Vesa Lampu

REAL-TIME FPGA IMPLEMENTATION OF A DIGITAL SELF-INTERFERENCE CANCELLER IN AN INBAND FULL-DUPLEX TRANSCEIVER

Faculty of Information Technology and Communication Sciences
Master of Science Thesis
September 2019

ABSTRACT

VESA LAMPU: Real-time FPGA Implementation of a Digital Self-interference Canceller in an Inband Full-duplex Transceiver
Tampere University of technology
Master of Science Thesis, 55 pages, 11 Appendix pages
September 2019
Master's Degree Programme in Electrical Engineering
Major: Electronics
Examiners: D. Sc. (Tech) Lauri Anttila, Prof. Mikko Valkama

Keywords: real-time, full-duplex, nonlinear, FPGA, self-interference, canceller, digital

Full-duplex is a communications engineering scheme that allows a single device to transmit and receive at the same time, using the same frequency for both tasks. Compared to traditionally used half-duplex, where the transmission and reception is divided temporally or spectrally, the spectral efficiency may theoretically be doubled in full-duplex operation. However, the technology suffers from a profound problem, namely the self-interference (SI) signal, which is the name given to the signal a node transmits and simultaneously also receives. Making the full-duplex technology feasible demands that the SI signal is mitigated with SI cancellers. Such cancellers reconstruct an estimate of the SI signal and subtract the estimate from the received signal, thus suppressing the SI. For the SI signal to be diminished as much as possible, canceller solutions should be deployed in both analog and digital domains. This thesis presents a digital real-time implementation of a novel nonlinear self-interference canceller, based on splines interpolation. This canceller utilizes a Hammerstein model to identify the SI signal, taking advantage of a FIR filter for the identification of the SI channel, and splines interpolation to model the nonlinear effects of the transceiver circuitry. The new canceller solution promises great reduction in computational complexity compared to traditional algorithms with little to no sacrifice in cancellation performance.

The algorithm was implemented for a National Instruments USRP SDR device using LabVIEW Communications System Design Suite 2.0. The LabVIEW program provides the required connectivity to the USRP platform, as the SDR lacks a user interface. In addition, the functionality of the SDR is determined in LabVIEW, by creating code that is then run on the USRP, or more specifically, on the built-in FPGA of the device. The FPGA is where the SI canceller is executed, in order to ensure real-time operation. Even though the USRP device employs a high-end FPGA with plenty of resources, the canceller implementation needs to be simplified nonetheless, for example by approximating magnitudes of complex values and by decreasing the sample rate of the canceller. With the simplifications, the implementation utilizes only 34.9 % of available slices on the FPGA and only 34.6 % of the DSP units. Measurements with the canceller show that it is capable of SI cancellation of up to 48 dB, which is on par with state-of-the-art real-time SI cancellations in literature. Furthermore, it was demonstrated that the canceller is capable of bidirectional communication in various circumstances.

TIIVISTELMÄ

VESA LAMPU: Reaaliaikainen digitaalisen itseishäiriön vaimentajan FPGA toteutus full-duplex lähetin-vastaanottimessa
Tampereen teknillinen yliopisto
Diplomityö, 55 sivua, 11 liitesivua
Syyskuu 2019
Sähkötekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Elektroniikka
Tarkastajat: TkT Lauri Anttila, Prof. Mikko Valkama

Avainsanat: reaaliaikainen, full-duplex, epälineaarinen, FPGA, itseishäiriö, vaimennus, digitaalinen

Full-duplex on tietoliikenneratkaisu, jossa yksi laite lähettää ja vastaanottaa tietoa yhtäaikaisesti, käyttäen samaa taajuutta. Verrattuna yleisesti käytettyihin half-duplex-ratkaisuihin, joissa lähetys ja vastaanotto on jaettu joko ajan tai taajuuden suhteen, full-duplex-toiminnassa voidaan spektrillinen tehokkuus teoreettisesti jopa tuplata. Full-duplex teknologia kärsii kuitenkin hyvin merkityksellisestä ongelmasta, niin kutsutusta itseishäiriöstä, eli signaalista, jonka laite lähettäessään myös samanaikaisesti vastaanottaa. Full-duplex-toiminnan edellytyksenä on, että itseishäiriötä vähennetään vastaanotetusta signaalista merkittävästi, käyttäen itseishäiriön vaimentajia. Näiden vaimentimien tarkoitus on luoda mallin pohjalta itseishäiriösignaalista estimaatti, ja vähentää estimaatti vastaanotetusta signaalista, jolloin itseishäiriö vaimenee. Vaimennuksen maksimoimiseksi vaimentimia tulisi käyttää sekä analogisella, että digitaalisella puolella. Tässä työssä on esitetty uudenlaisen digitaalisen epälineaarisen itseishäiriön vaimenninratkaisun reaaliaikatoteutus. Tämä uusi vaimennin perustuu niin kutsuttuun spline-interpolointiin, hyödyntäen Hammerstein mallia, jossa häiriökanava mallinnetaan FIR-filtterin ja lähettimen epälineaarisuudet spline-interpoloinnin avulla. Tämän uuden vaimentimen on tarkoitus olla laskennallisesti huomattavasti kevyempi kuin aiemmat vastaavat algoritmit, samalla kuitenkin vaimentavan itseishäiriötä yhtä tehokkaasti.

Algoritmi toteutettiin National Instrumentsin USRP SDR alustalle käyttäen LabVIEW Communications System Design Suite 2.0:sta. LabVIEW tarjoaa rajapinnan USRP:n ja tietokoneen välille, sillä USRP-laitteessa ei ole käyttöliittymää. Lisäksi USRP:n toiminta määritetään LabVIEW:ssa kirjoittamalla koodi, joka ajetaan USRP:n sisältämällä FPGA:lla. Varsinainen itseishäiriönvaimennin ajetaan juuri FPGA:lla reaaliaikaisuuden takaamiseksi. Vaikkakin USRP:n sisältämä FPGA:lla on suuri määrä resursseja, on vaimennintoteutusta yksinkertaistettava, esimerkiksi estimoimalla kompleksiluvun magnitudin ja alentamalla vaimentimen näytetaajuutta. Yksinkertaistuksien kanssa vaimennin käyttää vain noin 34.9 % lohkoista ja vain 34.6 % DSP-yksiköistä. Mittaukset osoittivat, että vaimennin kykenee alentamaan itseishäiriötä jopa 48 dB, ollen näin alan huipputulosten tasolla. Lisäksi osoitettiin, että vaimentimen avulla on mahdollista viestiä kaksisuuntaisesti erilaisissa olosuhteissa.

PREFACE

As I am writing this, more than a year has passed since I started working on my thesis. The project this thesis is based on was started in February 2018, and it was successfully completed at the end of the same year. One could argue that the thesis could have been finished a long time ago, and they would probably be right, as I have indeed taken my time with it. Nevertheless, here I am writing the preface, and it is time to give credit where it is due.

First and foremost, I would like to thank my supervisor Lauri Anttila for his excellent guidance throughout the project and thesis. It has definitely been a privilege working with him, the amount of things I have learned this past year and a half is astounding. As my supervisor, he has also been a huge factor in making the atmosphere of the work environment enjoyable, which is something I am really grateful for. Of the senior scholars, I would also like to thank my thesis' other examiner Mikko Valkama and Dani Korpi for providing me with instructions and a nice place to work in.

I would like to give enormous thanks to my colleague Pablo Pascual Campo for helping me along the way, and giving me support whenever I needed it. He also kept pushing me to finish the thesis, so it is in part thanks to him that I am writing this now, instead of sometime in the future.

A special thankyou goes to one of my lecturers, Olli-Pekka Lundén, who sparked the interest towards RF engineering in me. He is also the one who suggested me for the position in the project. Additionally, Matias Turunen played a major role in the measurements, and I thank him for that. Seyed Ali Hassani of the KU Leuven University was of great help with the LabVIEW code, big thanks to him as well.

Outside the academia, I would of course like to thank my family and especially my parents Taina and Tomi for providing me with the circumstances to reach this point, not just during my studies, but also throughout my life. I know I do not say it often enough, but I do appreciate everything you have done for me. My last thankyous go to all of my friends, family members and colleagues who were not mentioned, but who made the last six years of my life as enjoyable as they have been.

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 732174 (ORCA, Extension #4).

Tampere, September 16, 2019

Vesa Lampu

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

1-D	One Dimensional
A/D	Analog-to-Digital
CPU	Central Processing Unit
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
D/A	Digital-to-Analog
dB	Decibel
dBm	Decibel-milliwatt
DSP	Digital Signal Processing
EBD	Electrical Balance Duplexer
FD	Full-duplex
FDD	Frequency Division Duplexing
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
GHz	Gigahertz
HD	Half-duplex
I	In-phase component
IF	Intermediate Frequency
IIR	Infinite Impulse Response
ISM	Industrial, Scientific and Medical
I/Q	In-phase/Quadrature
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
LMS	Least Mean-Squares
LO	Local Oscillator
LUT	Lookup Table
MHz	Megahertz
ns	Nanosecond
PA	Power Amplifier
OFDM	Orthogonal Frequency-Division Multiplexing
Q	Quadrature-phase component
QAM	Quadrature Amplitude Modulation
RAM	Random Access Memory
RC	Resistor and Capacitor
RF	Radio Frequency
RLS	Recursive Least-Squares
SDR	Software Defined Radio
SER	Symbol Error Ratio
SI	Self-Interference
SINR	Signal-to-Interference-plus-Noise Ratio
TDD	Time division duplexing
TX	Transmitter
UI	User Interface
USRP	Universal Software Radio Peripheral
VI	Virtual Instrument
$\mathbf{1}$	Array of all ones
A	Amplitude
a	Real part of a complex number
B	Amount of bits representing a value

b	Imaginary part of a complex number
B_c	Amount of bits in the filter coefficients
c	Complex number
C	Channel capacity
$\mathbf{C}$	Spline basis matrix
d	Desired signal
$\bar{d}$	Mean of desired signal
DC	DC-offset
E	Statistical expectation operator
e	Error signal
F	A discrete signal
f_c	Carrier frequency
f_s	Sampling frequency
G	A discrete signal
$\mathbf{h}$	Array representation of taps of a static FIR filter
h	Tap of a static FIR filter
$\hat{h}$	Channel estimate
h_p	Channel coefficients in the memory polynomial model
i	Spline knot index
J	Cost function
j	imaginary unit
K	Decimation factor
L	Interpolation factor
l	Lag introduced to a signal in finding the cross-correlation
M	Memory of a digital filter
M_{post}	Post-cursor taps of a digital filter
M_{pre}	Pre-cursor taps of a digital filter
N	Number of symbols transmitted and received
N_e	Amount of misinterpreted symbols
N_i^P	Spline basis function of order P
N_{pp}	Number of partial products
$\mathbf{p}$	Cross-correlation vector
P	Spline order
P_{mp}	Nonlinearity order of the memory polynomial model
Q	Number of spline control points
$\mathbf{q}$	Control points of the splines model
q^{im}	Imaginary part of complex spline control points
q^{re}	Real part of complex spline control points
$\mathbf{R}$	Autocorrelation matrix
S	Amount of samples in a signal
s	Spline interpolated signal
s_i	Ideal symbol
s_r	Received symbol
$\mathbf{s}$	Vector containing past spline interpolated signals
$t_{x,i}$	Spline knot on the x-axis
u	Spline abscissa value
$\mathbf{w}$	Impulse response of an adaptive filter
$\mathbf{X}$	Diagonal matrix containing past values of x
$\mathbf{x}$	Array representation of FIR filter inputs
x	Input signal for FIR filter/SI canceller

x_e	Complex envelope of a signal
x^{SI}	SI model output in the memory polynomial model
x_c	Carrier wave
x_i	In-phase component of x_e
x_q	Quadrature component of x_e
xw	Element of array $\mathbf{X}^*$(n)$\mathbf{w}$(n)
y	Output of the FIR filter/Hammerstein model
$\tilde{z}$	Memory polynomial model imperfection
α	Term used for complex value magnitude approximation
β	Term used for complex value magnitude approximation
γ	Coefficient dependent on rounding mode
Δx	Distance between spline knots
δ	Weight value used for DC-offset determination
θ	Phase of signal
λ	Small term to be added for leakage factor
λ_{max}	Maximum eigenvalue of autocorrelation matrix
μ	Step-size of adaptive FIR filter algorithm
μ_q	Step-size of spline control point update
μ_w	Step-size of linear filter coefficient update
$\mathbf{\Sigma}$	Matrix collecting M past vectors $\mathbf{\Psi}$ as columns
Σ	Element of matrix $\mathbf{\Sigma}$
σ	Error from quantization for a given value
σ_d	Expected value of the mean square of the desired signal
σ_e	Error caused by quantization for signal e
σ_n	Error caused by quantization for noise
σ_x	Error caused by quantization for signal x
φ	Spline interpolation curve
$\mathbf{\Psi}$	Array containing the spline basis function indexed at i
ψ	Basis function in the memory polynomial model

1. INTRODUCTION

Full-duplex is a communications engineering scheme where a single device both transmits and receives information at the same time, at the same frequency. Practically all modern consumer devices nowadays are half-duplex devices, in which transmission and reception are divided by either time or frequency [16, pp. 453—454]. Traditionally, full-duplex technology has been seen very difficult to implement, due to the so-called self-interference signal [10]. The signal that a full-duplex device transmits, it naturally also receives, and this received signal is called the self-interference signal. The self-interference is orders of magnitude greater in power than the actual desired signals transmitted from other devices, and thus in order to utilize full-duplex operation, the self-interference signal has to be mitigated [28]. This mitigation has long been the bottleneck for the commercialization of full-duplex devices, however, in recent years leaps in the technology have been made. With full-duplex technology, spectral efficiency can be theoretically doubled compared to traditional half-duplex schemes, which is a significant matter considering the low amount of free frequency channels. Full-duplex can also, among other things, solve the well-known hidden node problem [40]. The upsides of full-duplex make it an attractive choice for future communication systems.

The cancellation of the self-interference signal can be divided into two broad categories: analog and digital cancellation [25]. As the names suggest, the analog implementations function in the analog domain, and the digital ones in the digital domain. On their own, neither implementation is sufficient in the cancellation of the self-interference. Thus, for the technology to be feasible, both are needed to achieve sufficient cancellation. Duplexers can be counted as analog self-interference cancellers, even though strictly speaking they do not cancel the self-interference, but rather just add isolation between the transmitter and receiver chains. In addition to duplexers, actual self-interference cancellers can be implemented in the analog domain. In these implementations, an estimate of the received signal is formed from the transmit one, taking into account the channel effects on the signal, and the estimate is subtracted from the received signal. The digital cancellers function on the same principle, but in the digital domain.

A digital self-interference canceller can be implemented in many environments. If the goal is to make the system real-time, an option is to implement the algorithm on an FPGA, as is the case in this work. An FPGA, short for Field-programmable Gate Array, is a reprogrammable microchip that can be programmed to execute a wanted algorithm. The difference between an FPGA and for example the central processing unit (CPU) of a computer is that the program is translated into physical signals that propagate on the FPGA in parallel. The logical operations are achieved by generic logic gates that modern FPGA have in the order of tens of thousands. In addition to an FPGA, a full-duplex device needs a transceiver. For this purpose, National Instruments' USRP device is utilized. The USRP (Universal Software Radio Peripheral) is a software defined radio device, the functionality of which can be programmatically defined. The USRP has a transceiver circuitry in addition to an FPGA where the actual logic is executed. The USRP is controlled in LabVIEW Communications 2.0 environment. LabVIEW is a graphical programming language developed by National Instruments, in which the functionality is defined with functional blocks and wires connecting the blocks, rather than text as in traditional programming languages. LabVIEW Communications 2.0 is a standalone piece of LabVIEW software intended to be utilized in communications applications. Two separate codes are made in LabVIEW Communications 2.0: a target code that defines the functionality of the FPGA and the USRP, and the host code, which collects and visualizes data from the USRP and controls the target code. The host code is executed on a computer in LabVIEW Communication 2.0. The target code is also made in the same program, but it is translated into a bit file, which is uploaded onto the USRP when the host code is run.

The implemented digital self-interference canceller is based on a novel spline-based Hammerstein model, introduced in [47]. The Hammerstein model is a cascaded discrete-time model, where a linear part follows a nonlinear one [41]. This structure emulates the modelled circuitry fairly well, with the nonlinear power amplifier (PA) followed by the linear channel. The most notable source of nonlinearity is the transmitter PA [29], which is driven close to the saturation point in order to maximize efficiency. By doing so, the amplifier causes noticeable nonlinear distortion on the transmit signal. In this implementation, the nonlinear behavior is modelled using splines. Splines are piecewise defined polynomial approximations, with certain continuity constraints between the pieces [3, p. 11]. After the nonlinear power amplifier, the signal propagates through the channel, which is modelled using a finite impulse response (FIR) filter. The response is controlled with the filter coefficients. The output of the Hammerstein model is supposed to correspond to the actual self-interference signal as closely as possible. The output is subtracted from the received signal, giving the error signal as a result. Both the splines and the linear filter are adaptive, and the coefficients of both are controlled with the information from the error signal. This algorithm is considerably lighter in terms of calculations required than previous algorithms, which utilize memory polynomial models [47, 54].

The algorithm is implemented in LabVIEW Communications 2.0, as a part of an existing transceiver code. The native frequency of the used USRP device is 120 MHz, and the transceiver code runs at that frequency as well. Even though the self-interference canceller is relatively light in computation, it still cannot be executed entirely within one clock cycle at that frequency. Furthermore, 120 MHz restricts the amount of operations that can be done in one clock cycle, therefore the algorithm was decided to run at 60 MHz. The original algorithm still cannot be executed completely even at this lower rate, but the timing constraints are easier to handle. The timing of the algorithm is done by means of pipelining, which means that chosen intermediate values are stored in the FPGA's registers. The algorithm has seven pipelining stages in total, therefore it takes seven clock cycles for certain inputs to produce the corresponding output.

The system was tested by transmitting OFDM signals, which is a widely used waveform in telecommunications. The amount of cancellation the algorithm provides was tested in two separate scenarios. In the first, the transmitter and receiver chains had their own antennas to isolate the chains. In the other scenario, a radio frequency (RF) canceller was utilized and the chains shared an antenna. Both scenarios were measured at multiple different transmit powers. The digital canceller is capable of providing up to around 48 dB of digital cancellation, when the system utilizes OFDM signals, with 10 MHz bandwidth, transmit at 2.4 GHz. With the RF canceller, the system is able to reduce the self-interference level close to the device's noise floor. In addition to these tests, the two setups were tested in a real bidirectional full-duplex scenario. It turns out that the digital canceller may be utilized in an actual communication device, as the information the other node sends is decipherable, since the noise added by the full-duplex operation is manageable. Thus, the algorithm may someday be used in an actual consumer device.

2. RADIO COMMUNICATIONS SYSTEM BASICS

This chapter delves into the basic theoretical information needed to understand the functionality of communications systems. In order to modify the signal digitally, the signal has to be represented in a discrete form. This means that the signal has to be sampled, which means taking a sample of the signal at predetermined intervals or at sampling frequency. It turns out that every sample of every signal can be represented with just two values, the in-phase (I) and quadrature (Q) components, and they can be joined together to form a complex valued number. The I- and Q-branches of the signal are at baseband, but the I/Q-modulator, which is responsible for the generation of the transmit signal, transforms the signals to passband centered on the carrier frequency [39, p. 3]. In the digital domain, these values, or samples, present the continuous-time signal for a preset time, determined by the sampling frequency. The signal can be processed in the digital domain for example, to extract information from it, or, as it is the case in this thesis, to cancel the self-interference signal from it.

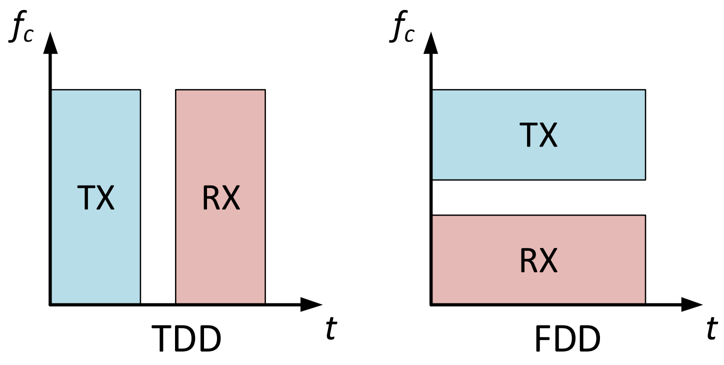


Figure 1. *Visual representation of TDD and FDD half-duplex schemes.*

There are numerous ways to establish a link between two radio devices. The simplest might be simplex, which means that one radio acts as a transmitter and the other as a receiver at all times. A bit more advanced technology is half-duplex, where both radios can act as both transmitter and receiver. The transmission and reception in half-duplex devices can be divided temporally or spectrally. Temporal or time division duplexing (TDD) allows the devices only to operate as a transmitter or a receiver at a time, while in spectral division or frequency division duplexing (FDD), a device acts as both transmitter and receiver at the same time, but in different frequency bands [20, p. 30]. Both TDD and FDD are conceptually presented in Figure 1. Taking the duplexing idea even further, in a full-duplex system both devices act as both transmitter and receiver simultaneously using the same frequencies. Using the same temporal and spectral resources allows more efficient data transmission, theoretically even doubling the spectral efficiency. However, full-duplex gives rise to a problem known as the self-interference signal, which has to be minimized [10, 25, 28, 47]. The end of this chapter considers some benefits of the full-duplex scheme.

2.1 In-phase and Quadrature Signals

A communications system consists of a transmitter and a receiver at its simplest form. Between the two there is a path known as *channel* through which the signal propagates from transmitter to receiver. The channel can be a wire or just air or free space, as is the case with wireless communication [58, pp. 762—763].

In order to convey information through the channel, a signal is needed. This signal, the *carrier,* can be an analog sinusoidal signal for example, although there are other options as well. The carrier on its own will not carry any information, but the information can be encoded to the carrier with *modulation*. In addition to allowing the information to be included to the carrier, modulation also allows the signal to be shifted in frequency, which is desired especially in wireless communications. Three basic ways of analogue modulation are amplitude-, frequency-, and phase-modulation which affect the amplitude, frequency and phase of the carrier respectively [58, p. 767].

Let us consider an amplitude and phase modulated sinusoidal carrier wave $x_c(t)$, which can be written as

$$x_c(t) = A(t)cos\big(2\pi f_c + \theta(t)\big), \tag{1}$$

where $A(t)$ is the amplitude of the signal over time, f_c the constant carrier frequency and $\theta(t)$ the phase of the signal over time. Amplitude modulation encodes data to $A(t)$ and phase- and frequency-modulations encode it to $\theta(t)$. By applying a trigonometric identity to Equation (1), the carrier can now be rewritten as

$$x_c(t) = A(t)\,cos\big(\theta(t)\big)\,cos(2\pi f_c) - A(t)\,sin\big(\theta(t)\big)\,sin(2\pi f_c) \tag{2}$$

$$= x_i(t)cos(2\pi f_c) - x_q(t)sin(2\pi f_c) \tag{3}$$

where $x_i(t)$ represents the *in-phase component* and $x_q(t)$ the *quadrature component* of the signal. Both of the components represent the amplitude of their respective sinusoidal signals over time. These components can also simply be denoted as I and Q respectively. By applying Euler's formula to Equation (3), the carrier can finally be written as

$$x_c(t) = \mathrm{Re}\big\{x_e(t)e^{j2\pi f_c t}\big\}, \tag{4}$$

where $x_e(t) = x_i(t) + jx_q(t) = A(t)e^{j\theta(t)}$ is the *complex envelope* of the real bandpass signal $x_c(t)$. Equation (4) signifies that the signal $x_c(t)$ can now be represented with the complex baseband signal $x_e(t)$, no matter how the signal is modulated. This in turn implies that theoretically all analog signals at frequency f_c can be represented only with the in-phase and quadrature components. The synthesis for $x_c(t)$ from $x_i(t)$ and $x_q(t)$ is illustrated in Figure 2 [52, p. 35]. The setup in Figure 2 is called an I/Q-modulator.

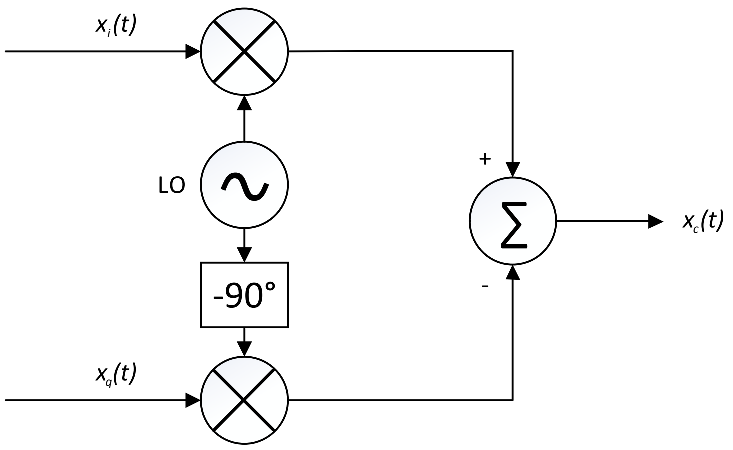


Figure 2. *Synthesis of $x_c(t)$ from $x_i(t)$ and $x_q(t)$ adapted from [52, p. 36].*

Figure 2 shows that the I/Q data will be multiplied with the carrier in a mixer, such that the I-branch will be multiplied with just the carrier and the Q-branch with a 90 degree shifted carrier. Thus, the I/Q data does not have to be at carrier frequency, as it is shifted to the desired frequency. This process of synthesizing the transmitted signal is also called *upconversion,* as the data signal is being converted to carrier frequency [52, p. 35]. Upconversion of the signal requires the carrier to be phase shifted exactly 90 degrees, and this is rarely the case. This problem causes an effect known as *I/Q-imbalance* or *mismatch*. The imbalance creates a mirror image of the signal in frequency domain, with respect to the local oscillator (LO) frequency. In time domain, the I/Q-imbalance can be modeled as a superposition of the baseband signal and its complex conjugate. The effects of the imbalance can be digitally mitigated, however [60].

2.2 Sampling and Resampling

In order to perform digital signal processing, the analog signal of interest has to be converted to digital form. This is done by *analog-to-digital (A/D)* conversion. A/D conversion *samples* a continuous-time signal with continuous amplitude. The end result is a discrete set of samples that represent the signal for the whole duration of the sampling period. The amplitude of the samples is also discrete and bound with a lower and upper limit. In other words, the samples are *quantized* [21, p. 101]. Quantization introduces distortion, often called *quantization noise,* to the signal since the samples rarely have the exact level of the possible representations. That is why the samples values have to be rounded, truncated or by some other means assigned a value representable in the digital domain. Using as many steps as possible for the outputs of the A/D converter decreases the effects of quantization noise [46, pp. 27—28]. Additionally, quantizing the signal may lead the output to *saturate* if the sample's value exceeds or drops below the maximum range of the possible outputs [26]. Once saturated, the output of the system will remain constant, yielding either the maximum or the minimum representable value. Both the quantization noise and saturation have to be taken into account when translating the signals from analog and digital domain.

Since the sample representing the signal remains constant for a set period of time, there is an upper-limit to which frequencies can be sampled with a certain sampling frequency. Conversely, there is a lower-limit to the sampling frequency when sampling a signal with a certain bandwidth. This lower-limit is known as the *Nyquist frequency,* and it is exactly half of the sampling frequency of the system. Signals with frequencies higher than the Nyquist frequency will be *aliased*, or mirrored with respect to the Nyquist frequency in frequency domain [44, p. 98]. Anti-aliasing filters restrict the input bandwidth of the input signals to satisfy the restriction imposed by the Nyquist frequency, which mitigates the aliasing effects in A/D converters.

In some instances, the sample rate of the signal has to be changed, in a process called *resampling*. Changing the sample rate does not change the signal the sequence of values represents, but rather the amount of samples the sequence holds. Making the sample rate higher is called *interpolation* and the reduction of it is called *decimation* [43, p. 305]. Decimation by an integer factor is arguably the easiest of the operations to understand. If a signal, sampled at frequency f_s, consists of S samples and the decimation factor is K, only every Kth sample from the original signal is kept, reducing the length of the signal to S/K. Thus, the sampling frequency is reduced to f_s/K [14, pp. 72—73]. In order to prevent aliasing, the signal has to be low-pass filtered before decimation. A system that low-pass filters and decimates the signal is called a *decimator* [14, p. 89].

Interpolation, contrary to decimation, adds samples to the original signal. If the signal sampled at frequency f_s again holds S samples and the interpolation factor is L, between every sample $L-1$ zero samples will be added to the signal. This makes the signal's new length SL and the new sample rate is Lf_s [14, pp. 97—98]. After adding the new values, their amplitudes have to be estimated. To achieve this, the signal has to be low-pass filtered, ideally using the sinc function. A simpler way for interpolation is it to linear interpolate the values of the new samples, assuming the continuous-time signal does not have any abrupt changes between the samples. Regardless of the execution, a system, which adds zero samples to the signal and low-pass filters it, is called an *interpolator* [14, pp. 103—110].

In addition to integer factors, interpolation and decimation are possible with non-integer factors as well. Changing the sampling frequency by a rational number, which can be written as L/K, where both L and K are integers, can be achieved, conceptually, by interpolating the signal by a factor of L and decimating it by a factor of K. It is also possible to scale the sampling frequency by a non-rational value, but that is out of the scope of this thesis.

2.3 Wireless Inband Full-duplex Systems

As it was stated at the beginning of the chapter, the simplest way to establish a radio connection is to have a single transmitter and a single receiver. This type of configuration is named *simplex.* An example of a simplex system is a radio receiver, which only receives the signals that a broadcaster sends. While simplex definitely has its uses, it is more common for radio systems to be *duplex.* Duplex means that all the devices connected to the system can act as both transmitter and receiver, a *transceiver*, which allows for two-way communication between the devices. Duplex can be divided into *half-duplex* and *full-duplex.* Half-duplex systems are either transmitters or receivers at a time over a certain resource, namely time or frequency. Full-duplex systems act as both transmitter and a receiver simultaneously while using the same resources [50, p. 672].

A prominent problem with full-duplex systems is the *self-interference* (SI) signal. Simultaneously transmitting and receiving over the same frequency band causes SI. Naturally, by doing so the signal that is being transmitted is also being received by the same device. This received signal is the SI signal and it is often orders of magnitude greater in power than the desired received signals. Thus, the desired signals are drowned in the SI and are impossible to decipher. Additionally, RF equipment is sensitive, especially on the receiver side in order to pick up even the weakest signals. If the SI gets to the receiving chain unattenuated, the components might be damaged or destroyed all together [30, p. 14].

Obviously, the SI signal needs to be attenuated or ideally cancelled completely. A way to achieve attenuation is to separate the transmitter and receiver chains, so that they do not share a common antenna [10]. This method adds *isolation* between the chains, but of course, the receiving antenna will still pick up some of the SI, but now it is attenuated. The same isolation effect can be achieved by using a *duplexer,* which now allows the usage of a shared antenna again. A duplexer separates the transmitting and receiving chains by routing the received signal from the antenna to the receiver and the signal from the transmit-chain to the antenna. The duplexer is most often a *circulator,* which is a passive device, but there are active duplexers as well, that require power and control from outside, such as an *electrical balance duplexer (EBD)* [37, 53]. Still, even with a duplexer, there is some leakage of the SI to the receiver since the isolation the duplexer provides is not perfect. Additional cancellation of the SI is therefore required.

Figure 3 illustrates a simplified model of a full-duplex transceiver, completed with an RF-canceller and a digital canceller. The transmitter chain consists of a digital-to-analog converter, I/Q-modulator (discussed in Section 2.1), an amplifier and a bandpass filter. After the transmitter chain, there is a power amplifier (PA), which amplifies the signal, in order for the receiver to be able to pick up the transmit signal. The receiver chain consists of almost the same components as the transmitter, except the I/Q-modulator is substituted with a demodulator and the digital-to-analog converter with an analog-to-digital one.

Both the transmitter and receiver chains share a common antenna, the utilization of which is made possible by a circulator.

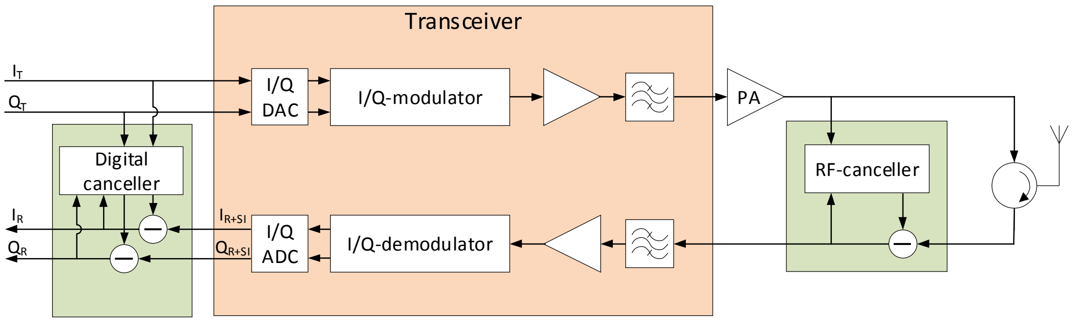


***Figure 3.** A full-duplex transceiver with an RF canceller and a digital canceller, adapted from [30, p. 19].*

The physical circuitry of the transceiver introduces imperfections and non-idealities to the signals propagating through the system. Most notable of these effects, as far as SI cancellation is concerned, is the nonlinearity induced by the amplifiers. The most significant source of nonlinearity is the PA on the output of the transmission chain, although other amplifiers and mixers of the system may add to it as well. Nonlinearity in the amplifiers is caused by saturation, where high inputs do not yield as high outputs as it would, if the system was linear. Thus, the system becomes nonlinear. Although the distortion is unwanted, the amplifiers are still used close to the nonlinear region in order to maximize power efficiency. Nonlinearity causes spectral regrowth, which can be seen in the spectrum of the signal as "widening" of the signal at its base [15]. Other imperfections of the physical circuitry include the I/Q-imbalance (discussed in Section 2.1) and quantization noise (discussed in Section 2.2).

The full-duplex transceiver introduced in Figure 3 contains two separate canceller blocks for cancelling the SI signal. These are the RF-canceller and the digital canceller, which function in the analog and digital domains respectively. Both of the blocks have a similar purpose; they estimate the effects the coupling channel and distortion from the physical circuitry have on the signal, and subtract the estimate from the received signal [30, p. 3]. This functionality is inherently different from the isolation that the duplexers provide, since isolation just prevents the transmit signal from leaking to the receiver, while the cancellation actually affects the received signal. Ideally, the estimate the canceller block produces matches the above-mentioned effects exactly and the SI signal is cancelled completely. This is not the case with real signals however, hence there is some residual power of the SI signal left over. Still, the power of the SI is decreased significantly and it is now possible to decipher the received signals of interest. The functionality of the digital canceller will be explained in more detail in the following sections.

There are numerous benefits to using full-duplex systems, most notable one being the theoretical doubling of spectral efficiency [1, 18]. This is due to both the receiver and the transmitter using the same frequency, therefore doubling the amount of data that is transferrable over the frequency band. As the frequency spectrum grows more and more crowded, being able to use the available frequencies more efficiently becomes more and more crucial. Full-duplex also simplifies the planning of links, as they only need one frequency for operation [38, pp. 6—7].

A problem full-duplex offers a solution to is the hidden node problem. The hidden node problem might occur when two devices try to contact a shared node, for example, when two Wi-Fi devices contact a wireless access point. The problem arises when the two client nodes do not know of each other's existence and send their data to the access point simultaneously. The messages will then collide and the messages would have to be sent again, and the same issue might occur. The nodes know to send new messages by using the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol, even in half-duplex systems. However, even using CSMA/CA results in some packet losses [17]. To combat this, a full-duplex system can be utilized. Full-duplex does not fix the issue on its own however, but rather it enables techniques that rely on simultaneous transmission and reception to ultimately overcome the problem. These techniques introduce new acknowledgment schemes for the transmitter and receiver, introduced for example in [4] and [62]. The techniques are similar to CSMA/CD, but they utilize the full-duplex capabilities of the nodes to detect the collisions.

Full-duplex systems can also be used in wireless backhauling solutions. Backhauling refers to the way an access node is connected to the backbone network. For example in a cellular phone case, the phones contact the base station, which acts as an access node and the access node then connects, or backhauls, to other such nodes forming the backbone of the network [11, pp. 61—62]. Especially in locations with highly dense networks, using wireless backhauling instead of wired one, the deployment costs are reduced. Full-duplex can be utilized for *self-backhauling* which means that the same frequency is used for the network's uplinks and downlinks, and for the backhauling, which saves spectral and temporal resources [31].

3. NONLINEAR DIGITAL SELF-INTERFERENCE CANCELLER

This chapter focuses on the theoretical aspects of a nonlinear digital self-interference canceller and introduces a state-of-the-art implementation of such an algorithm. A nonlinear self-interference canceller, as opposed to a linear one, considers the nonlinear effects the physical circuitry has on the signal. The most notable source of nonlinear distortion is the PA on the transmission chain, although other parts have nonlinear effects on the signal as well. The digital SI canceller can be ultimately reduced to a simple subtraction, where the transmitted signal is subtracted from the received signal. Unfortunately, it is not as straightforward as that, since the received signal has passed through the channel and some circuitry, so it is attenuated and distorted. Therefore, in order to cancel the received SI, the effects of the channel and the circuitry have on the signal have to be modelled. This problem can be simplified a bit since the most prominent effects are those from the channel and the nonlinearity the PA on the transmitter chain induces.

Traditionally, the effects from the channel and the nonlinearities are modeled in the digital domain using a *memory polynomial model*. The coefficients of the model take into account the memory effects of the devices in use. This means that a single sample cannot simply be handled independently as the previous values affect it as well. The memory polynomial model of the self-interference signal can be written as

$$x^{SI}(n) = \sum_{p=0}^{P_{mp}} \sum_{m=0}^{M} h_p(m)\psi\big(x(n-m)\big) + \tilde{z}(n), \qquad (5)$$

where $x^{SI}(n)$ is the nth sample of the self-interference signal x^{SI}, P_{mp} the nonlinearity order of the model, M the length of the memory of the model, $h_p(m)$ the channel coefficients that include the memory of the PA, multipath channel and possible RF cancellation, $\psi\big(x(n-m)\big)$ the pth order basis function and $\tilde{z}(n)$ the error from the imperfection of the model. In the digital canceller, the SI signal is approximated using (5), and the output is subtracted from the received signal [2] to produce the *error signal*. If the model matches the effects perfectly, the error signal will be zero, and the SI is cancelled completely. However, there is always some residual power left from the SI, as the model is never perfect.

The effects of the channel and the nonlinearities of the circuitry are hard to define, and they can even change abruptly. That is why the coefficients of the model cannot be constant, but rather adaptive, which means that the coefficients of the model are updated at every cycle of the algorithm. This is the most computationally demanding process of the algorithm, as the actual cancellation can be reduced to mere multiplication of the coefficients and the transmit signal, the product of which is then subtracted from the received signal. The detailed formulas for the updates of the memory polynomial model are omitted here, but can be found for example in [32], where the update is achieved with the least mean squares (LMS) method. Literature also has examples of the update of SI canceller algorithm being implemented using the recursive least squares (RLS) method, reported for example in [63].

As stated earlier, the nonlinear memory polynomial model incorporates the channel effects and the nonlinearities in a single expression. However, it is also possible to decouple these effects and model them separately, which makes them independent of one another. Thus, it is also possible to apply the updates to the model coefficients separately, and with different paces, which might be desirable in some systems. The following sections focus on tools for modeling the channel and nonlinearities. These models are then used to define the Hammerstein spline-based SI canceller.

3.1 Adaptive Linear Filtering

A physical filter is a frequency selective circuit that, ideally, only passes signals through unattenuated and undistorted on wanted frequencies. A physical example of such a filter is an RC circuit, which, depending on the component arrangement, can act as a high-pass or low-pass filter. The electrical properties of the components determine how the filter responds to different frequencies. A digital filter does not comprise of physical components as it is just a mathematical algorithm, but it aims to accomplish the same as a physical filter. Since the digital filter is just defined by some coefficients (discussed later), the response of the filter is easily adjustable. Furthermore, if the filter adjusts itself, the filter becomes *adaptive*. Additionally, a digital filter can have a phase response that is completely *linear,* a feat not possible for physical filters.

There are two basic classes of digital filters, *infinite impulse response (IIR)* and *finite impulse response (FIR)* filters. Out of the two classes, FIR has two major advantages over the IIR in terms of linear filtering required in the SI canceller. First and most importantly, the FIR is inherently stable, so it does not require analysis to confirm the stability. This is especially handy for adaptive filters, as the changes in the coefficients could make an IIR filter unstable. Secondly, FIR can have exactly linear phase response (but not necessarily), whereas the IIR generally has a nonlinear phase response [24, p. 321]. Thus, the remainder of this section focuses on FIR filters.

FIR filtering for real valued signals is defined by the following expression:

$$y(n) = \sum_{k=0}^{M} h(k)x(n-k)\,, \tag{6}$$

where $y(n)$ is the nth output of the filter, M is the number of filter coefficients or *taps*, often referred to as the *memory* of the filter, $h(k)$ the kth tap of the filter and $x(n-k)$ the input x delayed by k steps. The variable h also describes the impulse response of the filter. The FIR filter can be presented in a block diagram, which clarifies the operation. A block diagram presentation of a FIR filter is presented in Figure 4.

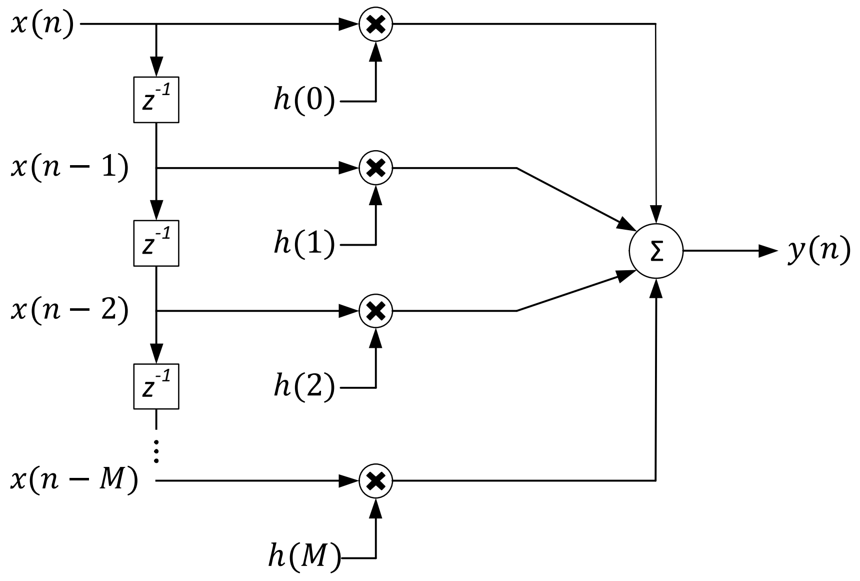


Figure 4. *A block diagram of an FIR filter.*

Figure 4 makes it clear that the output y is dependent on M past values and the present value of x and on the values of h. Indeed, the performance of the filter is adjustable by varying the coefficients of h, as it is the only variable in the designer has direct access to in addition to the amount of taps in the filter [34, pp. 34—36]. For simplification, the past and current values of x and the values of h can be compiled into separate vectors, $\mathbf{x} = [x(n)\ \ x(n-1) \dots x(n-M)]^T$ and $\mathbf{h} = [h(0)\ \ h(1) \dots h(M)]^T$. Now the FIR filtering operation can be expressed as

$$y(n) = \mathbf{h}^T\mathbf{x}, \tag{7}$$

which is equivalent to Equation (6).

For a static filter, there are several methods for acquiring the values of $\mathbf{h}$ to meet the criteria of the design. These methods include the window method, the optimal method and the frequency sampling methods, and they are among the most widely used procedures for filter design [24, p. 351]. These methods are computationally heavy however, and are designed to be only used once during the filter designing process. This is why the adaptive filtering takes another approach to determining the impulse response of the filter. These approaches include the *least mean squares (LMS)* and the *recursive least squares (RLS)* methods. Out of these two, the LMS offers less computational complexity and it is the most widely used adaptive filter algorithm [34, p. 46; 9, p. 71]. Therefore, the remainder of this chapter focuses on the LMS algorithm.

3.1.1 LMS Algorithm Derivation

A block diagram for an adaptive filter illustrated in Figure 5 shows two additional signals needed for the operation of the adaptive filter. These signals are the *desired signal* $d(n)$ and the *error signal* $e(n)$.

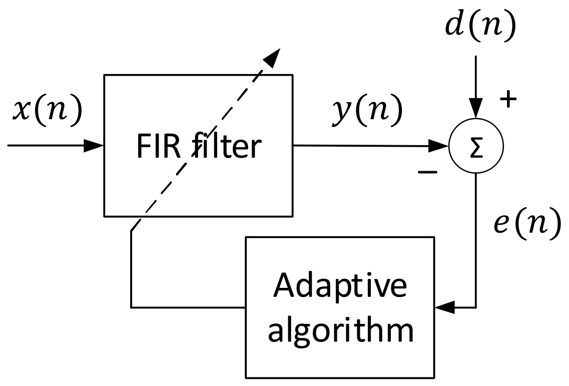


Figure 5. *A block diagram presentation of an adaptive filter.*

The desired signal presents the wanted behavior of the filter. It can be, for example, a measured signal. The difference between the desired signal and the filter output, the error signal, is a measure of how well the filter follows the wanted behavior. Smaller error signal signifies better performance. With the information on the performance from the error signal, the adaptive algorithm changes the filter coefficients accordingly [22, pp. 96—97].

In order to generalize the adaptive filter solution, let us consider complex-valued signals. Thus, the FIR filter solution of Equation (7) can be written as

$$y(n) = \mathbf{w}^H\mathbf{x}, \tag{8}$$

where $\mathbf{w}$ is the complex-valued impulse response of the filter (cf. **h**, but for adaptive filters $\mathbf{w}$ is more common notation), and $\mathbf{w}^H$ is the Hermitian transpose of $\mathbf{w}$. Now the error signal can be written as [22, p. 205]

$$e(n) = d(n) - y(n) = d(n) - \mathbf{w}^H\mathbf{x}. \tag{9}$$

Now we may write a *cost function* J for the filter [22, p. 97]:

$$J(\mathbf{w}) = E[|e(n)|^2] = E[e(n)e^*(n)], \tag{10}$$

where E is the statistical expectation operator. The LMS algorithm aims to minimize the mean square error, which is where the name of the algorithm comes from. Equation (10) can be then further expanded using Equation (9) [22, p. 107; 49, p. 187]:

$$J(\mathbf{w}) = {\sigma_d}^2 - \mathbf{w}^H\mathbf{p} - \mathbf{p}^H\mathbf{w} + \mathbf{w}^H\mathbf{R}\mathbf{w}, \tag{11}$$

where ${\sigma_d}^2$ is the expected value of the mean square of the desired signal $d(n)$, $\mathbf{p} = E[\mathbf{x}d^*(n)]$ the *cross-correlation vector* between $\mathbf{x}$ and $d^*(n)$ and $\mathbf{R} = E[\mathbf{x}\mathbf{x}^H]$ the *auto-correlation matrix* of the input. The minimization of the cost function is achieved with *method of steepest descent*, which means that the successive applied updates to the coefficients of the filter are done in the direction opposite of the gradient vector of the cost function. The method of steepest descent is defined as [22, p. 204; 49, p. 186]

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \frac{1}{2}\mu\,\boldsymbol{\nabla}\mathbf{J}(\mathbf{w}), \tag{12}$$

where $\mathbf{w}(n+1)$ is the new impulse response of the filter, $\mathbf{w}(n)$ the current impulse response, μ the *step-size* or *learning rate* of the algorithm and $\boldsymbol{\nabla}\mathbf{J}(\mathbf{w})$ the gradient vector of the cost function. The step-size is multiplied with a half for convenience that will be apparent later. The gradient of the cost function is [22, p. 206; 49, p. 187]

$$\boldsymbol{\nabla}\mathbf{J}(\mathbf{w}) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n). \tag{13}$$

In order to estimate the gradient, the simplest way is to substitute the vector $\mathbf{p}$ and matrix $\mathbf{R}$ with their respective instantaneous estimates, $\widehat{\mathbf{p}} = \mathbf{x}d^*(n)$ and $\widehat{\mathbf{R}} = \mathbf{x}\mathbf{x}^H$. Thus, the instantaneous estimate for the gradient vector is given as [22, p. 236]

$$\widehat{\boldsymbol{\nabla}}\mathbf{J}(\mathbf{w}) = -2\mathbf{x}d^*(n) + 2\mathbf{x}\mathbf{x}^H\widehat{\mathbf{w}}(n). \tag{14}$$

Combining Equations (12) and (14), we get the following estimating expression for the update of the filter coefficients:

$$\widehat{\mathbf{w}}(n+1) = \widehat{\mathbf{w}}(n) + \mu\mathbf{x}(n)\left(d^*(n) - \mathbf{x}^H\widehat{\mathbf{w}}(n)\right) \tag{15}$$

$$= \widehat{\mathbf{w}}(n) + \mu\mathbf{x}(n)e^*(n). \tag{16}$$

Equations (15) and (16) describe the LMS filter update algorithm [22, p. 236; 49, p. 204].

An important feature of the LMS filter to consider is its *convergence*. Although the use of an FIR filter guarantees stability of the filter, the adaption could still lead the filter coefficients to *diverge*. From a design standpoint, the only value the adaptive filter has, that is configurable by the designer, is the step-size μ in addition to the number of taps in the filter. It turns out, that the filter will converge if the step-size conforms to the following condition:

$$0 < \mu < \frac{1}{\lambda_{max}}, \tag{17}$$

where λ_{max} is the maximum eigenvalue of the autocorrelation matrix **R**. Typically, it is advisable to use values for μ that are closer to the lower bound of the range [9, pp. 75—78].

3.1.2 Finite Precision LMS

Thus far, we have only been considering infinite precision for all the values of the algorithm. However, if the filter is implemented in a digital environment, on an FPGA for example, the filter then falls victim to effects that arise from finite precision. These effects are caused by the A/D-conversion (discussed in Section 2.2) and the finite word-lengths of the values. The finite word-lengths cause errors in the intermediate values of the algorithm, as the values are either rounded or truncated to the nearest possible value. These errors accumulate and might cause instability and divergence of the algorithm. To combat this, the intermediate values of the algorithm should have word-lengths that are as long as possible for the implementation. However, even using sufficiently large word-lengths does not guarantee stability for the algorithm. The finite word-lengths might also cause *stall*, which means that the filter coefficients stop updating before convergence is achieved. This can be prevented by choosing a new lower bound for the step-size:

$$\frac{2^{-B_c}}{4\sigma_x\sqrt{\sigma_e^2+\sigma_n^2}} < \mu < \frac{1}{\lambda_{max}} \tag{18}$$

where B_c is the amount of bits in the filter coefficients (without the signed bit), and σ_x, σ_e and σ_n are the errors caused by quantization for signals x, e and noise respectively [9, pp. 96—97]. The errors are given as

$$\sigma^2 = \gamma\frac{2^{-2B}}{12}, \tag{19}$$

where B is the amount of bits representing the value excluding the signed bit and γ a coefficient dependent on the rounding mode. If the products are in full precision, i.e. they are only quantized after all operations, $\gamma = 1$, otherwise $\gamma = N_{pp} + 1$, where N_{pp} is the amount of partial products [9, pp. 93—94]. With finite word-lengths, the step-size is a critical matter, as for the convergence of the algorithm smaller values of μ yield better convergence, however the step-size cannot be too small as it could cause stall and significant error in the coefficients of the filter [9, p. 96].

The effects of finite word-length can also be reduced by adding a new term, called *leakage factor* to the filter coefficient update algorithm. With the added term, the LMS update algorithm introduced in Equations (15) and (16) can be written as

$$\hat{\mathbf{w}}(n+1) = (1+\mu\lambda)\hat{\mathbf{w}}(n) + \mu\mathbf{x}e^*(n), \tag{20}$$

where $(1+\mu\lambda)$ is the leakage factor and λ is a small real number that conforms to $0 \leq \lambda \leq \frac{1}{\mu}$ [34, p. 615], or for further improvement, a diagonal matrix [6]. It is worth noting that the added leakage factor term in Equation (20) adds more complexity to the algorithm, as it adds multiplications. For λ it holds that it cannot be too small, as the effect it provides becomes negligible and at the same time, λ cannot be too large either as it degrades the performance of the filter. The leakage factor thus provides a trade-off between performance and stability.

3.2 Splines Interpolation

A well-known example of polynomial approximation is the Taylor series, which approximates the whole curve of the function with a single expression. There is a profound problem with this approach, however: if the curve under investigation is badly behaving, i.e. it has abrupt changes in certain regions, the whole approximation suffers, not only in the region of the bad behavior [7, p. 17; 23, p. 23]. *Splines,* which are also a form of polynomial approximation, have a solution for this problem, as they are defined *piecewise.* The curve under investigation is divided into regions, and the curves within the regions are approximated separately. This way, a bad approximation in one region does not affect the rest of the approximations.

The points between the regions of the splines are called *knots*. While the knots can have arbitrary distances between consecutive ones, in the remainder of this section we will focus on splines with evenly spaced knots. A spline with evenly spaced knots is called a *uniform spline*. Between the knots, the splines are defined by polynomials with order P. In order to ensure smoothness of the splines approximations, the splines have to be differentiable up to $(P-1)th$ order, even at the knots [61, p. 538]. This is also true for B-splines [55], which are in focus in this work. Other spline interpolation schemes with different rules on the approximations, such as the Catmul-Rom splines [13] and natural splines [8] have been utilized for nonlinear system identification as well, but they are out of the scope of this thesis.

Spline basis function of order P for B-splines is defined recursively as [7, pp. 109—124]

$$N_i^P(u) = \frac{u-t_{x,i}}{t_{x,i+P}-t_{x,i}} N_i^{P-1}(u) + \frac{t_{x,i+P+1}-u}{t_{x,i+1+P}-t_{x,i+1}} N_{i+1}^{P-1}(u), \tag{21}$$

where u is the *abscissa* value between two consecutive knots, $t_{x,i}$ is the knot in the x-axis and i is an index for the knot in question. The abscissa value is the normalized value of the interpolated signal in the space between two consecutive knots. For the purposes of the self-interference canceller, we will define B-splines up to the second order. Since the B-spline basis functions are determined recursively, in order to define the rest of the splines, the basis function with order zero has to be determined. The zeroth order B-spline basis function is defined as

$$N_i^0 = \begin{cases} 1, & t_{x,i} \leq u \leq t_{x,i+1} \\ 0, & \text{otherwise} \end{cases} . \tag{22}$$

The zeroth order spline represents a unit-width box function [61, pp. 541—542]. By applying the B-spline basis function definition from Equation (22) to the recursive definition of P order B-splines introduced in Equation (21), the first order B-spline basis function can be defined as

$$N_i^1(u) = \frac{u-t_{x,i}}{t_{x,i+P}-t_{x,i}} \underbrace{N_i^0(u)}_{=1} + \frac{t_{x,i+P+1}-u}{t_{x,i+1+P}-t_{x,i+1}} \underbrace{N_{i+1}^0(u)}_{=1} \tag{23}$$

$$= \begin{cases} \frac{u-t_{x,i}}{t_{x,i+1}-t_{x,i}}, & t_{x,i} \leq u \leq t_{x,i+1} \\ \frac{t_{x,i+2}-u}{t_{x,i+2}-t_{x,i+1}}, & t_{x,i+1} \leq u \leq t_{x,i+2} \\ 0, & \text{otherwise} \end{cases} . \tag{24}$$

Similarly, the second order B-spline basis function can be defined as

$$N_i^2(u) = \frac{u-t_{x,i}}{t_{x,i+P}-t_{x,i}} N_i^1(u) + \frac{t_{x,i+P+1}-u}{t_{x,i+1+P}-t_{x,i+1}} N_{i+1}^1(u) \tag{25}$$

$$= \begin{cases} \frac{u-t_{x,i}}{t_{x,i+2}-t_{x,i}} \frac{u-t_{x,i}}{t_{x,i+1}-t_{x,i}}, & t_{x,i} \leq u \leq t_{x,i+1} \\ \frac{u-t_{x,i}}{t_{x,i+2}-t_{x,i}} \frac{t_{x,i+2}-u}{t_{x,i+2}-t_{x,i+1}} + \frac{t_{x,i+3}-u}{t_{x,i+3}-t_{x,i+1}} \frac{u-t_{x,i+1}}{t_{x,i+2}-t_{x,i+1}}, & t_{x,i+1} \leq u \leq t_{x,i+2} \\ \frac{t_{x,i+3}-u}{t_{x,i+3}-t_{x,i+1}} \frac{t_{x,i+3}-u}{t_{x,i+3}-t_{x,i+2}}, & t_{x,i+2} \leq u \leq t_{x,i+3} \\ 0, & \text{otherwise} \end{cases} . \tag{26}$$

The zeroth, the first and the second order spline basis functions are presented in Figure 6 for uniform splines.

We can define the distance between consecutive knots as Δx as we are considering uniform splines. Thus, we can write for example $t_{x,i+1} - t_{x,i} = \Delta x$, and similarly for the other knots. The second order B-spline basis function can therefore be finally written as

$$N_i^2(u) = \begin{cases} \frac{1}{2\Delta x}(u-t_{x,i})^2, & t_{x,i} \leq u \leq t_{x,i+1} \\ \frac{1}{2}+\frac{1}{\Delta x}(u-t_{x,i+1}) - \frac{1}{(\Delta x)^2}(u-t_{x,i+1})^2, & t_{x,i+1} \leq u \leq t_{x,i+2} \\ \frac{1}{2}-\frac{1}{\Delta x}(u-t_{x,i+2}) - \frac{1}{2(\Delta x)^2}(u-t_{x,i+2})^2, & t_{x,i+2} \leq u \leq t_{x,i+3} \\ 0, & \text{otherwise} \end{cases} , \tag{27}$$

which can be written in matrix form as

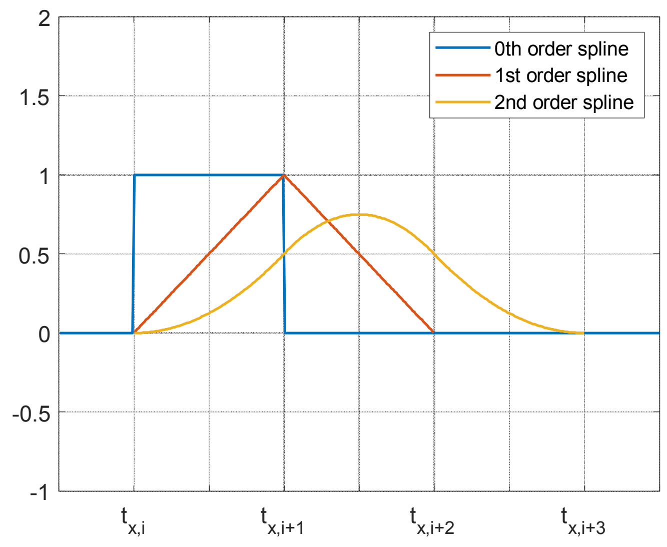


Figure 6. *The zeroth, first and second order B-splines basis functions for uniform splines.*

$$N_i^2(u) = \underbrace{[u^2 \quad u \quad 1]}_{\mathbf{u}^T} \underbrace{\begin{bmatrix} \frac{1}{2(\Delta x)^2} & \frac{-1}{(\Delta x)^2} & \frac{1}{2(\Delta x)^2} \\ \frac{-1}{\Delta x} & \frac{1}{\Delta x} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}}_{\mathbf{C}} \tag{28}$$

$$= \mathbf{u}^T\mathbf{C}, \tag{29}$$

where $\mathbf{C}$ is the spline basis matrix. If the distance between knots is defined as unity, the $\mathbf{C}$ matrix simplifies to [55]:

$$\mathbf{C} = \frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}. \tag{30}$$

Owing to these, the spline interpolation scheme that interpolates an input signal $x(n)$ can be written as a function of two local variables: the span index i and the abscissa value u, which is, as stated earlier, a normalized value of the input that lies between two consecutive knots. At instance n, these variables can be expressed as [56]:

$$i_n = \left\lfloor \frac{x(n)}{\Delta x} \right\rfloor + \frac{Q-1}{2} \tag{31}$$

$$u_n = \frac{x(n)}{\Delta x} - \left\lfloor \frac{x(n)}{\Delta x} \right\rfloor. \tag{32}$$

Additionally, the interpolation scheme is dependent on Q control points contained within the vector $\mathbf{q} \in \mathbb{R}^{Q\times 1} = [q_0 \cdots q_{Q-1}]^T$ that will define the spline interpolation curve $\varphi(u)$, that connects all the control points. The spline curve can be written as

$$\varphi(u) = \mathbf{\Psi}^T(\mathbf{1} + \mathbf{q}), \tag{33}$$

where $\mathbf{1}$ is a $Q \times 1$ vector of all ones and $\mathbf{\Psi} \in \mathbb{R}^{Q\times 1} = [0 \cdots 0\ \mathbf{u}^T\mathbf{C}\ 0 \cdots 0]^T$, where $\mathbf{u}^T\mathbf{C}$ is indexed such that the first element is at index i [47, 55].

Traditionally, spline interpolation, comprising the procedure presented previously in this section, has been applied for real-valued signal, such as in [55]. However, communication theory uses signals with complex values, as was discussed in Section 2.1, therefore spline theory needs to be redefined to comply with this. This procedure is addressed in the next section.

3.3 Adaptive Hammerstein Spline-based Self-Interference Canceller

In this section, we will combine the LMS adaptive linear filter and splines introduced earlier to define a nonlinear Hammerstein spline-based self-interference canceller. A Hammerstein system is a cascaded system that has a static nonlinear part followed by a linear one [57, p. 5]. This configuration matches well with the transceiver layout introduced in Figure 3. The nonlinear part, the PA, is followed by the linear channel, which are then modelled using the splines and the linear filter, respectively.

Figure 7 illustrates a block diagram of the Hammerstein system, completed with the SI canceller and the principle of the update of the splines and linear filter.

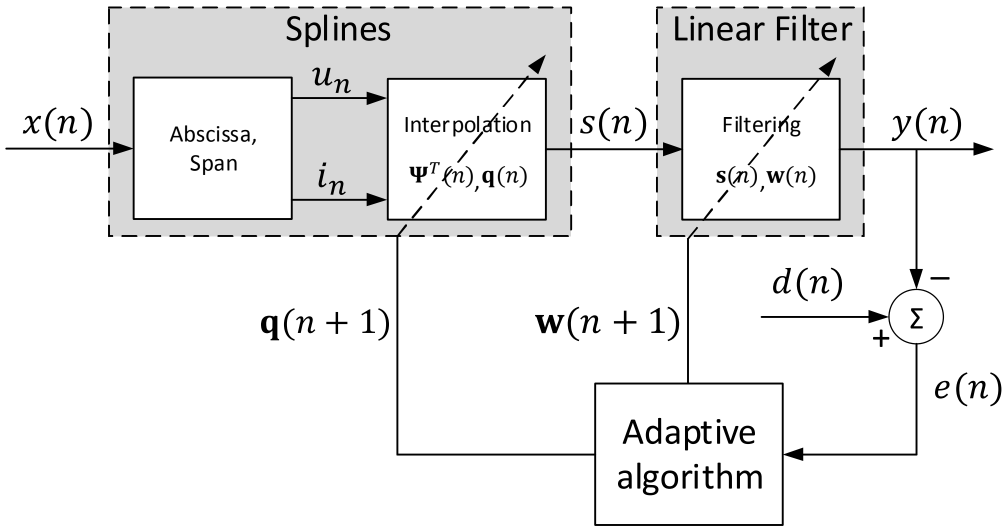


Figure 7. *Block diagram of a Hammerstein system and the adaptive self-interference canceller.*

In the Hammerstein spline-based SI canceller, the output of the linear filter $y(n)$ is again subtracted from the desired signal $d(n)$ to produce the error signal $e(n)$. With the signals presented in Figure 7, the linear filter output can be written as

$$y(n) = \mathbf{w}(n)^H \mathbf{s}(n), \tag{34}$$

where $\mathbf{w}(n)$ is the impulse response of the linear filter at iteration n and with memory M, and $\mathbf{s}(n) \in \mathbb{C}^{M\times 1} = \left[s(n + M_{pre}) \dots s(n) \dots s(n - M_{post})\right]^T$, where M_{pre} and M_{post} are the filter pre-cursor and post-cursor taps, respectively [47]. Similarly to the adaptive linear filter introduced in Section 3.1.1, the coefficients of the parts of the Hammerstein system have to be updated in order to achieve cancellation. The adaptive behavior is again based on the information of the error signal $e(n) = d(n) - y(n)$. In the system described in Figure 7, the coefficients to be updated are the filter taps $\mathbf{w}(n)$ and the control points of the splines $\mathbf{q}(n)$. Thus, the cost function of the system can now be written in terms of the filter taps and the control points [54], considering only the instantaneous error:

$$J(\mathbf{w}, \mathbf{q}) = e^*(n)e(n). \tag{35}$$

With the method of steepest descend, and by noting the decoupled nature of the system, the linear filter tap update algorithm can be written as

$$\mathbf{w}(n + 1) = \mathbf{w}(n) + \mu_w e^*(n)\mathbf{s}(n), \tag{36}$$

where μ_w is the step-size for the filter and $\mathbf{s}(n)$ the regression of the splines output. Equation (36) is equivalent to Equation (16), which was used to define an adaptive linear filter, with the exception of the input $\mathbf{x}(n)$ being replaced by the splines output $\mathbf{s}(n)$.

As mentioned in the previous section, the spline theory needs to be adapted for the use of complex values. For this purpose, the following lines present the extension to complex spline interpolation from [47]. In this context, the index and abscissa values can be redefined by replacing $x(n)$ with $|x(n)|$ in Equation (32) and removing $\frac{Q-1}{2}$ in Equation (31), as the absolute value makes the index value always positive. Additionally, Δx is defined as 1 for simplicity. Thus, the index and abscissa values can be expressed as

$$i_n = \lfloor |x(n)| \rfloor + 1 \tag{37}$$

$$u_n = |x(n)| - \lfloor |x(n)| \rfloor. \tag{38}$$

In order to model the complex behavior of the PA, two different real splines, modeling the I- (real) and Q-branches (imaginary) are used. However, for the sake of simplicity the two splines can be combined in one single complex valued expression that contain both of the branches. To this end, the interpolation scheme presented in Equation (33) is multiplied by the input signal $x(n)$ in order to retain its amplitude and phase and the control points are redefined as complex values: $\mathbf{q} = \mathbf{q}^{\mathrm{re}} + j\mathbf{q}^{\mathrm{im}}$. Hence, the splines nonlinear complex output can now be written as

$$s(n) = x(n)\boldsymbol{\Psi}^T(n)(\mathbf{1} + \mathbf{q}(n)). \tag{39}$$

The update algorithm for the control points can be determined in a similar manner to the linear filter. The method of steepest descend gives us:

$$\mathbf{q}(n+1) = \mathbf{q}(n) - \mu_q \frac{\partial J(\mathbf{w},\mathbf{q})}{\partial \mathbf{q}} \tag{40}$$

$$= \mathbf{q}(n) - \mu_q \left(e^*(n) \frac{\partial e(n)}{\partial \mathbf{q}} + e(n) \frac{\partial e^*(n)}{\partial \mathbf{q}} \right) \tag{41}$$

$$= \mathbf{q}(n) + \mu_q \left(e^*(n) \left[\frac{\partial y(n)}{\partial \mathbf{q}^{\mathrm{re}}} + j \frac{\partial y(n)}{\partial \mathbf{q}^{\mathrm{im}}} \right] + e(n) \left[\left(\frac{\partial y(n)}{\partial \mathbf{q}^{\mathrm{re}}} \right)^* + j \left(\frac{\partial y(n)}{\partial \mathbf{q}^{\mathrm{im}}} \right)^* \right] \right), \tag{42}$$

where μ_q is the step-size for the control points and $y(n)$ is the output of the linear filter. The partial derivatives of the filter output can be defined as

$$\frac{\partial y(n)}{\partial \mathbf{q}^{\mathrm{re}}} = \mathbf{\Sigma}(n)\mathbf{X}(n)\mathbf{w}^*(n) \tag{43}$$

$$\frac{\partial y(n)}{\partial \mathbf{q}^{\mathrm{im}}} = j\mathbf{\Sigma}(n)\mathbf{X}(n)\mathbf{w}^*(n), \tag{44}$$

where $\mathbf{\Sigma}(n) \in \mathbb{R}^{Q \times M} = \left[\mathbf{\Psi}\left(n + M_{pre}\right) \dots \mathbf{\Psi}(n) \dots \mathbf{\Psi}\left(n - M_{post}\right)\right]$ is a matrix that collects M past vectors $\mathbf{\Psi}(n)$ as its columns and $\mathbf{X}(n) = \operatorname{diag}\{x(n + M_{pre}), \dots, x(n - M_{post})\}$ is a diagonal matrix containing M past values of $x(n)$. With Equations (43) and (44), the control point update algorithm can be written as

$$\mathbf{q}(n+1) = \mathbf{q}(n) + \mu_q e(n)\mathbf{\Sigma}(n)\mathbf{X}^*(n)\mathbf{w}(n). \tag{45}$$

In conclusion, the Hammerstein spline-based canceller algorithm is a straightforward one. The splines output is first defined by Equation (39), after the abscissa and the span have been determined using Equations (37) and (38). Then, using the splines output, the linear filter output is determined by Equation (34), and the error signal is defined with the output. With the error signal, the updates for the linear filter coefficients and spline control points are applied after Equations (36) and (45), respectively [47].

The main merit of the Hammerstein spline-based canceller is its relative computational simplicity, compared to previous such algorithms [54], for example the memory polynomial model, introduced at the beginning of Chapter 3. This makes the spline-based model consume fewer resources when implemented on a given platform. The reduction of complexity is achieved without sacrificing performance much. Indeed, compared to the memory polynomial model, the reduction of complexity is 77 % in terms of multiplications needed, with practically the same performance [47]. This reduction in complexity is important for implementing the canceller in resource scarce platforms and making the technology commercially feasible.

4. DEVELOPMENT ENVIRONMENT

In this chapter, we take a look at the development environment in which the self-interference canceller is built in. The digital SI canceller operates in the digital domain, which would make a computer an attractive platform. However, since the canceller should work in real-time, it cannot be implemented on a computer, but rather it is implemented on a *Field-Programmable Gate Array (FPGA)* device. The FPGA is a chip that blurs the line between software and hardware, as the software that is programmed to be used on the FPGA, is actually physically wired within the chip. The FPGA in the implementation of this thesis lies inside a USRP, which is a software defined radio (SDR) device. In order to give the FPGA the program, it obviously need to be written first. This part of the implementation is performed in *LabVIEW Communications System Design Suite 2.0*. The code written with LabVIEW contains other signal processing code necessary for the operation as well, but the focus in this thesis is on the SI canceller implementation.

4.1 USRP and FPGA

In this work, the hardware for the self-interference canceller is a Universal Software Radio Peripheral. The Universal Software Radio Peripheral, or USRP for short, is a software defined radio (SDR) device developed by Ettus Research, a brand of National Instruments. An SDR is a communications device, the operation of which can be programmatically controlled. This control over the functionality is what makes SDRs such versatile platforms for communications engineering. If new functionality is to be added to a fully analog device, the whole system might have to be designed again. At least new physical parts would have to be added to the system, which increases the complexity and costs of the device. In an SDR platform this is not an issue, since new functionality can be just programmed, and it will be implemented as part of the system. This greatly reduces the deployment costs of the systems.

An SDR is not completely digital though. At least the *RF frontend,* which includes the transceiver part of the system, is analog, in order for the signals to be transmitted and received. Between the analog and digital domains, the signal is converted accordingly using A/D and D/A converters (discussed in Section 2.2). In an USRP device, the frontend includes direct-conversion transceivers for two channels. The digital signal processing part outputs samples, that after the A/D conversion are transmit by the transmitter. Figure 8 illustrates a high-level schematic of the USRP devices.

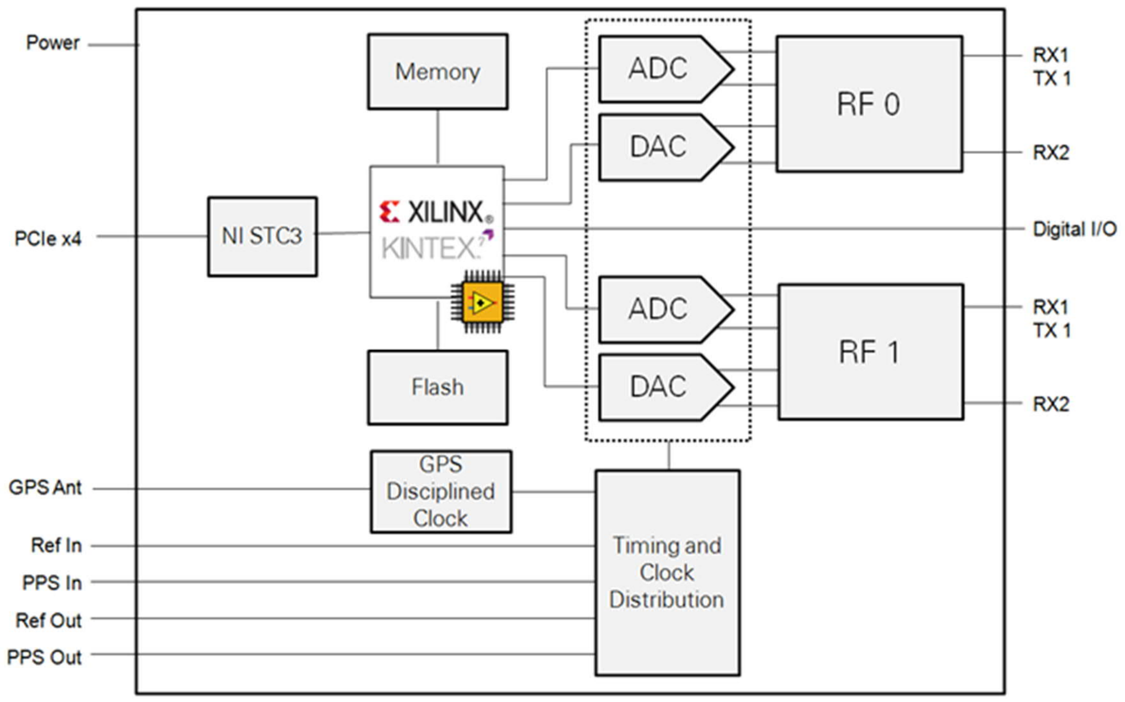


Figure 8. *High-level schematic of a USRP device [45].*

The digital signal processing in an USRP device is achieved with a Field-programmable Gate Array (FPGA). An FPGA is a digital chip that can be configured to perform wanted computations. In many ways, the FPGA is similar to a microchip, as they both can be programmed and after doing so they both can act autonomously. Modern FPGAs consist of numerous generic logic blocks, which in turn consist of generic logic cells implemented on silicon using transistors. By giving the FPGA orders on how to connect the logic cells and what kind of operation to perform on them, the functionality of the chip is defined. This makes the signals inside the FPGA propagate in parallel, and thus the FPGA is an ideal device for real-time applications, which differentiates it from microchips.

In addition to the logic cells, FPGAs also include Digital Signal Processing (DSP) blocks and Block RAM blocks. DSPs are used for more complex arithmetic, for example multiplications. Block RAM acts as internal memory for the device. The USRP contains a Xilinx Kintex-7 model XC7K410T FPGA, the specifications of which are shown in Table 1. The FPGA found in the USRP is compared to Kintex-7 model XC7325T, which can be found in a National Instruments FlexRIO device PXIe-7972, which is also programmable by LabVIEW.

Table 1. *Comparison between the FPGAs found in an USRP and PXIe-7972 devices [64].*

Element	USRP	PXIe-7972
Logic Cells	406 720	326 080
DSPs	1 540	840
Slices (4 LUTs & 8 flip-flops)	63 550	50 950
Block RAM (36Kb)	795	445

From Table 1 it can be seen that the FPGA found in USRP is superior in terms of resources to the one found in PXIe-7972 device. Especially crucial to the implementation are the DSP units, which are an important resource in digital signal processing. A memory polynomial self-interference canceller has been implemented on a PXIe-7972, and in that implementation the performance suffered from lack of available DSP units, described in [48]. Thus, the USRP offers a sufficient platform for the implementation of the self-interference canceller.

4.2 LabVIEW Communications System Design Suite 2.0

As crucial as the hardware for the implementation is its counter-part, the software. The software defines the behavior of the program by instructions on how to modify or transfer data. Although there are many alternatives, in this thesis the implementation of the self-interference algorithm is programmed with LabVIEW, more specifically with LabVIEW Communications System Design Suite 2.0. LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a *visual programming language,* developed by National Instruments. The LabVIEW language is based on a programming language called G, which is also a visual language. LabVIEW is mainly aimed at data acquisition and visualization, although the applications of the language have expanded over the years. Nowadays, there are numerous expansions and versions of LabVIEW, specifically tailored for certain types of applications.

As stated previously, LabVIEW is a graphical programming language. This means that the programmer does not have to write the wanted functionality like in text-based languages. In LabVIEW, all functionalities are defined as *functional blocks*, which are dragged and dropped on the coding platform. There are blocks for basic arithmetic, for example addition and multiplication, and more advanced functionality as digital filters and interfaces for hardware. Figure 9 illustrates a simple addition of two signed 32-bit integer variables, realized in LabVIEW.

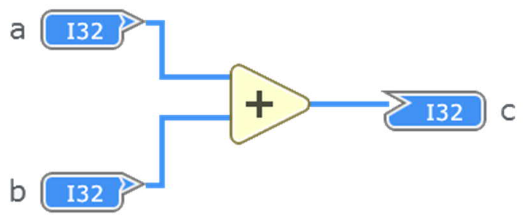


Figure 9. *Addition of two variables in LabVIEW.*

The code in Figure 9 can be realized in C++ as shown in Program 1.

```
int a;
int b;

int c = a + b;
```

Program 1. *Addition of two variables in C++.*

In text-based programming languages, the code is executed row by row as seen from Program 1, and so it is easy to keep track of the order of the operations performed. However, in LabVIEW the code is executed seemingly parallel (even though a CPU of a computer does the computation in series). In LabVIEW, a functional block is only executed when all data is available to it. This way, the user has more control over the flow of the program, yet with large programs, keeping track of the order of operations may become cumbersome.

An advantage LabVIEW has over widely used text-based languages is that it provides the user with a user interface (UI) without further programming needed. Indeed, two different views, the diagram, which defines the functionality, and the panel, which acts as the UI of the program, define a LabVIEW program. For example, for the code illustrated in Figure 9, LabVIEW automatically creates two controls for the variables *a* and *b* and an indicator for the variable *c*. Using these, the user of the program could easily find out the sum of two integer numbers. This is unlike for example in C++, where achieving this level of functionality would require substantial amount of code.

An important aspect of LabVIEW are the *Virtual Instruments (VI)*, which act similarly to functions in text-based programming languages. The VIs can thus be used to simplify the designs, which in LabVIEW can quickly become hard to read. VIs, like functions in traditional languages, make parts of the code reusable, since the same functionality can be achieved multiple times simply by calling the corresponding VI with the wanted functionality. This adds modularity to the programming tasks. The VIs are actually the parts of the program that hold both the front panel and the diagram that define the functionality of the code. On a diagram, the VIs are presented by blocks, similar to any other part of the program. *Controls* and *indicators* determine the connections of the VIs*,* which act as inputs and outputs of the VI, respectively. The VI that is run when the execution of the program starts is called the *top-level VI,* and all the VIs the top-level VI calls are called *subVIs*.

LabVIEW Communications System Design Suite 2.0 is a standalone version of Lab-VIEW NXG, aimed specifically at communications engineering. The code written in Lab-VIEW Communications is the same as in regular LabVIEW, but the functional block have an emphasis on communications engineering, with such blocks as Wave Generation and FFT Spectrum. LabVIEW Communications also has FPGA tools incorporated within it. While it is possible to control the USRP with LabVIEW Communication without the need to program the FPGA separately, in this work the FPGA is also programmed. The whole program is divided into two separate entities: the *host code* and *target code*. The host code collects and sends data to the target code, visualizes the data, and controls the operation of the target code. These operations are carried out on a computer, since the USRP lacks user-interface, and data visualization tasks are demanding and they do not need to be executed in real time. The target code is the code that is run on the FPGA of the USRP. The target code includes the transceiver code and the self-interference canceller. The target code is also written as LabVIEW code but when it is finished, it is converted into a bit file, which contains introductions for the FPGA on how to process the data. The bit file is uploaded to the USRP’s FPGA when the host code is run.

5. IMPLEMENTATION

This chapter presents the implementation of the self-interference canceller in LabVIEW Communications System Design Suite 2.0 in detail. The implementation, as discussed in Chapter 4, is written in LabVIEW Communications version 2.0, and an USRP acts as the target device. Apart from the actual code, this chapter considers issues that rise from the selected hardware and software environment and solutions to those problems. These issues include the quantization, pipelining and complex number magnitude approximation on the FPGA. Additionally, this chapter focuses on the actual SI canceller implementation on the target FPGA, but naturally, the canceller is not independent of other code. Additional code is required on the host side as well, to ensure the functionality of the SI canceller and to control the behavior of it. The other bit of code consists mainly of transceiver related functionality, both on the FPGA and host side codes. An overview of the SI canceller implementation on the FPGA is shown in Figure 10.

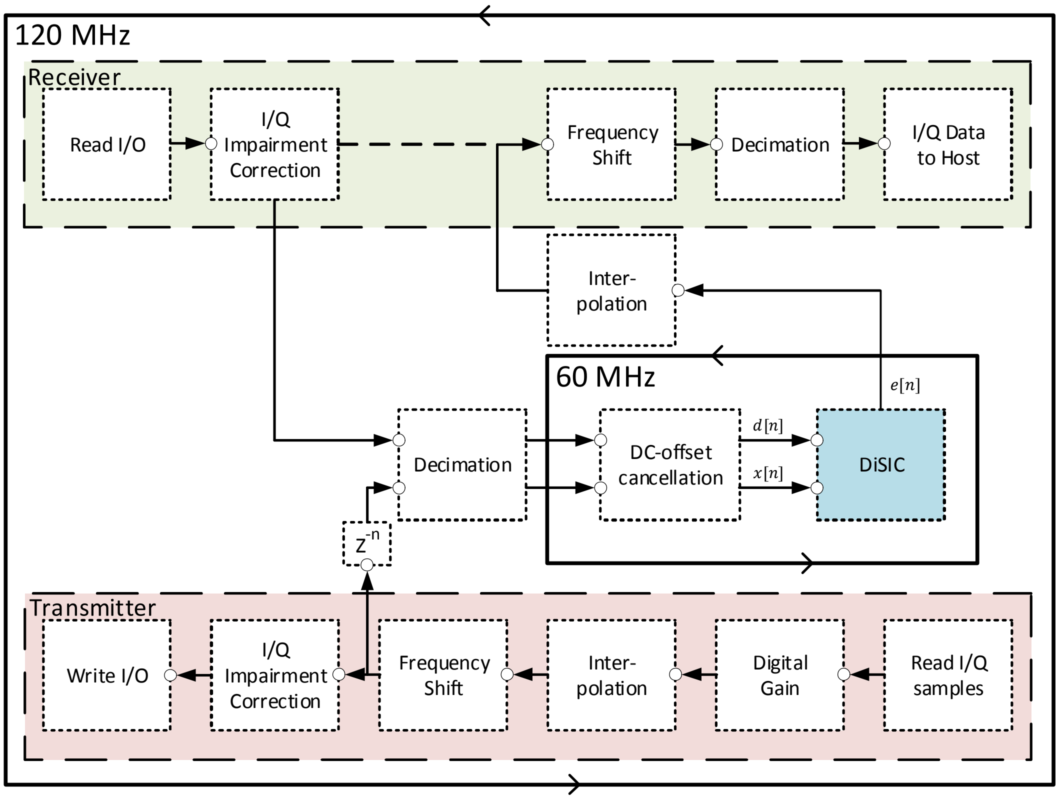


Figure 10. *Overview of the FPGA transceiver loop, complete with the SI canceller.*

Figure 10 shows that in addition to the actual canceller code, additional processing for the signals is needed. The SI canceller code, even though computationally lighter than previous such algorithms, cannot be executed completely in one clock cycle of 8.33 ns, which corresponds to the native A/D data clock at 120 MHz. Therefore, the canceller code is implemented in a 60 MHz loop, where the length of a clock cycle is 16.67 ns. The change of clock domain requires the input signals of the canceller to be decimated. The relative clock speed is chosen to be half of the original sampling frequency, so that the decimation operation is as easily implementable as possible. By choosing the SI canceller clock domain to be 60 MHz, the decimation only requires to take every other sample of the signals from the 120 MHz loop. This is sufficient for the signal $x(n)$ as it does not contain any noise or other frequencies. Input $d(n)$ needs to be properly decimated with a decimator, which includes a filter, to prevent aliasing. Conversely, the output of the canceller, the signal $e(n)$, needs to be interpolated with an interpolator back to 120 MHz. In addition to decimation, signal $x(n)$ has to be delayed to correspond to the propagation of the signal in the circuitry and the registers of the program.

Even though the canceller operates at 60 MHz, which offers considerably more operations to be executed in a single clock cycle compared to 120 MHz, not all of the required operations can be executed within one cycle. To get around this problem, the algorithm is *pipelined*. Pipelining is a programming method, where intermediate values of the algorithm are stored in internal registers of the device [36]. This way, the amount of operations required to be performed in one cycle can be mitigated and controlled. In LabVIEW Communications 2.0, pipelining can be achieved by using *feedforward* blocks, illustrated in Figure 11.

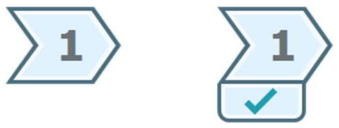


Figure 11. *Feedforward nodes in LabVIEW Communications 2.0.*

The feedforward node shown in Figure 11 holds the input value until next cycle, and the input is transferred to the output on the next cycle. The number on the node illustrates how many clock cycles the value will be delayed by. The feedback node on the right in Figure 11 is functionally the same, except it has an additional enable input. When the enable input is asserted, the block functions the exact same as the normal feedforward node. When the enable input is false, the node retains the values in the registers until enable is asserted again. This way, only valid data can be stored in the registers. Adding feedforward nodes obviously adds delay to the system, so the amount of them should be kept to a minimum, for the system to retain real-time operation.

There are two separate major physical clocks used within the USRP. These clocks are the A/D converter clock, called Data Clock and the FPGA Clock, which runs at 40 MHz. In order to make use of the I/O nodes within the code, the code has to be clocked with the Data Clock. Depending on the device, the Data Clock runs either at 120 MHz or at 200 MHz. As mentioned previously, the canceller code is run at 60 MHz, which is half of the 120 MHz that the used USRP runs the Data Clock at. However, it is only possible to derive the 60 MHz clock from the FPGA Clock, which is not necessarily in synchronization with the Data Clock. Thus, the canceller code needs to be run slightly faster than half of the Data Clock rate, 62 MHz in this work, in order to ensure that no data is dropped between the cycles. By doing so, the canceller loop will have non-valid values every now and then. Therefore, the data will have a handshaking protocol on it, to tell the upstream nodes whenever data is valid and can propagate forward in the code. The handshaking in the canceller code is implemented using feedforward nodes with enable. Furthermore, to prevent false values from affecting the updates of $\mathbf{w}$ and $\mathbf{q}$, the update only occurs when the first valid value has propagated through the algorithm, which in the implementation occurs after seven clock cycles. This is achieved with the handshake protocol in the program, which propagates an 'input valid' signal throughout the code.

A major general challenge in the implementation is, in addition to the timing, the quantization. As discussed in Section 2.2, in a digital environment the values of the algorithm are always of finite length, which gives rise to some problems. To ensure that the errors from the quantizations are low, as many bits should be used for the values of the signals as possible. There are limiting factors to this however, most notably the maximum bit widths of the DSP blocks in the FPGA. A single DSP unit in a Kintex-7 device can handle at maximum values that are 18 and 25 bits long. In order to avoid overflowing and thus using unnecessarily many DSPs, the inputs of the multiplication operations should conform to these bit widths. In addition to the DSPs limiting the bit widths, the maximum bit width that the FPGA can handle is 64 bits. The inputs are 16-bit signed integers, so the output should be the same type as well, which further restricts the possible values. Manipulating the values within the code is achieved by using reinterpret and type cast nodes, illustrated in Figure 12.

Figure 12. *Reinterpret, cast to fixed-point and cast to complex fixed-point nodes in LabVIEW Communications 2.0.*

The reinterpret node takes the bits that represents the input value and changes the interpretation of the value, without changing the bits. Thus, the input and output of the node have to have the same amount of bits. The casting nodes on the other hand take the value the input represents and change the type of value, so that, ideally, the value represented stays the same but the bits change. In the implementation, the cast nodes are mostly used to limit the bit widths of the values. In these situations, the output value cannot entirely match the accuracy of the input, since the bit widths are smaller on the output. To mitigate errors rising from this, the output values are rounded by using the half-to-even scheme.

Last general issue with LabVIEW in the implementation is the use of matrices. Even though LabVIEW Communications FPGA tools allow the use of two-dimensional matrices, these tools are cumbersome and difficult to use. The solution for this is to reduce the matrix operations of the SI canceller algorithm to corresponding 1-D array operations. This saves resources on the FPGA and gives more control for the user on the actual operations. It should be noted here, that in LabVIEW Communications array operations are always element-wise.

5.1 Complex Number Magnitude Approximation

For the purposes of the implementation of the algorithm described in Section 3.3, the absolute value or magnitude of a complex number has to be determined. The magnitude of a complex number c can be determined by the well known formula $|c| = \sqrt{a^2 + b^2}$, where a is the real part, and b is the imaginary part of the complex number. The issue of this computation is the demanding square root operation [42, p. 479]. On a computer, this operation is trivial, but on an FPGA or other resource scarce platforms, the operation has to be simplified, since the standard way of defining the square root is by using iterative methods, such as the Newton-Raphson algorithm [51]. Naturally, the reduction of the computation of the magnitude optimizes the performance even on platforms that have resources to spare.

In this thesis, the magnitude of a complex number will be determined by using the *αMax+βMin technique*, described for example in [42, pp. 480—482] and [12]. As the name suggests, with this technique the magnitude of a complex number c is approximated by

$$|c| \approx \alpha \cdot \max(|Re\{c\}, Im\{c\}|) + \beta \cdot \min(|Re\{c\}, Im\{c\}|), \tag{46}$$

where α and β are constants. By using the αMax+βMin technique, the demanding square root operation is now reduced to mere multiplications and additions, which greatly reduces the complexity of the computation compared to iterative algorithms. The technique requires the comparison of two real numbers, which is a straightforward operation, even on resource scarce platforms. The constants are assigned values $\alpha = 0.96043387$ and $\beta = 0.397824735$, which yield the least amount of error for an equiripple implementation, that utilizes only one value per constant [12]. These values yield an error for the approximation that is less than 4 %. The difference between the actual magnitude and the αMax+βMin approximation of a complex number is illustrated in Figure 13.

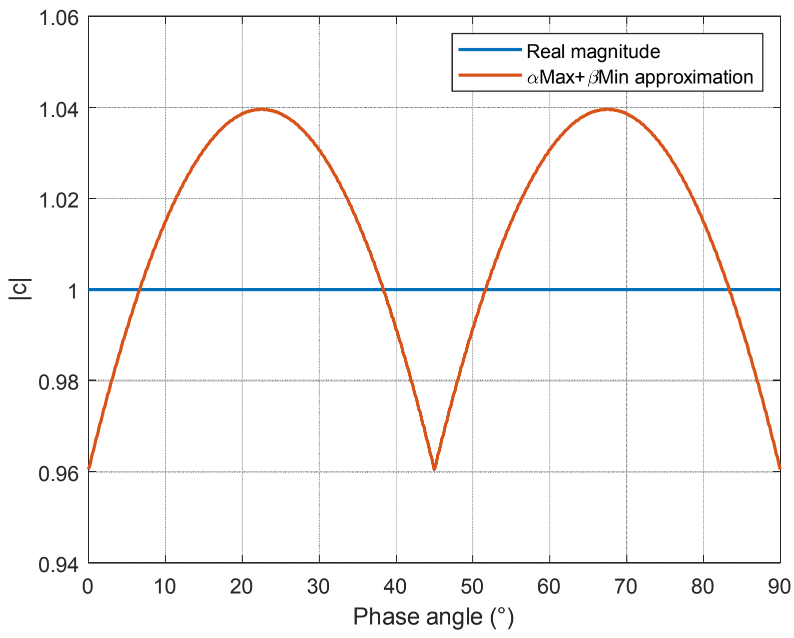


Figure 13. *The real magnitude and the αMax+βMin approximation of the magnitude of a normalized complex number.*

Figure 13 shows the magnitude of a complex vector that has unity magnitude, which is illustrated in the figure as a constant line. The magnitude is plotted against the phase angle of the complex number, and the curve will repeat every 90°. The same unity magnitude vector's magnitude is approximated with the αMax+βMin technique drawn in red. It can be seen from Figure 13 that the error of the approximation is indeed within 4 % of the actual value of the magnitude. The mean absolute error of the αMax+βMin technique is 2.4 % and the mean error is 1.3 % with the chosen parameter values. Therefore, the technique is suitable for usage in the implementation.

5.2 Delay Estimation

In order to cancel the SI signal from the received signal, the input signals d and x of the canceller have to have corresponding samples. This problem is trivial in a simulation environment, where timing restriction do not apply. However, in a real life implementation, various parts of the program and circuitry, such as registers and A/D conversion, add *delay* to the received signal. To cancel this effect, the transmitted signal has to be delayed accordingly before the samples are fed to the canceller.

Determining a delay between two correlating signals is a straightforward process. The *cross-correlation* of the signals tells about the similarity of two signals. Cross-correlation between discrete complex valued signals F and G is defined as

$$(F \star G)(l) = \sum_{m=-\infty}^{\infty} F^*(m)G(m+l) \tag{47}$$

where l denotes lag that is introduced to the signal G and F^* the complex conjugate of F. In practice, cross-correlation means that the signal G is delayed by one sample at a time and the product between F^* and G is calculated every iteration. The more similar the two signals, the higher the maximum value of the cross-correlation and the point of the maximum value of the cross-correlation denotes the lag between the two signals [27].

5.3 Target Code

This section introduces the FPGA implementation of the SI canceller in more detail. The 62 MHz loop, where the canceller is located is shown in Appendix A. In the canceller loop, the signals $x(n)$ (presented as 'x[n]') and $d(n)$ (presented as 'd[n]') are read from a joined FIFO 'DX_FIFO', carrying unsigned 64-bit values. The 32 upper bits of these represent $d(n)$ and lower 32 bits $x(n)$. The joined values are immediately written to a target-to-host FIFO 'dx to host'. This FIFO is used on the host side to determine the delay between the signals and the DC-offset of the received signal. Before entering the canceller, $d(n)$ is delayed in a Discrete Delay block using the input 'M1', to correspond to the pretaps of the linear filter. Using the 'Canceller on 60M?' control, the user can control whether the samples propagate to the canceller or if the received signal is just immediately transferred back to the 120 MHz transceiver loop. The values are sent back using the 'E_FIFO' FIFO. To validate the performance of the canceller, the same samples are also sent to the host by using the 'e to host' FIFO.

The SI canceller is presented in Appendix A as the gray block with a wave and a red cross, within the case structure. The diagram of the SI canceller is illustrated in Appendix B. The operation of the canceller is divided into four distinct functional blocks for convenience. The 'Splines Estimation' subVI calculates the spline interpolation of $x(n)$ ('x in') and gives out $s(n)$ (presented as 's[n]'), which is the input of the 'Linear Filter' subVI. The linear filter determines the error signal $e(n)$ and updates the filter coefficients. Finally, with the error signal, the spline control-points are updated in the 'Splines Update' subVI.

The splines interpolation is the first operation that is performed on the signal $x(n)$. The splines interpolation is achieved with the 'Splines Estimation' subVI, presented in Appendix C. First, the single unsigned 32-bit value 'x in' representing signal $x(n)$ is split into two 16-bit values, that are the I- and Q-branches of the signal. The branches are then reinterpreted as <4.12> signed fixed-point values. The notation <4.12> denotes that the number's presentation has 4 signed bits, in this case including the signed bit, and 12 fractional bits. This conversion is performed in order for the algorithm to use fixed-point arithmetic and for the implementation to comply with the MATLAB model made of the algorithm. The separate fixed-point values are then joined to make a single <4.12> complex fixed-point value.

In the top branch of the ‘Splines Estimation’ subVI, the absolute value or magnitude of the complex value $x(n)$ is determined using the αMax+βMin method introduced in Equation (46). The absolute values of the real and imaginary parts are determined and they are compared with each other and multiplied accordingly. The magnitude of $x(n)$ is then used to determine i and u after (37) and (38), respectively. The value for u is further used to determine u^2 and a five element array of $[u^2\ u^2\ u^2\ u\ u]$ is created. This array will be the input of the ‘Splines Row’ subVI. The ‘Splines Row’ subVI additionally takes i as an input. The output of the ‘Splines Row’ subVI is the new 11 element column for the splines matrix $\mathbf{\Sigma}(n)$. In order to save resources and reduce the complexity of the algorithm, the rows are just delayed according to input ‘w_ind’, which will be explained later. This way, the rows are stored in registers instead of a matrix until they are needed for the calculations. The other outputs of the ’Splines Row’ subVI are the three non-zero elements of the calculated row in an array. This array is fed to the lower branch of the subVI.

In the lower branch of the ‘Splines Estimation’ subVI the complex value of $x(n)$ is stored as the first element of regressive array ‘x_n’, holding past values of $x(n)$, the amount of which is determined by the register ‘M’. The array ‘x_n’ represents the diagonal matrix $\mathbf{X}(n)$ without the zeros. All computations in the implementation are adjusted accordingly. The array is fixed sized, capable of holding 60 elements, but the apparent size is controlled by the register ‘M’, by adding zero values after index indicated by ‘M’. At the same time, the three values from the control-point array ‘q_i’ corresponding to i are taken with the ‘q subarray’ subVI. The subVI simply takes three consecutive values from the given array, starting from the given index. In LabVIEW, indexing starts from 0. After taking the corresponding values, the splines output $s(n)$ is determined after Equation (39). The splines output is cast as <5.13> type complex fixed-point value. The calculation is simplified by omitting the zero values from the $\mathbf{\Psi}(n)$ array. Since the array multiplication is element-wise, the resulting array’s elements are added up with a Sum Array Elements block. This result is finally multiplied with $x(n)$ to produce the splines output $s(n)$.

The calculation of the new spline matrix column in the ‘Splines Row’ subVI is presented in Appendix D. The calculation of $\mathbf{u}^T\mathbf{C}$ is performed in this subVI. From Equations (29) and (30) it follows that $\mathbf{u}^T\mathbf{C}$ is defined as

$$\mathbf{u}^T\mathbf{C} = [u^2\ u\ 1]\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \left[\left(\frac{u^2}{2} - u + \frac{1}{2}\right)\ \left(-u^2 + u + \frac{1}{2}\right)\ \left(\frac{u^2}{2}\right)\right]. \quad (48)$$

The first operation in the subVI is multiplication of the five element array ‘u array’, which is described above, and a constant array corresponding to the non-zero values of matrix $\mathbf{C}$. The output of the multiplication is a five element array $\left[\left(\frac{u^2}{2}\right)\ (-u^2)\ \left(\frac{u^2}{2}\right)\ (-u)\ (u)\right]$. These elements are then taken accordingly to create a three element described in Equation (48). This three-element array is the ‘splines reduced’ output. For the ‘splines row’ output, the $\mathbf{u}^T\mathbf{C}$ array is inserted into an 11-element array of zeros at index determined by i, thus creating the vector $\mathbf{\Psi}(n)$, which is the new column for matrix $\mathbf{\Sigma}(n)$ as well.

On the FPGA, all array lengths have to be fixed. This is why the FPGA code does not support the Array Subset functional block, as the output array can have varying sizes. Instead, taking a subarray has to be implemented as seen in 'q subarray' subVI in Appendix E. In the 'q subarray' subVI, three elements from the array $\mathbf{q}$ are taken separately, starting from the input 'index'. After the elements have been taken, they are used to form a three-element array, which acts as the output 'subarray'. With this approach, even if the input 'index' exceeds the length of the input array 'q', the returned elements are just zeroes, and the output array has a fixed length of three. This is unlike using the Array Subset block, which would return an array with less elements.

After the 'Splines Estimation' subVI, the splines output $s(n)$ propagates to the subVI 'Linear Filter'. The subVI applies the linear filtering to the signal, and updates the filter taps. This subVI is illustrated in Appendix F. First, the signal $s(n)$ is placed in a regressional array $\mathbf{s}(n)$, indicated by 's_n'. Again, the array has a fixed length of 60 elements, but the apparent size can be altered with the register 'M', similarly to the 'x_n' array in 'Splines Estimation' subVI. The array 's_n' is then multiplied with the conjugate of filter taps array 'w_n' according to Equation (34), to produce the model output signal $y(n)$. To produce this product, the complex conjugate of every element in the filter tap is taken, and the element-wise products are added together to produce a single value. The summation of the element-wise product array elements is achieved with the 'Sum Array Elements' subVI. The output $y(n)$ is then subtracted from the desired signal $d(n)$ to produce the error signal $e(n)$. Before this, the DC components of both I- and Q-branches are cancelled by a simple subtraction. The signal $d(n)$ (presented as 'd in') is, similarly to $x(n)$, interpreted as signed <4.12> complex fixed-point number. After the error signal is determined, in the top branch it is fed as is to the output 'e(fxp) out'. Below this, the full accuracy error signal is cast as signed <4.12> complex fixed-point value. This value is further split into its real and imaginary parts, which in turn are reinterpreted as signed 16-bit numbers, to follow the number representations of the inputs. Finally, the separate branches are joined to form a single unsigned 32-bit value, where the high bits represent the imaginary (Q) part and low bits the real (I) part of the error signal.

The update of the filter taps is performed as described in Equation (36). The fully accurate error signal is first conjugated and bit shifted according to the input 'mu_w'. By substituting the multiplication $\mu_w e^*(n)$ with a bit shift, DSPs are saved for other operations. After this, the remaining calculations for the update are performed, with multiplication of $\mu_w e^*(n)$ with the corresponding array $\mathbf{s}_n$, indicated again by 's_n', and finally adding the result to $\mathbf{w}(n)$, indicated by 'w_n'. The result is applied to the register 'w_n' only when the inputs 'update w?' and 'input valid' are asserted. If the user so desires, the update of the filter coefficients may be stopped with the 'update w?' control, which is set in the host code. Furthermore, it is possible to reset the array of $\mathbf{w}(n)$ by writing true to the 'reset q and w' register, which is read within the case structure where update of $\mathbf{w}(n)$ is applied.

In order to determine the product of $\mathbf{w}(n)^H\mathbf{s}(n)$, the elements from the element-wise multiplication have to be added together. This operation is performed in the 'Sum Array Elements' subVI, shown in Appendix G. LabVIEW provides the user with a built-in function that sums the array elements together, which is also utilized in the 'Sum Array Elements' subVI. The issue with using this functional block to determine the sum of the array elements at once is that it provides overhead, and the operation cannot be scheduled for larger arrays, even at relatively low frequencies. Thus, in order to be able to use the full 60 elements of the $\mathbf{w}(n)$ and $\mathbf{s}(n)$ arrays, the summation is divided into 12 parallel sums of five element subarrays. Here, the Array Subset functional block may be utilized, as the starting indices of the subarrays are fixed. The sums of the subarrays are then added together, to determine the total sum of the elements of the array, thus producing $\mathbf{w}(n)^H\mathbf{s}(n)$.

After the error signal is determined in the linear filter, the signal is used to update the spline control points in the 'Splines Update' subVI. In addition to the error signal, the 'Splines Update' subVI takes the current value of $\mathbf{w}(n)$, indicated as 'w_n in', as an input from the 'Linear Filter' subVI, and $\mathbf{q}(n)$ and $\mathbf{X}(n)$, indicated respectively as 'q_i in' and X in, from the 'Splines Estimation' subVI. The 'Splines Update' subVI is illustrated in Appendix H. The update of the control points is determined in accordance to Equation (45). In the implementation, the calculation of matrix multiplication $\mathbf{\Sigma}(n)\mathbf{X}^*(n)\mathbf{w}(n)$ is simplified by only considering five most significant taps of the filter. The most significant taps are usually around the $M_{pre} + 1$ tap [47], so the first the taps $M_{pre} - 1$, M_{pre}, $M_{pre} + 1$, $M_{pre} + 2$ and $M_{pre} + 3$ are taken from the $\mathbf{w}(n)$ array with the 'w subarray' subVI. The subVI is similar to 'q subarray' but instead of two consecutive values of the index input, five consecutive values are taken instead to form a five element array. The corresponding values are taken from the $\mathbf{X}(n)$ array similarly with the 'X subarray' subVI. The values to be taken are determined by the input 'w_ind'. Similarly to the control 'M1' in the canceller loop, 'w_ind' is controllable from the host, and it is determined to be $M_{pre} - 1$. Therefore, the five element subarrays from and the array 'x in' correspond to the wanted values near the most significant tap. The subarray of 'x in' is conjugated and multiplied with the subarray of 'w_n' to produce a reduced version of $\mathbf{X}^*(n)\mathbf{w}(n)$, with only five elements. The product is further multiplied with $\mathbf{\Sigma}(n)$ in 'Splines x X*w' subVI to finally produce $\mathbf{\Sigma}(n)\mathbf{X}^*(n)\mathbf{w}(n)$. Meanwhile, the error signal is bit shifted the same as in the 'Linear Filter' subVI to produce $\mu_q e(n)$, instead of multiplying it with μ_q in order to save DSPs. The remaining steps of the update are applied by multiplying $\mathbf{\Sigma}(n)\mathbf{X}^*(n)\mathbf{w}(n)$ with $\mu_q e(n)$, the product of which is then added to the current values of the control points. Similarly to the filter tap update in 'Linear Filter' subVI, the update of the control points is controlled by 'update q?' input, which is asserted from the host code. Additionally, the 'input valid' input has to be asserted for the update to take place. It is also possible to reset the control points with the 'reset q and w' register, the same as with the filter taps.

The calculation of $\mathbf{\Sigma}(n)\mathbf{X}^*(n)\mathbf{w}(n)$ is performed in the subVI 'Splines x X*w' is presented in Appendix I. The calculation can be simplified as stated earlier, if only values near the most significant tap are taken into account. This is why it is sufficient to keep track of five columns of the matrix $\mathbf{\Sigma}(n)$. The columns of the spline matrix $\mathbf{\Sigma}(n)$ are regressional, which means that a new column calculated in the 'Splines Estimation' subVI will be the new first column of the matrix, while the last column will be discarded. The new column is of no interest immediately however and thus it is delayed in the 'Splines Estimation' subVI, until it corresponds to the five columns near the most significant tap of the filter. The columns of the spline matrix are stored in registers 'splines_1' through 'splines_5' and the columns are regressed by writing the array from the register to the next after it is read. The reduced calculation of $\mathbf{\Sigma}(n)\mathbf{X}^*(n)\mathbf{w}(n)$ can be written as:

$$\begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} & \Sigma_{1,3} & \Sigma_{1,4} & \Sigma_{1,5} \\ \Sigma_{2,1} & \Sigma_{2,2} & \Sigma_{2,3} & \Sigma_{2,4} & \Sigma_{2,5} \\ \Sigma_{3,1} & \Sigma_{3,2} & \Sigma_{3,3} & \Sigma_{3,4} & \Sigma_{3,5} \\ & & \vdots & & \\ \Sigma_{11,1} & \Sigma_{11,2} & \Sigma_{11,3} & \Sigma_{11,4} & \Sigma_{11,5} \end{bmatrix} \begin{bmatrix} xw_1 \\ xw_2 \\ xw_3 \\ xw_4 \\ xw_5 \end{bmatrix} = \tag{49}$$

$$\begin{bmatrix} \Sigma_{1,1}xw_1 + \Sigma_{1,2}xw_2 + \Sigma_{1,3}xw_3 + \Sigma_{1,4}xw_4 + \Sigma_{1,5}xw_5 \\ \Sigma_{2,1}xw_1 + \Sigma_{2,2}xw_2 + \Sigma_{2,3}xw_3 + \Sigma_{2,4}xw_4 + \Sigma_{2,5}xw_5 \\ \Sigma_{3,1}xw_1 + \Sigma_{3,2}xw_2 + \Sigma_{3,3}xw_3 + \Sigma_{3,4}xw_4 + \Sigma_{3,5}xw_5 \\ \vdots \\ \Sigma_{11,1}xw_1 + \Sigma_{11,2}xw_2 + \Sigma_{11,3}xw_3 + \Sigma_{11,4}xw_4 + \Sigma_{11,5}xw_5 \end{bmatrix}, \tag{50}$$

where Σ represents the elements in the spline matrix $\mathbf{\Sigma}(n)$, and xw the elements in the reduced array $\mathbf{X}^*(n)\mathbf{w}(n)$. Since only the columns of the spline matrix are stored in registers, the rows have to be formed with the build array blocks within the subVI. The five element rows are then multiplied element-wise with the reduced $\mathbf{X}^*(n)\mathbf{w}(n)$ array. The elements of the resulting array are then summed together, to form the eleven elements of the resulting array in (50). In the final step, the elements are built into an array that is passed as the output of the subVI.

The Hammerstein spline-based SI canceller is a low complexity approach to the SI cancellation problem. The reduced complexity is highlighted mainly in the comparatively low number of DSP units required for the implementation. The resource utilization of the algorithm implementation is shown in Table 2.

Table 2. *Resource usage of the SI canceller on the FPGA.*

Element	Total	Used	Percentage (%)
Registers	508 440	37 778	7.3
DSPs	1 540	533	34.6
Block RAM	795	3	0.4
LUTs	254 200	67 632	26.6
Total Slices	63 550	22 183	34.9

Table 2 shows that with 60 taps of memory in the linear filter, the implementation uses only 34.6 % of the available DSP. This value could be easily reduced for more resource scarce platforms, as the performance of the canceller does not greatly decline even with reduced memory. Additionally, the amount of LUTs, which handle the logic operations on the FPGA, is at acceptable levels with only 26.6 % of the available ones used by the algorithm. Even though the implementation utilizes the pipeline technique and values such as the spline matrix row, linear filter taps and spline control points are stored in register, the register usage of the algorithm is merely 7.3 %.

5.4 Host Code

The host code running on a desktop computer provides the user-interface of the FPGA program, since the USRP does not provide direct interaction with the user. The purpose of the host code is to control the program on the FPGA and visualize measured data that is transferred to the computer. The control and visualization of the transceiver code and related data is omitted here, and the focus is on the SI canceller. Most of the variables mentioned in Section 5.3 are assigned values with the Write FPGA Control functional block, which as the name suggests, writes values to the control values of the FPGA VI. This is unlike regular LabVIEW, where instead of controls, the variables have to be assigned using other means. The controllable values are shown in Appendix J, along with the user-interface of the program. The left side of the UI is dedicated to controlling the transceiver and visualizing data read from the FPGA, while the right side controls the SI canceller. The visualized data includes a spectrum and a received power plot. The memory, pretaps and the learning rates of the SI canceller can be assigned values to the users liking, but in order to ensure the best performance of the canceller, the delay between transmit and received signals need to be known exactly. Furthermore, the DC-offset of the received signal needs to be mitigated. These issues are handled in a host code loop, illustrated in Appendix K. The delay between the transmit and received signal is determined by using the cross-correlation, introduced in Equation (47). The length of the read signals is subtracted from index of the maximum value, which gives the delay between the two signals. The user then changes the ‘x Delay’ control, until the delay is shown to be zero. The DC-offset of the received signal is determined simply by determining the mean of the signal. In order to avoid abrupt changes, the previous values affect the measured DC-offset according to

$$DC(n+1) = \delta DC(n) + (1-\delta)\bar{d}, \tag{51}$$

where DC is the determined DC-offset, δ is a value between 0.9 and 0.99, and $\bar{d}$ is the mean of the received signal. Value of 0.95 is chosen for δ in this implementation. The mean of the received signal is determined from a set of 65536 samples. This is the depth of the FIFO, which transfers the data between the FPGA and the host PC. The DC-offset is determined separately for the real (I) and imaginary (Q) parts of the signal. As opposed to the delay correction of the signal $d(n)$, the measured DC-offset values are written to the FPGA every cycle of the loop automatically.

6. EXPERIMENTS AND RESULTS

The functionality of the self-interference canceller implementation introduced in the previous chapter is validated with measurements in this chapter. The verification of the performance is divided into two main parts: functional validation, which validates the SI cancellation capabilities of the canceller in three scenarios, and full-duplex operation, which studies the system's ability to function in an actual full-duplex communication scenario. As mentioned previously in Section 2.3, in order to achieve sufficient cancellation of the SI for the full-duplex system to be feasible, both analog and digital cancellation methods have to be utilized. In the measurements, the digital canceller is fixed as the one described in the previous chapter, running on USRP-2953R devices' FPGAs. The analog cancellation is achieved by two different methods: two separate antennas and an RF canceller, which also incorporates a circulator, allowing the use of a single antenna. The two setups are presented in Figure 14.

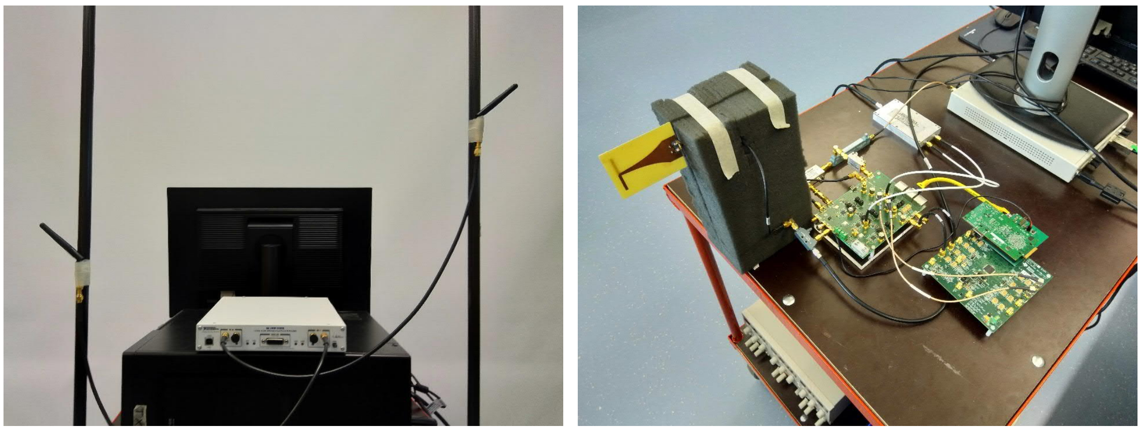


Figure 14. *The two setups used in the measurements: the two-antenna node (left) the RF canceller node (right).*

The two-antenna setup achieves the analog cancellation of the SI by adding isolation between the transmitter and receiver chains. In the setup, the isolation is approximately 33 dB. The antennas are commercial off-the-shelf monopole antennas designed for 2.4 GHz ISM band. In the RF canceller setup, the analog cancellation is achieved with the analog RF canceller. The RF canceller is an adaptive one, and it is controlled with a laptop. Like the digital canceller, the RF canceller estimates the effects of the channel and nonlinearities, and subtracts the estimate of the SI signal in a combiner. The RF canceller has three taps of memory. The RF canceller is connected to a circulator, which in turn is connected to a designated custom dipole antenna, which is also designed for 2.4 GHz ISM band. The RF canceller is capable of suppressing the SI signal by up to 48 dB, and in addition, the use of circulator adds around 20 dB isolation between the transmission and reception chains, making the maximum amount of total achievable isolation in the order of 70 dB [59].

6.1 Signal-to-interference-plus-noise Ratio, Symbol Error Rate and Sum Rate

For the purposes of analyzing the performance of the full-duplex system, three figures of merit are introduced. These figures are the signal-to-interference-plus-noise ratio (SINR), symbol error rate (SER) and channel capacity. In order to utilize the metrics, the signal has to carry *symbols* as its payload. To this end, QAM-signals are used. An M-QAM can carry M different symbols, which make up the *alphabet* of that particular QAM-signal. The symbols in the alphabet represent $log_2(M)$ bit binary numbers. The alphabet can be presented in a *constellation,* which a presentation of the symbols in Cartesian coordinates. The coordinates are translated into complex values, which allows the use of I/Q-modulation for the symbols [65, pp. 447—481]. For example, in a 16-QAM signal, the alphabet holds 16 different symbols, and the symbols are 4 bit long. Furthermore, the constellation has 16 points, all of which are assigned a different symbol they represent. An example of a 16-QAM signals alphabet is presented in Figure 15, with possible assigned symbols shown on top of the points.

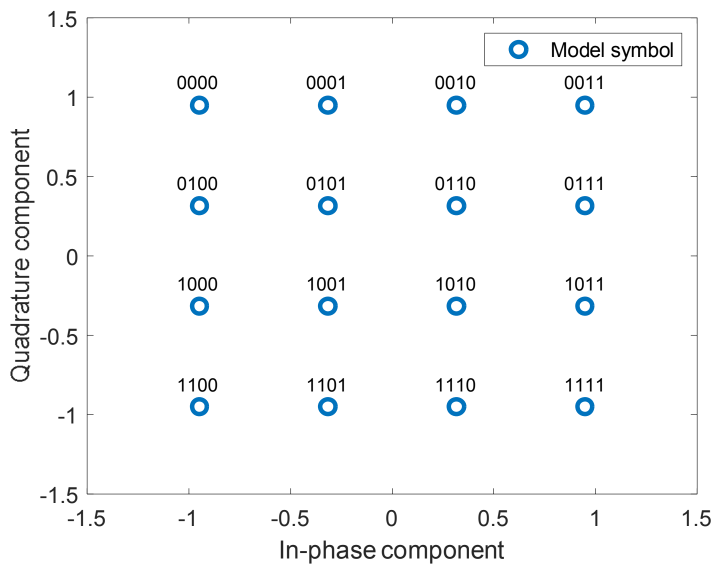


Figure 15. *Constellation of a 16-QAM alphabet, with the symbols shown.*

SINR can be estimated by the magnitude of the error of the received symbols compared to the ideal ones. Ideally, the samples sent by the transmitter would be received on the receiver side with only channel effects on the samples. However, the received signal also includes noise and interference [66], and the samples are distorted even further. Since the system usually transmits and receives more than one symbol, the SINR can be defined as the average of the deviations from the ideal values. An estimate of the SINR in decibels can be therefore be formally written as

$$\mathrm{SINR} = 10log_{10}\left(\frac{\frac{1}{N}\sum_{k=1}^{N}\left|s_{i,k}\hat{h}_k\right|^2}{\frac{1}{N}\sum_{k=1}^{N}\left|s_{i,k}\hat{h}_k - s_{r,k}\right|^2}\right)\ (\mathrm{dB}), \tag{52}$$

where N is the amount of symbols sent and received, $S_{i,k}$ the ideal symbol at index k, $S_{r,k}$ the received symbol at index k and $\hat{h}_k$ the channel estimate for the kth symbol. It should be noted, that the SINR figure is indeed an estimate, as the calculation of it relies on determining the channel response, which in itself is an approximation. The channel can be estimated using the received and transmit signals, using for example the least squares method, which gives more accurate results than LMS. Assuming the channel model is adequate, $S_{i,k}\hat{h}_k$ presents the received symbols without any interference or noise. Since these effects are inevitably contained within the real received signal, determining $S_{i,k}\hat{h}_k - S_{r,k}$ only leaves the interference and noise, assuming these are the only distorting factors in the system. Higher values for SINR denote better performance, as the magnitude of the error induced by interference and noise gets smaller.

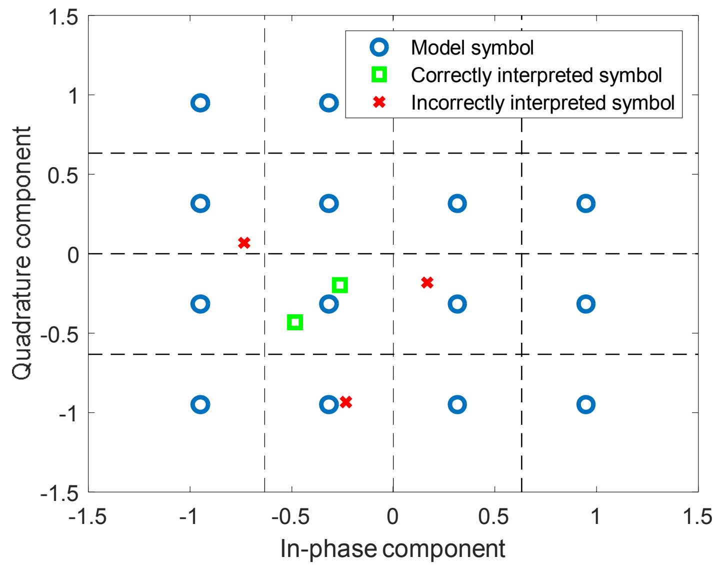


Figure 16. *Constellation of potential received symbols, supposed to represent symbol 1001 in Figure 15.*

SER is a measure of the frequency of errors in the constellation of the received symbols. An error in the constellation occurs when a received symbol is misinterpreted. The interpretation of the symbols can be achieved simply by dividing the constellation into squares, centered at the ideal symbols. Now, if the received symbol is not located within the square of the intended symbol, the received symbol is interpreted incorrectly, while symbols within the intended square are correctly interpreted [65, pp. 447—481]. SER can be defined as the portion of the deficits compared to the total number of symbols received. Formally, it can be expressed as:

$$\mathrm{SER} = \frac{N_\mathrm{e}}{N}, \tag{53}$$

where N_e is the amount of misinterpreted symbols. Figure 16 illustrates a constellation of five possible symbols being interpreted in the alphabet introduced in Figure 15. Supposedly, all of the received symbols should represent the symbol 1001, but only the symbols marked as green squares achieve this, while the red crosses are misinterpreted. Assuming only these five symbols were transmit, the SER of this particular system would be 0.60, since, out of five symbols, three were misinterpreted.

Finally, the channel capacity C for a single node can be determined with Shannon's capacity theorem [16, p. 100]:

$$C = log_2\left(1 + 10^{\frac{\text{SINR(dB)}}{10}}\right) \text{ (bits/s/Hz)}, \tag{54}$$

where SINR(dB) is the measured SINR figure of the node in decibels. The capacity presented in (54) is normalized, as it does not take into account the bandwidth of the system. Considering only two nodes with capacities C_1 and C_2 in the system and assuming the two devices transmit exactly half of the time, the sum rate or the total capacity in half-duplex operation, C_{HD}, can be determined as $C_{HD} = \frac{1}{2}(C_1 + C_2)$. Similarly, assuming that both nodes transmit constantly in full-duplex operation, the sum rate in this scenario can be written as $C_{FD} = C_1 + C_2$, where C_{FD} is the total capacity in full-duplex operation.

6.2 Functional Validation

The aim of the functional validation test is to validate that the digital canceller is capable of cancelling the SI from the received signal. To this end, the digital canceller is measured in two distinct scenarios, first being the two-antenna node and the other the RF canceller node, introduced earlier. The measurements with the two-antenna node and the RF canceller fully utilized are actual operation environments for the canceller, and thus display the performance of the implemented algorithm. The experiments were conducted in a laboratory environment, by placing the node under test in the center of the room, in order to minimize reflections from other objects. The recorded data consists of the received signal d right before the digital canceller and the error signal e just after the canceller, as illustrated in Appendix A. The measurements were conducted in 2.4 GHz frequency using 10 different transmitter (TX) powers, from 0 dBm to 18 dBm in intervals of 2 dBm. The signals used are 10 MHz bandwidth random OFDM signals, and they are frequency shifted to a digital intermediate frequency (IF) of -15 MHz. The payload consists of 16-QAM symbols.

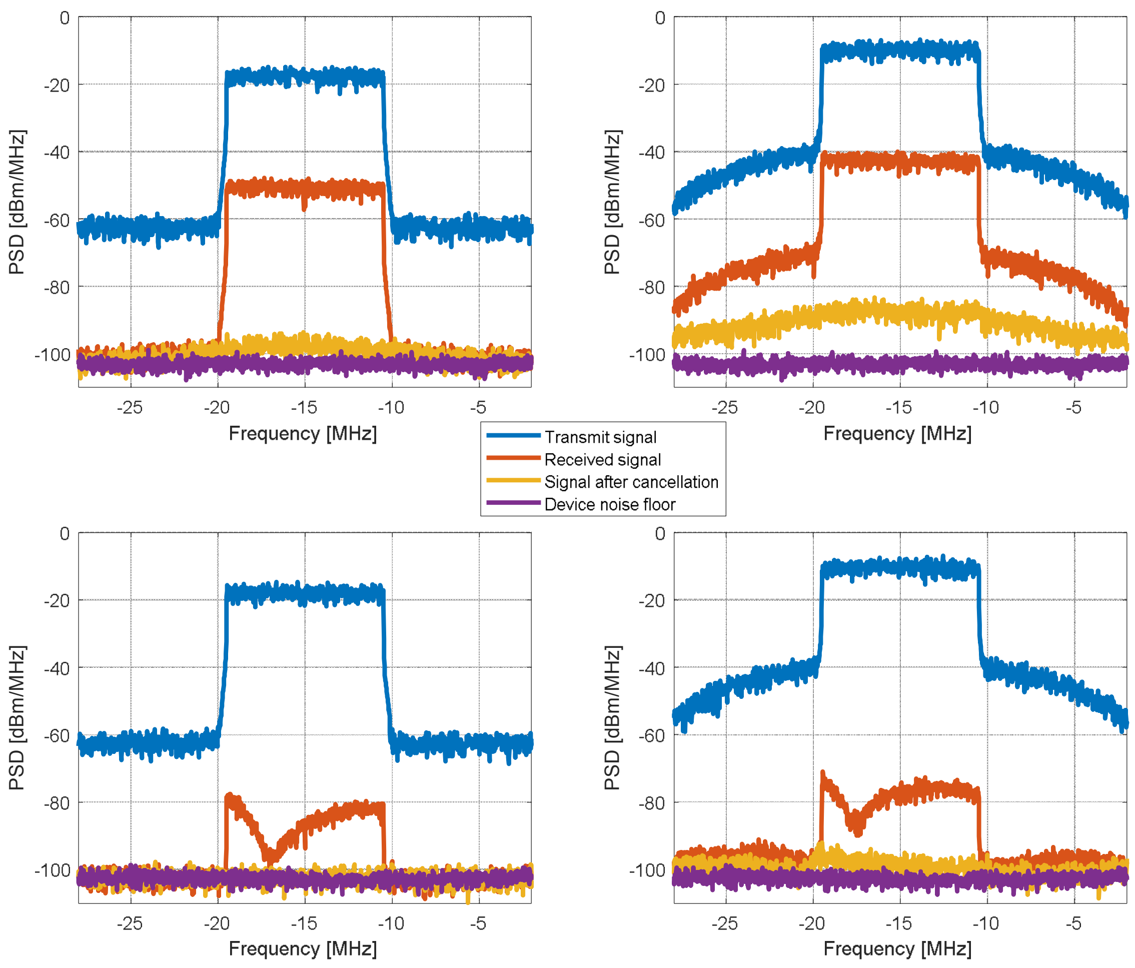


Figure 17. *Power spectral densities in four measurement cases: Two-antenna node with 8 dBm TX power (top left) and 16 dBm TX power (top right), and RF canceller node with 8 dBm TX power (bottom left) and 16 dBm TX power (bottom right).*

Four power spectral densities of the measurements are shown in Figure 17, with two transmit powers (8 dBm and 16 dBm) for both of the nodes. The plots illustrate the transmit signal of the node, the received signal before the digital cancellation, the signal after the cancellation stage, and the device noise floor for reference. The spectra in Figure 17 show that the digital canceller is capable of providing further cancellation to the SI signal in addition to the analog cancellation. Moreover, the measurements with 16 dBm transmit power illustrate that the canceller indeed suppresses the nonlinearities of the received signal. This further improves the overall inband cancellation result. Since the RF canceller is capable of cancelling the nonlinear effects of the received signal as well, the cancellation of the nonlinearities in the RF canceller node with 16 dBm transmit power is less impressive. Still, the digital canceller is able to provide additional cancellation on top of the analog one, reducing the residual power of the SI canceller close to the noise floor. In the two-antenna node, the residual inband power is still approximately 17 dB over the noise floor, yet the inband cancellation of the SI signal is around an impressive 45 dB. In the case of 8 dBm transmit power, the residual power is reduced close to the noise floor on both of the nodes, with the RF canceller node being less than 1 dB away. This is due to the initial power of the SI being relatively low and having little nonlinear distortion. These results indicate that with enough analog suppression of the SI, the system is able to almost fully cancel out the SI signal with the digital canceller, even in the presence of severe nonlinear distortion.

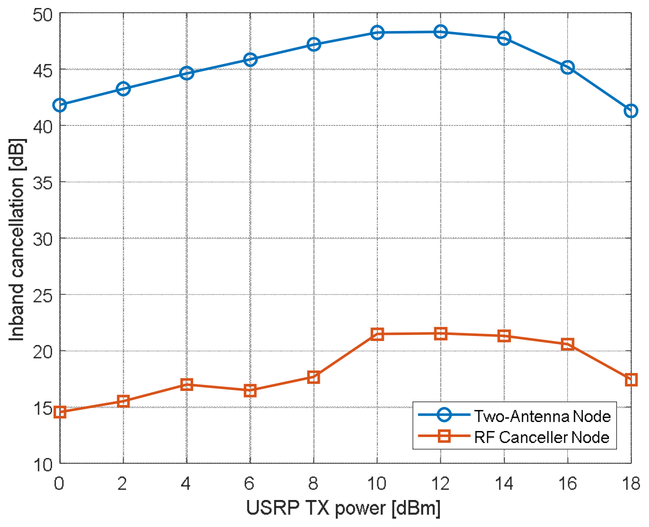


Figure 18. *Inband cancellation of the SI signal with different transmit powers in both measurement cases.*

The inband cancellations with different transmit powers in both measurement scenarios are presented in Figure 18. The difference between the cancellations provided by the digital canceller in the two nodes is a consequence of the RF canceller providing more isolation between the transmitter and receiver chains than the two-antenna node. The two antennas provide a fixed isolation of around 33 dB in all of the measurement cases, whereas the isolation of the RF canceller fluctuates between 62 dB and 68 dB in the measurements. This is due to the RF canceller being an active device, adapting to even the smallest changes in its surroundings. As mentioned previously, the residual power of the SI is close to the device noise floor in the RF node, which limits the achievable inband cancellations. In fact, the achieved digital cancellation of the SI is limited by the device noise floor, since it is physically impossible to push it below this limit. Additionally, the performance if the canceller is hindered by the application of the αMax+βMin technique, which only approximates the magnitude of the input signal. With the received power relatively high in the two-antenna scenario, the digital canceller still has a lot of power to cancel, even after analog suppression. This is why the inband cancellation in the two-antenna scenario lies between 40 dB and 50 dB, with the peak cancellation being around 48 dB achieved with 10 and 12 dBm transmit powers. This level of cancellation exceeds similar real-time realizations, described for example in [5] and [33], where the reported digital cancellations are 43 dB and up to 35 dB, respectively. More recently, [19] and [35] have utilized machine learning and neural networks, respectively, to obtain digital cancellations of 50 dB and 45 dB, respectively. Thus, the implemented digital canceller in this work is among the state-of-the-art solutions in the field, in terms of achieved cancellation.

Considering the antennas produce around 33 dB of isolation between the chains, the maximum combined suppression of the SI in the node is in the order of 80 dB. With transmit powers less than 6 dBm, the SI canceller is able to reduce the residual power level of the SI signal to within 2 dB of the device noise floor. Both of the cancellation results rise until the optimal operation point is achieved at 12 dBm, after which the cancellation results start to decline due to increasing nonlinearity of the system. It should be noted that with the two-antenna measurements, there is always some low amount on residual SI left on the cancelled signal. In the RF canceller scenario, the inband cancellation of the digital canceller remains low compared to the antenna node measurements, but that is due to the combined cancellation reaching the noise floor with negligible difference. The noise floor is reached with the RF canceller node with transmit powers up to and including 8 dBm, after which the residual power is around 1 to 2 dB from the noise floor. With 18 dBm transmit power, the difference goes up to around 5 dB. The maximum achieved digital cancellation with the RF node is around 22 dB with 10 dBm transmit power, and with 14 dBm transmit power, the node is capable of providing total isolation of up to 90 dB. This figure combines the 68 dB of isolation the combination of the RF canceller and the circulator provides, and digital cancellation of 21 dB of the digital canceller.

6.3 Bidirectional Full-duplex Operation

In order to illustrate the communication capabilities of the implemented canceller, the utilized nodes were used in a communication link. The measurements consider the received signal from the opposing node in half-duplex (HD) and full-duplex (FD) scenarios, and the quality of the link is determined by the figures of merit introduced in Section 6.1. The measurements were carried out again in a laboratory environment to ensure the least amount of interference from external sources. The measurements were conducted with a center frequency of 2.4 GHz. The transmit signals were random OFDM signals with a bandwidth of 20 MHz and 64-QAM symbols as the payload. The utilized transmit powers are the same as in the functional validation measurements: from 0 to 18 dBm with intervals of 2 dBm. Two sets of measurements were carried out, one with the nodes in the same room, and another with a wall separating the nodes. This will give evidence that the canceller is capable of operating in various circumstances. The measurement case with a line-of-sight between the nodes acts as a proof-of-concept measurement, whereas the case where the line-of-sight is obstructed by a wall resembles more an actual communications scenario. In both cases, the separation of the nodes was 4 meters.

6.3.1 Line-of-sight case

Figure 19 illustrates the SINR and SER results of the two nodes with the nodes in the same room. The SER result is presented in a semi-logarithmic plot, as the obtained values have a large range. For the cases with no misinterpretations in the symbols, the SER is assigned value $2.2204 \cdot 10^{-16}$, which is the used spacing between double precisions numbers in MATLAB. All results are an average of around 20 measurements, in order to minimize the effects of random variation.

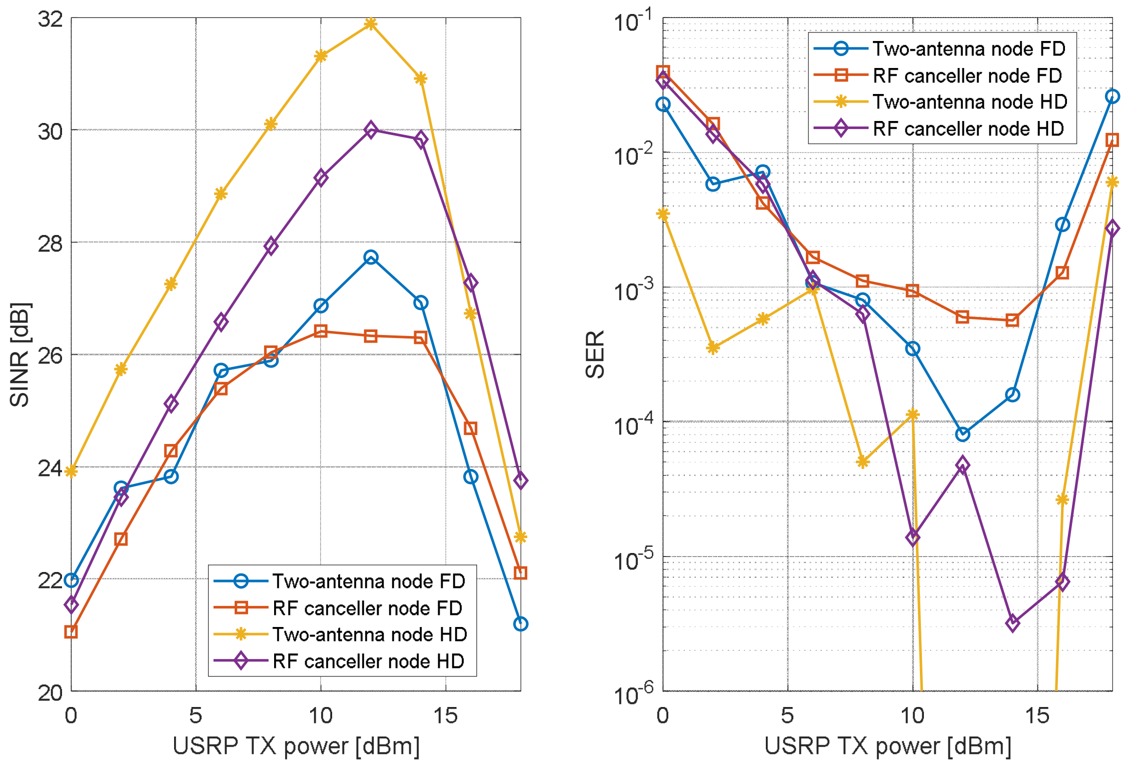


Figure 19. *SINR and SER of both nodes in half-duplex and full-duplex scenarios, with different transmit powers, the nodes in the same room.*

Overall, the results in Figure 19 show that the employment of the FD scheme has negative effects on the quality of the received signal. Since the nodes are not similar and the other node includes two antennas, the setup and the environment are not symmetrical for the nodes. Hence, it is more sensible to compare the HD and FD operations within the same node, as opposed to comparing all of the performances together. Both nodes showcase smaller differences in the SINR results between the respective HD and FD cases in lower transmit powers, which is reasonable, as the residual power of the SI after digital cancellation is close to noise floor in these cases. Higher transmit powers leave more power of the SI on the received signal, and thus the SINR difference rises up to around 4 dB in transmit powers between 10 and 14 dBm. With transmit powers even higher than these, the nonlinear distortion is heavy in both HD and FD cases, which deteriorates both performances. From all of the results, it can be seen that the SINR rises until 12 or 14 dBm after which it starts to drop, due to nonlinear behavior of the transmitters. The SER results are especially sensitive to random variations in the data as results are low, hence the SER curves do not display as clear curves as the SINR results in Figure 19. Still, higher SINR results generally denote better SER results as well, and thus the SER results are at their lowest when the SINR results are at their highest. The SER results show that only the two-antenna node is capable of interpreting the symbols perfectly with transmit powers of 12 and 14 dBm, while the other cases have misinterpretations with all powers. As is the case in with the SINR, the difference between the SER results in respective HD and FD cases are lower with low transmit powers, with the SERs of the RF canceller being nearly identical. The gap again rises with powers between 10 and 16 dBm, while the 18 dBm cases exhibit again less difference between the HD and FD cases. All of the cases exhibit relatively low SER values, however, which suggests the link is capable of transferring most of the information to the receiving node intact. The SINR and SER results combined demonstrate that with the adequate operating conditions, the information can be conveyed successfully in a bidirectional FD scenario, with acceptable decline in performance. Furthermore, practical communications systems utilize channel coding that can correct misinterpreted symbols, thus mitigating the need to transfer all data intact. Indeed, in a practical communications system resources are wasted if the SER of the uncoded signal drops to zero, which is not a desired feature. Thus, reaching SER values of zero is not necessary for the documented system to be a viable option for an actual communication implementation.

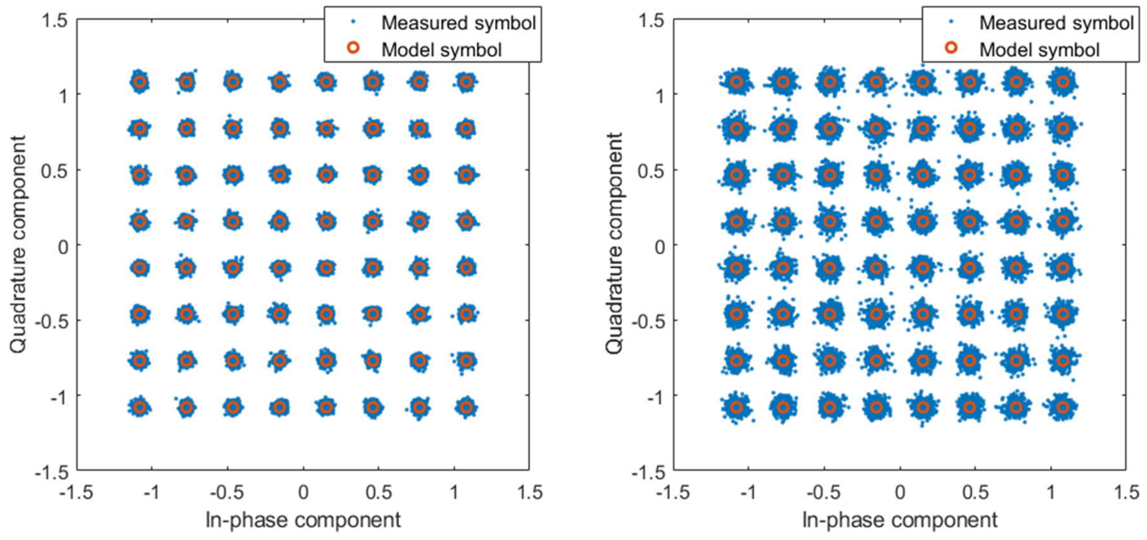


Figure 20. Constellations of the received symbols in the two-antenna node, in HD (left) and in FD (right) operations with 12 dBm TX power, both nodes in the same room.

Figure 20 illustrates the difference between the HD and FD operations in terms of the constellations. The exhibited case is where the difference between the schemes is at its highest, in the two-antenna node with 12 dBm transmit power. Clearly, the FD scheme adds noise to the received signal, which was also seen in the SINR results in Figure 19.

Due to this, the measured symbols are scattered on a wider circumference around the model symbols, which in turn makes it more probable for the symbols to be misinterpreted. This effect is reflected on the SER results. However, as was stated earlier, the amount of noise the FD scheme adds to the link is still within reasonable limits, which the constellations in Figure 20 emphasize.

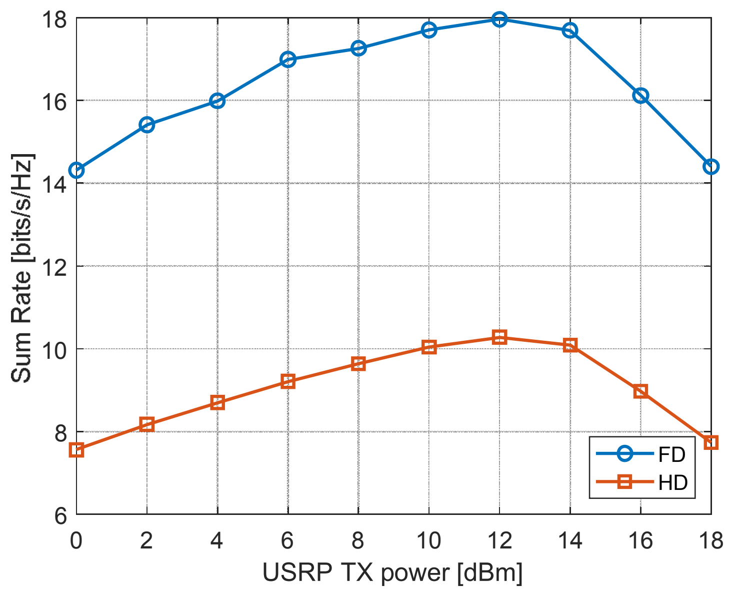


Figure 21. *Sum rate with various TX powers in FD and HD operations, both node in the same room.*

Finally, Figure 21 shows the sum rates of the HD and FD operations with the utilized TX powers. Despite the added noise in the FD scheme, the system is clearly capable of taking better advantage of the channel, with the sum rate of the FD operation being approximately 1.8 times greater on average than the HD one across all of the measurements. Furthermore, the difference between the attained sum rates stays relatively constant throughout the measurements. Therefore, the utilization of the FD scheme now saves considerable amount of resources. These results further prove the favorable communication capabilities of the implemented digital canceller.

6.3.2 Through the wall case

The SINR and SER results of the measurement case with the wall separating the two nodes are presented in Figure 22, similarly to the results shown in Figure 19. The plots in Figure 22 show similar results to the ones presented in Figure 19, with the SINR rising until after around 14 dBm the results start to drop. As a whole, the SINR results with the wall between the nodes are lower than without the obstruction, which is to be expected, since the obstruction attenuates the signals.

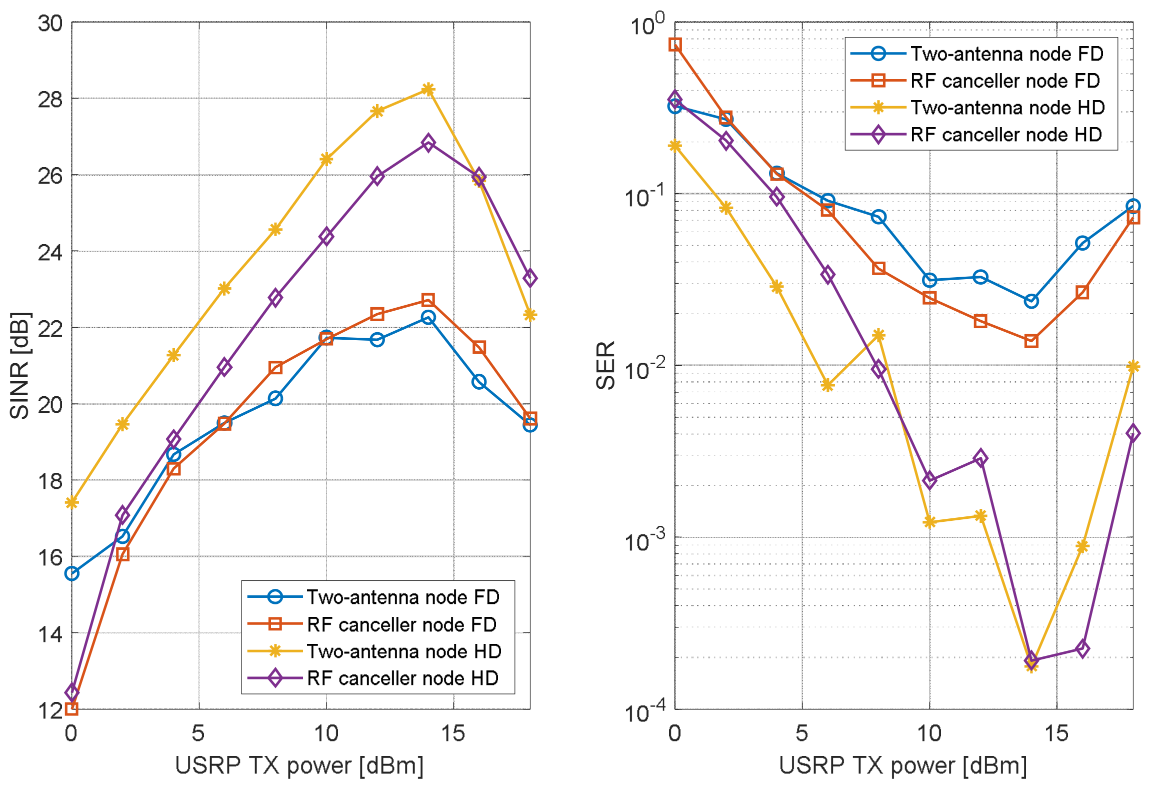


Figure 22. *SINR and SER of both nodes in half-duplex and full-duplex scenarios, with different transmit powers, the nodes separated by a wall.*

Again, it is most reasonable to compare the SINRs within one node, as the environment is not symmetrical. Similarly to the previous case, the SINR results are closer to each other in both nodes with lower transmit powers, but with optimal powers, the difference grows significantly. This effect is also present in the previous case, but not as nearly as distinctly as in this one. The SINR difference between HD and FD schemes is around 2 dB in the two-antenna node with powers less than 8 dBm and afterwards the difference rises up to some 6 dB at 14 dBm transmit power. The difference is less than 1 dB in the RF canceller node with low powers, and only rises up to around 4 dB at 14 dBm transmit power. These results are reflected in the SER plot, with all SER results being practically same at low transmit powers. The difference between HD and FD schemes is nearly identical to the previous test case, with the exception that all results have dropped by an order of magnitude. With higher powers, the SER results in HD cases drop, but before that, the results do not differ greatly from each other. In this case, none of the measurement results demonstrate perfect interpretation, as the SER never drops to zero, but, as was stated earlier, that is not an issue. These results indicate that the wall has an effect on the received signal quality, and the proximity of an obstacle reduces the functionality of the SI canceller, thus deteriorating the FD operation. Still, the link is able to convey messages over it, with acceptable degrading of the performance, as was the case in the first measurement scenario. Conclusively, the SI canceller is capable of suppressing the SI signal in a manner, which does not excessively interfere with the communication link.

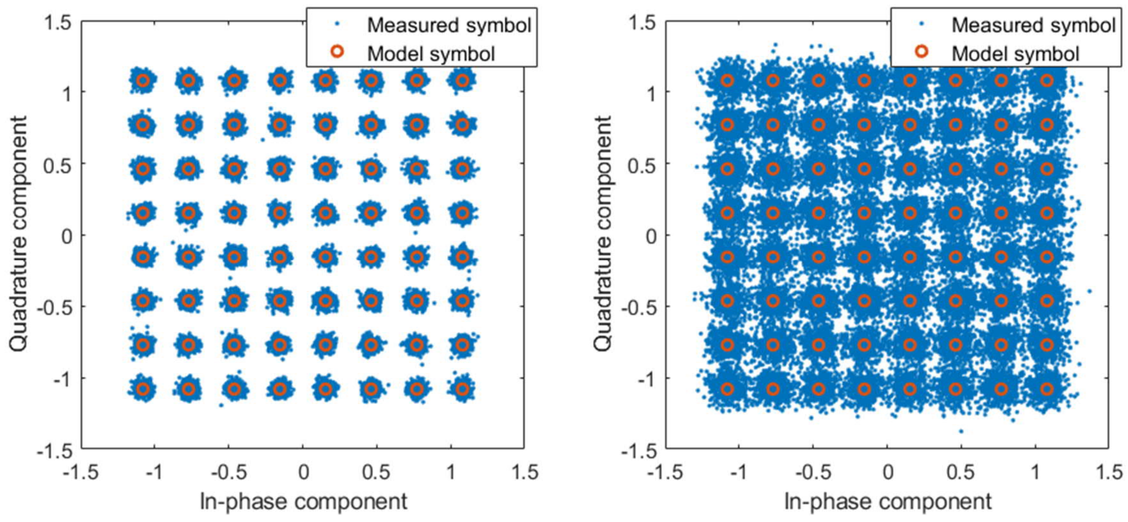


Figure 23. *Constellations of the received symbols in the two-antenna node, in HD (left) and FD (right) operations with 14 dBm TX power, nodes separated by a wall*

Figure 23 presents two constellations similarly to Figure 20, but the data is taken from the 14 dBm transmit power case with the line-of-sight between the nodes obstructed. In this case, the difference between the SINR and SER results is at its highest. The HD case is relatively unaffected by the obstruction, compared to the previous measurement case, while, the noise added by the FD scheme is even more evident in the constellation, yet the symbols still form visible clusters around their respective model symbols. This visually suggests that the information carried by the signal in the FD case could be deciphered by an advanced decoding technique.

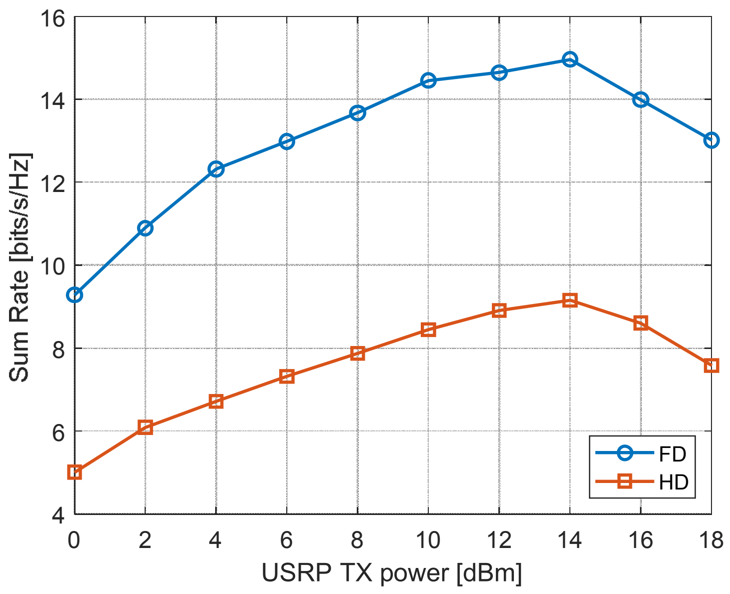


Figure 24. *Sum rate with various TX powers in HD and FD operations, nodes separated by a wall*

The sum rates of the through the wall measurements for HD and FD operations are illustrated in Figure 24. As was the case with the line-of-sight case, the sum rate of the FD operation is considerably higher than the HD one, with the gap between the two staying nearly constant. On average, the sum rate of the FD scheme is 1.7 times greater than in the HD scheme, which is approximately the same as in the line-of-sight case. Thus, it can be concluded that the implemented system is capable of operating in various circumstances, even ones that resemble real-life communications scenarios, with little deterioration in the performance.

7. SUMMARY AND FUTURE WORK

This chapter summarizes this thesis chapter by chapter and presents possible future endeavors. It is highly likely that the future solutions in communications engineering will rely on full-duplex technology, taking into account the rapid developments in the field. The work presented in this thesis aimed to address one major issue with the SI canceller solutions, which is the high complexity of the traditionally used algorithms. At the same time, the performance of the SI canceller should not suffer from the reduced complexity, i.e. the system should achieve sufficient cancellation of the SI for a link between two such devices to be feasible. The issue of decreasing the algorithm complexity was addressed by implementing a Hammerstein spline-based digital self-interference canceller, which promises great reduction in complexity compared to traditional memory polynomial implementations, while maintaining adequate levels of SI suppression. These qualities of the algorithm were proven with the implementation and measurement results presented in this thesis.

Chapter 2 shows how a typical transceiver operates in principal. The transceiver functions on the boundary between the digital and analog domains. In the digital domain, the signals are presented as in-phase and quadrature components, which can be interpreted as the real and imaginary parts of a complex number in the baseband frequency. These complex values are quantized, meaning they may only have a value from a finite set. On the boundary of the domains, the digital signals are translated into analog ones in a D/A converter, and vice versa in an A/D converter. The transmitted signals are amplified in the PA stage, which adds considerable nonlinear distortion to the signal as it is operated near the nonlinear region to maximize efficiency. In the case of a full-duplex device, while the transmitter chain is transmitting, the receiver receives signals at the same time and the transmit signal thus leaks into the receiver chain, which is called the self-interference signal. The SI signal is the most prominent issue in full-duplex technology, as it drowns out all other signals of interest, and in worst case might damage the receiver side of the device. The SI problem can be alleviated by SI cancellers, which may be operated in the analog or digital domains.

A novel digital nonlinear SI canceller solution is presented in Chapter 3. This spline-based model utilizes splines and a linear filter in cascaded Hammerstein system to reconstruct the SI signal, and removes the approximation from the received signal by a simple subtraction. The whole algorithm is adaptive, which means that the algorithm learns the behavior of the system it is modelling independently. The linear filter models the channel effects on the signal. In this work, an adaptive FIR filter is utilized, the coefficients of which are updated with the well-known LMS algorithm. The splines, which are a form of polynomial approximation, model the nonlinear effects of the PA. For the purposes of communications engineering, the traditional spline theory is expanded to conform to the use of complex values.

Chapters 4 and 5 introduce the implementation environment and the proper implementation in detail. The algorithm is implemented on an FPGA to achieve real-time operation. The software used to develop the code for the FPGA is LabVIEW Communications System Design Suite 2.0, which utilizes the LabVIEW visual programming language. LabVIEW Communications also offers effortless connection to USRP SDR devices, which acts as the transceiver of the system. Furthermore, the USRP also incorporates a high-end Xilinx Kintex-7 FPGA, which is the FPGA target of the implementation. The FPGA incorporates functionality besides the SI canceller, which runs at 120 MHz. In order to minimize the effort of FPGA programming, the SI canceller is run at lower rate of 60 MHz. This reduction requires the input signals to be decimated from 120 MHz and the output to be interpolated back to 120 MHz. Even at reduced clock frequency, the algorithm cannot be executed completely within one cycle, and thus, the algorithm is pipelined with seven pipeline stages. The computations within the algorithm are simplified for example by only considering the most significant columns of the spline matrix and by using the αMax+βMin technique to determine the magnitude of a complex value. In the end, the implemented algorithm only utilizes 34.6 % of available DSP48s, which emphasizes the low complexity of the algorithm.

Finally, in Chapter 6 the operation of the implemented system is verified with measurements. The implemented SI canceller is capable of suppressing the SI by up to 48 dB using two antennas as the isolator between the transmission and reception chains. The demonstrated performance surpasses the documented performances of other such implementations. In conjunction with the analog RF canceller, the system is capable of reducing the SI signal to very near the noise floor in most of the measurement cases. Additionally, it is shown that the implemented canceller is capable of bidirectional full-duplex operation. The measurements demonstrate that the full-duplex operation deteriorates the received signals when compared to the half-duplex operation, but most of the received information is still decipherable.

Even though the implementation proved successful, there are still areas in which it could be improved. Most notably, the canceller could be implemented within the 120 MHz loop, which would free resources associated with the decimation and interpolation tasks. More importantly, the increased sample rate would allow the use of signals with higher bandwidth. This way the system would be more resembling of an actual communications device. The increase of the sample rate could be implemented for example by computing the spline interpolation and linear filtering in the 120 MHz loop, while the adaptive algorithm could still be run separately on a 60 MHz loop. Additionally, the whole algorithm could be implemented on an independent SDR device, as the current implementation is heavily reliant on the LabVIEW host program. Still, the nonlinear SI canceller algorithm and the implementation in this work take us one step closer to making the full-duplex technology commercially feasible.

REFERENCES

[1] M. Al-Imari, “Theoretical analysis of full-duplex system with power control,” *2016 International Symposium on Wireless Communication Systems (ISWCS)*, Poznan, 2016, pp. 461-465.

[2] L. Anttila, D. Korpi, V. Syrjälä and M. Valkama, “Cancellation of power amplifier induced nonlinear nonlinear self-interference in full-duplex transceivers”, in *Proc. 47th Asilomar Conference on Signals, Systems and Computers,* Nov. 2013, pp. 1193-1198.

[3] K-E. Biebler and M. Wodny, *Splines and Compartment Models : An Introduction*, World Scientific Publishing Co Pte Ltd, 2013.

[4] W. Choi and H. Lim, “Immediate acknowledgement for single-channel full-duplex wireless networks,” *2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, Las Vegas, NV, 2012, pp. 477-478.

[5] M. Chung, M. S. Sim, J. Kim, D. K. Kim and C. Chae, “Prototyping real-time full duplex radios,” in *IEEE Communications Magazine*, vol. 53, no. 9, pp. 56-63, September 2015.

[6] J. Cioffi, “Limited-precision effects in adaptive filtering,” in *IEEE Transactions on Circuits and Systems*, vol. 34, no. 7, pp. 821-833, July 1987.

[7] C. de Boor, *A practical guide to splines,* Springer, 1978.

[8] E. J. Dempsey and D. T. Westwick, "Identification of Hammerstein models with cubic spline nonlinearities," in *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 2, pp. 237-245, Feb. 2004.

[9] P. S. R. Diniz, *Adaptive filtering: Algorithms and practical implementation*, Boston: Springer, 2008.

[10] M. Duarte and A. Sabharwal, “Full-duplex wireless communications using off-the-shelf radios: Feasibility and first results,” *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 2010, pp. 1558-1562.

[11] M. Ergen, *Mobile broadband: Including WiMAX and LTE*, Springer, 2009.

[12] A. Filip, “Linear approximations to √x2+y2 having equiripple error characteristics,” in *IEEE Transactions on Audio and Electroacoustics*, vol. 21, no. 6, pp. 554-556, December 1973.

[13] M. Gasparini, L. Romoli, S. Cecchi and F. Piazza, “Identification of Hammerstein model using cubic splines and FIR filtering,” *2013 8th International Symposium on Image and Signal Processing and Analysis (ISPA)*, Trieste, 2013, pp. 354-359.

[14] O. Gazi, *Understanding Digital Signal Processing*, Springer, 2018.

[15] K. M. Gharaibeh, *Nonlinear distortion in wireless systems: modeling and simulation with Matlab*, pp. 1-8, John Wiley & Sons, 2011.

[16] A. Goldsmith, *Wireless Communications.* Cambridge University Press, New York, NY, USA, 2005.

[17] S. Gollakota and D. Katabi, “Zigzag decoding: Combating hidden terminals in wireless networks,” in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication,* pp. 159-170.

[18] S. Goyal, P. Liu, S. S. Panwar, R. A. Difazio, R. Yang and E. Bala, “Full duplex cellular systems: will doubling interference prevent doubling capacity?,” in *IEEE Communications Magazine*, vol. 53, no. 5, pp. 121-127, May 2015.

[19] H. Guo, J. Xu, S. Zhu and S. Wu, “Realtime Software Defined Self-Interference Cancellation Based on Machine Learning for In-Band Full Duplex Wireless Communications,” *2018 International Conference on Computing, Networking and Communications (ICNC)*, Maui, HI, 2018, pp. 779-783.

[20] L. Hanzo, *3G, HSPA and FDD Versus TDD Networking.* 2008.

[21] M. Hayes, *Schaum's outline of theory and problems of digital signal processing.* New York: McGraw-Hill, 1998.

[22] S. Haykin, *Adaptive filter theory,* Prentice hall, 1986.

[23] K. Höllig, *Finite Element Methods with B-Splines,* 2003.

[24] E.C. Ifeachor and B.W. Jervis, *Digital signal processing: A practical approach.* Reading: Addison-Wesley, 1993.

[25] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti and P. Sinha, "Practical, real-time, full duplex wireless," in *MobiCom ’11*, New York, NY, pp. 301-312, 2011.

[26] A. N. Karanicolas, Hae-Seung Lee and K. L. Barcrania, “A 15-b 1-Msample/s digitally self-calibrated pipeline ADC,” in *IEEE Journal of Solid-State Circuits*, vol. 28, no. 12, pp. 1207-1215, Dec. 1993.

[27] C. H. Knapp and G. Clifford Carter, "The generalized correlation method for estimation of time delay," in *IEEE Transactions on Acoustics, Speech, and Si0nal Processing,* vol. ASSP-24, no. 4, pp. 320-327, August 1976.

[28] D. Korpi, T. Riihonen, V. Syrjälä, L. Anttila, M. Valkama and R. Wichman, “Full-Duplex Transceiver System Calculations: Analysis of ADC and Linearity Challenges,” in *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3821-3836, July 2014.

[29] D. Korpi *et al.*, “Full-duplex mobile device: pushing the limits,” in *IEEE Communications Magazine*, vol. 54, no. 9, pp. 80-87, September 2016.

[30] D. Korpi, *Full-duplex wireless: Self-interference modeling, digital cancellation and system studies*, Doctoral Dissertation, Tampereen Teknillinen Yliopisto, 2017.

[31] D. Korpi, T. Riihonen, A. Sabharwal and M. Valkama, “Transmit Power Optimization and Feasibility Analysis of Self-Backhauling Full-Duplex Radio Access Systems,” in *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4219-4236, June 2018.

[32] D. Korpi, Y. Choi, T. Huusari, L. Anttila, S. Talwar and M. Valkama, “Adaptive Nonlinear Digital Self-Interference Cancellation for Mobile Inband Full-Duplex Radio: Algorithms and RF Measurements,” *2015 IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, 2015, pp. 1-7.

[33] D. Korpi, M. AghababaeeTafreshi, M. Piilila, L. Anttila and M. Valkama, “Advanced architectures for self-interference cancellation in full-duplex radios: Algorithms and measurements,” *2016 50th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 2016, pp. 1553-1557.

[34] B. Kovačević, Z. Banjac and M. Milosavljević, *Adaptive digital filters*, Berlin: Springer, 2013.

[35] Y. Kurzo, A. Burg and A. Balatsoukas-Stimming, “Design and Implementation of a Neural Network Aided Self-Interference Cancellation Scheme for Full-Duplex Radios,” *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, 2018, pp. 589-593.

[36] M. Lam, “Software pipelining: An effective scheduling technique for VLIW machines”, in *Proc. ACM SIGPLAN Conf. Programming Languages Des. Implement.,* Atlanta, GA, June 1988, pp. 318–328.

[37] L. Laughlin, C. Zhang, M. A. Beach, K. A. Morris and J. L. Haine, “Passive and Active Electrical Balance Duplexers,” in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 1, pp. 94-98, Jan. 2016.

[38] T. Le-Ngoc and A. Masmoudi, *Full-Duplex Wireless Communications Systems: Self-Interference Cancellation,* Springer, 2017.

[39] Y. Li, *In-Phase and Quadrature Imbalance: Modeling, Estimation, and Compensation,* Springer, 2014.

[40] S. Liu, L. Fu and W. Xie, “Hidden-node Problem in Full-duplex Enabled CSMA Networks,” in *IEEE Transactions on Mobile Computing*.

[41] Y. Liu and E. Bai, “Iterative identification of Hammerstein systems,” in *Automatica,* vol. 43, no. 2, pp. 346-354, 2007.

[42] R. G. Lyons, *Understanding digital signal* processing (2nd ed.). Prentice hall, 2003.

[43] U. Meyer-Bease, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, 2001.

[44] B. Mulgrew, P. M. Grant and J. Thompson, *Digital signal processing: Concepts and applications* (2nd ed.). Basingstoke: Palgrave Macmillan, 2003.

[45] “Overview of the NI USRP RIO Software Defined Radio”, National Instruments, 30.07.2019. Available: https://www.ni.com/fi-fi/innovations/white-papers/14/overview-of-the-ni-usrp-rio-software-defined-radio.html

[46] M. Parker, *Digital signal processing 101: Everything you need to know to get started* (2nd ed.). Newnes, 2017.

[47] P. Pascual Campo, D. Korpi, L. Anttila and M. Valkama, “Nonlinear Digital Cancellation in Full-Duplex Devices Using Spline-Based Hammerstein Model,” *2018 IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1-7.

[48] M. Piililä, *Real-time FPGA implementation of nonlinear self-interference cancellation in full-duplex radio transceiver,* Master of Science Thesis, Tampereen Teknillinen Yliopisto, 2017.

[49] A. D. Poularikas, *Adaptive filtering: Fundamentals of least mean squares with MATLAB,* CRC Press, 2014.

[50] D. M. Pozar, *Microwave engineering* (2nd ed.). New York: Wiley, 1998.

[51] R. V. W. Putra, "A novel fixed-point square root algorithm and its digital hardware design," *International Conference on ICT for Smart Society*, Jakarta, 2013, pp. 1-4.

[52] T. S. Rappaport, *Millimeter wave wireless communications*. Upper Saddle River, NJ: Prentice Hall, 2015.

[53] A. Sabharwal, P. Schniter, D. Guo, D. W. Bliss, S. Rangarajan and R. Wichman, "In-Band Full-Duplex Wireless: Challenges and Opportunities," in *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 9, pp. 1637-1652, Sept. 2014.

[54] M. Scarpiniti, D. Comminiello, R. Parisi and A. Uncini, "Hammerstein uniform cubic spline adaptive filters: Learning and convergence properties," in *Signal Processing,* vol. 100, pp. 112-123, 2014.

[55] M. Scarpiniti, D. Comminiello, R. Parisi and A. Uncini, "Nonlinear spline adaptive filtering," in *Signal Processing,* vol. 93, no. 4, pp. 772-783, 2013.

[56] M. Scarpiniti, D. Comminiello, R. Parisi and A. Uncini, "Novel Cascade Spline Architectures for the Identification of Nonlinear Systems," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 7, pp. 1825-1835, July 2015.

[57] P. Śliviński, *Nonlinear system identification*, Springer, 2013.

[58] N. Storey, *Electronics: A systems approach* (2nd ed.). Harlow: Addison-Wesley, 1998.

[59] J. Tamminen *et al.*, "Digitally-controlled RF self-interference canceller for full-duplex radios," *2016 24th European Signal Processing Conference (EUSIPCO)*, Budapest, 2016, pp. 783-787.

[60] M. Valkama, M. Renfors and V. Koivunen, "Advanced methods for I/Q imbalance compensation in communication receivers," in *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2335-2344, Oct. 2001.

[61] M. Vetterli, *Foundations of signal processing,* Cambridge University Press, 2014.

[62] L. Wang, K. Wu and M. Hamdi, "Combating Hidden and Exposed Terminal Problems in Wireless Networks," *IEEE Transactions on Wireless Communications,* vol. 11, *(11),* pp. 4204-4213, 2012.

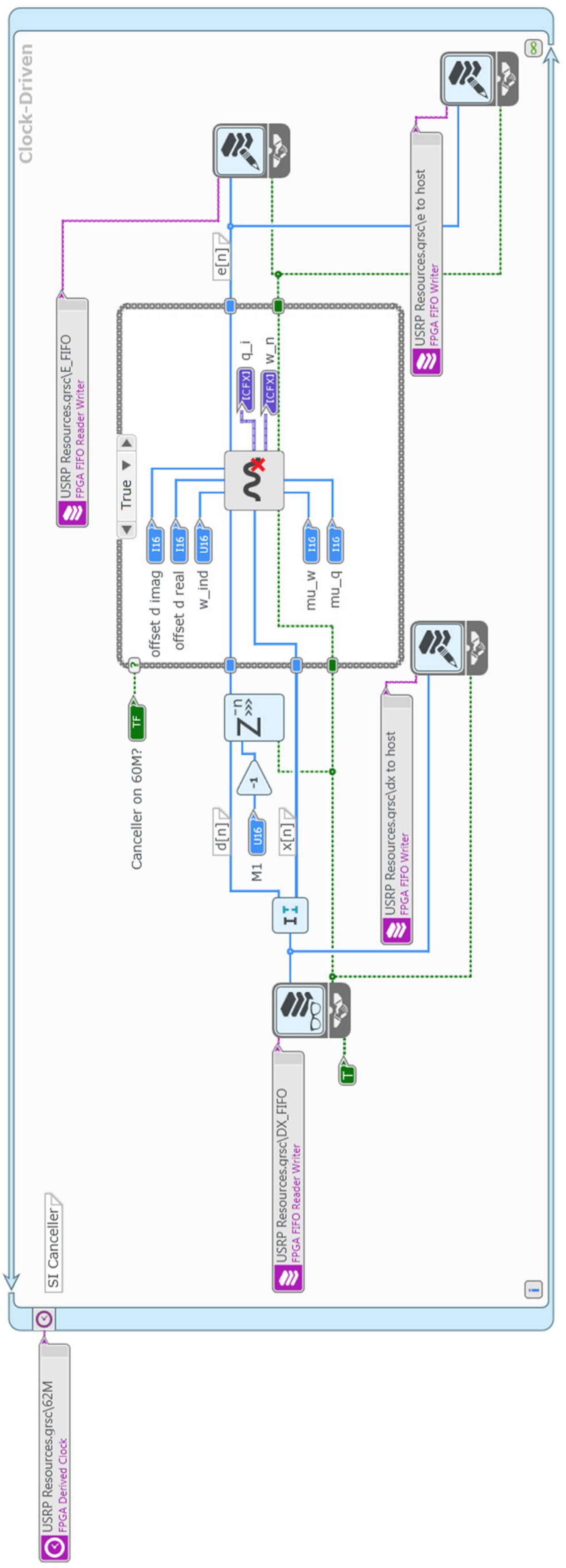
[63] D. Wu, C. Zhang, S. Gao and D. Chen, “A digital self-interference cancellation method for practical full-duplex radio,” *2014 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Guilin, 2014, pp. 74-79.

[64] “7 Series FPGAs Data Sheet: Overview”, Xilinx, 2018.

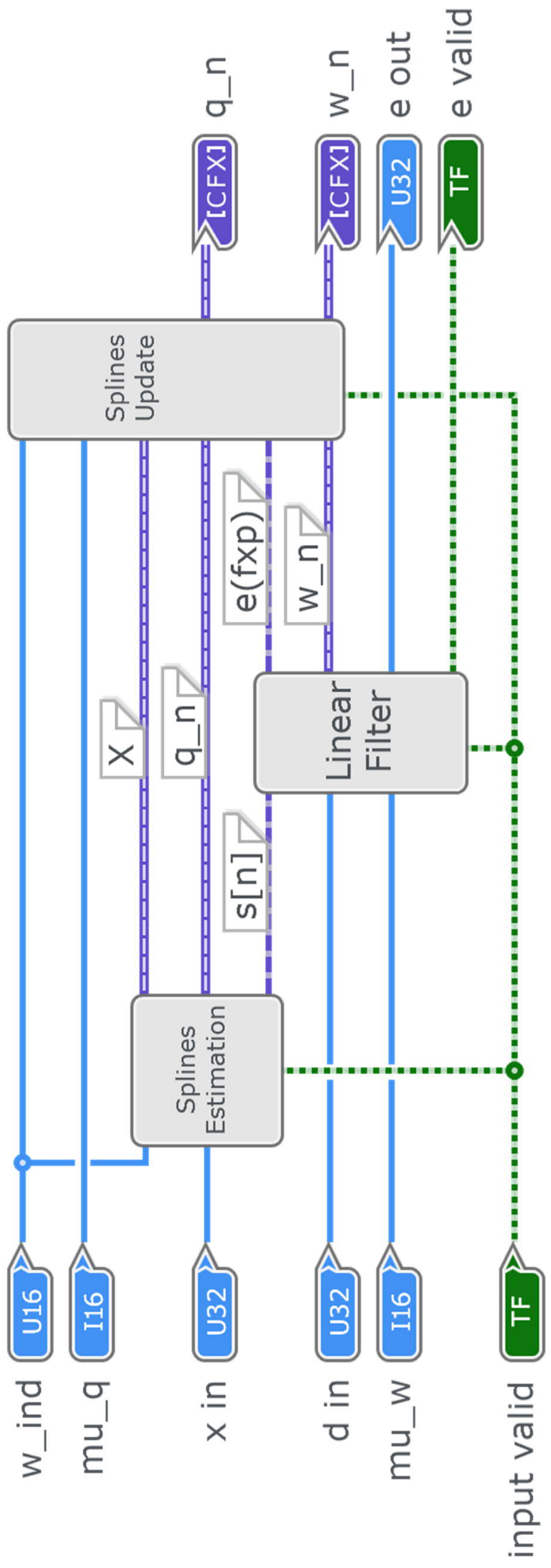
[65] F. Xiong, *Digital Modulation Techniques.* (2nd ed.) 2006.

[66] R. Y. Yen, H. Liu and W. K. Tsai, “QAM Symbol Error Rate in OFDM Systems Over Frequency-Selective Fast Ricean-Fading Channels,” in *IEEE Transactions on Vehicular Technology*, vol. 57, no. 2, pp. 1322-1325, March 2008.

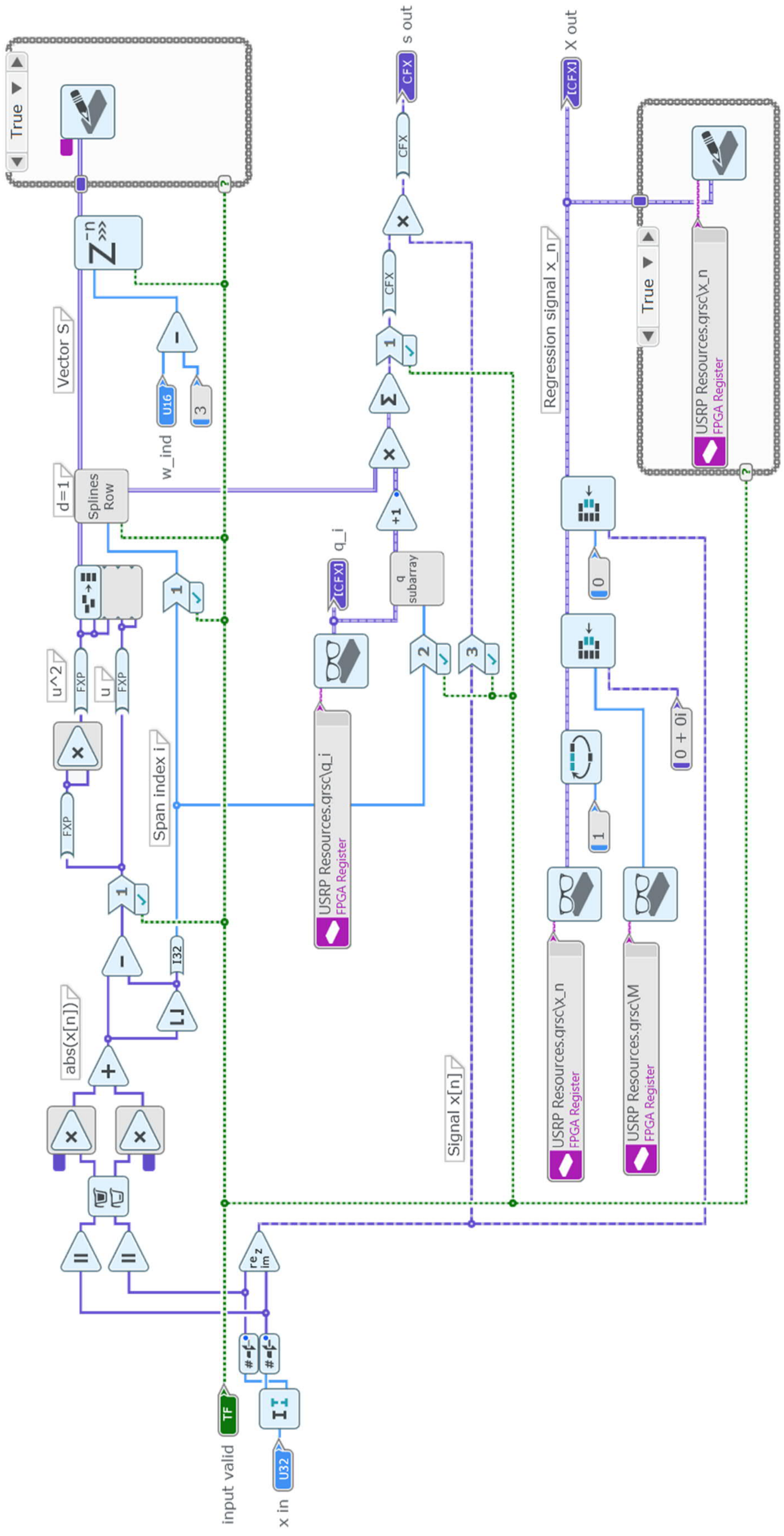
APPENDIX A: SI CANCELLER LOOP



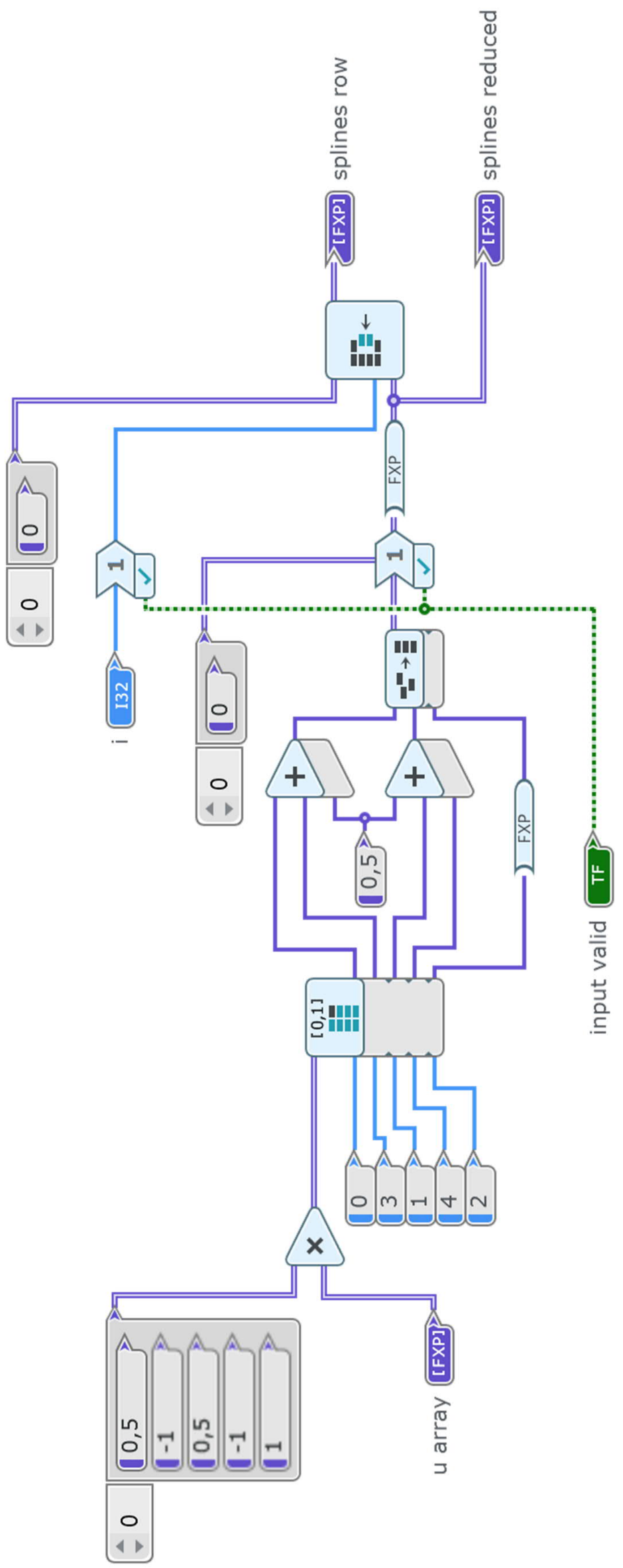
APPENDIX B: ‘SI CANCELLER’ VI



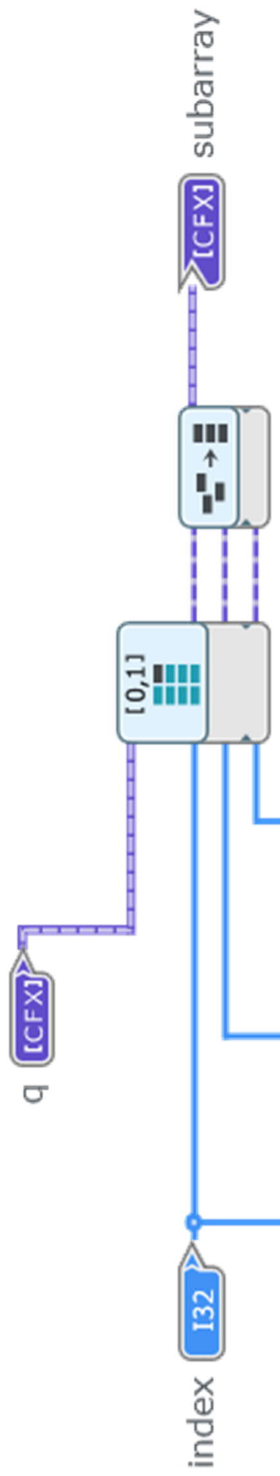
APPENDIX C: 'SPLINES ESTIMATION' SUBVI



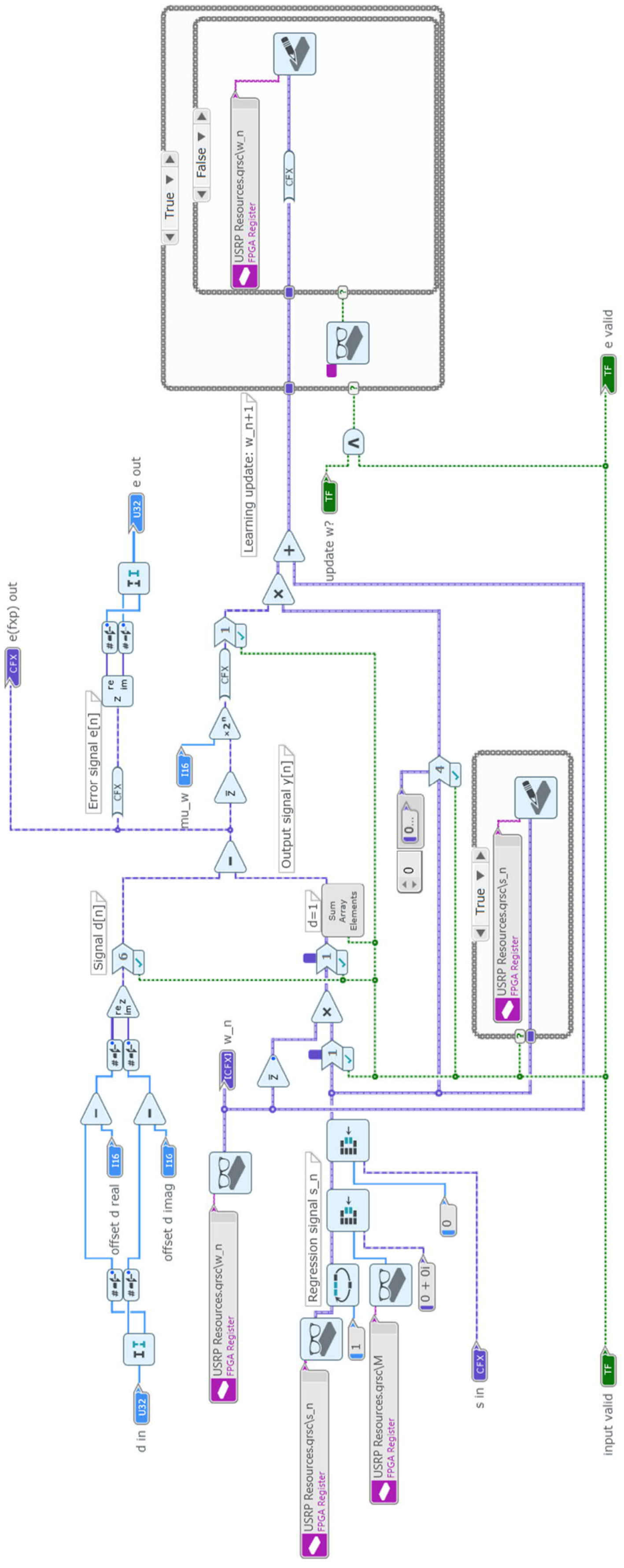
APPENDIX D: ‘SPLINES ROW’ SUBVI



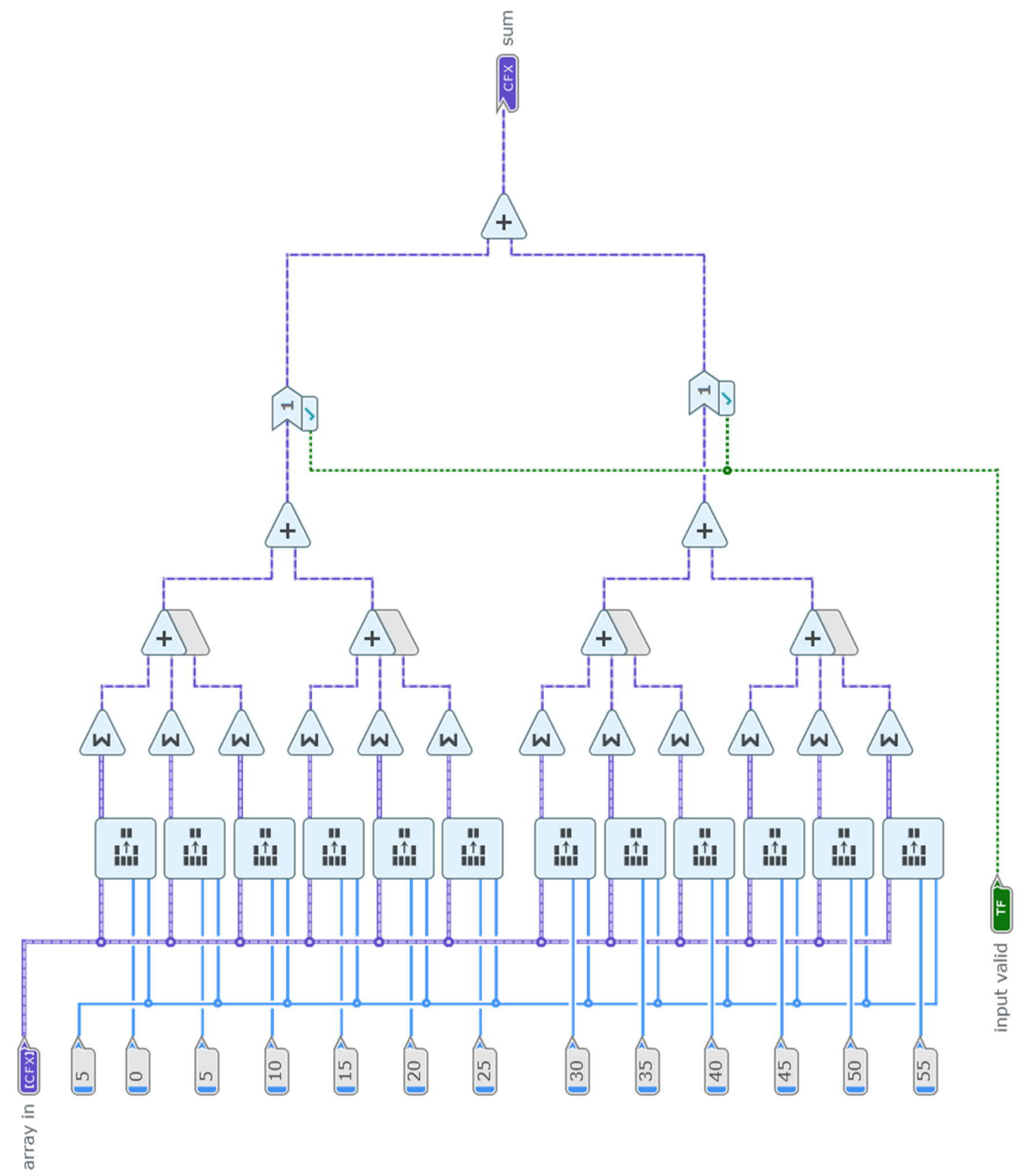
APPENDIX E: ‘Q SUBARRAY’ SUBVI



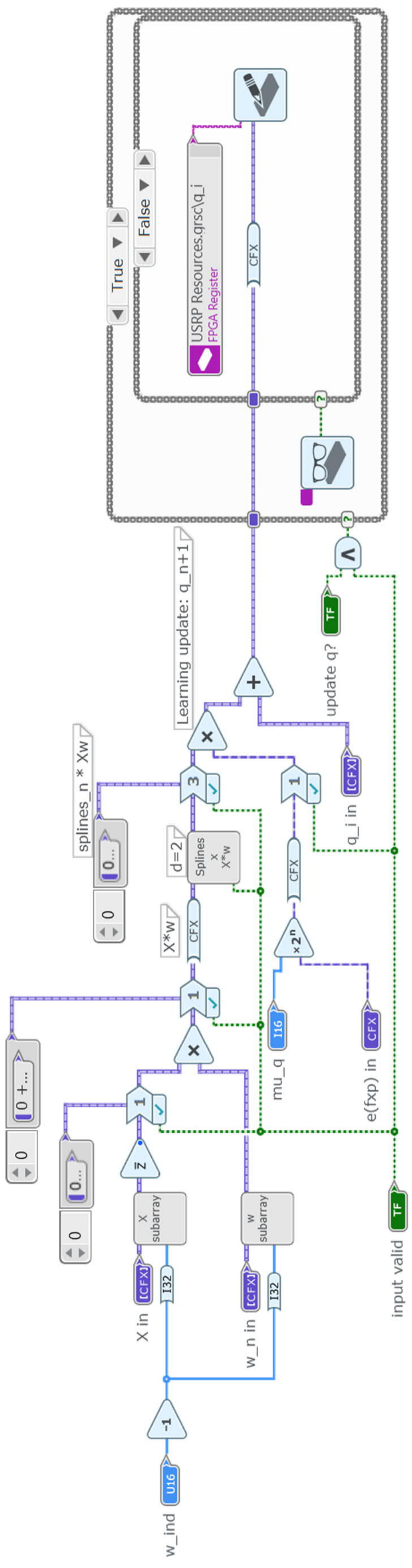
APPENDIX F: ‘LINEAR FILTER’ SUBVI



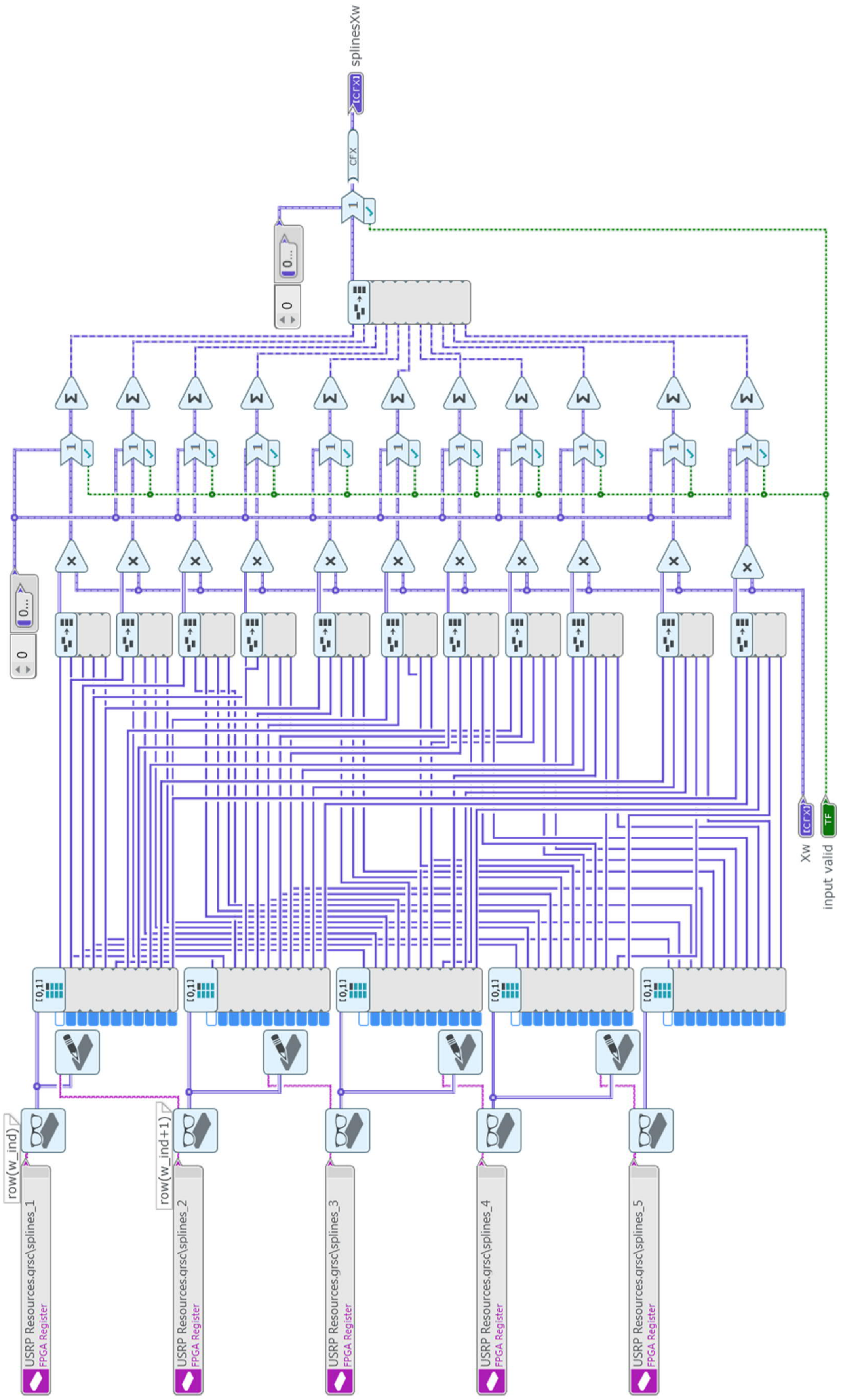
APPENDIX G: ‘SUM ARRAY ELEMENTS’ SUBVI



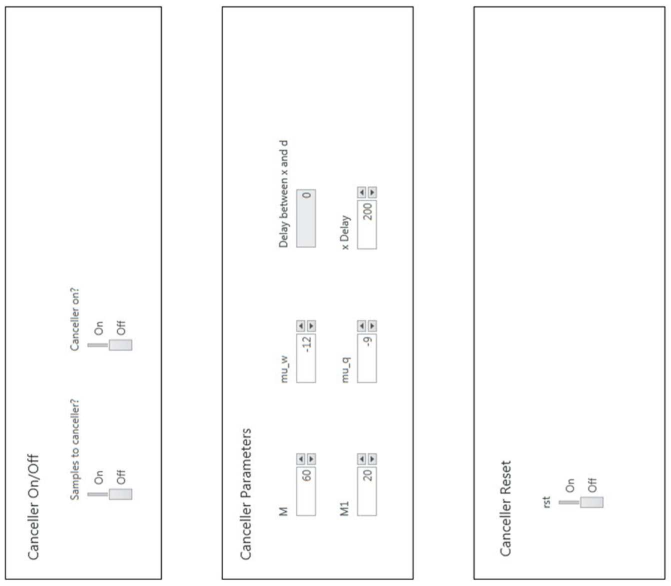
APPENDIX H: 'SPLINES UPDATE' SUBVI



APPENDIX I: ‘SPLINES X X*W’ SUBVI

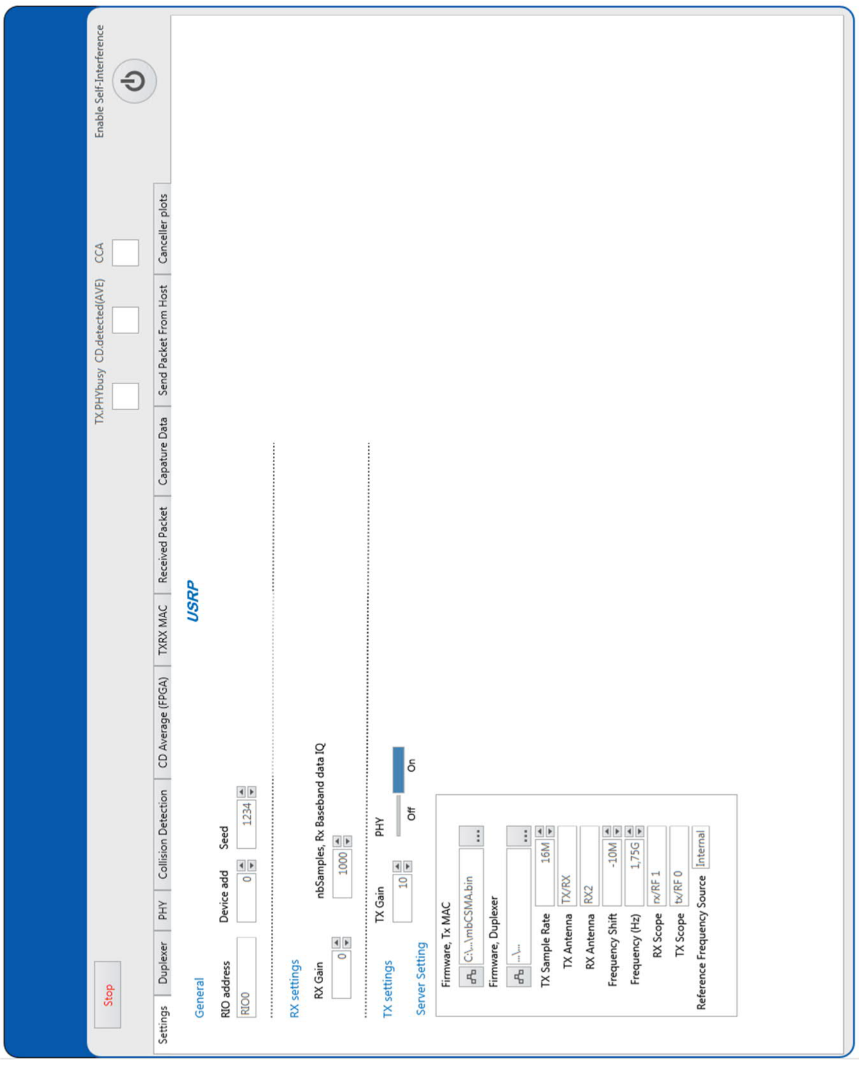

splinesXw
Xw
input valid
row(w_ind)
row(w_ind+1)
USRP Resources.qrsc\splines_1
FPGA Register
USRP Resources.qrsc\splines_2
FPGA Register
USRP Resources.qrsc\splines_3
FPGA Register
USRP Resources.qrsc\splines_4
FPGA Register
USRP Resources.qrsc\splines_5
FPGA Register

APPENDIX J: UI OF THE HOST PROGRAM


Canceller On/Off
Samples to canceller? On Off
Canceller on? On Off

Canceller Parameters
M 60
mu_w -12
Delay between x and d 0
M1 20
mu_q -9
x Delay 200

Canceller Reset
rst On Off



USRP

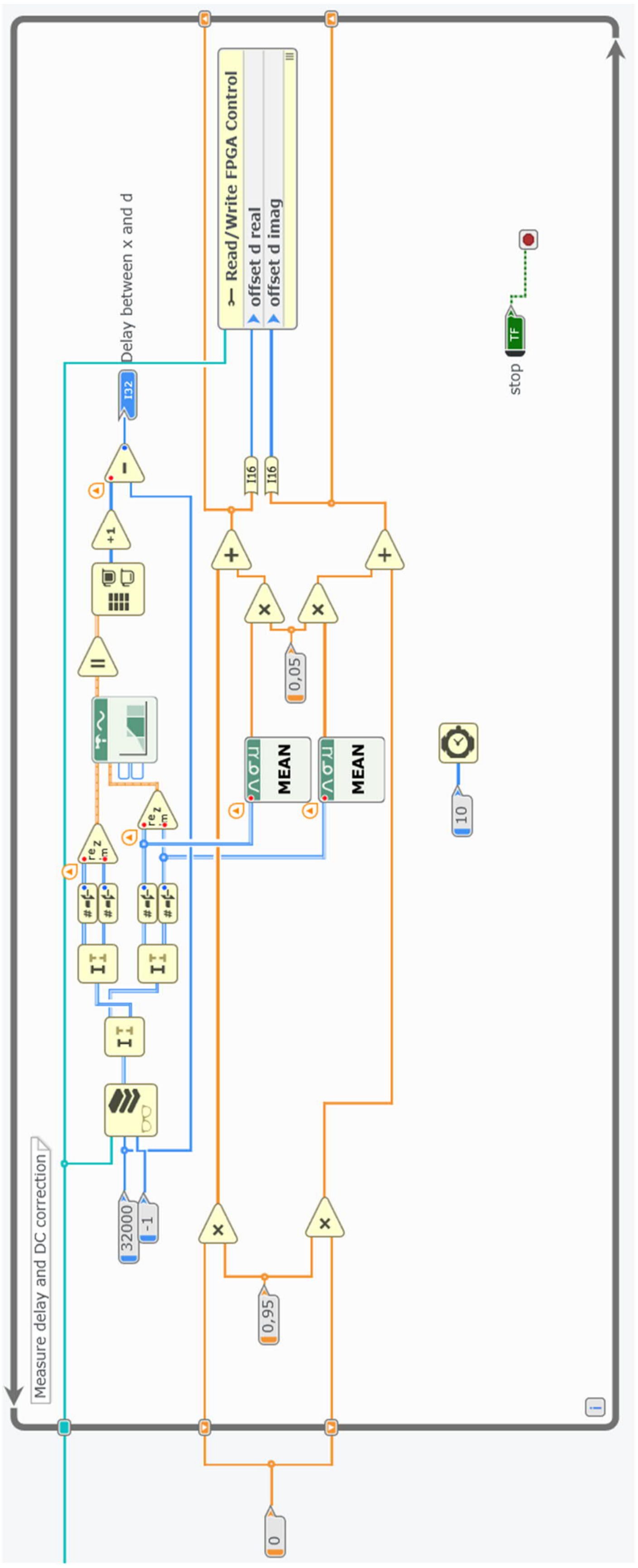
General
RIO address RIO0
Device add 0
Seed 1234

RX settings
RX Gain 0
nbSamples, Rx Baseband data IQ 1000

TX settings
TX Gain 10
PHY Off On

Server Setting
Firmware, Tx MAC C:\...\mbCSMA.bin
Firmware, Duplexer ...\...
TX Sample Rate 16M
TX Antenna TX/RX
RX Antenna RX2
Frequency Shift -10M
Frequency (Hz) 1,75G
RX Scope rx/RF 1
TX Scope tx/RF 0
Reference Frequency Source Internal

APPENDIX K: HOST CODE FOR DETERMINING DELAY AND DC-OFFSET


Read/Write FPGA Control
offset d real
offset d imag
Delay between x and d
I32
I16
I16
+1
0,05
MEAN
MEAN
10
TF
stop
32000
-1
0,95
0
Measure delay and DC correction