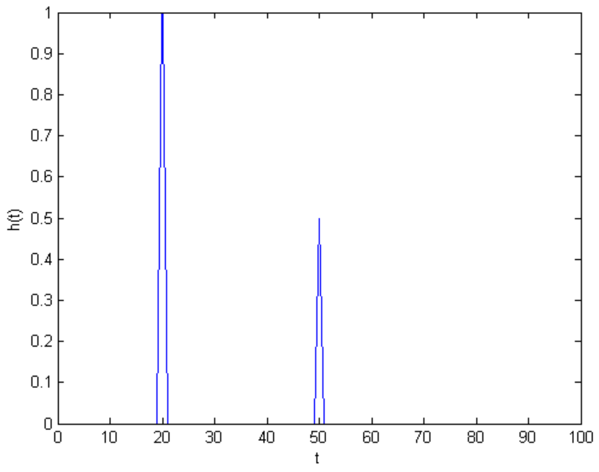


Figure 2.2: Example of an impulse response for a two-path channel.

2.2 System models

In many areas of engineering, it is useful to make a mathematical model of a real life scenario to be able to gain an understanding of how the real world scenario behaves. In this chapter, system models that offer a mathematical explanation for how wireless communications system, both single antenna and multiple antenna systems, behaves is presented.

2.2.1 SISO - Single Input Single Output

To be able to understand multiple antenna systems, it is essential to first understand the basic single antenna (SISO) systems as it lays some ground principles that the multiple antenna system models build upon. With one transmitter, one receiver and a time-invariant channel, such as a wire, the relationship between the transmitted signal $s(t)$ and the received signal $r(t)$ can be expressed as

$$r(t) = h(t) * s(t) + n(t), \tag{2.4}$$

where $h(t)$ is the channel impulse response, $s(t)$ is the transmitted signal, $n(t)$ is the random process modelling the noise and $*$ denotes convolution.

In wireless systems the channel can often not be seen as time-invariant since the channel will change as the propagation environment changes. Because of this, the impulse response of the channel becomes time-variant [6, p. 11]

$$y(t) = \int_{-\infty}^{\infty} x(t-\tau)h(t,\tau)d\tau \tag{2.5}$$

The channel $h(t,\tau)$ is referred to as a time-variant channel where τ denotes the delay.

2.2.2 MIMO - Multiple Input Multiple Output

In the 2000s, the use of wireless devices has increased dramatically [7] and in order for the wireless systems to be able to cope with the increased demands it has become necessary to develop new techniques to make more efficient use of the radio spectrum and to increase the bit rate of the system [8, p. 6]. One way of achieving this is to introduce the concept of MIMO, in which there are multiple antennas at the receiver and the transmitter [3, p. 465]. One of the advantages of MIMO is that beamforming can be used, in which the transmitted energy is focused in a certain direction which increases the received SNR in that direction. Another advantage is spatial multiplexing, where the data is transmitted on different antennas at the same time. The receiver can then recover each part by the use of signal processing. Another advantage is that MIMO introduces spatial diversity [8, p. 6]. Since the receive- and transmit antennas are separated in space the channel will affect the signal arriving at the different antennas differently, and by combining the signals received at each antenna a high SNR can be achieved even though the received signal strength may be low on some of the antennas.

A time-varying MIMO channel is described by the channel matrix

$$\mathbf{H}(t,\tau) = \begin{bmatrix} h_{11}(t,\tau) & \dots & h_{1M_{RX}}(t,\tau) \\ \vdots & \ddots & \vdots \\ h_{M_{TX}1}(t,\tau) & \dots & h_{M_{TX}M_{RX}(t,\tau)} \end{bmatrix}, \tag{2.6}$$

where $h_{ij}(t,\tau)$ is the impulse response between transmit antenna i and receive antenna j, and M_{TX} and M_{RX} is the number of transmit and receive antennas respectively.

The relationship between the transmitted signals and the received signals in a MIMO system can be described as

$$\mathbf{r}(t) = \mathbf{H}(t,\tau) \star \mathbf{s}(t) + \mathbf{n}(t) = \int_{-\infty}^{\infty} H(t,\tau)s(t-\tau) + n(t)d\tau, \tag{2.7}$$

where $\mathbf{r}$ is the vector of received signals, $\mathbf{s}$ is the vector of transmitted signals and $\mathbf{n}(t)$ is the noise. For a 2x2 MIMO system, $\mathbf{r} = \begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix}$ is the received signals at antenna 1 and 2, $\mathbf{s} = \begin{bmatrix} s_1(t) \\ s_2(t) \end{bmatrix}$ is the transmitted signal from antenna 1 and 2, and $\mathbf{H}$ is the 2x2 channel matrix.

From this it can be seen that $r_1(t)$ and $r_2(t)$ can be written as [8, p. 34]

$$r_1(t) = \int_{-\infty}^{\infty} s_1(t-\tau)h_{11}(t,\tau) + n(t)d\tau + \int_{-\infty}^{\infty} s_2(t-\tau)h_{21}(t,\tau) + n(t)d\tau \tag{2.8a}$$

$$r_2(t) = \int_{-\infty}^{\infty} s_1(t-\tau)h_{12}(t,\tau) + n(t)d\tau + \int_{-\infty}^{\infty} s_2(t-\tau)h_{22}(t,\tau) + n(t)d\tau \tag{2.8b}$$

2.3 Channel sounding

When developing channel models for different types of wireless systems it is necessary to know how the wireless channel behaves. Early measurements conducted in the 1960s only measured the received signal strength [3, p. 145], but as seen in section 2.1, the received signal strength is just one of the properties of a wireless channel.

The principle of a channel sounder is quite simple. The transmitter transmits a sequence, the signal travels through the wireless channel to the receiver, which by knowing the original transmitted sequence, can estimate the channels impulse response. In order for the channel sounder to be efficient, the transmitted signal needs to fulfill a few criteria. The bandwidth of the signal should be sufficiently large so that the delay resolution becomes high enough, since the bandwidth is proportional to the inverse of the delay resolution [3, p. 146]. The length of the sequence should be short enough so that the channel stays constant for one period of the signal, so that the channel can be seen as time-invariant for the duration of one sequence. The frequency response of the signal should be flat, so that the entire frequency spectrum of the signal can be estimated with equal quality [3, p. 146]. In this chapter we will describe two time domain channel sounding principles. We will also introduce the concept of MIMO channel sounding, as well as present two types of signals possible to use as sounding signals.

2.3.1 Impulse sounder

One type of sounder is the impulse sounder, in which the transmitter transmits a short pulse every T seconds. The receiver simply interprets the received signal as the channel impulse response [8, p. 118]. In order to sound over large bandwidth, the pulses transmitted should be narrow in time with high amplitude in order for the frequency response of the signal to be as flat as possible. Due to the properties of antennas, amplifiers, mixers etc, the type of pulse needed can be difficult to generate. Also, because to the shape of the pulse, the sounder can cause a significant amount of interference in frequency [9, p. 171], since the transmitted pulse gives a wide frequency response because of its similarity to a Dirac delta-function.

2.3.2 Correlative sounder

Instead of transmitting short pulses, a correlative sounder transmits a sequence with good autocorrelation properties. The receiver then takes the received signal $r(t)$ and correlates it with the transmitted signal $s(t)$. A simple noise free, time-invariant system can be expressed as

$$r(t) = h(t) * s(t). \tag{2.9}$$

If the continuous autocorrelation function for $s(t)$

$$R_{s,s}(\tau) = \int s(t)s^*(t-\tau)dt, \tag{2.10}$$

is equal to the Dirac's delta function

$$\delta(t) = \begin{cases} 1 & t = \infty \\ 0 & k \neq 0 \end{cases}, \tag{2.11}$$

or if the discrete autocorrelation function

$$R_{s,s}[k] = \sum_{n \in Z} s[k]s^*[n-k], \tag{2.12}$$

equal to the Kronecker delta

$$\delta[k] = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}, \tag{2.13}$$

the channel impulse response $h(t)$ can be reliably estimated by cross-correlating $r(t)$ with $s(t)$ [3, p. 151]

$$r(t) \circledast s(t) = h(t) * s(t) \circledast s(t) \implies h(t) = r(t) \circledast s(t), \tag{2.14}$$

where the $\circledast$ denotes discrete cross-correlation.

One issue with correlative sounder is that the hardware in the transmitter and the receiver also contribute to the distortion of the transmitted signal, so (2.9) doesn't suffice when describing the entire system. A more accurate description is instead

$$r(t) = h_{TX}(t) * h(t) * h_{RX}(t) * s(t), \tag{2.15}$$

where $h_{TX}(t)$ impulse response for the transmitter, and $h_{RX}(t)$ is the impulse response from the receiver. Because of this, simply cross correlating $r(t)$ with $s(t)$ is not enough, as it is required to remove $h_{TX}(t)$ and $h_{RX}(t)$ as well [10, p. 1467]. In a practical system, this can be done by connecting the transmitter to the receiver with a cable, and estimating the impulse response of the system. The saved impulse response is then removed from the impulse response obtained when sounding the channel by deconvolving the total estimated impulse response with that of the system.

2.3.3 MIMO channel sounding

The challenge that arises when performing channel sounding in a MIMO system is that the received signals at each receive antenna will be a combination of all of the transmitted signals, as described in subsection 2.2.2. This makes it necessary to differentiate between the signals transmitted from each antenna. In this project the differentiation is made by transmitting different signals with good cross correlation properties from each antenna in order to minimize the interference between them, so that the channels between all antenna pairs can be estimated. In other words

$$R_{s_1,s_2}(\tau) = \int_{-\infty}^{\infty} s_1(t)s_2^*(t-\tau)dt, \tag{2.16}$$

should be as low as possible, where s_1 and s_2 are the sequences transmitted from the two antennas.

2.4 Sounding sequences

When performing correlative channel sounding, there are a number types of sequences that can be used. Two of them are the Zadoff-Chu- and m-sequences, which both will be presented below.

2.4.1 The Zadoff-Chu sequence

The Zadoff-Chu-sequence is a complex-valued discrete time sequence where the elements in the sequence are defined as

$$u_i[k] = (-1)^{ik} e^{\frac{j\pi i^2 k}{N}},\ 0 \leq i \leq N-1, \tag{2.17}$$

where N, the length of the sequence, is an odd, positive integer, $0 < k < N$, $\gcd(k, N) = 1$.

The reason why Zadoff-Chu sequences are suitable for channel sounding is due to its lack of variation in amplitude, and its autocorrelation-(2.18) and cross correlation(2.19) properties which can be seen in Figure 2.3 [see 11, pp. 151, 152]

$$R_s(\tau) = \begin{cases} N \text{ for } \tau \equiv 0 \text{ mod } N \\ 0 \text{ otherwise} \end{cases}, \tag{2.18}$$

$$R_{s_{k_1},s_{k_2}}(\tau) = \sqrt{N}, \text{ for } \tau \equiv 0 \mod N, \tag{2.19}$$

if $\gcd(|k_1 - k_2|, N) = 1$ where k_1 and k_2 are the value of k in Equation 2.17 for the respective Zadoff-Chu sequence.

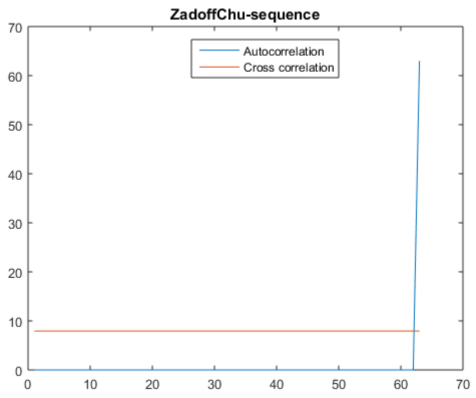


Figure 2.3: The autocorrelation of a Zadoff-Chu sequence, and a cross correlation between two Zadoff-Chu sequences with $k_1 = 25$ and $k_2 = 29$.

It is their cross correlation properties makes them particularly interesting in MIMO channel sounding, since two Zadoff-Chu-sequences transmitted from two antennas interfere with each other with a factor of $\frac{1}{\sqrt{N}}$. Another reason that Zadoff-Chu-sequences are suitable for channel sounding is its flat frequency response. This means that fluctuations in the received power will be equally visible over the entire frequency spectrum.

2.4.2 m-sequences

A binary m-sequence is a sequence consisting of or 1s and -1s, with the length $N = 2^n - 1$, where n is a positive integer. In order for a binary sequence to be considered an m-sequence, they need to fulfill a number of conditions [11, p. 119]. They need to be balanced, ie. the number of 1s must not differ from the number of -1s more than one. A subsequence of 1s or -1s is called a run, and 1/2 of all runs in the sequence must be of length 1, 1/4 of all runs must be of length 2, 1/8 of all runs must be of length 3 etc. The last condition is that the autocorrelation function for the sequence must be

$$R_s(\tau) = \begin{cases} N & \text{for } \tau \equiv 0 \mod N \\ -1 & \text{otherwise} \end{cases}. \tag{2.20}$$

The frequency response of an m-sequence can be seen in Figure 2.4.

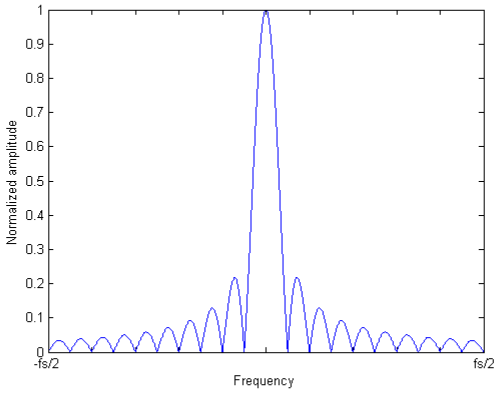


Figure 2.4: Frequency spectrum of an arbitrary m-sequence.

One way of performing MIMO sounding with m-sequences, is to delay the m-sequence transmitted from one of the antennas by $N/2$ samples in relation to the sequence transmitted from the other antenna, so the result of the cross correlation at the receiver shows two peaks, one at 0 and one at $\frac{N}{2}$ samples as shown in Figure 2.5, assuming a noise free one path channel. This limits the maximum delay that can be unambiguously detected to $\frac{N}{2}$ samples. If the sequences are transmitted continuously, the receiver has no way of knowing which peak corresponds to which transmitted sequences. This could possibly be solved by for example not transmitting continuously, but leaving a small window between each transmitted sequence so that the receiver can discern which peak corresponds to which transmit antenna.

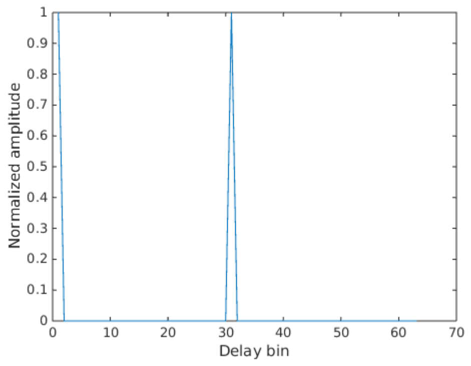


Figure 2.5: Cross correlation of two m-sequences shifted $\frac{N}{2}$ samples.

2.4.2.1 Construction of m-sequences

m-sequences are constructed using a Linear Feedback Shift Register (LFSR) of length k, where the feedback function of the LFSR must be a primitive polynomial in the Galois field $GF(2^k)$, for an m-sequence of length $N = 2^k - 1$ [12, p. 3].

Feedback shift registers are made up by a series of elements, forming a queue. For each clock cycle, the queue is pushed one step forward and a new value for the first step is created from some or all of the queue values from the last iteration. The value pushed out of the queue is the output for each iteration. This produces a binary sequence with the values 1 and 0, and the 0:s are replaced with -1:s.

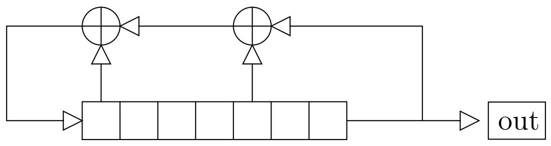


Figure 2.6: Example of a 7-bit shift register.

The feedback function of an LFSR describes what positions that are used when computing the new element. It is often represented in the form of a polynomial [13]

$$p(x) = x^k + x^{k-1} + \ldots + x + 1, \tag{2.21}$$

where k_n denotes which position in the shift register that is added. As an example, the LFSR in Figure 2.6 has 7 bits and the feedback function

$$p(x) = x^7 + x^4 + x + 1 \tag{2.22}$$

indicates that new elements are calculated by modulo-2 adding the first, last and the third element of the LFSR [14, p. 17].

2.5 Estimating the impulse responses

Although wireless channels generally are time-variant, for this application it is assumed that the channel is time-invariant for the entire length of one sequence.

Since the equipment used for measuring the channel also have its own impulse response, it has to be taken into account as well. From (2.7) we get the new equation for the received signal at antenna i

$$r_i(t) = \sum_{j=1}^{M_{TX}} s_j(t) * h_j^{TX}(t) * h_{ij}(t) * h_i^{RX}(t) + n(t), \tag{2.23}$$

where $n(t)$ is the additive noise process, $h_{TXj}(t)$ is the impulse response of the j:th transmitter chain, $h_{RXi}(t)$ is the impulse response of the i:th receiver chain, $s_j(t)$ is the signal transmitted from the j:th transmit port, and $h_{ij}(t)$ is the impulse response from the j:th transmit port to the i:th receiver port. Using the commutative property of the convolution operator, (2.23) can be rewritten as

$$r_j(t) = \sum_{i=1}^{M_{TX}} s_i(t) * h_{ij}^d(t) * h_{ij}(t) + n(t), \tag{2.24}$$

$$\text{where } h_{ij}^d(t) = h_i^{TX}(t) * h_j^{RX}(t). \tag{2.25}$$

The two transmitted signals are removed from the received signal by cross correlation.

$$\hat{h}_{ij}^{tot} = \sum_{i=1}^{M_{TX}} r_j(t) \circledast s_i(t) + n(t),\ j = 1 \ldots M_{RX}. \tag{2.26}$$

If the cascade of the hardware responses $h_{ij}^d(t)$ are known, one way of removing the measurement system from the channel measurement is by deconvolution, which in the frequency domain can be done by

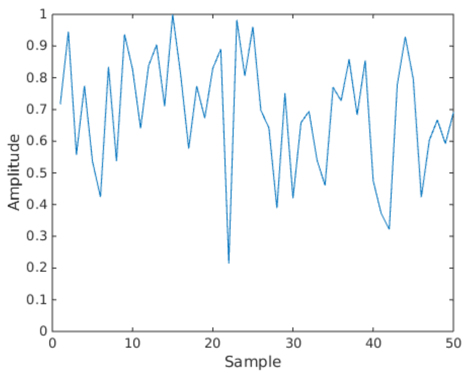
$$\hat{h}_{ij}(t) = \mathcal{F}^{-1}\left[\frac{\mathcal{F}[\hat{h}_{ij}^{tot}(t)]}{\mathcal{F}[h_{ij}^d(t)]}\right], \tag{2.27}$$

where $\mathcal{F}$ is the Fourier transform.

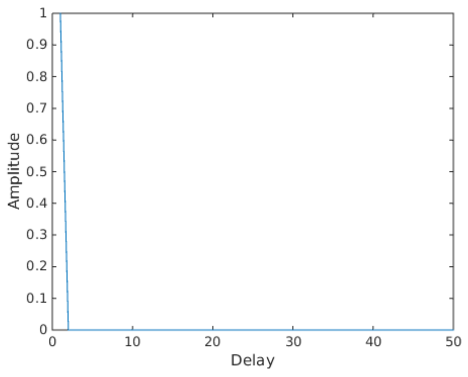
As can be seen below, if $\hat{h}_{ij}^{tot}(t)$ and $h_{ij}^d(t)$ closely resembles Dirac delta functions, cross correlation may be used to remove the systems inpulse response h_{ij}^d from the total impulse response $\hat{h}_{ij}^{tot}$

$$\hat{h}^{ij}(t) = \hat{h}_{ij}^{tot}(t) \circledast h_{ij}^d(t). \tag{2.28}$$

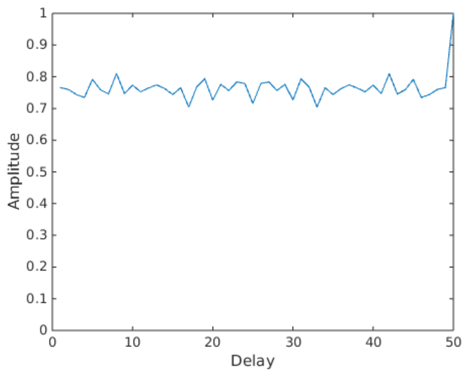
As a sequence approaches a Dirac delta function, the result of cross correlation approaches the result of a deconvolution. Figure 2.7b and Figure 2.7c shows the deconvolution and cross correlation respectively of a sequence of 50 random numbers used as an example sequence Figure 2.7a. As can be seen there is quite a large difference between the two, the deconvolution is a normalized Dirac delta function, while the cross correlation average is around 0.8, apart from the spike at 50 lag where it's equal to 1. If we instead look at Figure 2.8 we can see that the cross correlation more closely resembles the deconvolution when the peak is significantly higher than the rest.



(a) Original sequence

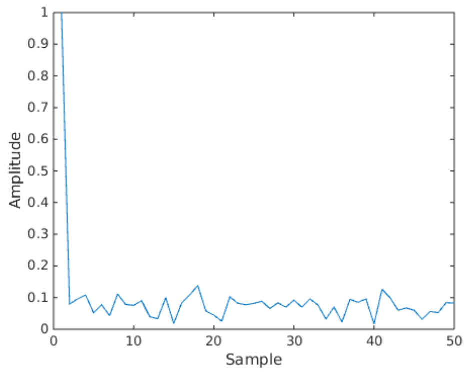


(b) Deconvolution

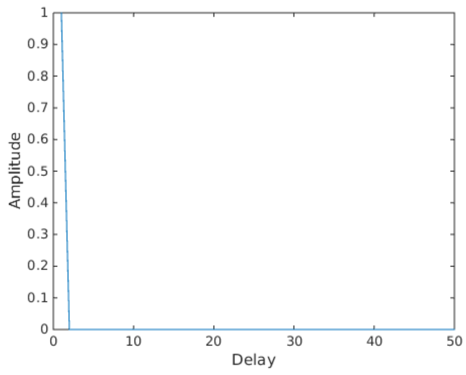


(c) Cross correlation

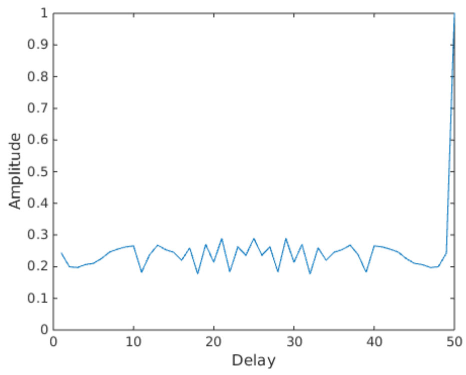
Figure 2.7: Deconvolution and cross correlation done for an example sequence of 50 random normalised values.



(a) Original sequence



(b) Deconvolution



(c) Cross correlation

Figure 2.8: Deconvolution and cross correlation done for an example sequence of 50 random normalised values, with a Dirac delta function.

Chapter 3

Tools overview

3.1 USRP

Software Defined Radio (SDR) is a concept in the area of radio devices, in which tasks (such as baseband processing algorithms) that have traditionally been done with hardware components are implemented in either software or reprogrammable hardware such as an FPGA. This makes SDR platforms more versatile when it comes to experimenting with wireless systems and technologies, making it a useful tool in for example prototyping new wireless systems [15].

The advantage of implementing the processing on a computer is that higher level programming languages can be used, while the FPGA is available to do processing at a much higher rate than a computer but at the cost of being more complex and restricting to do the implementation on.

One example of SDR devices are the USRPs, which is a series of devices manufactured by Ettus Research and National Instruments. The device used in this project is the NI-USRP 2953R. It can tune its center frequency from 1.2 to 6 GHz, with a bandwidth of 40 MHz. It is equipped with two RF chains (RF0 and RF1) with 2 antenna ports each, one transmit antenna (TX1) and one receive antenna (RX2) [16]. The transmit antenna can be used for both transmitting and receiving, though not at the same time. The USRP is equipped with a Kintex 7410T FPGA from Xilinx. A list of few of the specifications can be seen in Table 3.1, and a more exhaustive list can be found in [17].

In Figure 3.1 a block diagram of the NI USRP-2953R is shown, which shows a basic view of the components used between the antennas and the host computer controlling the USRP. The data to be transmitted is generated by the FPGA, either directly or it is sent to the FPGA from the host computer. It is sent to the digital upconverter, which re-samples the signal to work with the digital-analog converter, all done on the onboard FPGA. Then, outside the FPGA, a low pass filter cleans up the signal from high frequency components and noise before a mixer upconverts the signal to the selected frequency. The signal is then amplified and transmitted from the antenna. The receiver part of the USRP works in the same way, but all operations are reversed except the amplifying, as can be seen in the diagram.

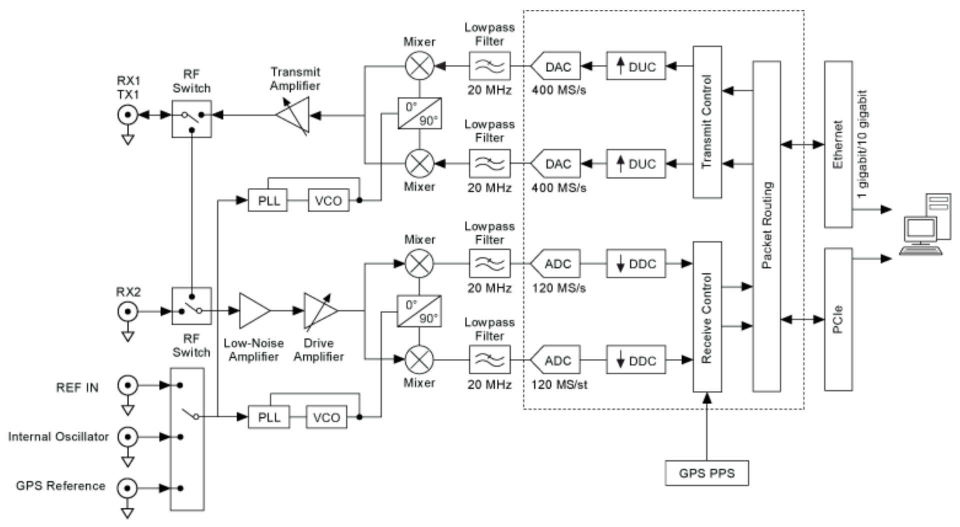


Figure 3.1: NI USRP-2953R block diagram [18], the functionality inside the square is implemented on the FPGA.

Logic cells	406 720
DSP slices	1540
Block memory	28 620 Kb

Table 3.1: Kintex 7 410T specifications

The FPGA can be used to process the data on the USRP instead of on the host computer, reducing the number of calculations needing to be performed by the host computer. By using lookup tables, which are able to store and retrieve bits, depending on the input to the table, logic blocks can be created. These blocks can, depending on the values stored, behave like any logic gate like AND or OR, etc. Logic blocks can be grouped together to perform multiplications or simultaneously perform several additions [19], and these groups are referred to as DSP slices.

By putting many such logic blocks on the FPGA and redirectable paths between them, different implementations can be created by changing the values stored in the lookup tables and by changing the paths between the blocks. If needed, the logic blocks can be used as RAM memory along with the dedicated block memory that is built into the FPGAs. A very important purpose of the FPGA is to give the flexibility of being able to compile an application in just a couple of hours and still get the benefits of an integrated circuit, like being able to do single cycle timed loops, which are loops where all operations are able to run in a single clock cycle. An FPGA also have I/O ports, which allows it to receive input values from external data sources.

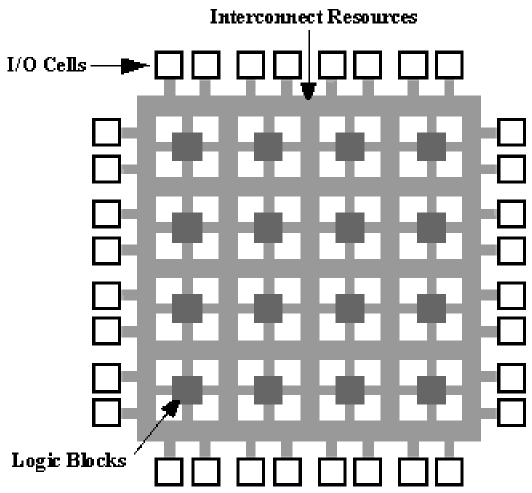
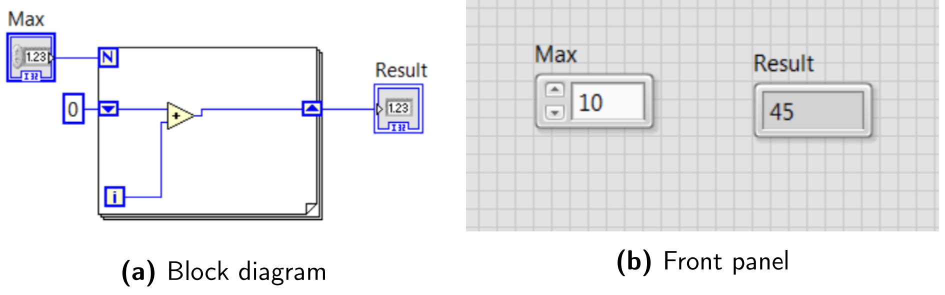


Figure 3.2: An example of a small FPGA.

3.2 Programming environment

3.2.1 LabVIEW

LabVIEW is a graphical programming platform developed by National Instruments. A LabVIEW program (referred to as a Virtual Instrument, or VI) consists of two parts, the block diagram and the front panel. The front panel contains the user interface and the block diagram contains the program logic. Elements on the Front panel are referred to as controls or indicators, where controls can be used for input and indicators for output [20]. Figure 3.3 is an example of a LabVIEW VI that calculates $\sum_{i=0}^{N-1} i$, where N equals 10, as defined in the "Max" control, and outputs it to the "Result" indicator.



(a) Block diagram **(b)** Front panel

Figure 3.3: A simple LabVIEW example showing addition, a loop, shift registers and the front panel.

Block are connected using wire. The colors of the wires differ depending on what datatype the wire is, and the thickness of them denotes whether it is a scalar value, an array or a matrix [21].

3.2.1.1 LabVIEW FPGA

In order to program the FPGA, the LabVIEW FPGA module is used. LabVIEW FPGA programming interface is similar to usual LabVIEW programming, although there are some differences. The more notable difference in regards to this project is the lack of floating point numbers, no complex numbers, no division and no modulus operator [22] in LabVIEW FPGA. One central component in LabVIEW FPGA is the Single Cycle Timed Loop (SCTL). It's guaranteed to execute one loop iteration in one clock cycle, compared to a normal while loop which will take 3 clock cycles to execute one loop iteration [23].

In order to run an FPGA VI, is it first compiled to a bitfile, which loaded onto the FPGA from a VI running on a host computer. In this project, the host computer is connected to the USRP via a PCIe-cable as illustrated in Figure 3.4.

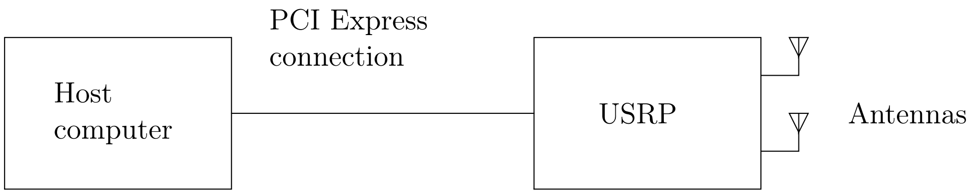


Figure 3.4: The hardware setup used

3.2.1.2 Communication between host computer and FPGA

Data can be transported between the host computer and the FPGA in two ways, either by using front panel controls and indicator on the FPGA VI, and write to/read from these on the host, or to use DMA FIFOs. There are three types of FIFOs:

- Host-to-target, used to stream data from the host computer to the FPGA
- Target-to-host, used to stream data from the FPGA to the host computer
- Target-scoped, used to stream data inside the FPGA, eg. between two parallel running loops

Figure 3.6 shows an example where data is read from the host and written to a target-scoped FIFO in the upper SCTL, and is read from a FIFO and written to the host using a target-to-host FIFO in the lower SCTL. One of the features of target-scoped FIFOs is that they can transport data between SCTLs with different clock rates [24].

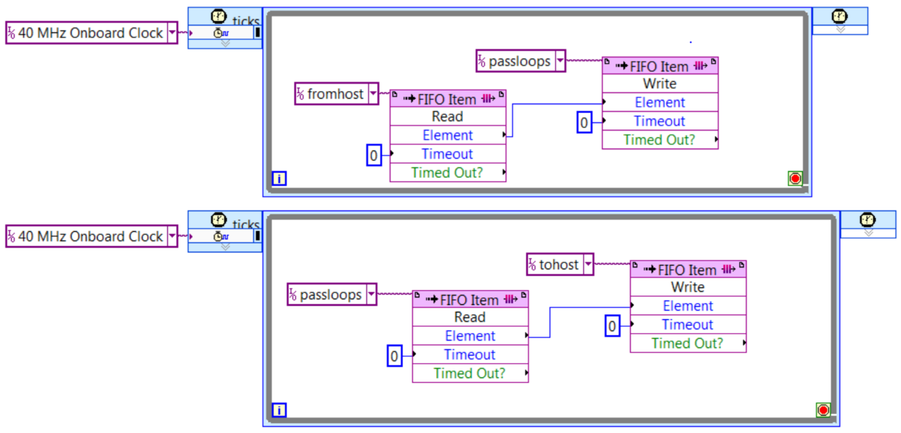


Figure 3.5: Example showing the use of FIFOs

3.2.1.3 Storing and accessing data inside the FPGA VI

For data that needs to be accessed from a sequence arbitrarily and not in a sequential manner, the preferred method is to use memories. Memories can either be implemented in block memory or by using the LUTs, depending on size of the memory and whether the memory needs to be accessed from different clock domains [25]. If the memory is implemented in block memory, it takes a minimum of one iteration to read from the memory whereas memory implemented in LUT doesn't have this latency. Below is an example of an FPGA VI where data is read from a memory from the indices $0 \ldots 9$ and written to a target-scoped FIFO.

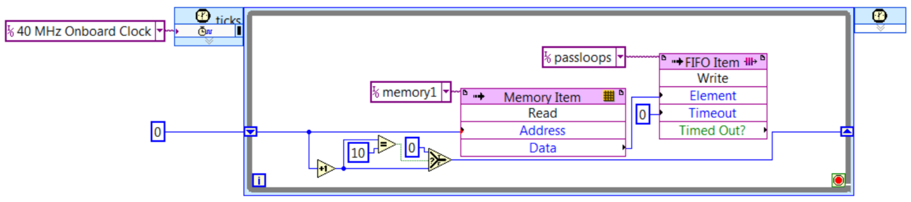


Figure 3.6: Example of reading from a memory in LabVIEW FPGA

Chapter 4

Proposed implementation

In this chapter the FPGA implementation of the cross-correlator at the receiver, for both Zadoff-Chu sequences and m-sequences, as in (2.26) is described. Since the sequences are transmitted continuously from the transmitter, the cross correlation in this application is equal to circular cross correlation.

4.1 General approach

One important constraint of FPGA development is that the FPGA has limited space for programming logic, meaning that the number of operations that can be performed in parallel on the FPGA is limited. In order to be able to handle long sequences, the circular cross-correlation needs to take up a small amount of the available resources, even when the sequence length N is large.

Thus, if we want to be able to perform circular cross-correlation for sequences of arbitrary lengths, we need to implement it in such a way that the number of operations needed to be performed in parallel does not increase as the sequence length, N, increases. The algorithm for achieving this is described the example below.

If we take, for example, the m-sequence $x_1 = x_2 = (1\ {-1}\ 1)$ the circular cross-correlation equals

$$x_1(3) \times x_2(1) + x_1(2) \times x_2(2) + x_1(1) \times x_2(3) = 1 \times 1 + (-1) \times (-1) + 1 \times 1 = 3 \tag{4.1}$$

$$x_1(3) \times x_2(3) + x_1(2) \times x_2(1) + x_1(1) \times x_2(2) = 1 \times 1 + (-1) \times 1 + 1 \times (-1) = -1 \tag{4.2}$$

$$x_1(3) \times x_2(2) + x_1(2) \times x_2(3) + x_1(1) \times x_2(1) = 1 \times (-1) + (-1) \times 1 + 1 \times 1 = -1 \tag{4.3}$$

If we imagine a scenario in which the transmitter transmits x_1 repeatedly, and that x_1 is known at the receiver as $\hat{x}$ and the receiver receives exactly the same, discrete values that are transmitted, the received values will be $\ldots 1\ {-1}\ \ 1\ \ 1\ {-1}\ \ 1 \ldots$. This scenario could obviously never occur, as explained in chapter 2, but it will do as a model for describing our implementation.

Let the received samples be $r = 1\ -1\ 1\ 1\ -1\ 1\ 1\ -1\ldots$ and $\hat{x} = (1\ -1\ 1)$. $N = 3$ denotes the length of $\hat{x}$. τ denotes the number of samples $\hat{x}$ is shifted, $sum_{-1} = 0$ and $i = k + \tau \mod N$. For the first iteration $\tau = k = 0$ and k is incremented by 1 after each processed sample. In order to calculate the circular cross-correlation for one period, the following steps are executed.

The following is done in order to calculate (4.1).
For the first received sample $r_0 = 3$, take $\text{sum}_0 = \text{sum}_{-1} + r_0 \times \hat{x}_i = 0 + 1 \times 1 = 1$.
For the second received sample $r_1 = 2$, $\text{sum}_1 = \text{sum}_0 + r_1 \times \hat{x}_i = 1 + -1 \times -1 = 2$.
For the third received sample $r_2 = 1$, $\text{sum}_2 = \text{sum}_1 + r_2 \times \hat{x}_i = 2 + 1 \times 1 = \mathbf{3}$.

In order to calculate (4.2), τ is incremented in order to induce the shift in $\hat{x}$. $\tau = 1$ and k is reset to 0.
For the fourth received sample $r_4 = 3$, take $\text{sum}_0 = \text{sum}_{-1} + r_0 \times \hat{x}_i = 0 + 1 \times -1 = -1$.
For the fifth received sample $r_5 = 2$, $\text{sum}_1 = \text{sum}_0 + r_1 \times \hat{x}_i = -1 + -1 \times 1 = -2$.
For the sixth received sample $r_6 = 1$, $\text{sum}_2 = \text{sum}_1 + r_2 \times \hat{x}_i = -2 + 1 \times 1 = -1 = \mathbf{-1}$.

The calculations are done in the same manner for (4.3), where $\tau = 2$ and $k = 0$.
For the seventh received sample $r_7 = 3$, take $\text{sum}_0 = \text{sum}_{-1} + r_0 \times \hat{x}_i = 0 + 1 \times 1 = 1$.
For the eighth received sample $r_8 = 2$, $\text{sum}_1 = \text{sum}_0 + r_1 \times \hat{x}_i = 1 + -1 \times 1 = 0$.
For the ninth received sample $r_9 = 1$, $\text{sum}_2 = \text{sum}_1 + r_2 \times \hat{x}_i = 0 + 1 \times -1 = \mathbf{-1}$.

This concludes $R_{x_1,x_2}(\tau)$ for $0 \leq \tau \leq N - 1$. In our implementation, the algorithm then starts over from $\tau = 0$ as we continuously perform circular cross-correlation of the received signal.

4.2 LabVIEW implementation

The LabVIEW implementation of the algorithm is based on the "NI-USRP Streaming" example project that is bundled with LabVIEW, and the FPGA portion is heavily based on the "Streaming (Xcvr)" VI, which in its original state configures the USRP for receiving and transmitting on a specified center frequency, and the received samples are passed on directly to the host computer using two target-to-host FIFOs.

Our implementation is based on this VI, and the received samples are sampled from the radio interfaces at RF0 and RF1 inside a SCTL, and written to two target-scoped FIFOs instead, as seen in Figure 4.1, for further processing on the FPGA. The I- and Q-components are quantized by 16 bits, and merged into one 32-bit integer with the I-component in the lower bits and the Q-component in the higher.

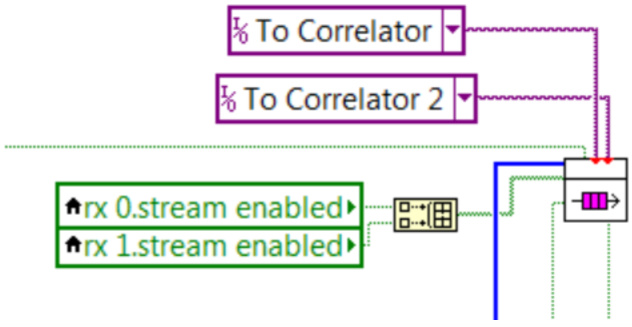


Figure 4.1: Samples being written to FIFOs from a SCTL loop where the blue wire is a 2x1 array containing one sample from RF0 and one from RF1

For the correlator SCTLs to begin executing, it requires knowledge of a few values, the type of sequence, the length of the sequences and the sequences themselves. Thus, before the correlator SCTLs start executing, these values are supplied to the FPGA from the host. The type of sequence and the sequence length is written to the FPGA from the host using Front panel indicators on the LabVIEW FPGA VI and the sequences are written to the FPGA from the host using two host-to-target FIFOs.

On the FPGA VI, the Front panel indicators storing the sequence length and type of sequence, both have a default value of 0. Before the SCTLs performing the circular cross-correlation is executed, the FPGA application waits for these values to be defined, as seen in Figure 4.2. The loop will exit when the values in the indicators are no longer 0.

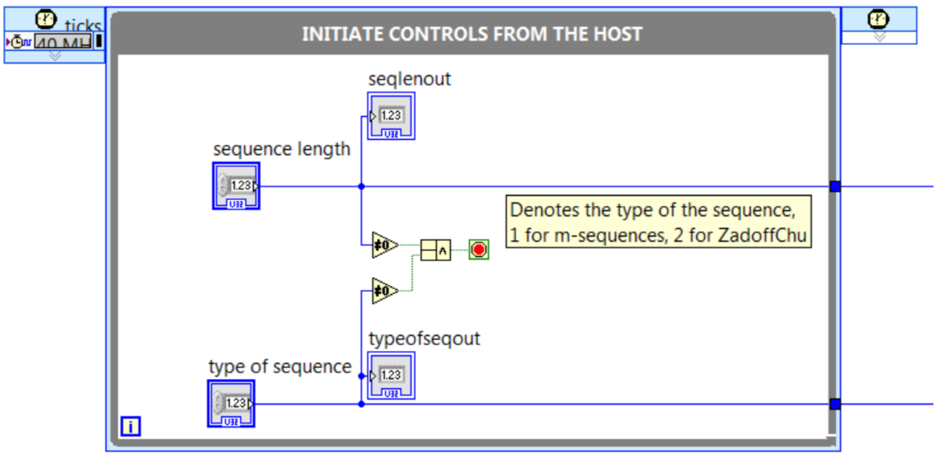

ticks
40 MHz
INITIATE CONTROLS FROM THE HOST
seqlenout
sequence length
Denotes the type of the sequence,
1 for m-sequences, 2 for ZadoffChu
typeofseqout
type of sequence

Figure 4.2: The loop that waits for the type of sequence and sequence length to be specified from the host

After Figure 4.2 have exited, the contents of the FIFOs storing s_0 and s_1 are written to two memories each, one to be used in the SCTL performing the circular cross-correlation for the samples from RF0, and one to be used in the SCTL for the samples from RF1.

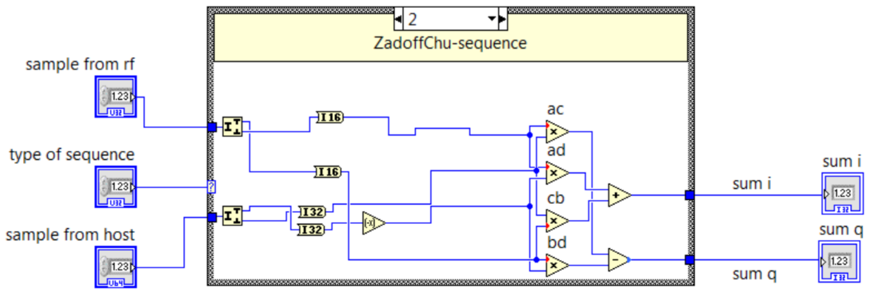
The samples in the sequences are represented by one unsigned 64-bit integer. In the case of the Zadoff-Chu sequence the real and imaginary part of each element is represented as 32-bit integers and merged into a unsigned 64-bit integer with the real component in the higher bits and the imaginary component in the lower. In the case of the m-sequence, each element of the m-sequence is represented as a 32-bit integer and written to both the higher and lower part of a 64-bit unsigned integer. After the sequences are taken from from the FIFOs and stored in the memories, the SCTLs that performs the circular cross-correlation starts executing.

4.2.1 Correlator loops

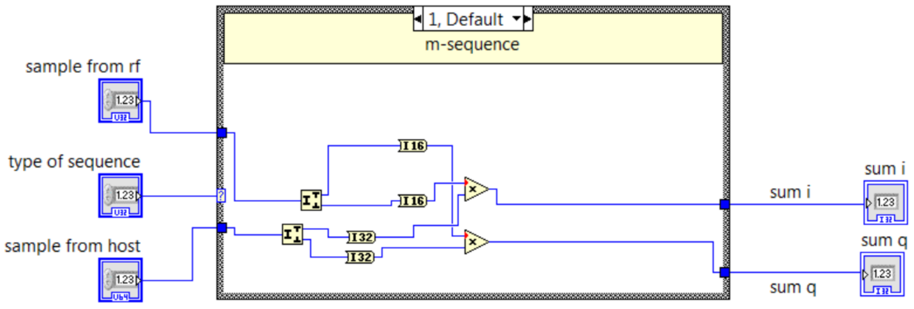
The circular cross-correlation is performed in two SCTL, one for the samples at RF0 and one for the samples at RF1. The implementation of the two SCTLs are nearly identical, they only differ in which FIFO they read the received samples from. Each of the SCTLs circularly cross-correlates the received samples with both of the transmitted sequences, s_0 and s_1, and they functions as follows.

For every iteration, one element from the FIFO containing the received samples is read as a 32-bit unsigned integer. As mentioned in section 4.1, the I- and Q-components are represented by one 16-bit integer respectively, so the integer from the FIFO is split up into I- and Q-components as 16-bit integers. These are multiplied with the corresponding element in s_0 and s_1. The way that the multiplication is done differs depending on whether the sequence used is an m-sequence or a Zadoff-Chu sequence. In the case of an m-sequence the I- and

Q-component is simply multiplied with the corresponding element in s_0 and s_1 as seen in Figure 4.3a. In the case of the Zadoff-Chu sequence the I- and Q-component are treated as the real and imaginary part of a complex number, which is multiplied with the conjugate of the corresponding element in s_0 and s_1 as can be seen in Figure 4.3a.



(a) Multiplication for Zadoff-Chu sequence



(b) Multiplication for m-sequence

Figure 4.3: The multiplication for different sequence types

The resulting I-component is added to the sums of I-components from the previous iteration, as described in section 4.1 and the resulting Q-component is added to the sums of Q-components in the same manner. If the calculations for one value of τ is completed, the sums for the I- and Q-components are merged into a 64-bit integer and written to the host using a host-to-target FIFO, and set to zero for the next loop iteration. A flow chart of the implementation of circular cross-correlation with one sequence $\mathbf{s}$ can be seen in Figure 4.4. Each correlator loop implements Figure 4.4 for both sequences s_0 and s_1 in parallel.

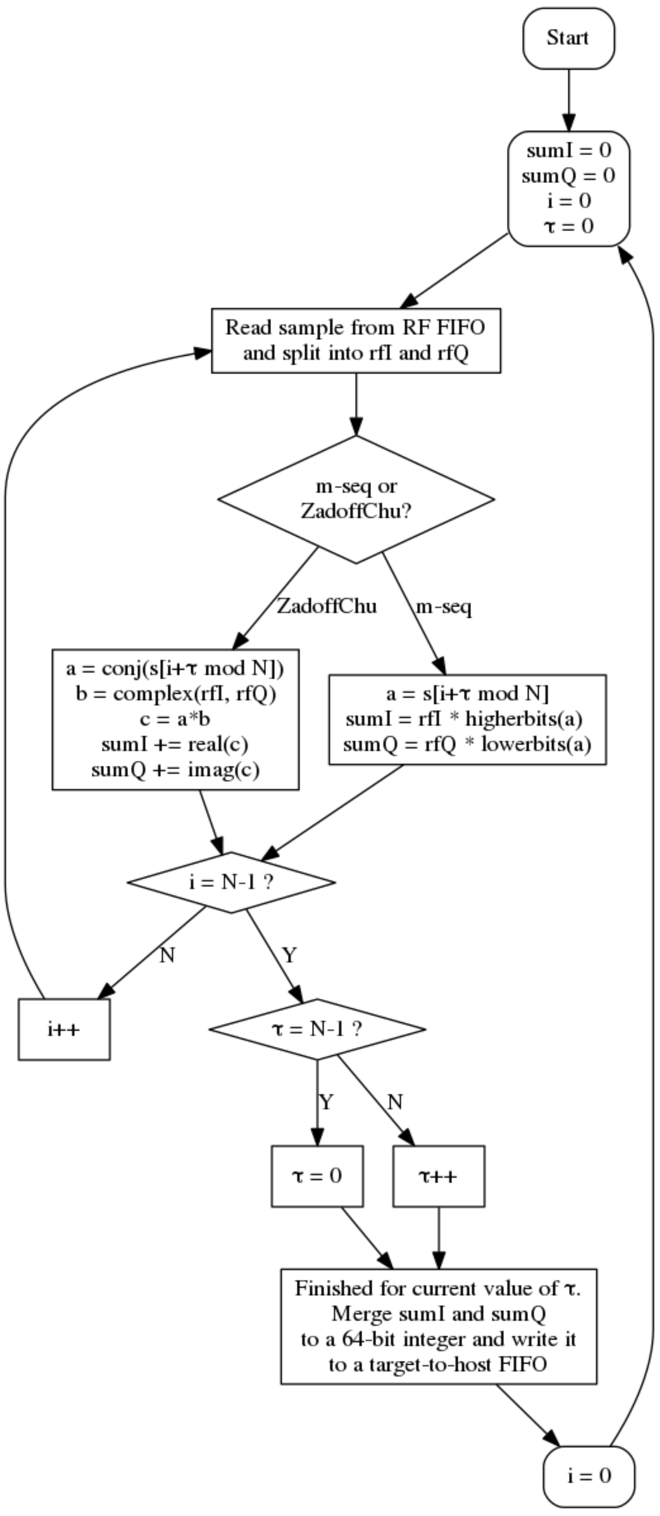
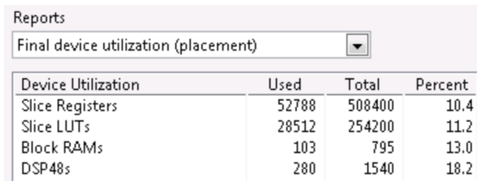


Figure 4.4: A flow chart describing the loops performing circular cross-correlation, where **s** is the sequence stored on the FPGA

4.2.2 Resource utilization and scaling

Figure 4.5a shows the device utilization for the original "Streaming (Xcvr)" VI, and Figure 4.5b show the device utilization with our additional processing. As can be seen, the device utilization has increased slightly, but lack of device resources is no issue.

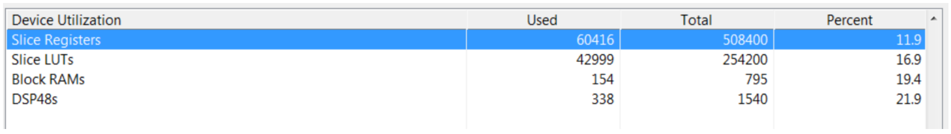
Figure 4.5



Reports
Final device utilization (placement)

Device Utilization	Used	Total	Percent
Slice Registers	52788	508400	10.4
Slice LUTs	28512	254200	11.2
Block RAMs	103	795	13.0
DSP48s	280	1540	18.2

(a) Device utilization of the original "Streaming Xcvr (FPGA)" VI



Device Utilization	Used	Total	Percent
Slice Registers	60416	508400	11.9
Slice LUTs	42999	254200	16.9
Block RAMs	154	795	19.4
DSP48s	338	1540	21.9

(b) Estimated device utilization after the correlation were added

One of the most notable increase is from 11.2% to 16.9% for LUTs (Lookup tables). The reason for this is that the memories that is used to store the sequences s_0 and s_1 on the FPGA are implemented in Lookup tables, meaning that they also consume resources that could be used for programming logic[25]. This could pose a problem since an increase in the number of sequences used also leads to an increase in the number of memories needed by one memory block per correlator loop. It could also be a problem if the sequence lengths increases dramatcially, since that would require larger memories. One way to remedy this is to implement the memories in block memory instead of Lookup tables, since block memory does not use programming logic for storage[25].

Since every correlator loop implements Figure 4.4 once per transmitted sequence, the number of performed operations, Figure 4.3b and Figure 4.3a, between the received sample and the corresponding elements in the stored sequences s_i will increase with linearly with an increased number of transmitted sequences. This will cause problems as the number of sequences used increases, as the number of DSP slices used for operations such as multiplication and addition (listed as "DSP48s" in Figure 4.5) is limited.

The issue of storing the sequences on the FPGA could possibly be circumvented by letting the FPGA itself generate the elements of the sequences. The generation of m-sequences is well suited for implementing on an FPGA, since the operations required are quite simple. It might prove more difficult to generate a Zadoff-Chu sequence, since the there are no floating point numbers and no exponential function e^x available in LabVIEW FPGA.

Chapter 5

Validation of the implementation

5.1 2-path channel

In order to verify that the implementation is correct, a simulated 2-path channel is created. This is done using two USRP:s, one as transmitter and one as a receiver. First, the devices transfer function is calculated and stored, then one of the transmit ports on the transmitting USRP is connected to a power splitter, and from the power splitter, a 0.5 m cable is connected to a power combiner, and between the remaining ports on the power splitter and power combiner, a approximately 30 m long cable is connected. The power combiner is then connected to the receiving USRP, and an m-sequence is transmitted. The 2-path channel is sounded by cross-correlating the received signal with transmitted m-sequence, on the FPGA, and the result is saved to the hard drive, after the device transfer function has been removed from the result.

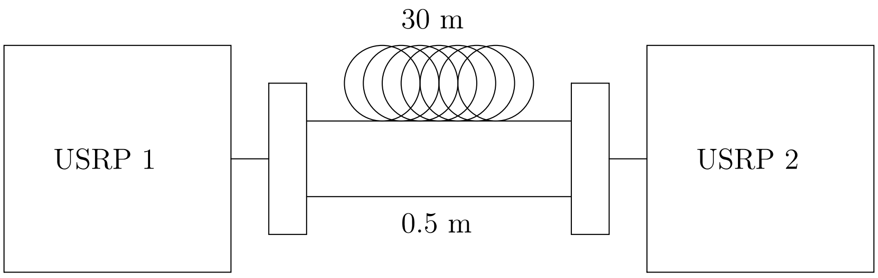
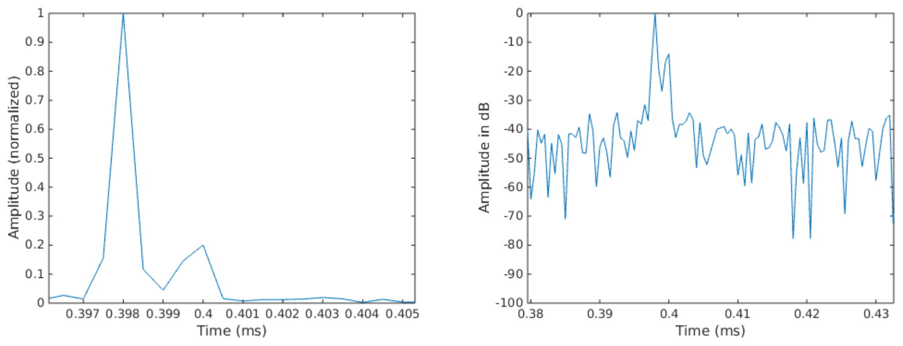


Figure 5.1: The simulated 2-path channel setup

The sequence transmitted is an m-sequence of length 1023, the sample rate is 20 MHz, and the carrier frequency is 1.5 GHz. The result of the sounding can be seen in Figure 5.2, and as can be seen there are two peaks. The first, and taller, one is from the short path and the weaker, second one is from the longer path. The reason why the second peak is weaker is due to the fact that the longer cable attenuates the signal approximately 13 dB as can be seen from Figure 5.2. The



(a) Normalized amplitude **(b)** Normalized amplitude (dB)

Figure 5.2: Snapshot of the estimated impulse response for a 2-path channel, using a m-sequence of period 1023

difference in length between the cables, l, can be approximated by

$$l = v_p \times \frac{k}{f_s} \tag{5.1}$$

where v_p is the propagation speed of the signal in the cable, f_s is the sample rate, and k is the number of samples the second peak is delayed in relation to the first peak. The propagation velocity v_p is between 69 percent and 80 percent of the speed of light depending on the material in the cable [26].

By looking at Figure 5.2 we can see that $k = 4$. This means that the difference in length of the cables can be approximated to

$$l = 0.69 \times c \times \frac{4}{20 \times 10^6} \approx 41m \tag{5.2}$$

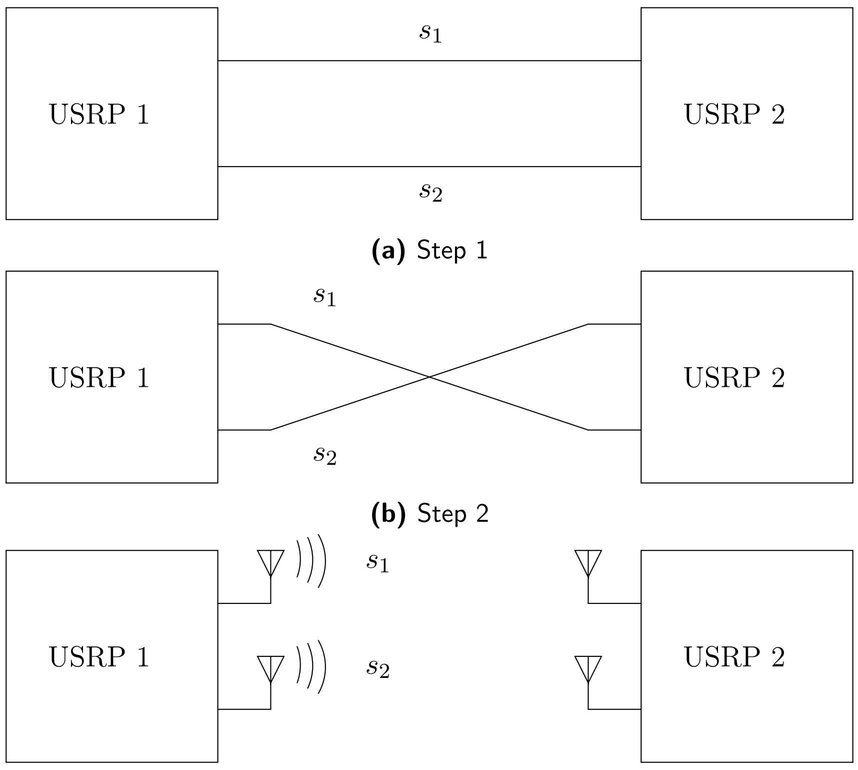
If we take into account the delay resolution $\Delta\tau = \frac{1}{BW}$, the ambiguity of (5.2) becomes

$$\Delta l = \frac{1}{BW} v_p = \frac{1}{20 \times 10^6} \times 0.69c = 10.34 \text{ m} \tag{5.3}$$

5.2 2x2 MIMO

In this example, a practical example of the MIMO channel sounding described in section 2.5 is performed for the 2x2 MIMO case. The setup is using two USRPs, one for transmitting and one for receiving. The sounding process is as follows, and visualized in Figure 5.3

1. Cables are connected from RF0 and RF1 on the transmitter, to RF0 and RF0 on the receiver, respectively. The sequence s_1 is transmitted from RF0, and s_2 is transmitted from RF1. The receiver performs circular cross-correlation on each received signal with the transmitted sequence, obtaining



(a) Step 1

(b) Step 2

(c) Step 3

Figure 5.3: Visualization of the MIMO channel sounding process

$h_{d11}(t)$ and $h_{d22}(t)$ as described in section 2.5 and the results are saved to the hard drive of the host computer.

2. The connected cables are crossed, so that RF0 is connected to RF1, and RF1 is connected to RF0. s_1 and s_2 is the transmitted, and the receiver performs circular cross-correlation as previously, obtaining $h_{d12}(t)$ and $h_{d21}(t)$.

3. The cables are disconnected from one end, and antennas are connected on the loose ends of the cables, and the antenna ports. s_1 and s_2 is transmitted from RF0 and RF1, respectively, and the receiver performs circular cross-correlation on the FPGA, for each received signal with both s_1 and s_2 as described in (2.26).

In these three stages, it is important that the USRPs is not power cycled between the steps, which would result in the transfer function of the system changing. To counter this, the sounder was implemented to be able to do multiple soundings without needing a restart.

Measurements are made using both Zadoff-Chu sequences and m-sequences. In the case of Zadoff-Chu sequences the sequence parameters is $k_1 = 23$, $N_1 = 1023$,

$k_2 = 37$, and $N_2 = 1023$. For the m-sequences the length $N = 1023$ is used, and s_2 is shifted 511 samples in relation to s_1. In each case, $N \times 5 = 5115$ samples are saved.

In Figure 5.4a and Figure 5.5a the estimated channel impulse response $\hat{h}_{11}^{chan}(t,\tau)$ is presented, and it is obtained in the following way. The first 1023 samples from the systems impulse response $h_{d11}(t)$ is removed from the impulse response $h_{11}^{tot}(t,\tau)$ obtained in step 3 in Figure 5.3 by deconvolution according to

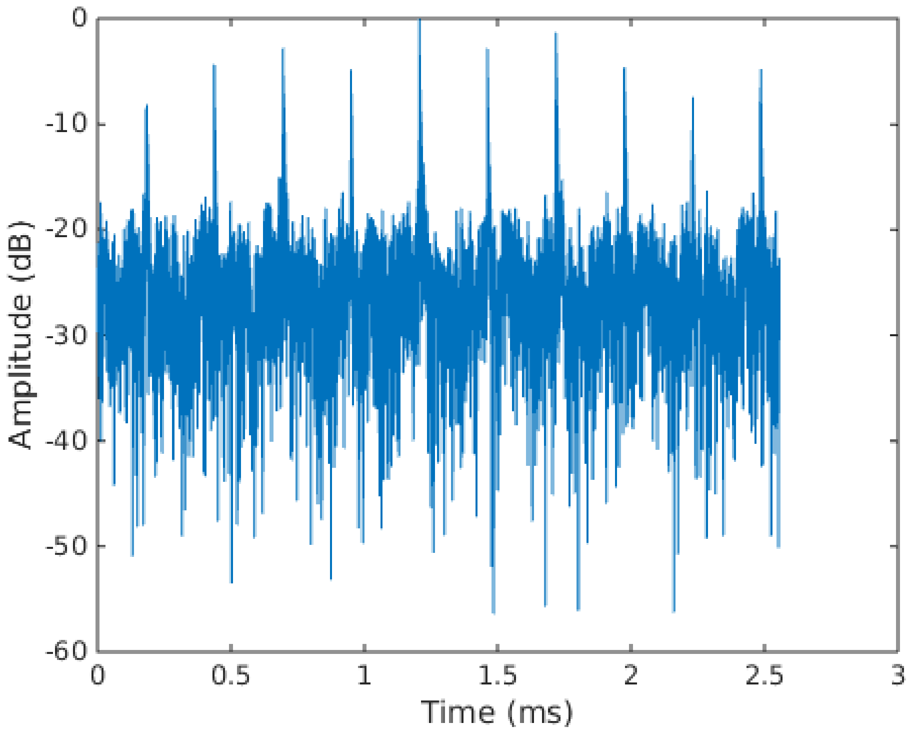
$$\hat{h}_{11}^{chan}(t,\tau) = \mathcal{F}^{-1}\left[\frac{\mathcal{F}[h_{11}^{tot}(t,\tau)]}{\mathcal{F}[h_{11}^{d}(t)]}\right]. \tag{5.4}$$

The following $4 * 1023$ samples are treated in the same way, creating the graphs seen in Figure 5.4a and Figure 5.5a.

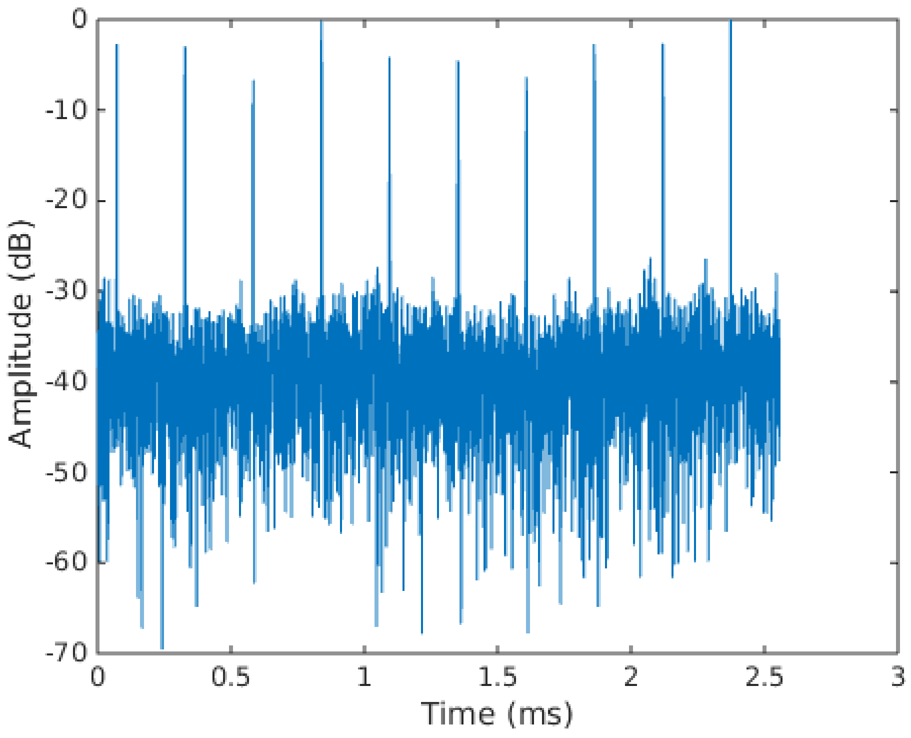
For the m-sequences, as explained in subsection 2.4.2, the expected results are two peaks for each $N = 1023$ number of received samples with one peak 511 samples after the first peak. This is due to the sequence transmitted from the second antenna is delayed 511 samples in relation to that transmitted from the first antenna, as well as the cross-correlation properties of m-sequences explained in subsection 2.4.2. Our estimated channel impulse response when using m-sequences can be seen in Figure 5.4a, where there are two peaks per N samples. Since we capture $5 \times N = 5115$ samples, the plot shows 10 peaks. By analyzing the plots in MATLAB we are able to see that the peaks are 511 samples apart.

For the Zadoff-Chu sequences the expected result is one peak per N samples, since the transmitted sequences are two different Zadoff-Chu sequences. The peak-to-average ratio of the cross-correlation of two different Zadoff-Chu sequences is $\frac{1}{\sqrt{N}}$ as seen in subsection 2.4.1. As seen, our estimated channel impulse response when using Zadoff-Chu sequences presented in Figure 5.5a matches the expected result of one peak per N samples, as we capture $N * 5 = 5115$ samples.

Figure 5.4b and Figure 5.5b shows the estimated channel impulse response $\hat{h}_{11}^{chan}$ obtained by circularly cross-correlating the systems estimated impulse response with the estimated total impulse response. As described in section 2.5, deconvolution and circular cross-correlation yields the same result when the sequence is a Kronecker delta. Since our channel is subjected to noise and interference between the transmitted sequence, the estimated channel impulse response in Figure 5.4b and Figure 5.5b will not be of the same quality as when deconvolution is used. When the channel impulse response is estimated using m-sequences, the result matches the result obtained from the deconvolution fairly well. We attribute this to the fact that the interference between the two m-sequences is zero at all points except for 511 samples after the initial peak. For Zadoff-Chu sequences, the interference between the two transmitted sequences is constant at a ratio of $\frac{1}{\sqrt{N}}$, making the channel impulse response that is estimated using circular cross-correlation look quite from that estimated using deconvolution. As can be seen from Figure 5.5b and Figure 5.5a, while Figure 5.5b do contain the expected 5 spikes it also contains e.g. one unexplained -30 dB component.

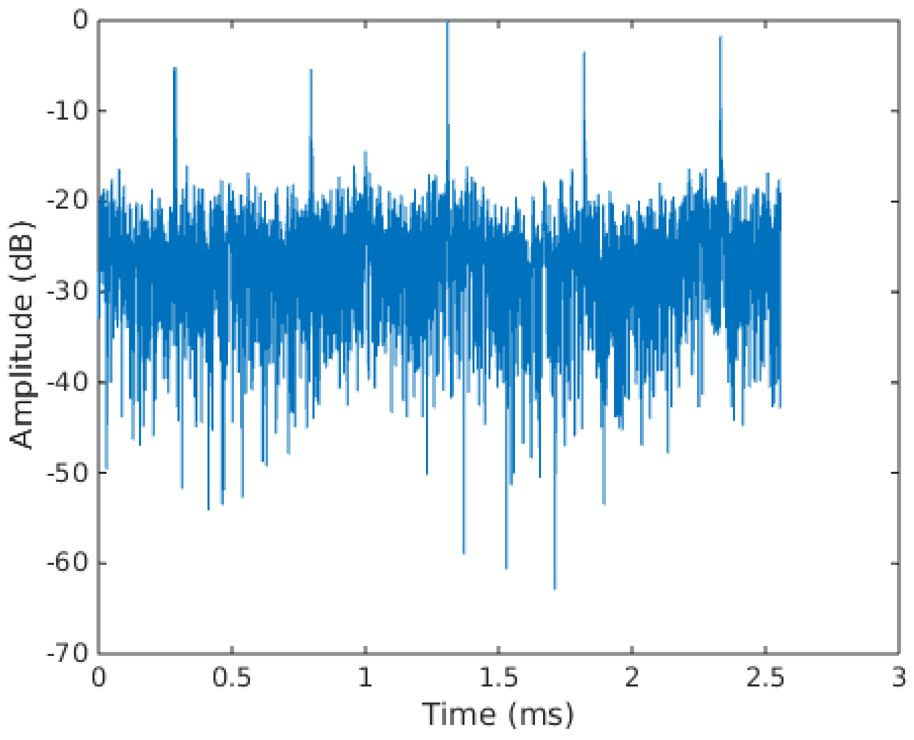


(a) The estimated channel impulse response with the systems impulse response removed by deconvolution

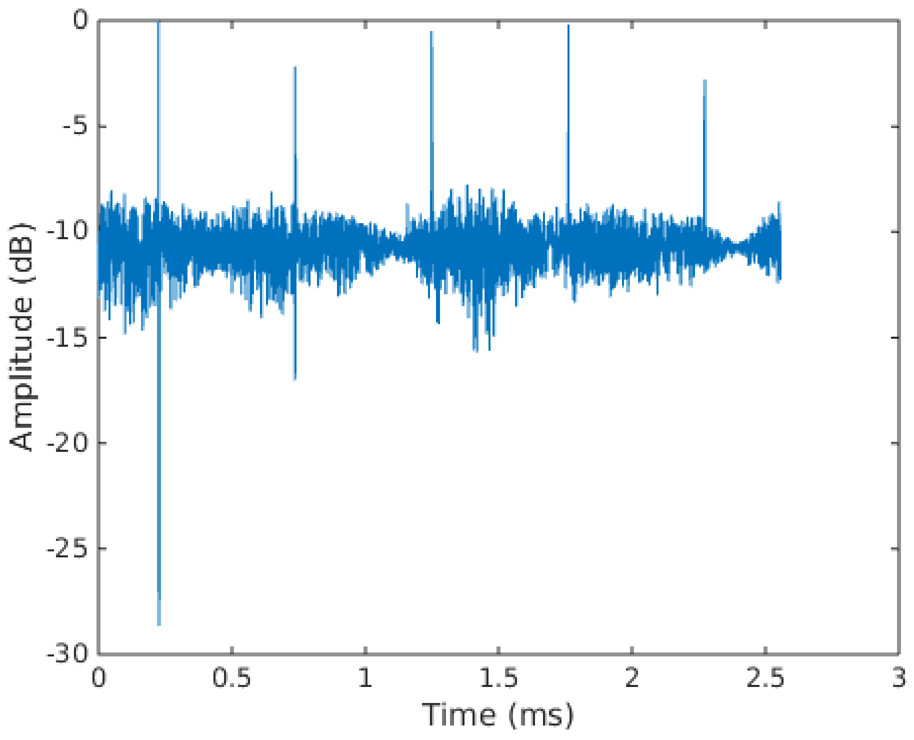


(b) The estimated channel impulse response with the systems impulse response removed by cross-correlation

Figure 5.4: MIMO sounding using a m-sequence of period 1023



(a) The estimated channel impulse response with the systems impulse response removed by deconvolution



(b) The estimated channel impulse response with the systems impulse response removed by cross-correlation

Figure 5.5: MIMO sounding using a Zadoff-Chu sequence with $N = 1023$, $k_1 = 23$, $k_2 = 37$

Chapter 6

Conclusion

In this thesis, a 2x2 MIMO channel sounder implemented in LabVIEW FPGA have been proposed and evaluated. A few different approaches were tried during the development, in order to make the FPGA implementation possible. The channel sounder was implemented for two different types of sequences, Zadoff-Chu sequences and m-sequences. This was useful both for evaluating the sounder itself and also to evaluate and compare the sequences themselves, and their channel sounding properties, but required a more advanced implementation to work on the USRP. In the end, the sounder produced a consistent result, using deconvolution, for both Zadoff-Chu and m-sequences as can be seen in Figure 5.5a and Figure 5.4a respectively.

Overall, both sequences resembles the expected results, the floor of the Zadoff-Chu sequence in Figure 5.5a is slightly higher than that of the m-sequence in Figure 5.4a and this can most likely be attributed to that in addition to the noise, the Zadoff-Chu sequence transmitted from one antenna also suffers from interference from the Zadoff-Chu sequence transmitted from the other antenna with a factor of $\frac{1}{\sqrt{N}}$ as mentioned in subsection 2.4.1. With the throughput and accuracy of the sounder, along with the implementation of both sequences, the goals stated in section 1.2 were eventually reached.

6.1 Future work

This thesis implements a relatively basic MIMO channel sounding and has room for improvement and expansion. Currently, few antennas and few sequences are used and an interesting continuation would be to significantly increase the number of transmit antennas and the number of different sequences, especially with regards to Zadoff-Chu sequences. Both to see how the FPGA implementation handles the increased demands, and also how the Zadoff-Chu sequences performs as the number of sequences increases. Adding the capability of cycling between transmitting and receiving automatically would make for a more flexible channel sounder. In addition, GPS could be used to synchronize the local oscillators of multiple USRP:s, as well for deciding when one USRP should transmit and receive.

In terms of sequences used for the sounding, there are more than the m- and Zadoff-Chu sequences that was not used at all in this thesis. Gold codes and Kasami sequences could have been evaluated and compared as well. As for the hardware, the sounding in this thesis was carried out over short range and with stationary USRPs. The sounding could be also performed over longer distances with stronger antennas and the USRPs placed on moving vehicles. Further, the FPGA application could be extended to remove the system transfer function from the total transfer function, something which is currently done on the host computer.

6.2 Issues encountered

A Zadoff-Chu sequence consist of complex numbers where the real and imaginary components are floating point numbers are bounded by $[-1, 1]$. Since LabVIEW FPGA doesn't support floating point number, one need to be careful when converting the Zadoff-Chu sequence to integers, so that no precision is lost, as that will influence the auto- and cross-correlation properties of the sequence. It is unclear if the conversion really is the source of the errors we experienced, since the error should be very low for that conversion, so it is a possibility that the problems were due to a bug in our implementation.

Our initial LabVIEW FPGA implementation had some issue. The sequences was written to the FPGA in the same way as in chapter 4, and the received samples were written to FIFOs. These FIFOs were read from in another SCTL. In the SCTL, the received samples were buffered in an N-length array. In every iteration, the buffered array and the sequence were multiplied element wise, and the elements in the resulting array were added. In the next iteration, the element from the receiver was pushed onto the buffered array, and the last element discarded. There are several issues with this approach, which ultimately lead us in a different direction which resulted in the implementation described in chapter 4. Since LabVIEW FPGA doesn't support variable length arrays, the length of the sequence had to be hardcoded into the VI meaning that for each sequence length one individual bitfile would be needed. It is also troublesome for LabVIEW FPGA to handle large arrays. While the implementation worked for sequences of shorter length (eg. 63), it became problematic when moving to longer sequences, as in the synthesis step of the compilation the FPGA would run out of DSP-slices from performing all of the multiplications. The compile process also took quite a long time (a few hours). This, combined with our lack of experience with FPGA programming and LabVIEW FPGA, made for quite a slow development process.

Bibliography

[1] Qing Rao et al. “AR-IVI - Implementation of In-Vehicle Augmented Reality”. In: *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on* (2014).

[2] U.S. Department of Transportation. *Safercar*. 2015. URL: www.safercar.gov.

[3] Andreas F. Molisch. *Wireless Communications Second Edition*. Wiley, 2011.

[4] Gerald Matz and Franz Hlawatsch. *Fundamentals of Time-Varying Communication Channels*. Academic Press, 2011.

[5] Wikimedia Commons. *2 Ray Ground Reflection*. By Derekjc (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons. 2014. URL: https://upload.wikimedia.org/wikipedia/commons/9/95/2-Ray_Ground_Reflection.png.

[6] Franz Hlawatsch and Gerald Matz. *Wireless Communications over Time-Varying Channels*. Academic Press, 2011.

[7] ITU. *Mobile-cellular subscriptions*. 2014. URL: https://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2015/Mobile_cellular_2000-2014.xls.

[8] Nelson Costa and Simon Haykin. *Multiple-Input Multiple-Output Channel Models, Theory and Practice*. Wiley, 2010.

[9] Sana Salous. *Radio Propagation Measurement and Channel Modelling*. Wiley, 2013.

[10] Gerald Matz et. al. “Bounds on the systematic measurement errors of channel sounders for time-varying mobile radio channels”. In: *Vehicular Technology Conference, 1999. VTC 1999 - Fall. IEEE VTS 50th* (1999).

[11] Solomon W. Golomb and Guang Gong. *Signal Design for Good Correlation.* Cambridge University Press, 2005.

[12] Abhijit Mitra. "On the Construction of m-Sequences via Primitive Poynomials with a Fast Indeitification Method". In: *International Journal of Electrical and Computer Engineering* (2008).

[13] New Wave Instruments. *Linear Feedback Shift Registers.* 2010. URL: http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm.

[14] Andreas Klein. *Stream Ciphers.* Springer, 2013.

[15] National Instruments. *Prototyping Next Generation Wireless Systems with Software Defined Radios.* 2015. URL: http://www.ni.com/white-paper/14297/en/.

[16] National Instruments. *DEVICE SPECIFICATIONS NI USRP-2953R.* 2014. URL: http://www.ni.com/pdf/manuals/374197a.pdf.

[17] Ehab Mohsen. *Balancing Performance, Power, and Cost with Kintex-7 FPGAs.* 2013. URL: http://www.xilinx.com/support/documentation/white_papers/wp432-K7-Perf-Power-Cost.pdf.

[18] National Instruments. *NI USRP-2953R Block Diagram.* Image courtesy of National Instruments and is not to be further distributed without their consent. 2015. URL: http://zone.ni.com/reference/en-XX/help/373380D-01/usrphelp/2953_block_diagram/.

[19] Xilinx. *7 Series DSP48E1 Slice user guide.* 2015. URL: http://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf.

[20] National Instruments. *LabVIEW Environment Basics.* 2015. URL: https://www.ni.com/getting-started/labview-basics/environment.

[21] National Instruments. *Dataflow Programming Basics.* 2015. URL: http://www.ni.com/getting-started/labview-basics/dataflow#Wires.

[22] National Instruments. *Unsupported LabVIEW Features (FPGA Module).* 2015. URL: http://zone.ni.com/reference/en-XX/help/371599L-01/lvfpgaconcepts/fpgamisc/.

[23] National Instruments. *Single-Cycle Timed Loop FAQ for the LabVIEW FPGA Module.* 2015. URL: http://digital.ni.com/public.nsf/allkb/722A9451AE4E23A586257212007DC5FD.

[24] National Instruments. *Implementing Multiple Clock Domains (FPGA Module)*. 2012. URL: http://zone.ni.com/reference/en-XX/help/371599H-01/lvfpgaconcepts/implementing_domains/.

[25] National Instruments. *Storing Data on an FPGA Target (FPGA Module)*. 2012. URL: http://zone.ni.com/reference/en-XX/help/371599H-01/lvfpgaconcepts/fpga_storing_data/.

[26] Axon' Cables. *COAXIAL CABLES*. 2015. URL: http://www.axon-cable.com/publications/Coaxial-cables.pdf.

Notes