TAMPERE UNIVERSITY OF TECHNOLOGY

OLLI MYLLÄRI

DIGITAL TRANSMITTER I/Q CALIBRATION: ALGORITHMS AND REAL-TIME PROTOTYPE IMPLEMENTATION

Master of Science Thesis

Examiners: Prof. Mikko Valkama and
M.Sc. Lauri Anttila
Examiners and topic approved in the Faculty
of Computing and Electrical Engineering
Council Meeting on 7th of April, 2010

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
MYLLÄRI, OLLI: Digital Transmitter I/Q Calibration: Algorithms and Real-Time Prototype Implementation
Master of Science Thesis, 85 pages, 19 appendix pages.
June 2010
Major: Digital Transmission
Examiners: Prof. Mikko Valkama and M.Sc. Lauri Anttila

Keywords: I/Q imbalance, digital pre-distortion, real-time implementation, USRP, low-IF, direct-conversion

Nowadays, the direct-conversion and the low-IF transceiver principles are seen as the most promising architectures for future flexible radios. Both architectures employ complex I/Q mixing for up- and downconversion. Consequently, the performance of the transceiver architectures can be seriously deteriorated by the phenomenon called I/Q imbalance. I/Q imbalance stems from relative amplitude and phase mismatch between the I- and Q-branches of the transceiver, thus resulting in self-interference or adjacent channel interference. This thesis addresses details of the real-time prototype implementation of the transmitter unit realizing a widely-linear least-squares-based I/Q imbalance estimation algorithm and a corresponding pre-distortion structure as previously proposed by Anttila et al.

First transceiver architectures and radio transmitter principles are discussed with special emphasis on I/Q imbalance related aspects. Thereafter, the imbalance estimation principle itself is reviewed and a recursive version of it is derived. Then the implementation platform and software are introduced. After that, implementation details are discussed and implementation-related practical issues are addressed. Finally, simulation results and comprehensive RF measurement results from the real-time prototype implementation are presented.

The work done in this thesis realizes a real-time prototype implementation of the WL-LS I/Q imbalance estimation algorithm and corresponding pre-distortion structure. In addition, the implementation is shown to give consistent results with Matlab simulations and it can operate on general purpose processors.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
MYLLÄRI, OLLI: Lähettimen digitaalinen I/Q kalibrointi: algoritmeja ja reaali-aikainen prototyyppitoteutus
Diplomityö, 85 sivua, 19 liitesivua.
Kesäkuu 2010
Pääaine: Digitaalinen siirtotekniikka
Tarkastajat: Prof. Mikko Valkama ja DI Lauri Anttila

Avainsanat: I/Q epätasapaino, digitaalinen esivääristys, reaali-aikainen totetutus, USRP, matalavälitaajuus, suoramuunnos

Nykyaikana suoramuunnos- ja matalavälitaajuuslähetin-vastaanotin periaatteet nähdään lupaavimpina arkkitehtuureina tulevaisuuden joustaville radioille. Molemmat arkkitehtuurit käyttävät taajuusmuunnoksissa kompleksista I/Q taajuus-sekoitusta. Tästä johtuen mainittujen lähetin-vastaanotinarkkitehtuurien suorituskykyä huonontaa ilmiö nimeltä I/Q epätasapaino, mikä johtuu suhteellisesta amplitudi ja vaihe epäsovituksesta modulaattorin I- ja Q-haarojen välillä. Tämän vuoksi signaaliin muodostuu itseishäiriötä tai viereisen kanavan häiriötä heikentäen radiotaajuisen signaalin puhtautta. Tässä diplomityössä esitellään reaaliaikaisen lähetin-vastaanotinprototyypin toteutus, jossa on käytössä Lauri Anttilan aiemmin julkaisema laajasti lineaariseen pienimmän neliösumman menetelmään perustuva I/Q epätasapainon estimointi algoritmi ja siihen liittyvä esivääristysrakenne.

Aluksi esitellään lähetin-vastaanotinarkkitehtuurit ja niihin liittyvät pääperiaatteet painottaen I/Q epätasapainoon liittyviä asioita. Tämän jälkeen johdetaan I/Q epätasapainon estimointiin käytettävän algoritmin rekursiivinen versio ja esitellään toteutukseen käytettävä kehitysalusta ohjelmistoineen. Tämän jälkeen käydään läpi toteutuksen yksityiskohdat ja siihen liittyvät käytännön ilmiöt. Lopuksi esitellään simulaatiotulokset ja kokonaisvaltaiset radiotaajuusmittaukset reaali-aikaisesta prototyyppitoteutuksesta.

Diplomityöprojektin tuloksena on radiolähettimen reaali-aikainen prototyyppi toteutus, jossa on käytössä laajasti lineaariseen pienimpään neliösummaan perustuva I/Q epäsovituksen estimointi ja vähentämis algoritmi. Implementaatio tuottaa yhdenmukaisia tuloksia Matlab simulaatioiden kanssa ja pystyy toimimaan yleiskäyttöisen suorittimen laskentateholla.

PREFACE

The research leading to this thesis was supported by the Academy of Finland, the Finnish Funding Agency for Technology and Innovation (Tekes) and the Technology Industries of Finland Centennial Foundation. It has been part of a strategic research project in the Tekes GIGA Technology Programme called "DIRTY-RF: Advanced Techniques for RF Impairment Mitigation in Future Wireless Radio Systems". Moreover, the research work presented in this thesis has been carried out during the years 2009-2010 at the Department of Communications Engineering, Tampere University of Technology, Finland.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Mikko Valkama for giving me an opportunity to join and work in his group with interesting and challenging topics, and his guidance, advice and support during the work. I would also like to express my deepest gratitude to M.Sc. Lauri Anttila for fruitful talks and his invaluable guidance, advice, and co-operation during the thesis project. Furthermore, the atmosphere at the Department of Communications Engineering has been the most pleasant and inspiring. Therefore, I would like to thank all the current and earlier personnels of the department for giving me a possibility to become a member of this wonderful team.

Finally, I wish to express my warmest thanks to my parents Markku and Outi Myllärі for their help, support and love throughout my studies. At last, I deeply thank my fiancée Mari Paju for love, support and patience during this work and especially during the everyday life.

On 5th of May 2010, in Tampere, Finland.

Olli Myllärі

CONTENT

LIST OF ABBREVIATIONS

ACI	adjacent channel interference
ADC	analog-to-digital converter
AGC	automatic gain control
ANSI	American National Standards Institute
AM	amplitude modulation
ARLS	approximate recursive least squares
AWGN	additive white Gaussian noise
BB	baseband
BER	bit-error rate
BPF	bandpass filter
CFO	carrier frequency offset
CIC	cascade-integrator comb
CR	cognitive radio
DA	data-aided
DAC	digital-to-analog converter
DC	direct current
DDC	digital down-converter
DSP	digital signal processing
DUC	digital up-converter
EVM	error vector magnitude
FARLS	fast approximate recursive least squares
FE	front-end
FFT	fast Fourier transform
FIR	finite impulse response

FPGA	field programmable gate array
FT	Fourier transform
GN	Gauss-Newton recursive least squares
GNSS	Global Navigation Satellite System
GPP	general purpose processor
GUI	graphical user interface
HB	half-band
I	in-phase
ICI	inter-carrier interference
IF	intermediate frequency
IFFT	inverse fast Fourier transform
IFT	inverse Fourier transform
IMD	intermodulation distortion
I/O	input/output
I/Q	in-phase/quadrature
IR	image reject
IRR	image rejection ratio
ISI	inter-symbol interference
LMS	least mean squares
LNA	low noise amplifier
LO	local oscillator
LP	lowpass
LS	least-squares
LUT	look-up table
MFI	mirror frequency interference

MIMO	multiple-input and multiple-output
ML	maximum likelihood
MSE	mean square error
NDA	non-data-aided
NLMS	normalized least mean square
OFDM	orthogonal frequency division multiplexing
PA	power amplifier
PAPR	peak-to-average power ratio
PC	personal computer
PGA	programmable gain amplifier
PLL	phase-locked loop
PSK	phase shift keying
PM	phase modulation
Q	quadrature
QAM	quadrature amplitude modulation
RF	radio frequency
RLS	recursive least-squares
RRC	root-raised-cosine
RSSI	received signal strength indication
RX/TX	receiving/transmitting
SH	sample-and-hold
SD	secure digital
SDR	software defined radio
SNR	signal-to-noise ratio
SWIG	simplified wrapper and interface generator

TCXO	temperature compensated crystal oscillator
TDD	time division duplex
UHD	universal hardware driver
USB	Universal Serial Bus
USRP	Universal Software Radio Peripheral
WL	widely-linear
WL-LS	widely-linear least-squares

LIST OF SYMBOLS

$\mathbf{0}$	Zero matrix or vector
$\mathbf{a}$	Interpolation weight vector
$\mathbf{c}$	Time domain cross-correlation
$C(f)$	Frequency domain cross-correlation
D	Imbalance filter impulse response delay
$\mathbf{D}(n)$	FARLS diagonal matrix
DC_{est}	DC offset estimate
$e(n)$	Error in adaptive algorithms
f	Frequency
f_c	Center frequency or carrier frequency
f_{IF}	Frequency of IF stage
f_s	Sampling frequency
g	Feedback loop gain
g_T	Relative gain or amplitude imbalance of LO signal
$\hat{\mathbf{g}}$	Concatenated imbalance filter impulse response estimate
$\hat{\mathbf{g}}_1$	Non-conjugate imbalance filter impulse response estimate
$\hat{\mathbf{g}}_2$	Conjugate imbalance filter impulse response estimate
$\hat{\mathbf{g}}_1^0$	Zero-padded non-conjugate imbalance filter impulse response estimate
$\hat{\mathbf{g}}_2^0$	Zero-padded conjugate imbalance filter impulse response estimate
$g_1(t)$	Non-conjugate imbalance filter impulse response
$g_2(t)$	Conjugate imbalance filter impulse response
$g_{1,p}(t)$	Modified non-conjugate imbalance filter impulse response

$g_{2,p}(t)$	Modified conjugate imbalance filter impulse response
$\tilde{g}_1(t)$	Observable non-conjugate imbalance filter impulse response
$\tilde{g}_2(t)$	Observable conjugate imbalance filter impulse response
$G_1(f)$	Non-conjugate imbalance filter transfer function
$G_2(f)$	Conjugate imbalance filter transfer function
$\tilde{G}_1(f)$	Observable non-conjugate imbalance filter transfer function
$\tilde{G}_2(f)$	Observable conjugate imbalance filter transfer function
$h_{fb}(t)$	Feedback loop impulse response
$h_I(t)$	Non-ideal in-phase filter impulse response
$h_Q(t)$	Non-ideal quadrature filter impulse response
$h_T(t)$	Relative non-ideal filter impulse response
$H_{fb}(f)$	Feedback loop transfer function
$\mathbf{I}$	Identity matrix
$IRR_{dB}(f)$	Image Rejection Ratio in decibels
$\mathbf{k}(n)$	Kalman gain vector
$l(t)$	LO signal
L_b	Length of observed data sequence
N_{bits}	Number of bits in numerical representation
N_g	Length of imbalance filter impulse response
N_w	Length of pre-distortion filter impulse response
OSF	Over-sampling factor
$\hat{p}$	Integer delay estimate

$\mathbf{P}(n)$	Inverse of covariance matrix
$r(t)$	Transmitter output / RF signal
R_b	Bitrate
t	Time
$\mathbf{u}(n)$	Input vector for adaptive algorithms
$w(t)$	Pre-distorter impulse response
$\hat{\mathbf{w}}$	Pre-distortion filter impulse response estimate
$W_{OPT}(f)$	Optimum pre-distorter transfer function
$x(t)$	Baseband equivalent of the imbalanced RF signal
$x_p(t)$	Baseband equivalent of the pre-distorted RF signal
$y(t)$	Observed feedback loop data sequence
$\mathbf{y}(n)$	Observed feedback loop data block at time instant n
$Y(f)$	Fourier transform of $\mathbf{y}$(n)
$\mathbf{y}_{mean}$	Complex mean value of $\mathbf{y}$
$\mathbf{y}_{comp}$	DC offset free version of $\mathbf{y}$
$z(t)$	Ideal baseband equivalent transmit signal
$z_p(t)$	Pre-distorted transmit signal
$\mathbf{z}(n)$	Original transmitted data sequence
$\mathbf{Z}(n)$	Convolution matrix of original transmitted data sequence
β	FARLS gain vector
δ	RLS initialization factor
$\delta(t)$	Impulse function

θ	Feedback loop phase
λ	RLS forgetting factor
μ	LMS Step-size variable
τ	Fractional-delay estimate
ϕ_T	Relative phase mismatch of LO signal
	ARLS gain variable
ω	Digital normalized frequency
$\Im[.]$	Imaginary part of the number
$\Re[.]$	Real part of the number
$(.)^T$	Transpose of a matrix or vector
$(.)^{-1}$	Inverse of a matrix
$(.)^*$	Complex conjugate
$(.)^H$	Hermitian transpose of a matrix or vector
$(.)^+$	Pseudo-inverse of a matrix
$\|.\|$	Norm of a vector
$\wedge(.\|.)$	Log-likelihood function
$\mathcal{F}\{.\}$	Fourier transform
$\mathcal{F}^{-1}\{.\}$	Inverse Fourier transform
$Tr\{.\}$	Trace of a matrix

1. INTRODUCTION

1.1 Motivation and Background

In recent years, the wireless communication sector has experienced unprecedented growth as new emerging standards offer higher throughput, more reliable data transfer and more diverse services. There are currently close to 4.6 billion cellular users worldwide and this number is expected to grow as transmission rates grow [42]. This rapid proliferation constitutes a challenging future for the wireless communication industry.

The diversity of wireless standards creates a need for multi-standard terminals which will support all existing as well as emerging wireless radio systems [35]. One particularly interesting approach in designing a multi-standard wideband transceiver is to build a flexible system which utilizes one common radio frequency (RF) front-end and can be programmed to operate in all communication modes [29, 15, 25, 72, 62, 63, 88], such a concept is also known as software defined radio (SDR). However, the design of such a device poses many technical challenges which must be overcome to enable its operation. The major challenge in this regard is the efficient and undistorted use of spectrum, a scarce and valuable resource [35]. Wireless terminals operate at frequencies of several gigahertz and the spectrum in this frequency range is already crowded.

The current trend in implementing future wireless radio transceivers is to use the direct-conversion [2, 1, 59] or the low-IF [61, 1] transceiver architectures. However, there are still a number of practical issues to be overcome before these transceiver architectures can be fully implemented in future wideband flexible transceiver units. In both architectures many transceiver functions have been moved from analog parts towards the digital signal processing (DSP) parts, enabling a low-cost, simple, less power-consuming and highly integrable transceiver unit [56]. One practical problem, however, is the sensitivity of such simplified analog front-ends to imperfections in the used radio electronics [52].

Both of the above-mentioned transceiver architectures are based on the analog com-

plex in-phase/quadrature (I/Q) up- and downconversion, which renders them vulnerable to amplitude and phase mismatch between the in-phase (I)- and quadrature (Q) branches [21, 38, 5, 49, 89]. As a result, there is crosstalk between mirror frequencies which, depending on the transceiver architecture selected, yields self-interference or adjacent channels interference, when interpreted in the frequency domain. Other major nonidealities on the transmitter side are local oscillator (LO) leakage, power amplifier (PA) nonlinearity and phase noise, which will also contribute to signal deterioration [30]. Moreover, future wireless systems will demand higher transmission rates, which involves wider bandwidths, higher order modulations and utilization of non-constant envelope modulation schemes and multi-carrier techniques. In addition, signals with wider bandwidths and higher order modulations, and multi carrier signals (e.g. orthogonal frequency division multiplexing (OFDM)) are especially sensitive to analog front-end (FE) nonidealities [31].

1.2 Scope and Outline of the Thesis

This thesis provides a general overview of significant RF impairments but it focuses particularly on I/Q imbalance. On the other hand, although the thesis mainly addresses the transmitter side of the transceiver, many functions are reciprocal and to some extent applicable on the receiver side. The approach here is to calibrate the transmitter with the help of DSP and digital pre-distortion, rendering the transmitter more tolerant to mismatches in the analog circuits. I/Q imbalance is described in detail with mathematical derivations and clarifying illustrations. Moreover, the I/Q imbalance calibration algorithm applied is extensively discussed. The ultimate goal is to develop a real-time transmitter prototype employing the I/Q imbalance estimation and mitigation algorithm deduced from theoretical derivations and computer simulations. In addition, details of the implementation of the real-time prototype are described.

The thesis is divided into seven chapters. After the introductory section, in Chapter 2, the basics of signal representations and the concept of bandpass transmission are reviewed. Further, different mixing schemes are examined, as well as the functionality of different radio transmitter architectures of interest. Chapter 3 provides an understanding of I/Q imbalance as a phenomenon and of the kind of effects it has on transmitter performance. In addition, not only mathematical models but also estimation and mitigation schemes for I/Q imbalance on the transmitter side are discussed. Next, Chapter 4 describes the development environment for real-time implementation. Chapter 5 addresses all significant implementation details and discusses implementation-related practical aspects. Chapter 6 presents all re-

sults achieved and draws a comparison between theoretical simulation and real-time implementation results. Finally, Chapter 7 summarizes the thesis and its achievements.

In addition, there are three appendices which give further information of the GNU Radio environment. Appendix A discusses general practices which have been considered useful during the work. Thereafter, Appendix B addresses a few examples of GNU Radio Python flow graphs, GNU Radio Companion flow graphs and GNU Radio signal processing block source files. Finally, in Appendix C short guide for updating the firmware of the USRP2 is given.

2. BANDPASS TRANSMISSION AND RADIO TRANSMITTER PRINCIPLES

Future wireless systems will be required to support higher data rates for a large number of simultaneous users. In addition, these co-existing users will be employing a variety of mobile terminal equipment with multiple wireless transmission technologies. This trend has made multi-standard flexible transceivers desirable, giving rise to greater demands on the design of future communication systems [12]. Moreover, a multi-standard flexible transceiver should be low-cost, portable and highly integrable. One formerly prevalent transceiver architecture is the *superheterodyne* principle [61], which is not however suited for the design of flexible transceivers. This makes for closer interest in sophisticated transceiver architectures, which move DSP parts closer to the RF front end, making the transceiver more flexible. Current state-of-the-art radio transceivers employ advanced DSP techniques to meet the given demands. In other words, a number of functionalities of a transceiver which have traditionally been implemented with analog circuits are now being taken over by digital signal processors.

The purpose of this chapter is to give a brief introduction to the traditional and modern transceiver architectures, as well as to reveal the imperfections and impairments that are encountered in their constituent blocks. The focus is on the transmitter architectures, as the whole thesis is concerned specifically with the transmitter side. The chapter commences with different representations of signals in the time and frequency domain and real and complex-valued signals are introduced in Section 2.1. A number of mixing or frequency translation techniques are then discussed in Section 2.2. Thereafter, principle of bandpass transmission is reviewed in Section 2.3. Some of the most desirable transmitter architectures are depicted in Section 2.4. Finally, and overview of the fundamental RF impairments encountered in wireless transceivers is presented in Section 2.5.

2.1 Real and Complex Signals

The fundamental objective of a telecommunication system is to transport information, essentially bits, from point A to point B. This information is represented with signals which may be in the form of voltage, current or an electromagnetic wave; this information-bearing signal can be described in the time domain and in the frequency domain, and there exists a relation between these representations [36, 69].

In general, communication signals can have either real or complex representation. Complex-valued signals are in practice signals which have two different real-valued signals carrying the real and imaginary parts. A complex-valued signal $z(t)$ is presented mathematically as

$$z(t) = z_I(t) + jz_Q(t) \tag{2.1}$$

where $\Re[z(t)] = z_I(t)$ is called the *in-phase* part of the complex-valued signal and $\Im[z(t)] = z_Q(t)$ the *quadrature* part.

Any signal which is a function of time is called a *time domain signal* and it has an equivalent frequency domain representation under certain conditions. These conditions are the Dirichlet's conditions which state that $z(t)$ should be absolutely integrable and partially monotonic [33]. An analog time domain signal is frequently called a *continuous time signal* and the corresponding sampled signal is called a *discrete time signal* [47]. Transformation from the time domain to frequency domain is called Fourier transform (FT). If a continuous time signal is denoted by $z(t)$, its corresponding FT [33] is

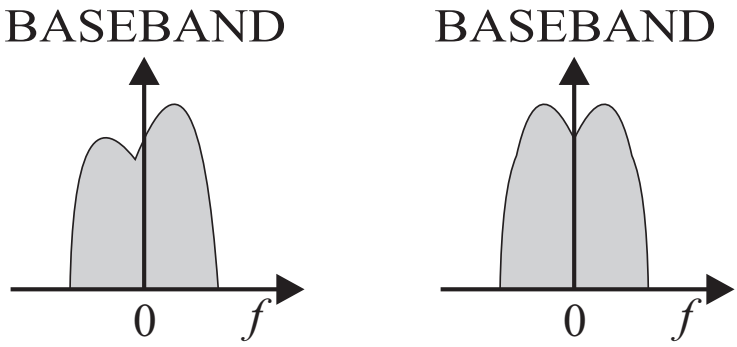
$$Z(f) = \mathcal{F}\{z(t)\} = \int_{-\infty}^{\infty} z(t)e^{-j2\pi ft}\,dt. \tag{2.2}$$

The magnitude of the Fourier transformed signal $|Z(f)|$ as a function of frequency is called the *amplitude spectrum* of the signal $z(t)$. Similarly, the argument of the $Z(f)$ as a function of frequency is called the *phase spectrum* of the signal $z(t)$. Fourier transform $Z(f)$ of the real-valued time domain signal $z(t)$ obeys Hermitian symmetry in the frequency domain, i.e. $Z(-f) = Z^*(f)$ [33], see Figure 2.1(b). In contrast, if $z(t)$ is complex-valued, its FT $Z(f)$ does not comply with this symmetry as is depicted in Figure 2.1(a) [33]. In other words, the amplitude spectrum of a real-valued signal is symmetric with respect to zero frequency, while that of the complex-valued signal does not obey this feature. Congruent transformation from the frequency domain to the time domain representation for the signal obtained from equation (2.2) is called as inverse Fourier transform (IFT), which can be formulated

[33] in the following terms:

$$z(t) = \mathcal{F}^{-1}\{Z(f)\} = \int_{-\infty}^{\infty} Z(f)e^{j2\pi ft}\,df. \tag{2.3}$$

Usually signals inside the digital parts of the transmitter are complex-valued, this being convenient in signal processing. Complex-valued arithmetics are also very useful and powerful tool for DSP. Moreover, positive and negative frequencies can be processed independently. In contrast, the medium of transmission in wireless telecommunication is always real-valued and a complex-valued signal cannot be transmitted over the real channel directly. For this reason, the concept of I/Q mixing or quadrature up/down conversion was initially introduced.



(a) Complex-valued signal (b) Real-valued signal

Figure 2.1: Frequency domain examples of complex-valued and real-valued signals.

2.2 Frequency Translations

In many cases, like in wireless communication, a physical transmission medium is incapable of transmitting frequencies at zero or very low frequencies. Consequently, the baseband signal has to be translated to a frequency range which is free from other signals and suitable to the communication system. This frequency translation is accomplished by mixing the baseband signal with the LO signal. There are two different approaches to the mixing operation, namely real mixing and complex mixing. The following two subsections discuss the way these mixing techniques work and their main differences.

2.2.1 Real Mixing

Real mixing is the traditional mixing technique widely employed in transceivers during the era of RF communication. It is based on multiplying a real-valued signal

with a real-valued sinusoidal signal, which is usually called an LO signal, and this signal is usually generated by an LO and a following phase-locked loop (PLL) circuit. The resulting output signal has a spectrum similar to that of the original signal, but translated up and down by f_{LO}, where f_{LO} is the frequency of the LO signal, called *carrier frequency*. On the other hand, f_c is called a *center frequency* and depending on the transmitter architecture it can be different from f_{LO}. The real mixing process can be described by the following equation

$$r(t) = z(t)cos(2\pi f_{LO}t) = z(t)\frac{1}{2}\left(e^{j2\pi f_{LO}t} + e^{-j2\pi f_{LO}t}\right). \tag{2.4}$$

The Fourier transform of the above equation yields the frequency domain result as

$$R(f) = \frac{1}{2}\left(Z(f - f_{LO}) + \frac{1}{2}\left(Z(f + f_{LO}\right), \tag{2.5}$$

where it can be clearly seen that the resulting bandpass signal has symmetric frequency components at $-f_{LO}$ and f_{LO}. Figure 2.2 comprises a general block diagram of the real mixer and the corresponding spectrum illustration before and after the mixing procedure.

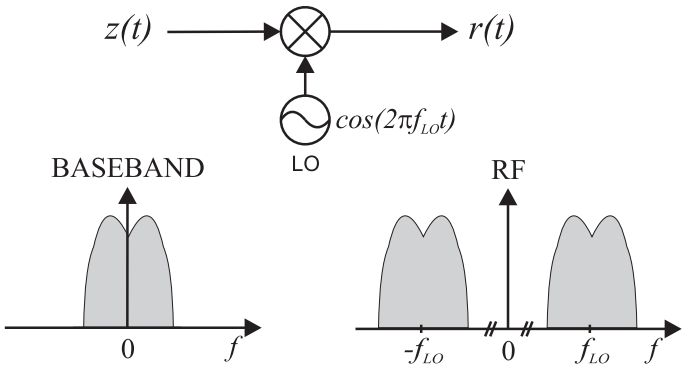


Figure 2.2: Real frequency translation.

2.2.2 Complex Mixing

The complex mixing approach performs the frequency translation with a complex-valued sinusoidal LO signal of frequency f_{LO}. The desired real- or complex-valued input signal is multiplied by the complex-valued LO signal to obtain the corresponding bandpass signal. The complex mixing technique results in single frequency translation without symmetry in spectral illustration. Using phasor notation, a complex-valued LO signal can be denoted $e^{j2\pi f_{LO}t}$ and with the well-known Euler theorem it can be shown to be a pair of orthogonal real-valued signals $cos(2\pi f_{LO}t) + jsin(2\pi f_{LO}t)$. The complex mixing process can be described by the

following equation

$$r(t) = z(t)e^{j2\pi f_{LO}t} = z(t)\left(cos(2\pi f_{LO}t) + jsin(2\pi f_{LO}t)\right). \tag{2.6}$$

The Fourier transform of the above equation yields the frequency domain result as

$$R(f) = Z(f - f_{LO}), \tag{2.7}$$

where it can be seen that the resulting bandpass signal does not have symmetric frequency components at frequencies $-f_{LO}$ and f_{LO} like the real mixing process, but only has a single energy concentration at frequency f_{LO}. Figure 2.3 comprises a general block diagram of the complex mixer and corresponding spectrum illustration before and after the mixing procedure.

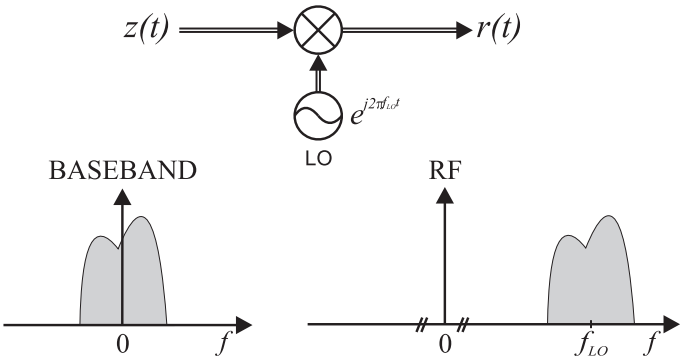


Figure 2.3: Complex frequency translation.

The complex mixing is in practice realized with real calculations as follows:

$$\begin{aligned}
r(t) &= z(t)e^{j2\pi f_{LO}t} && (2.8)\\
&= z(t)\left[cos(2\pi f_{LO}t) + jsin(2\pi f_{LO}t)\right] && (2.9)\\
&= z(t)cos(2\pi f_{LO}t) + jz(t)sin(2\pi f_{LO}t) && (2.10)\\
&= z_I(t)cos(2\pi f_{LO}t) - z_Q(t)sin(2\pi f_{LO}t) \\
&\quad + jz_Q(t)cos(2\pi f_{LO}t) + jz_I(t)sin(2\pi f_{LO}t) && (2.11)\\
&= r_I(t) + jr_Q(t), && (2.12)
\end{aligned}$$

where it can be seen that the same information is carried in $r_I(t)$ and $r_Q(t)$. This is also illustrated in Figure 2.4.

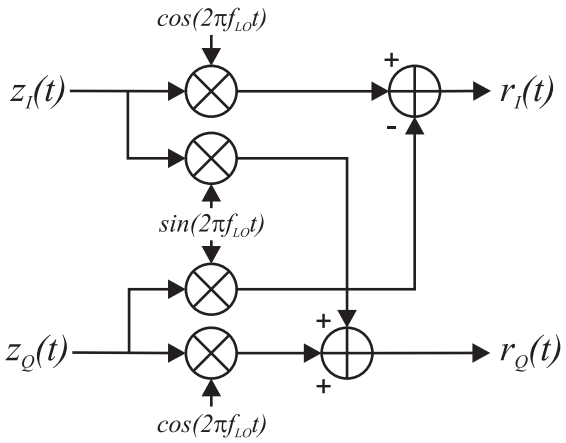


Figure 2.4: Complex frequency translation with real calculations.

2.3 Bandpass Transmission

In the context of wireless communication, a signal whose spectral density is concentrated in the frequencies around the origin (i.e. $f_c = 0$) is often referred to as a *baseband* or *low-pass signal* [16]. On the other hand, a signal with a spectrum centered on a non-zero frequency f_c, where f_c is usually called *carrier frequency*, is called a *bandpass signal* [16]. If we denote the complex baseband signal as $z(t)$, where $z(t) = z_I(t) + jz_Q(t)$, the corresponding analytic bandpass signal $s(t)$ is

$$s(t) = z(t)e^{j2\pi f_{LO}t}. \tag{2.13}$$

I/Q modulation is used in wireless transmission systems to convey a complex-valued signal over the real valued channel [71]. It is based on a technique where real and complex components of the signal are modulated by two trigonometric functions which have an exactly 90-degree phase difference. In practice this is done in such a way that the real component of the signal is modulated with the cosine wave and the complex component with the sine wave, and these two separately modulated signals are subtracted from each other, as seen in Figure 2.5. Using the lowpass-to-bandpass transformation the *quadrature carrier form* of the equation (2.13) is [64]

$$\begin{aligned} r(t) &= 2Re\{z(t)e^{j2\pi f_{LO}t}\} \\ &= 2z_I(t)\cos(2\pi f_{LO}t) - 2z_Q(t)\sin(2\pi f_{LO}t) \\ &= z(t)e^{j2\pi f_{LO}t} + z^*(t)e^{-j2\pi f_{LO}t}. \end{aligned} \tag{2.14}$$

It should be noted that taking real part of the complex signal corresponds to choosing the real-valued terms from the (2.11) or (2.13). Moreover, the frequency domain

representation of (2.14) is

$$R(f) = Z(f - f_{LO}) + Z^*(-f - f_{LO}). \tag{2.15}$$

According to (2.14) and (2.15), two real-valued signals $z_I(t)$ and $z_Q(t)$ can be transmitted over the same bandwidth, resulting in increased spectral efficiency.

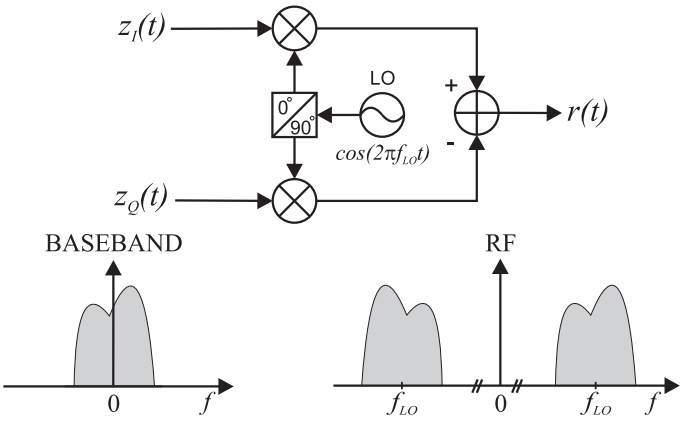


Figure 2.5: Basic I/Q Mixer.

If recovery of the transmitted RF signal $r(t)$ is considered, the bandpass-to-lowpass transformation is needed [64]. Moreover, in the frequency domain the FT $R(f)$ of the signal $r(t)$ has Hermitian symmetry about the zero frequency but not about the carrier frequency f_{LO}, as shown in Figure 2.5. It appears that the signal concentrated at frequency f_{LO} carries exactly the same information as the signal concentrated at frequency $-f_{LO}$ [83]. They are merely mirror images of each other. Either one of the frequency components in (2.15) or (2.14) can be chosen for further processing without loss of data.

2.4 Transmitter Architectures

Recently, most communication transmitters have been based on the *superheterodyne* principle, which consists of multiple stages with amplifiers, RF and intermediate frequency (IF) filters, mixers and frequency synthesizers to provide sufficient band limitation for the desired frequency band, and to deal with non-idealities caused by analog parts in the transmitters chain [61, 55]. In consequence, the superheterodyne architecture is impractical for an integrated modern multi-standard communication system. Consequently, there has been increasing interest in the *direct-conversion* or *homodyne*, the *low-IF* transmitter architectures, and, the ultimate goal, the *all-digital* architecture. In general, the current trend in the evolution of transceiver architectures has been for the DSP part to move closer to the analog FE. In this

sense, the superheterodyne principle and the direct-sampling architecture are the two extremities. The low-IF and the direct-conversion approaches offer a high level of integration and low complexity, and promise multi-standard operation. On the other hand, both approaches are more vulnerable to mismatches between different analog components.

In the following, the main currently used and future transmitter architectures are discussed and their advantages and disadvantages highlighted. First the superheterodyne transmitter architecture is introduced in Subsection 2.4.1, where after low-IF and direct-conversion architectures are addressed in Subsections 2.4.2 and 2.4.3, respectively. At the end of this section, the all-digital architecture is discussed in Subsection 2.4.4.

2.4.1 Superheterodyne

The superheterodyne architecture is a classical transmitter architecture widely used in RF communication transceivers [55]. Hence, there are a number of different variants of the conventional set-up. The architecture is based on multiple mixing and filtering stages to provide sufficient spectral characteristics for the transmitted waveform. As a result, this architecture has a complicated and power-consuming structure, comprised of discrete analog components. Consequently, its integrability level is very low and multi-standard operability is restricted by the IF frequencies [56]. On the other hand, operation of the superheterodyne architecture is robust and it has superior I/Q matching due to the low operating frequency in the IF stages. Moreover, it avoids DC offset and LO leakage, as well as the $1/f$-noise problems [56]. The basic structure of the superheterodyne transmitter architecture founded on quadrature frequency translation can be seen in Figure 2.6. [61]

First, the baseband signal generated in the DSP parts of the transmitter is converted to an analog signal, and up-converted to IF by quadrature mixing. Thereafter, the IF signal is band-pass filtered before final up-conversion to the carrier frequency. It should be noted that, in general, frequency of the LO2 is fixed and the desired center frequency on the system frequency band is tuned with IF from LO1. Then, after final up-conversion, the RF signal is again filtered to reject up-conversion images and LO leakage, and amplified before radiation from the antenna.

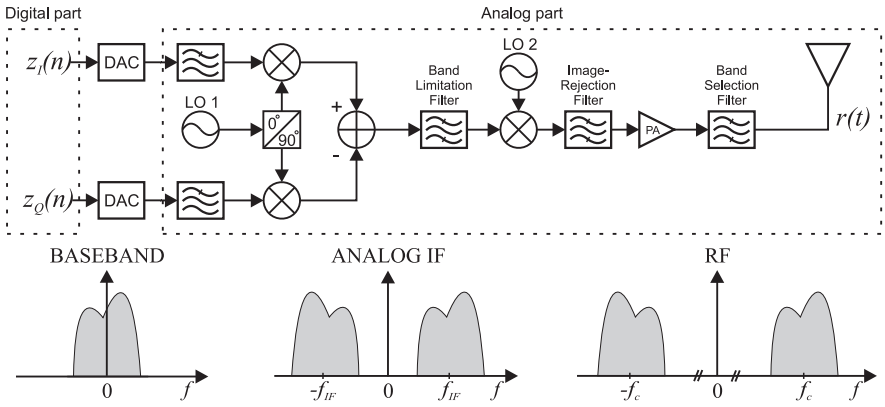


Figure 2.6: Block diagram of the traditional superheterodyne transmitter architecture.

2.4.2 Low-IF

The low-IF transmitter architecture usually consists of two larger functional parts, as a number of transmitter functionalities are already performed in the digital domain. Moreover, these segments comprise DSP-based and analog signal processing-based parts. The low-IF architecture has significantly decreased the required number of analog components, which yields higher integrability and lower cost [61, 1]. In addition, the low-IF architecture consumes less power than the superheterodyne. Similarly to the superheterodyne transmitter architecture, the low-IF principle overcomes the DC-offset and $1/f$-noise problems [56]. In contrast, the low-IF architecture suffers from mirror image problems due to mismatch between the analog I and Q branches. This is one of the most problematic drawbacks of this architecture. In addition, the low-IF architecture suffers from IF-dependent LO leakage [56]. One significant advantage from the system-point-of-view is that center frequency of the communication system can be tuned by adjusting the digital low-IF without changing analog LO frequency. In that case, the specifications of a digital-to-analog converter (DAC) has to meet requirements set by the IF. In general, the DSP part consists of generation of the desired signal waveform, channel filtering and digital up-conversion to the low IF. Thereafter, DACs convert the discrete-time digital signal to the continuous-time analog signal. The analog part consists of I/Q up-conversion to the desired RF center frequency, band-selection filter and power amplifier. A general block diagram of the low-IF transmitter architecture is given in Figure 2.7 [3].

In the low-IF transmitter architecture the digital baseband signal is first up-converted to digital low IF by digital complex mixing. Thereafter, the signal is fed through DAC and up-converted with a quadrature mixer to the desired carrier frequency. Finally, the signal is bandpass-filtered and amplified before radiation from the antenna.

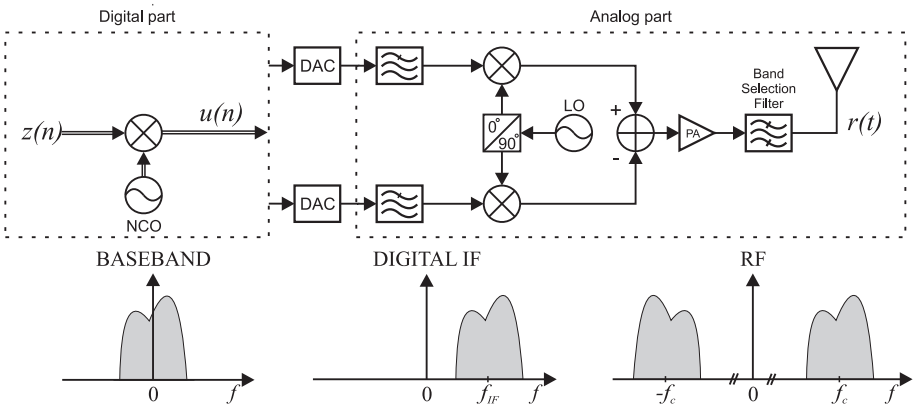


Figure 2.7: Block diagram of the low-IF transmitter architecture.

2.4.3 Direct-conversion

The direct-conversion architecture is also termed *homodyne* or *zero-IF* architecture. As the name indicates, this architecture performs up-conversion directly from baseband to RF frequencies [2]. The homodyne architecture is highly integrable, while majority of the transmitter functionalities are performed in digital domain. This makes homodyne an attractive choice for future multi-standard transceivers due to its integrability, low power consumption and low cost [1, 59]. On the other hand, the direct-conversion architecture is extremely vulnerable to the non-idealities of the remaining analog RF components [56]. In addition, these components should be very low-cost, which makes for inferior performance in the components. Another drawback of the architecture is that the signal center frequency is directly the LO frequency which sets higher quality demands on the LO. A general block diagram of the direct-conversion transmitter architecture can be seen in Figure 2.8. [59, 4]

In the direct-conversion transmitter architecture the desired digital baseband signal is first converted to an analog continuous-time signal. Thereafter, the signal is directly up-converted by a quadrature mixer to the carrier frequency f_c. Also with this architecture the signal is finally bandpass-filtered and amplified before radiation from the antenna.

2.4.4 All-Digital

The ultimate goal of the SDR or cognitive radio (CR) is the all-digital transceiver architecture. Basically in this architecture up-conversion of the desired signal to the RF frequencies is performed in the digital domain. The sampling frequency of the DAC should be high enough to generate a continuous-time analog signal which is directly on the desired RF center frequency f_c without reconstruction problems. As

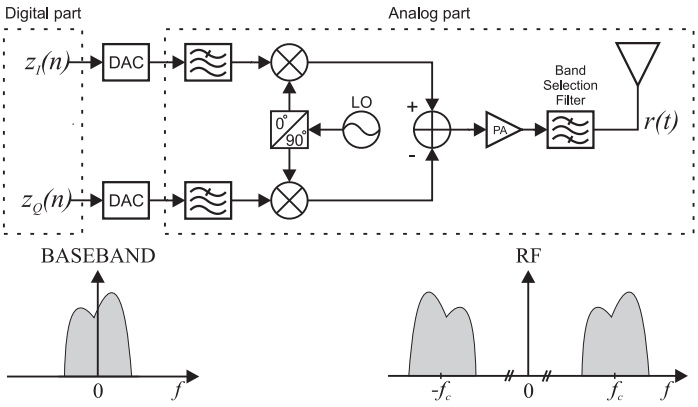


Figure 2.8: Direct-conversion transmitter architecture.

a result, this architecture is clearly the most highly digitized and it has the highest level of integrability. On the other hand, as stated above, a higher digitalization level means higher susceptibility to the impairments of the analog RF FE and difficulties with DACs become more significant. The all-digital architecture is usually based on a band-limited over-sampling approach [80]. A general block diagram of the direct-conversion transmitter architecture is given in Figure 2.9. [90, 85]

In the all-digital architecture the desired signal is up-converted to the desired carrier frequency f_C inside the digital parts of the transmitter. The RF signal is then converted from a digital discrete-time signal to an analog continuous-time signal. Thereafter come the PA and bandpass filter before radiating the signal out from the antenna.

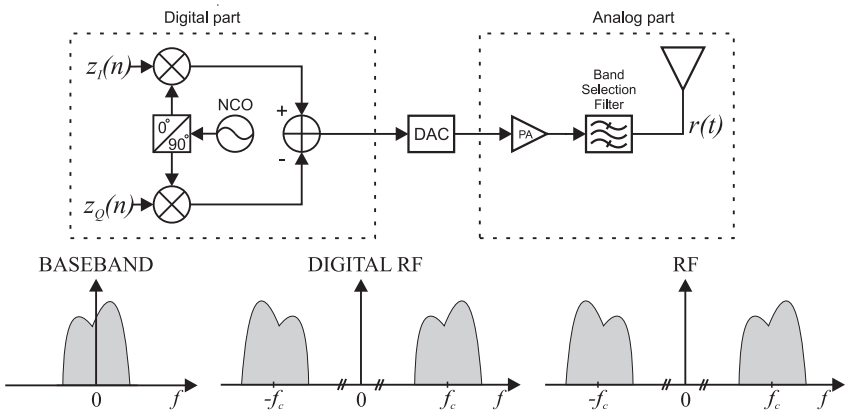


Figure 2.9: All-digital transmitter architecture.

2.5 RF Impairments in Different Transmitter Architectures

This section describes in brief all the significant RF impairments and nonlinearities present in radio transceivers. The main impairments and non-idealities degrading

the performance of future radio transmitters are I/Q mismatch, LO leakage, LO phase noise, and PA nonlinear distortion. In addition, DAC reconstruction phenomena and analog filter impairments also contribute to the signal deterioration.

First non-idealities and impairments in the mixers are addressed in Subsection 2.5.1. Thereafter, non-idealities in the PA stage of the transceiver are discussed in Subsection 2.5.2. At the end of the section, basic non-idealities in the analog-to-digital converter (ADC) and DAC stage are presented in Subsection 2.5.3.

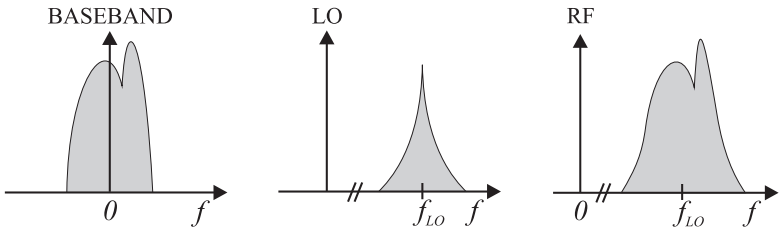
2.5.1 Impairments in Mixing Stage

The fundamental function of a mixer is to translate the original signal from baseband or IF to an RF carrier frequency, while keeping the characteristics of the original signal unchanged. This is achieved by multiplying the original signal by the LO signal, which is a pure undistorted sine wave. In practice, however, the signal is not pure sinusoid. Typical impairments and non-idealities in regard to LO and mixer are phase noise, I/Q imbalance, and LO leakage. In addition, the frequency offset of the LO can be considered an non-ideality but it does not originate from a single source.

Phase noise

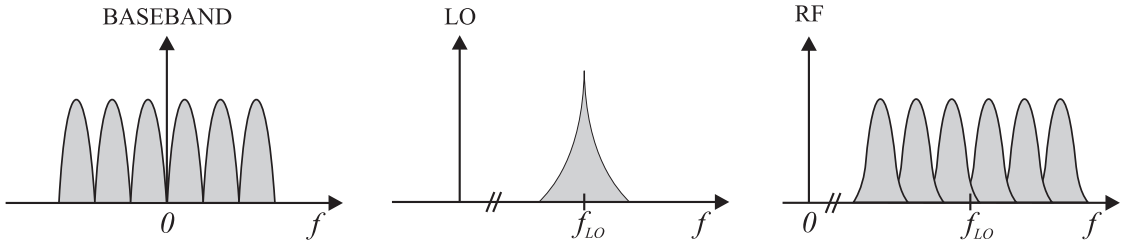
In practice, the local oscillator signal is not a pure sine signal at a single frequency due to phase noise and other imperfections in the oscillators. In the frequency domain the non-ideal LO signal is a spread version of a pure sine wave. An LO signal with phase noise can be seen in Figures 2.10(b) and 2.11(b) [53]. The effect of phase noise is a phase modulation of the local oscillator signal which is directly transferred to the original signal [43, 53]. From the point of view of the transmitted signal, mixing the impaired LO signal with the ideal baseband or intermediate frequency signal produces an RF signal with the phase noise of the LO superimposed on it. This impaired mixing results in in-band as well as out-of-band distortion [53, 30].

The effect of the phase noise on a single carrier signal can be seen from amplitude spectra in Figure 2.10 where the original undistorted signal is in Figure 2.10(a), the LO signal with phase noise in Figure 2.10(b) and the resulting signal to be transmitted in Figure 2.10(c). In practice, the in-band effect of a phase noise for single-carrier signal shows as a phase ripple as can be seen from the constellation plot of the signal $s(t)$ in Figure 2.12(a). This phenomenon is even more severe when using multicarrier waveforms where multiple adjacent sub-carriers start to interfere each



(a) Original signal (b) LO signal with phase noise (c) Output signal

Figure 2.10: Frequency domain illustration of local oscillator phase noise effects on a single-carrier signal.



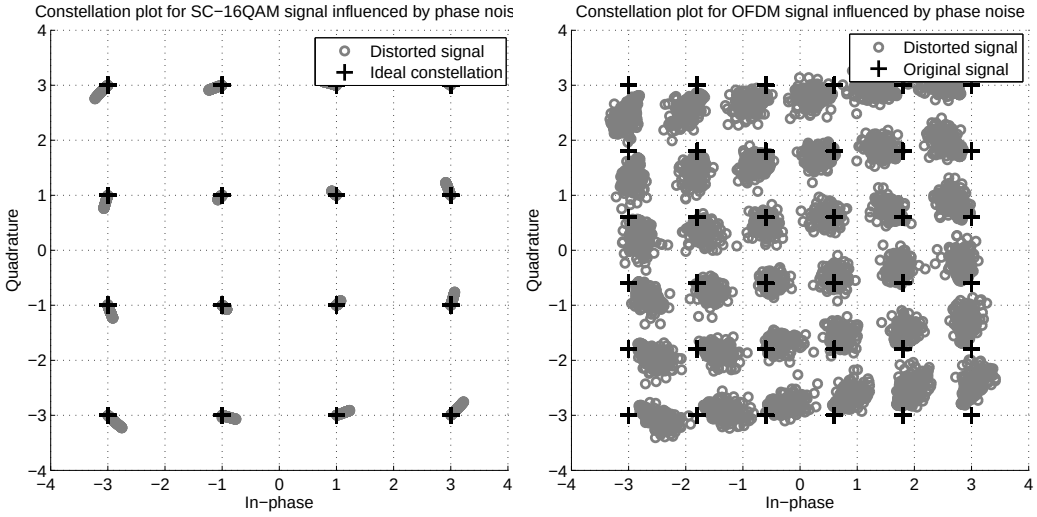
(a) Original signal (b) LO signal with phase noise (c) Output signal

Figure 2.11: Frequency domain illustration of local oscillator phase noise effects on a multicarrier signal.

other [76]. For multicarrier signals, phase noise shows in two ways which are common phase error, comparable to single-carrier signal, and inter-carrier interference (ICI). Frequency domain representation of phase noise effect for a multicarrier signal can be seen in Figure 2.11, where, again, $x(t)$ is the original undistorted signal, $l(t)$ the LO signal with phase noise and $s(t)$ the resulting signal to be transmitted. The corresponding constellation plot of the signal $s(t)$ in a multicarrier case can be seen in Figure 2.12(b).

I/Q Imbalance

I/Q imbalance is a result of finite image band attenuation in the transmitter. It depends on the relative amplitude and phase mismatch between the I and Q branches of the quadrature modulator. In addition, low-pass filters, DACs and other analog circuitry also contribute to the I/Q imabalance effects. In a direct-conversion transmitter I/Q imbalance creates self-interference and degradates the signal quality. In contrast, in low-IF transmitter architecture, I/Q imbalance is manifested as adjacent channel interference. [78, 17, 21, 26, 18, 8] This phenomenon is discussed in greater detail in Section 3.



(a) Single carrier 16-QAM (b) OFDM

Figure 2.12: Illustration of the LO signal phase noise in-band effect on the constellation.

LO Leakage

Usually in all transceivers the LO signal is conducted from the oscillator to the RF output signal. This phenomenon is called *LO leakage* and it introduces an undesirable spurious signal in the transmitted signal at the frequency of the LO signal [30]. The presence of an LO signal in the transmitted signal causes in-band interference in the case of receiver architectures which convert the baseband signal directly to the given RF center frequency [70, 2, 56]. On the other hand, LO leakage results in adjacent channel interference in low-IF transmitter architecture [30, 56]. An illustration of LO leakage in low-IF transmitter architecture is given in Figure 2.13.

2.5.2 Non-linearities in Power Amplifiers

The RF signal has to be always amplified before radiation from the antenna to attain a sufficient output power level. As a result, PA is one of the important primary components in any radio transceiver unit and is by nature nonlinear. As mentioned above, PA is responsible for amplifying the transmitted signal in such away, that it arrives at the receiver with a sufficient power level for successful detection. In addition, the efficiency should be maximized, especially on the terminal side, in order to maximize battery life. For this reason, linear PAs cannot be applied since their power efficiency is very poor. It is, thus, necessary to use more efficient nonlinear PAs and to drive them at or near the full power range [45]. Due to the driving of the PA in the nonlinear region, nonlinear distortion, both harmonic and intermodulation

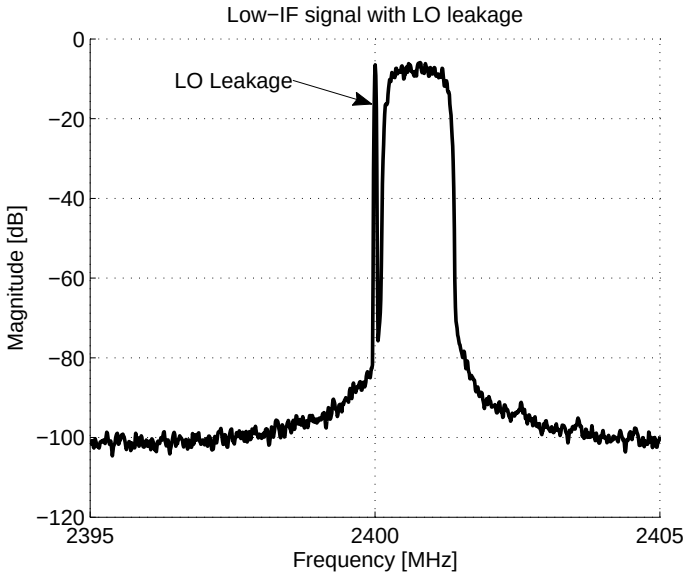


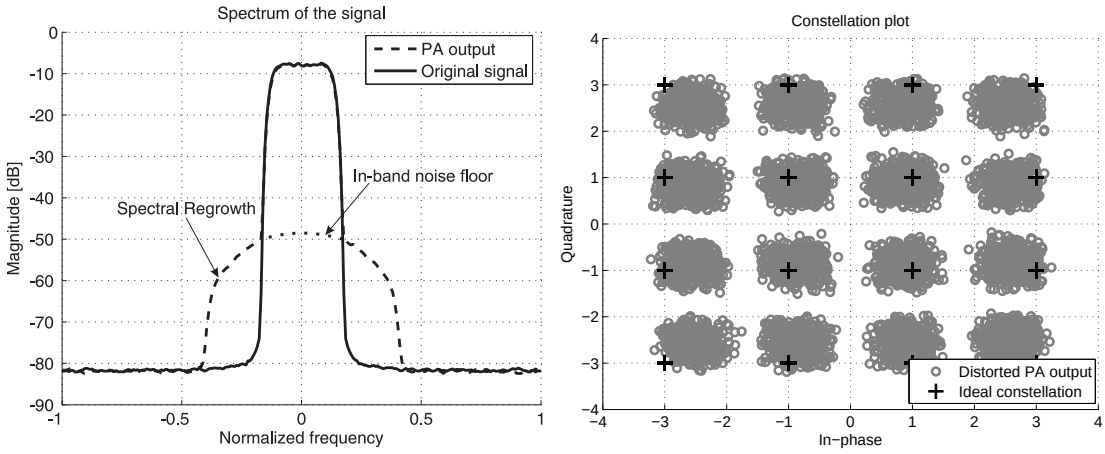
Figure 2.13: Frequency domain illustration of LO leakage. Low-IF signal with 16-QAM modulation, 8 times over-sampling, 22 % roll-off, 10 MHz sampling frequency and 2.4 GHz LO frequency.

distortion (IMD), is produced in the PA output [45]. As a result, in the frequency domain, the signal at the output of the PA contains not only the original signal frequency contents but also additional frequency components. The effect of these additional frequency components can be seen as in-band distortion, which results in an elevated noise floor and thus a increased bit-error rate (BER) [44]. Additionally, the spreading of the transmitted signal spectrum, called *spectral regrowth*, causes out-of-band distortion which interferes with adjacent channel signals [44].

The AM-AM conversion is a conversion between the normalized input amplitude of the PA to its normalized output amplitude and it is always present in PAs. In a linear PA, this AM-AM graph should be a straight line. A strictly memoryless PA can be modeled by its AM-AM characteristics.

The nonlinear distortion can be characterized as memoryless, quasi-memoryless or memory-containing, depending on the waveform used. For narrowband input signals, the PA does not show memory effects and the power amplifier can be regarded as a memoryless or quasi-memoryless system [46]. In the strictly memoryless case, the phase difference between the input and output signals is constant. Further, in the quasi-memoryless case, there is a varying phase difference without memory effects between input and output. As the bandwidth of the signal increases, the time span of the PA memory becomes comparable to the time variations at the input signal and the PA begins to show memory effects.

A conversion from amplitude modulation on the input signal to phase modulation on



(a) Spectra of original signal and distorted PA output.

(b) Constellations of original signal and distorted PA output.

Figure 2.14: Illustration of nonlinear PA effects on the transmitted signal. PA model was the Wiener model of a class AB amplifier. Single-carrier signal with 16-QAM modulation, 8 times over-sampling and 25 % roll-off.

the output signal is known as AM-PM conversion [46, 44]. In practice, this means that the delay in the PA varies over its input amplitude and results in a phase modulation. A quasi-memoryless PA can be characterized with one set of AM-AM and AM-PM conversions [46] and a PA with memory effects can be modeled with multiple sets of AM-AM and AM-PM conversions [44]. Moreover, the behavior of PAs with memory can be modeled for example with the Volterra model, Wiener, Hammerstein and the Wiener-Hammerstein models [40].

PAs can be made linear via many approaches. Common strategies are linear amplification using nonlinear components (LINC), envelope elimination and restoration (EER), peak-to-average power ratio (PAPR) reduction, feedforward linearization, and digital pre-distortion.

2.5.3 Non-idealities in Digital-to-Analog and Analog-to-Digital Converters

Among the most important parts of a radio transceiver are the DACs and ADCs. The DAC is used to interface the digital part of a transmitter with its analog circuits, namely analog front-end and, conversely, ADCs are used as an interface between the analog front-end and the digital part of the transceiver. The non-idealitites associated with DACs and ADCs are quantization noise, sampling clock offset, sampling jitter, and reconstruction phenomena. All these non-idealitites are briefly described in this section.

Quantization Noise

Quantization noise in a DAC and ADC occurs due to the limited number of bits which can be used to represent a sample value of the signal. A large number of bits is desirable to reduce quantization noise, but this increases the cost and power consumption of the DAC and ADC. Quantization noise appears as an additive noise process on top of the desired original signal. By reason of the additive nature of quantization noise, it is equally spread over the frequency span of the signal, and the impact of the quantization noise can thus be reduced by sampling the signal at a higher sampling frequency than the Nyquist theorem describes [47]. In other words, if higher over-sampling factors are used, the effect of quantization noise can be mitigated. The effect of quantization noise can be further reduced by using the so-called *delta-sigma* DACs, which shape the frequency spectrum of the quantization noise away from the desired signal band.

In general the maximum signal-to-noise ratio (SNR) of a signal can be calculated in decibels by the formula [57]

$$SNR_{MAX} = 6.02b + 4.76 - CF_{dB} + 10\log\left(\frac{f_S}{2f_B}\right)[dB], \tag{2.16}$$

where CF_{dB} is the crest-factor of the signal in decibels, f_S the sampling frequency and f_B the useful signal bandwidth. The rule-of-thumb for quantization noise is that every additional bit in the DAC or ADC increases the SNR of the desired signal by 6.02 dB.

Clipping

In the time domain, clipping can be seen as cutting the signal peaks which exceed the voltage range of the ADC. Usually this is a result of imperfect signal conditioning in the radio transceiver. Clipping results in odd order harmonics and intermodulation products and severe interference may occur if these new frequency components appear on a weak desired signal band. [47]

Sampling Jitter

In practical circuits, sampling is affected by uncertainty in the clock. Additionally, the delay between the logic generating the sampling phase and effective sampling is to some extent unpredictable [13]. This phenomenon is called *sampling jitter* and it

occurs due to phase noise in the sampling clock. As a consequence, the instants at which DAC converts the digital samples to an analog signal are not evenly spaced. This also applies to ADCs. As already stated, the main source of sampling clock jitter lies in instabilities in the LO clock and the buffer between the clock source and the DAC or ADC [47]. The impact of sampling jitter is that the actual sampling point is shifted from its ideal position, thus reducing SNR and BER. Sampling jitter has the greatest impact on bandpass signals, as the frequencies of the input signals are high, making the jitter an important parameter.

Sampling Clock Frequency Offset

One source of complication is sampling clock frequency offset. In transmitters, all the clocks and local oscillators are usually driven by one common reference clock and, in practice, the accuracy of the signals at each stage depends on the reference signal. If the sampling frequency of the DAC or ADC is offset by some factor with respect to the ideal sampling instant, the sampling points which are supposed to be taken at $(1, 2, ..., M)T_s$ are then taken at $(1, 2, ..., M)(1 + \delta)T_s$, such that a time shift $n\delta T_s$ appears on every nth sample [74]. In a way, this can be seen as a frequency offset it the output signal. Due to this, the signal is not sampled at the optimum sampling instant, which degrades performance of the analog-to-digital of digital-to-analog conversion.

Reconstruction Phenomena in DACs

Reconstruction of a digital signal waveform is usually done with a cascade of a sample-and-hold (SH) circuit and a lowpass (LP) reconstruction filter. A SH circuit of a DAC outputs a staircase-like analog waveform which can be seen in frequency domain as additional high-frequency terms. Following reconstruction filter should be able to remove all high-frequency terms from the SH output to smoothen the desired signal while keeping the original signal waveform. As a consequence, reconstruction filter may not be able to attenuate high frequency components adequately if signal bandwidth is a large fraction of sampling frequency. This may create signal folding on top of the desired signal. [57]

3. TRANSMITTER I/Q IMBALANCE AND DIGITAL PRE-DISTORTION CALIBRATION

Communication transmitters based on the analog domain I/Q up-conversion principle encounter a common problem in amplitude and phase mismatch. Although this complication is mainly inflicted by the I/Q modulators, which employ the principle of having equal gain and an exact 90-degree phase difference between I- and Q-branches, other analog FE components such as DACs and filters also contribute, in general, to the imbalance effects.

In an ideal transmitter, analog circuits in different branches have equal characteristics, but in practice, due to hardware manufacturing tolerances, a perfectly amplitude- and phase-balanced analog FE is not achievable. In addition, the electrical characteristics of analog components undergo short-time deviation due e.g. to temperature variation and, similarly, they are subject to change over the long term due to aging. These physical limitations result in a finite attenuation of the image signal and degradation of signal quality in I/Q processing.

One approach which might be thought to overcome these problems is to improve the quality of separate analog components to a level where the system performance loss due to the residual impairments remains acceptable [24, 79, 81]. However, such an approach may not be feasible in future radio transmitter architectures for the following reasons. The first drawback is that designing high-quality analog components satisfying all transmitter specifications will result in particularly expensive radio implementation. Additionally, sufficient and robust performance can only be realized over a narrow frequency band and practically only over a short period. These issues constrain the performance and flexibility of the transmitter.

Another desirable and feasible solution is to use DSP techniques to compensate the I/Q imbalance effects. The DSP-based calibration methods allow some errors in the analog design and have the advantage of achieving good performance without modifying the original transceiver architecture. In addition, DSP-based approaches offer the possibility to follow time-variant changes in the transmitter FE [18, 26, 17,

8].

The purpose of this chapter is to give a conception of I/Q imbalance and its effect in different transmitter architectures, and of how it can be efficiently mitigated. The chapter begins with transmitter signal models in narrowband frequency-independent and wideband frequency-selective cases. Moreover, a description of I/Q imbalance in general and the concept of image rejection ratio (IRR) is set out. Thereafter, mirror frequency interference (MFI) and its effects in different transmitter architectures are discussed. Finally, I/Q mismatch compensation schemes are derived and discussed.

3.1 Transmitter Signal Models

I/Q imbalance effects can be either frequency-independent or frequency-selective depending on the bandwidth of the desired signal and properties of the used electronics [8, 92]. In general, wideband signals usually experience frequency-selective I/Q imbalance effects, which means that I/Q imbalance parameters vary over the desired frequency band. On the other hand, narrowband signals experience only constant amplitude and phase mismatch.

In this section transmitter signal models for frequency-independent and frequency-selective I/Q imbalance effects are addressed and their effects on the desired signal are evaluated.

3.1.1 Narrowband Frequency-Independent Model

The narrowband frequency-independent behavior of I/Q imbalance stems from relative frequency-flat differences between the analog components of the I/Q modulator. A conceptual illustration of the frequency-independent transmitter model can be seen in Figure 3.1.

If the above-mentioned imbalance model is considered as a transmitter signal model, the imbalanced complex LO signal is modeled as [83]

$$x_{LO}(t) = \cos(\omega_c t) + j g_T \sin(\omega_c t + \phi_T), \tag{3.1}$$

where g_T and ϕ_T are transmitter amplitude and phase imbalances, respectively.

To obtain a better and more illustrative conception of the I/Q mismatch effect the model in (3.1) is further expanded. Denote the ideal baseband equivalent transmit

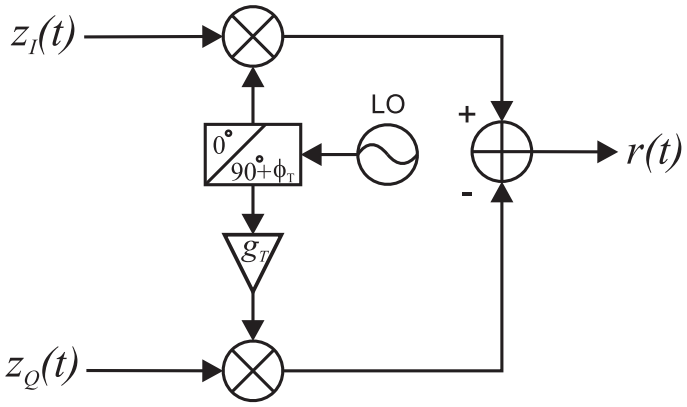


Figure 3.1: Block diagram of the narrowband frequency independent transmitter model.

signal as $z(t) = z_I(t) + jz_Q(t)$. Then the corresponding I/Q up-converted signal $r(t)$ is

$$r(t) = \Re\{z(t)x_{LO}(t)\} = z_I(t)\cos(\omega_c t) - z_Q(t)g_T\sin(\omega_c t + \phi_T). \tag{3.2}$$

This can be expanded with the trigonometric identity $\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$. Using the trigonometric identity and applying Euler's theorem, (3.2) becomes

$$\begin{aligned} r(t) &= [z_I(t) - z_Q(t)g_T\sin(\phi_T)]\cos(\omega_c t) - z_Q(t)g_T\cos(\phi_T)\sin(\omega_c t) && (3.3)\\ &= \left[z_I(t) - z_Q(t)g_T\frac{e^{j\phi_T} - e^{-j\phi_T}}{2j}\right]\frac{e^{j\omega_c t} + e^{-j\omega_c t}}{2} \\ &\quad -z_Q(t)g_T\frac{e^{j\phi_T} + e^{-j\phi_T}}{2}\frac{e^{j\omega_c t} - e^{-j\omega_c t}}{2j}. && (3.4)\end{aligned}$$

With further manipulations and rearranging the terms, the above equation can be written as

$$r(t) = \left[z_I(t) - z_Q(t)g_T\frac{e^{j\phi_T}}{j}\right]\frac{e^{j\omega_c t}}{2} + \left[z_I(t) + z_Q(t)g_T\frac{e^{-j\phi_T}}{j}\right]\frac{e^{-j\omega_c t}}{2}. \tag{3.5}$$

As is known, branch terms $z_I(t)$ and $z_Q(t)$ can also be expressed with the help of $z(t)$ and its conjugate in the following way:

$$z_I(t) = \frac{z(t) + z^*(t)}{2}, \quad z_Q(t) = \frac{z(t) - z^*(t)}{2j}. \tag{3.6}$$

Substituting (3.6) to (3.5), the imbalanced transmitted signal becomes

$$\begin{aligned} r(t) &= \left[\frac{z(t)+z^*(t)}{2} - \frac{z(t)-z^*(t)}{2j} g_T \frac{e^{j\phi_T}}{j}\right] \frac{e^{j\omega_c t}}{2} \\ &\quad + \left[\frac{z(t)+z^*(t)}{2} + \frac{z(t)-z^*(t)}{2j} g_T \frac{e^{-j\phi_T}}{j}\right] \frac{e^{-j\omega_c t}}{2} \qquad (3.7) \\ &= [g_1 z(t) + g_2 z^*(t)] \frac{e^{j\omega_c t}}{2} + [g_1 z(t) + g_2 z^*(t)]^* \frac{e^{-j\omega_c t}}{2} \qquad (3.8) \\ &= \Re\left\{[g_1 z(t) + g_2 z^*(t)]\, e^{j\omega_c t}\right\}, \qquad (3.9) \end{aligned}$$

where non-conjugate and conjugate term weights g_1 and g_2 are

$$g_1 = \frac{1 + g_T e^{j\phi_T}}{2}, \quad g_2 = \frac{1 - g_T e^{j\phi_T}}{2}. \qquad (3.10)$$

Corresponding baseband equivalent of the imbalanced RF signal for narrowband signal model can be denoted as

$$x(t) = g_1 z(t) + g_2 z^*(t). \qquad (3.11)$$

From (3.10) and (3.11) it can be clearly seen that original signal is represented by positive and negative frequencies weighted with g_1 and g_2, respectively. As a result, the original signal on positive frequencies is partly manifested on negative mirror frequencies. Moreover, if we consider the situation where gain factor $g_T = 1$ and phase difference $\phi_T = 0$, we obtain $g_1 = 1$ and $g_2 = 0$. This yields a perfectly balanced signal which has no mirror frequency term present. The ratio between the squared amplitudes of g_1 and g_2 is called *mirror image attenuation* or *image rejection ratio* and can be stated on a decibel scale as

$$IRR_{dB} = 10 \log_{10}\left(\frac{|g_1|^2}{|g_2|^2}\right). \qquad (3.12)$$

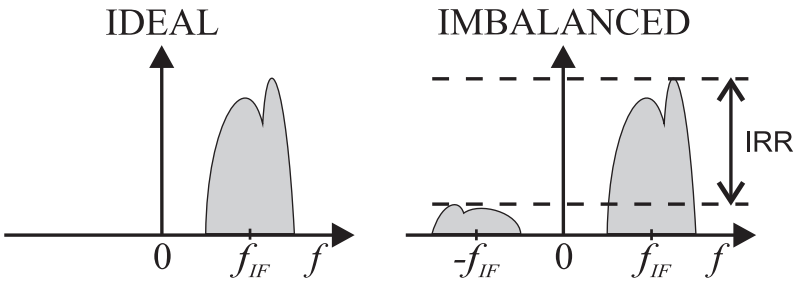


Figure 3.2: Illustration of IRR in frequency domain in case of low-IF transmitter. Ideal spectrum on the left and imbalanced spectrum on the right.

In general, IRR figure describes how much the mirror image is attenuated compared to the desired signal. A conceptual spectral illustration of the IRR is seen in Figure 3.2. In case of low-IF transmitter, a numerical illustration of this ratio with different amplitude and phase imbalances can be seen in Figure 3.3. As will be noted, both imbalance factors have an extensive influence on the IRR.

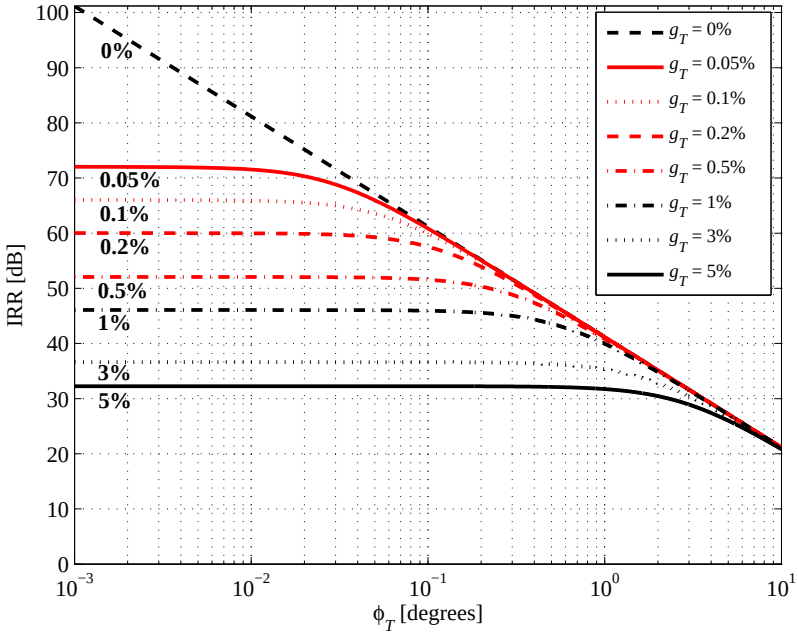


Figure 3.3: Image Rejection Ratio as a function of amplitude and phase imbalance. g_T and ϕ_T denote amplitude- and phase imbalance, respectively.

3.1.2 Wideband Frequency-Selective Model

In a wideband system context, the overall effective I/Q imbalance effects can vary as a function of frequency over the whole frequency band and wider bandwidths are more vulnerable to the frequency-selective behavior of I/Q imbalance [26, 17, 8, 92]. Consequently, the I/Q imbalance can be modeled as frequency-selective relative amplitude and phase difference between the I and Q branches. As stated earlier in this chapter, g_T and ϕ_T are the I/Q mixer amplitude and phase imbalances, respectively. In addition, the non-ideal filter transfer functions in the I and Q branches are modeled with the filters $h_I(t)$ and $h_Q(t)$. A conceptual block diagram of the frequency-selective I/Q imbalance model can be seen in Figure 3.4.

In the following the wideband frequency-selective signal model is derived using the frequency-independent I/Q imbalance model from (3.1) as a starting-point. For further analysis, denote the ideal baseband equivalent transmit signal as $z(t) =$

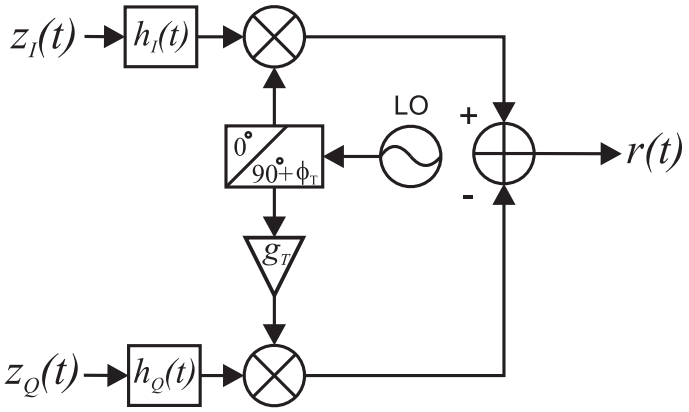


Figure 3.4: Block diagram of the wideband frequency-selective transmitter model.

$z_I(t) + jz_Q(t)$. Then the corresponding I/Q up-converted signal $s(t)$ is [8]

$$r(t) = (h_I(t) * z_I(t)) \cos(\omega_c t) - (h_Q(t) * z_Q(t))\, g_T \sin(\omega_c t + \phi_T). \tag{3.13}$$

Using the trigonometric identity $\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta)+\cos(\alpha)\sin(\beta)$, the above equation can be further expanded and it becomes

$$\begin{aligned} r(t) &= [h_I(t) * z_I(t)] \cos(\omega_c t) \\ &\quad - [h_Q(t) * z_Q(t)]\, g_T \left[\sin(\omega_c t)\cos(\phi_T) + \cos(\omega_c t)\sin(\phi_T)\right] \\ &= [h_I(t) * z_I(t) - \{h_Q(t) * z_Q(t)\}\, g_T \sin(\phi_T)] \cos(\omega_c t) \\ &\quad - [h_Q(t) * z_Q(t)]\, g_T \cos(\phi_T) \sin(\omega_c t). \end{aligned} \tag{3.14}$$

For further derivation the above equation has to be opened more and the terms have to be rearranged. By Euler's theorem (3.14) is tranformed to

$$\begin{aligned} r(t) &= \left[h_I(t) * z_I(t) - g_T \{h_Q(t) * z_Q(t)\} \frac{e^{j\phi_T} - e^{-j\phi_T}}{2j}\right] \frac{e^{j\omega_c t} + e^{-j\omega_c t}}{2} \\ &\quad - \left[g_T \{h_Q(t) * z_Q(t)\} \frac{e^{j\phi_T} + e^{-j\phi_T}}{2}\right] \frac{e^{j\omega_c t} - e^{-j\omega_c t}}{2j}. \end{aligned} \tag{3.15}$$

Substituting (3.6) into (3.15) and rearranging the result, we obtain

$$\begin{aligned}
r(t) &= \left[h_I(t) * \frac{z(t)+z^*(t)}{2} - g_T\left\{h_Q(t) * \frac{z(t)-z^*(t)}{2j}\right\}\frac{e^{j\phi_T}-e^{-j\phi_T}}{2j}\right] \\
&\quad \cdot\frac{e^{j\omega_c t}+e^{-j\omega_c t}}{2} \\
&\quad + \left[h_I(t) * \frac{z(t)+z^*(t)}{2} - g_T\left\{h_Q(t) * \frac{z(t)-z^*(t)}{2j}\right\}\frac{e^{j\phi_T}+e^{-j\phi_T}}{2}\right] \\
&\quad \cdot\frac{e^{j\omega_c t}-e^{-j\omega_c t}}{2j} \\
&= \left[h_I(t) * \frac{z(t)+z^*(t)}{2} - g_T\left\{h_Q(t) * \frac{z(t)-z^*(t)}{2j}\right\}\frac{e^{j\phi_T}}{j}\right] \\
&\quad \cdot\frac{e^{j\omega_c t}}{2} \\
&\quad + \left[h_I(t) * \frac{z(t)+z^*(t)}{2} + g_T\left\{h_Q(t) * \frac{z(t)-z^*(t)}{2j}\right\}\frac{e^{-j\phi_T}}{j}\right] \\
&\quad \cdot\frac{e^{-j2\omega_c t}}{2} \qquad (3.16) \\
&= [g_1(t) * z(t) + g_2(t) * z^*(t)]\,\frac{e^{j\omega_c t}}{2} \\
&\quad + [g_1(t) * z(t) + g_2(t) * z^*(t)]^*\,\frac{e^{-j\omega_c t}}{2} \qquad (3.17) \\
&= \Re\left\{[g_1(t) * z(t) + g_2(t) * z^*(t)]\,e^{j\omega_c t}\right\}, \qquad (3.18)
\end{aligned}$$

where the imbalance filter impulse responses $g_1(t)$ and $g_2(t)$ are defined as

$$g_1(t) = \frac{h_I(t) + g_T e^{j\phi_T} h_Q(t)}{2} \qquad (3.19)$$

and

$$g_2(t) = \frac{h_I(t) - g_T e^{j\phi_T} h_Q(t)}{2}. \qquad (3.20)$$

The corresponding baseband equivalent of the imbalanced RF signal is then [83]

$$x(t) = g_1(t) * z(t) + g_2(t) * z^*(t). \qquad (3.21)$$

On the other hand, the frequency-selective nature can also be considered as being only relative imbalance between the I and Q branches [8], thus branch filters $h_I(t)$ and $h_Q(t)$ can be considered as one relative imbalance filter $h_T(t)$. This can be seen by taking FT of (3.21), thereby the baseband equivalent signal becomes in frequency

domain as follows:

$$\begin{aligned} X(f) &= G_1(f)Z(f) + G_2(f)Z^*(-f) && (3.22)\\ &= \frac{H_I(f) + g_T e^{j\phi_T} H_Q(f)}{2} Z(f) + \frac{H_I(f) - g_T e^{j\phi_T} H_Q(f)}{2} Z^*(-f) && (3.23)\\ &= H_I(f) \left[\frac{1 + g_T e^{j\phi_T} \frac{H_Q(f)}{H_I(f)}}{2} Z(f) + \frac{1 - g_T e^{j\phi_T} \frac{H_Q(f)}{H_I(f)}}{2} Z^*(-f) \right]. && (3.24) \end{aligned}$$

It can be seen that $H_I(f)$ is a common factor for non-conjugate and conjugate branches, thus it can be neglected. As a result, (3.22) becomes

$$X(f) = \frac{1 + g_T e^{j\phi_T} H_T(f)}{2} Z(f) + \frac{1 - g_T e^{j\phi_T} H_T(f)}{2} Z^*(-f), \qquad (3.25)$$

where $H_T(f) = H_Q(f)/H_I(f)$. Consequently, the overall signal model is simplified slightly. This means that equation (3.19) becomes

$$g_1(t) = \frac{\delta(t) + g_T e^{j\phi_T} h_T(t)}{2} \qquad (3.26)$$

and the equation (3.20) simplifies to

$$g_2(t) = \frac{\delta(t) - g_T e^{j\phi_T} h_T(t)}{2}. \qquad (3.27)$$

The corresponding signal model can be seen as a block diagram in Figure 3.5. This signal model is used in all Matlab simulations in this thesis.

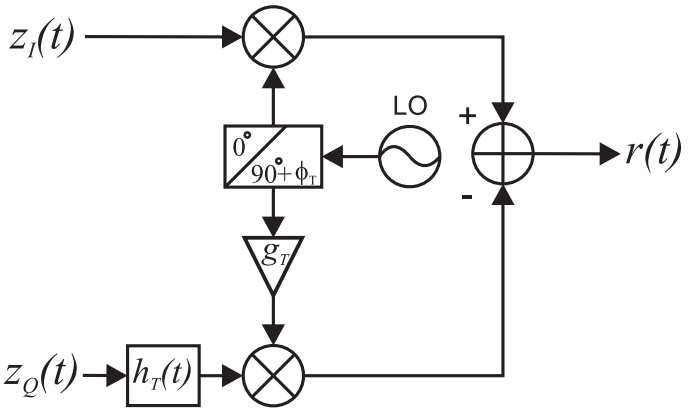


Figure 3.5: Block diagram of the wideband frequency-selective transmitter model with only one relative non-ideal branch filter causing the frequency-selective behavior of the I/Q imbalance.

Also with frequency-selective signal model, the mirror frequency attenuation can be found from (3.22) to be the relation between $G_1(f)$ and $G_2(f)$ in decibel scale as

follows:

$$IRR_{dB}(f) = 10\log_{10}\left(\frac{|G_1(f)|^2}{|G_2(f)|^2}\right). \tag{3.28}$$

Furthermore, in (3.28) and (3.43) FT $G_1(f)$ of $g_1(t)$ is

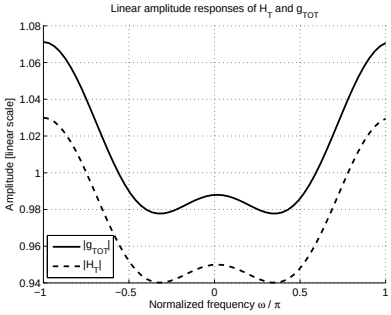
$$\begin{aligned} G_1(f) &= \frac{1}{2}\left[1 + g_T e^{j\phi_T} H_T(f)\right] && (3.29)\\ &= \frac{1}{2}\left[1 + g_T \|H_T(f)\| e^{j\phi_T} e^{j\arg(H_T(f))}\right] && (3.30)\\ &= \frac{1}{2}\left[1 + g_T \|H_T(f)\| e^{j(\phi_T + \arg(H_T(f)))}\right] && (3.31)\\ &= \frac{1}{2}\left[1 + g_{TOT}(f) e^{j\phi_{TOT}(f)}\right] && (3.32) \end{aligned}$$

and FT $G_2(f)$ of $g_2(t)$ can be formulated as

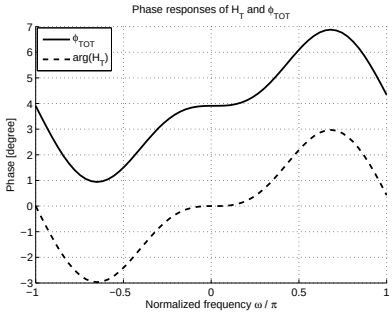
$$\begin{aligned} G_2(f) &= \frac{1}{2}\left[1 - g_T e^{j\phi_T} H_T(f)\right] && (3.33)\\ &= \frac{1}{2}\left[1 - g_T \|H_T(f)\| e^{j\phi_T} e^{j\arg(H_T(f))}\right] && (3.34)\\ &= \frac{1}{2}\left[1 - g_T \|H_T(f)\| e^{j(\phi_T + \arg(H_T(f)))}\right] && (3.35)\\ &= \frac{1}{2}\left[1 - g_{TOT}(f) e^{j\phi_{TOT}(f)}\right]. && (3.36) \end{aligned}$$

From (3.29)-(3.36) can be seen that total amplitude imbalance $g_{TOT}(f)$ is a product of modulator amplitude imbalance and amplitude response of the relative I/Q imbalance filter $h_T(t)$. Amplitude response of $h_{TOT}(f)$ can be found in Figure 3.6(a). Similarly, total phase mismatch $\phi_{TOT}(f)$ is a sum of modulator phase mismatch and phase response of $h_T(t)$. Phase response of $\phi_{TOT}(f)$ is in Figure 3.6(b).

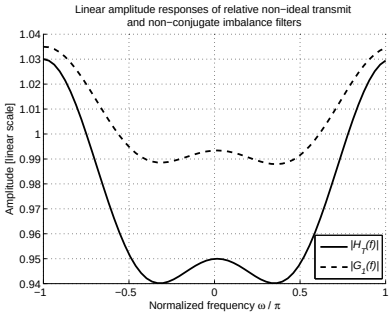
Throughout this thesis, most of the simulations are performed with frequency-selective I/Q imbalance model with gain imbalance 4 per cent, phase imbalance 4 degrees and non-ideal relative branch filter length N_T 3. In Figure 3.6, can be seen how amplitude and phase responses of imbalance filters form from non-ideal relative branch filter response and modulator amplitude and phase imbalance. Corresponding front-end IRR graph can be found in Figure 3.6(g).



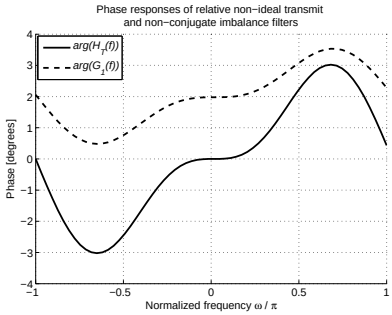
(a) Amplitude responses of $H_T(f)$ and $g_{TOT}(f)$.



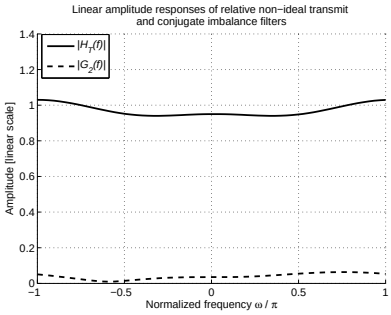
(b) Phase responses of $H_T(f)$ and $\phi_{TOT}(f)$.



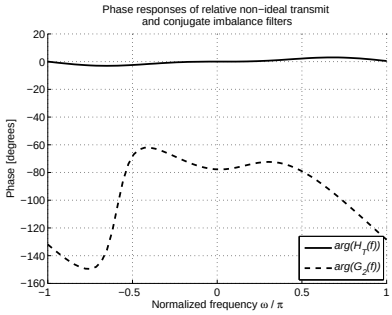
(c) Amplitude responses of $H_T(f)$ and $G_1(f)$.



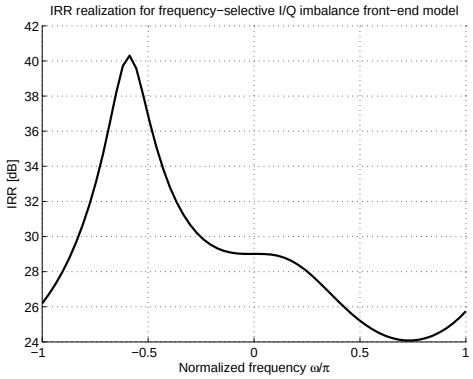
(d) Phase responses of $H_T(f)$ and $G_1(f)$.



(e) Amplitude responses of $H_T(f)$ and $G_2(f)$.



(f) Phase responses of $H_T(f)$ and $G_2(f)$.



(g) Resulting imbalanced FE IRR.

Figure 3.6: Formation of frequency-selective I/Q imbalance model with gain imbalance 4 per cent, phase imbalance 4 degrees and imbalance model length 3. Non-ideal relative branch filter impulse response $h_T = (0.97, -0.04, 0.02)$.

3.2 Mirror Frequency Interference Problem

Recent radio transmitters such as direct-conversion and low-IF utilizing I/Q signal processing are both vulnerable to mismatches between the I and Q branches. Although both use the quadrature mixing approach, the effects of MFI are different, though severe, for these architectures. In the following, the MFI problem is illustrated in the cases of direct-conversion and low-IF transmitters.

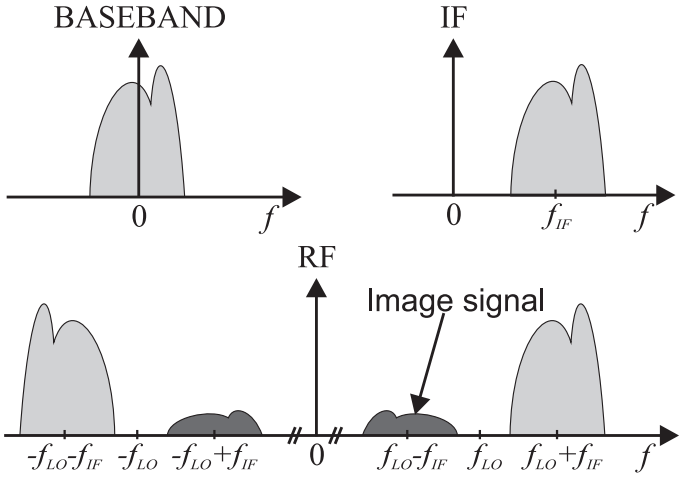


Figure 3.7: Illustration of I/Q imbalance effects in low-IF transmitter.

In the case of the low-IF transmitter the MFI problem is referred to as adjacent channel interference (ACI) because the mirror images originating in the transmitter are located equally far from the LO frequency f_{LO} as the desired signal. Namely, mirror images are centered at frequency $f_{MI} = f_{LO} - f_{IF}$. As a result, the mirror images cause interference to adjacent channels located in the frequency range from $f_{LO} - f_{IF} - B/2 = f_{MI} - B/2$ to $f_{LO} - f_{IF} + B/2 = f_{MI} + B/2$, where B is the useful bandwidth of the desired signal [87, 61, 1]. An illustration of this can be seen in Figure 3.7. If a multichannel wideband scenario is considered, there are multiple communication signals with different power levels and some of the signal can be even 60-100 dB stronger than the desired weaker signal. As a result, the image interference may entirely mask the desired weak signal on the adjacent frequency channel. Thus, the image attenuation of 30-40 dB provided by modern analog electronics [56] is clearly not sufficient for architectures with low-IF and clearly some kind of additional DSP-based processing is needed.

In contrast, the MFI problem in the direct-conversion transmitter causes self-interference in such that the mirror images resulting in the transmitter are located on the carrier frequency f_c, on the same frequency band as the desired signal [56, 17, 8, 73]. Thus, the image attenuation requirements for a direct-conversion transmitter are not as high as for a low-IF transmitter if adjacent channel interference is considered.

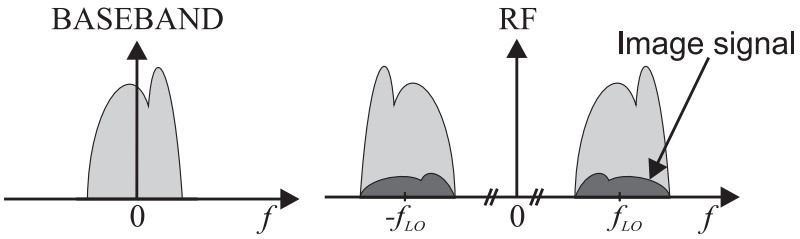
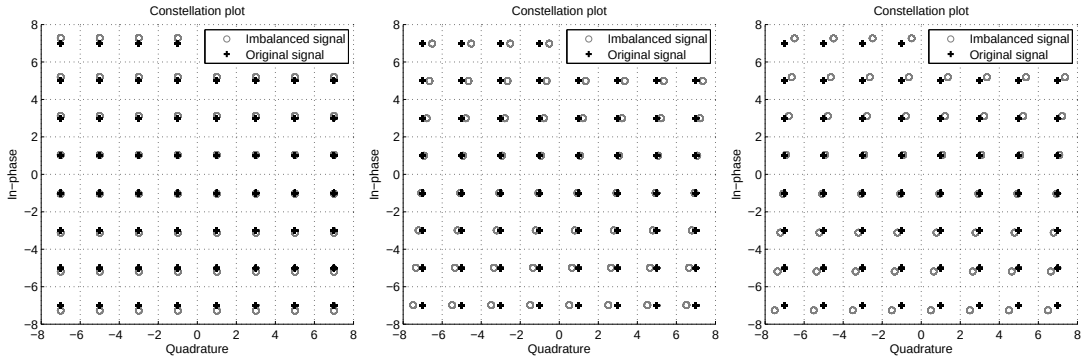


Figure 3.8: Illustration of I/Q imbalance effects in direct-conversion transmitter.

Of course, in-band distortion should be adequately attenuated. An illustration of self-interference in the frequency domain is given in Figure 3.8. The in-band effect of frequency-independent I/Q imbalance for direct-conversion transmitter is two-fold. The relative amplitude mismatch, greater than one, spreads the constellation vertically, as can be seen in Figure 3.9(a). On the other hand, relative amplitude mismatch, smaller than one, spreads the constellation horizontally. Moreover, the relative phase mismatch skews the constellation as illustrated in Figure 3.9(b). The combined effect of the amplitude and phase mismatch can be seen in Figure 3.9(c). Similar behavior is also seen with frequency-selective I/Q imbalance model, the only difference being that frequency-selectivity creates a conjugate inter-symbol interference to the signal which is seen as a noise-like behavior in the constellation plots. The in-band effect of I/Q imbalance with frequency-selective transmitter model can be seen in Figure 3.10.

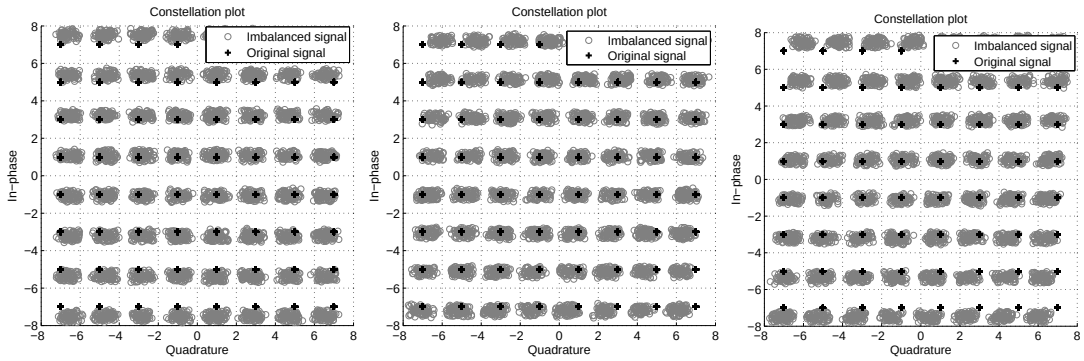


(a) Effect of amplitude imbalance (b) Effect of phase imbalance (c) combined effect of phase and amplitude imbalance

Figure 3.9: Effect of I/Q imbalance in direct-conversion transmitter. Single-carrier signal with 64-QAM modulation, 5 times over-sampling and 25 % roll-off. Non-frequency-selective I/Q imbalance model with gain imbalance 4 per cent and phase imbalance 4 degrees.

3.3 Transmitter I/Q Mismatch Estimation and Compensation

This section discusses digital pre-distortion-based transmitter I/Q imbalance calibration. The section briefly discusses known state of the art DSP-based I/Q imbalance



(a) Effect of amplitude imbalance (b) Effect of phase imbalance (c) combined effect of phase and amplitude imbalance

Figure 3.10: Effect of frequency-selective I/Q imbalance in direct-conversion transmitter. Single-carrier signal with 64-QAM modulation, 5 times over-sampling and 25 % roll-off. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees.

estimation and mitigation schemes. Thereafter, the widely-linear least-squares (WL-LS) I/Q imbalance estimation and calibration algorithm is mathematically described in detail and different possible recursive versions of it are formulated.

3.3.1 I/Q Imbalance Estimation and Mitigation Schemes

In addition to the WL-LS approach, there are also multiple other approaches to mitigate transmitter I/Q imbalance effects. Some of them consider only frequency-independent I/Q mismatch behavior and others estimate parameters for frequency-selective signal models. In this section a few algorithms are briefly described and discussed. Their differences to the WL-LS algorithms are also presented in a fairly general manner.

In [26] Ding et al. address a frequency-dependent modulator imbalance modeling and compensation scheme which is clearly the most similar to the WL-LS approach. The imbalance model in the paper is similar to the above frequency-selective model in Subsection 3.1.2. The difference is that the approach by Ding et al. does not include separate estimation of the imbalance branch filter impulse responses. Instead, the pre-distortion filter coefficients are estimated in one single step. The algorithm estimates the post-inverse of the quadrature modulator from the feedback loop signal and assumes that the corresponding optimum pre-inverse is equal to the estimated post-inverse. Finally, the algorithm uses the pre-inverse as a pre-distorter. Moreover, the algorithm can be used for direct-conversion and low-IF transmitter architectures.

Another approach by Windisch and Fettweis [87] is a method to compensate frequency-

independent modulator mismatch without complicated model fitting. The transmitter model used in this approach is quite similar to the frequency-independent model discussed in Subsection 3.1.1. The estimation stage relies on finding the actual amplitude and phase mismatch parameters from the feedback loop signal. They start the estimation fixing the residual phase mismatch and adapting the amplitude mismatch. After the amplitude mismatch estimate convergences it is fixed and phase mismatch is adapted in similar way. The approach is only applicable for low-IF transmitter architecture.

The work by Cavers and Liao in [17] proposes a frequency-independent I/Q imbalance estimation and mitigation scheme with an asymmetric cross-coupled compensation structure to gain reduced number of calculations. The I/Q imbalance estimation structure in the approach relies on transmitting four separate sinusoids with different phases on the channel center frequency and adapting the pre-distorter by measuring the corresponding modulator outputs from the feedback loop. It should be noted that this approach is only capable of calibrating a direct-conversion transmitter.

Approach by Vasudev and Oliver in [84] addresses an I/Q imbalance compensator structure similar to the WL-LS approach but the parameter estimation is done in a completely different manner. They have chosen to estimate frequency-selective I/Q imbalance parameters by sending complex sinusoids on selected frequencies over the whole desired frequency span and measuring the corresponding modulator outputs. Proper pre-distorter coefficients are then derived from the measurements. In consequence, this approach cannot be considered as an on-line calibration scheme. The algorithm is applicable for direct-conversion and low-IF transmitter architectures.

Last approach discussed here is proposed by Angrisani et al. found in [7]. They address I/Q imbalance parameter estimation with error vector magnitude (EVM) figure analysis. In practice, symbols are blindly recovered from the feedback loop signal, and constellation points are averaged and clustered in a way that mean deviation from the ideal symbol locations can be measured with EVM. The actual parameter estimation from EVM figures becomes a system of equations which is then solved with minimum-averaged-squared error method. The approach can only be used with direct-conversion transmitters.

From now on this thesis concentrates entirely on the WL-LS I/Q imbalance estimation and compensation approach found in [8] as it will be the algorithm to implement.

3.3.2 Widely-Linear Least-Squares Approach

The formulation in this section is based on the article written by Anttila et al. found in [8]. The aim of I/Q imbalance mitigation is to remove the conjugate term in (3.21) by properly pre-distorting the ideal transmitted data. Based on the above imbalance model the following pre-distorter of the form [8]

$$z_p(t) = z(t) + w(t) * z^*(t) \tag{3.37}$$

can be used. Here $w(t)$ represents the pre-distorter impulse response. From (3.21) and (3.37) it follows that the pre-distorted and imbalanced baseband equivalent signal is

$$x_p(t) = g_{1,p}(t) * z(t) + g_{2,p}(t) * z^*(t), \tag{3.38}$$

where the modified non-conjugate imbalance filter impulse response is

$$g_{1,p}(t) = g_1(t) + g_2(t) * w^*(t) \tag{3.39}$$

and the modified conjugate imbalance filter impulse response is

$$g_{2,p}(t) = g_2(t) + g_1(t) * w(t). \tag{3.40}$$

From (3.40) it can be derived that the optimum solution for the pre-distorter is the solution to

$$g_{2,p}(t) = g_2(t) + g_1(t) * w_{OPT}(t) = 0, \forall t. \tag{3.41}$$

The solution can be seen more intuitively as follows after FT

$$W_{OPT}(f) = \frac{-G_2(f)}{G_1(f)}. \tag{3.42}$$

In addition, if non-ideal filter impulse responses $g_1(t)$ and $g_2(t)$ are considered known, the IRR for estimated pre-distortion filter coefficients can be calculated as

$$IRR_{dB}(f) = 10 \log_{10} \left(\frac{|G_1(f) + G_2(f)W^*(-f)|^2}{|G_2(f) + G_1(f)W(f)|^2} \right). \tag{3.43}$$

A transmitter block diagram with digital pre-distortion structure realizing the above derivation can be seen in Figure 3.11.

In practice, however, the solution in (3.42) cannot be used directly to determine optimum pre-distortion filter coefficients because $g_1(t)$ and $g_2(t)$ are considered to be unknown. Thus, practical imbalance parameter estimation schemes using feedback

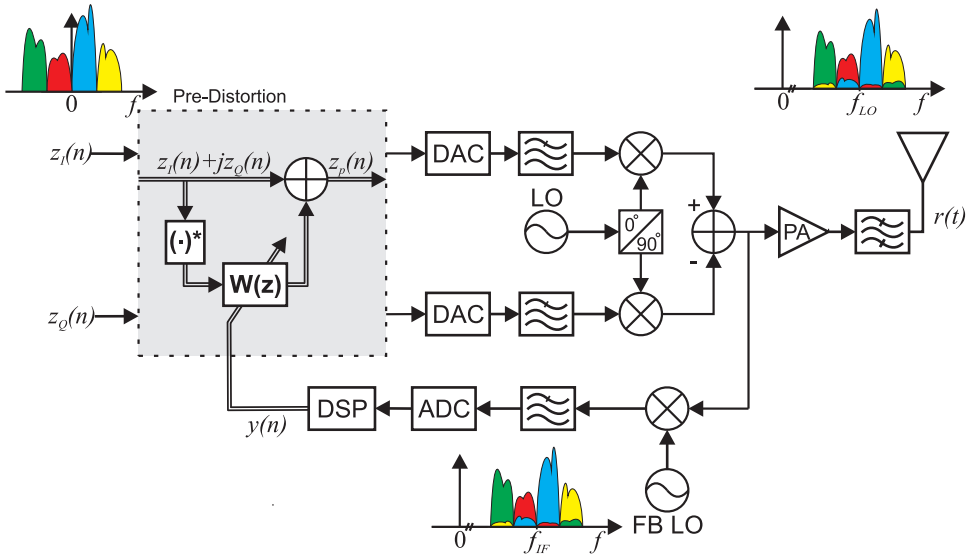


Figure 3.11: Block diagram of the estimation and pre-distortion structure with widely linear least-squares approach.

from RF back to transmitter digital parts are next addressed. Even if $g_1(t)$ and $g_2(t)$ could be considered known exactly, it would not ensure that finite length optimum pre-distortion filter $w_{OPT}(t)$ would give infinite IRR for the transmitter. In Figure 3.12 is reasonably long optimum pre-distortion filter impulse response on logarithmic scale which has only four to five dominating coefficients but it should be noted that there are also other non-zero coefficients, though they are insignificant. Furthermore, the optimum pre-distortion filter truncation effect can be seen in Figure 3.13.

First, a model for the observable feedback signal $y(t)$ is derived. The feedback loop employs real frequency translation to avoid any additional I/Q imbalance which may be produced in the case of quadrature down-conversion. The observable feedback signal, after translation to baseband and low-pass filtering, and after removal of the possible frequency offset and delay between the original signal and the feedback signal, can be shown to be

$$\begin{aligned} y(t) &= g e^{j\theta} h_{fb}(t) * \left(g_1(t) * z_p(t) + g_2(t) * z_p^*(t) \right) \\ &= \tilde{g}_1(t) * z_p(t) + \tilde{g}_2(t) * z_p^*(t), \end{aligned} \tag{3.44}$$

where g, θ, and $h_{fb}(t)$ denote the unknown feedback loop gain, phase, and impulse response, respectively. In equation (3.44), $\tilde{g}_1(t)$ and $\tilde{g}_2(t)$ represent observable imbalance filters and are of the form

$$\tilde{g}_1(t) = g e^{j\theta} h_{fb}(t) * g_1(t) \tag{3.45}$$

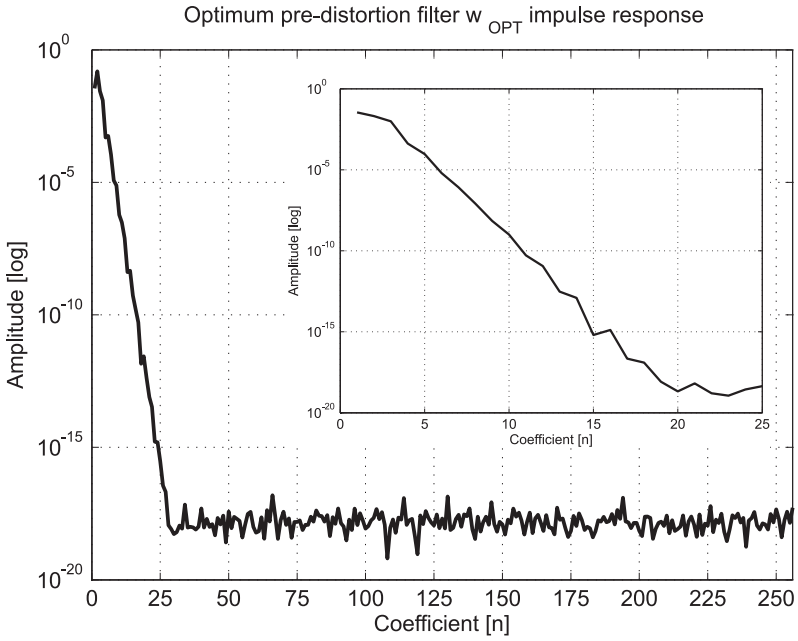


Figure 3.12: Optimum pre-distortion filter $w_{OPT}(t)$ impulse response of length 256.

and

$$\tilde{g_2}(t) = ge^{j\theta}h_{fb}(t) * g_2(t). \tag{3.46}$$

This shows that the actual imbalance filters $g_1(t)$ and $g_2(t)$ cannot be directly estimated. However, it may be noted that the optimum pre-distortion filter can still be found from the observed feedback loop signal. To show this, evaluate equation (3.42) using the Fourier transforms $\tilde{G_1}(f)$ and $\tilde{G_2}(f)$ of the observable filters and denoting FT of the feedback loop impulse response with $H_{fb}(f)$. This yields

$$-\frac{\tilde{G_2}(f)}{\tilde{G_1}(f)} = -\frac{ge^{j\theta}H_{fb}(f)G_2(f)}{ge^{j\theta}H_{fb}(f)G_1(f)} = -\frac{G_2(f)}{G_1(f)} = W_{OPT}(f). \tag{3.47}$$

This shows that the actual imbalance model with actual imbalance filters $g_1(t)$ and $g_2(t)$ does not have to be estimated. The optimum pre-distorter can be calculated with observed imbalance filters $\tilde{g_1}(t)$ and $\tilde{g_2}(t)$, because the feedback loop response is canceled in the pre-distorter coefficient calculations.

In general, the estimation of $\tilde{g_1}(t)$ and $\tilde{g_2}(t)$ is based on least-squares (LS) model fitting where original transmitted signal is adapted to the observed feedback loop signal. Next all the equations for the block LS method are derived, which in practice gives a better understanding of the way the estimation is performed. Next, the same equations are derived for the recursive least-squares (RLS)- and Gauss-Newton recursive least squares (GN)-based estimation method, which are more suitable for real-time implementation and give practically the same performance as the block

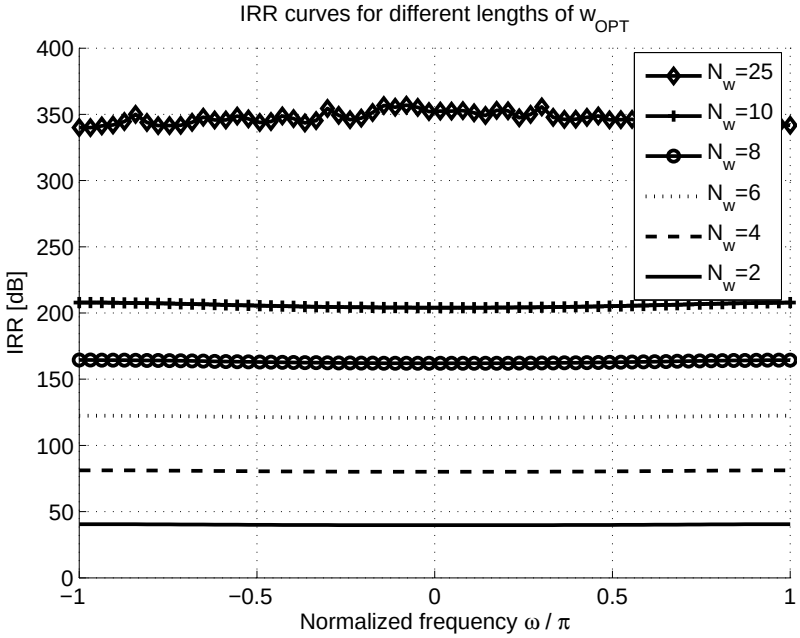


Figure 3.13: IRR as a function of normalized frequency with different optimum predistorter lengths. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees.

LS-based method [36, 77]. The same estimation problem can also be solved with mean square error (MSE) algorithms, e.g. least mean squares (LMS) [77] or normalized least mean square (NLMS) [77], but, as will be shown, these give less accurate estimation results than the RLS algorithm. Also approximate versions of the RLS algorithm, namely approximate recursive least squares (ARLS) [19] and fast approximate recursive least squares (FARLS) [20], are considered in view of their fairly good performance. On the other hand, the plain LMS algorithm evinces such weak performance that it will not even be considered. In contrast, the NLMS algorithm achieves reasonably good estimation accuracy and will be considered as a potential approach. The estimation equations for NLMS, ARLS, and FARLS are also given.

Block Least-Squares Approach

From now on, we switch to discrete time and vector-matrix notations for convenience. First, we derive an estimator for $\tilde{\mathbf{g}}_1$ and $\tilde{\mathbf{g}}_2$, which includes also feedback loop response, using a time-domain model fitting approach. Vectors $\tilde{\mathbf{g}}_1$ and $\tilde{\mathbf{g}}_2$ include the I/Q modulator and feedback loop responses. The aim is to find estimates $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$ which give the best fit between the original data sequence $z(n)$ and the observed feedback data sequence $y(n)$. The observed feedback data sequence can be

formulated for auto-correlation data windowing as [8]

$$\mathbf{y}(n) = \mathbf{Z}(n)\hat{\mathbf{g}}_1 + \mathbf{Z}^*(n)\hat{\mathbf{g}}_2 = [\mathbf{Z}(n) \;\; \mathbf{Z}^*(n)] \begin{bmatrix} \hat{\mathbf{g}}_1 \\ \hat{\mathbf{g}}_2 \end{bmatrix} = \mathbf{Z}_b(n) \begin{bmatrix} \hat{\mathbf{g}}_1 \\ \hat{\mathbf{g}}_2 \end{bmatrix}. \tag{3.48}$$

Here $\mathbf{y}(n) = [y(n) \; y(n-1) \; \cdots \; y(n-L_b+1)]^T$, L_b denotes the length of the observed feedback data sequence, and $\mathbf{Z}(n)$ is the *convolution matrix* formed from the original transmitted data sequence $\mathbf{z}(n)$. In general, if the LS solution is formed by the auto-correlation data windowing method the estimation process gives biased results [36]. This also applies to the WL-LS estimation approach. Consequently, covariance and pre-windowing methods should be utilized, as they have been shown during the work to be the best methods for the estimation process [48]. See Figure 3.15 for examples of the IRR curves with different data windowing methods [36]. For the covariance method the *convolution matrix* $\mathbf{Z}(n)$ of the original signal is formulated as follows

$$\mathbf{Z}(n) = \begin{bmatrix} z(n-N_g+1) & z(n-N_g+2) & \cdots & z(n) \\ z(n-N_g) & z(n-N_g+1) & & z(n-1) \\ \vdots & & \ddots & \vdots \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ z(n-L_b+1) & z(n-L_b) & \cdots & z(n-L_b-N_g+1) \end{bmatrix}, \tag{3.49}$$

where N_g denotes the length of estimated imbalance filters $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$. The corresponding observed feedback data sequence is

$$\mathbf{y}(n) = [y(n-N_g+1) \; y(n-N_g) \; \cdots \; y(n-L_b+1)]^T. \tag{3.50}$$

The imbalance filter coefficients $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$ best describing the data in the LS sense can then be solved from

$$\begin{bmatrix} \hat{\mathbf{g}}_1 \\ \hat{\mathbf{g}}_2 \end{bmatrix} = \mathbf{Z}_b^+(n)\mathbf{y}(n). \tag{3.51}$$

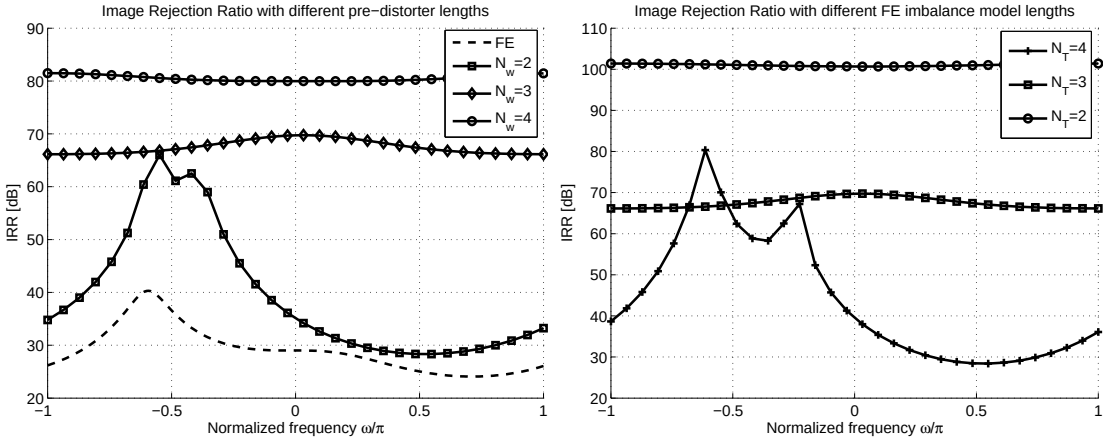
In equation (3.51) $\mathbf{Z}_b^+(n) = \left(\mathbf{Z}_b^H(n)\mathbf{Z}_b(n)\right)^{-1}\mathbf{Z}_b^H(n)$ is the *pseudo-inverse* of $\mathbf{Z}_b(n)$.

After estimating imbalance filter coefficients $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$ another model-fitting is needed to solve pre-distortion filter coefficients from (3.41). In practice, they can be solved with LS model-fitting with the equation

$$\mathbf{w} = -(\hat{\mathbf{G}}_1^H\hat{\mathbf{G}}_1)^{-1}\hat{\mathbf{G}}_1^H\hat{\mathbf{g}}_2^0, \tag{3.52}$$

where $\hat{\mathbf{G}}_1$ is a *convolution matrix* formed from $\hat{\mathbf{g}}_1$, and $\hat{\mathbf{g}}_2^0 = [\hat{\mathbf{g}}_2^T 0 \cdots 0]^T$ is a

zero-padded version of $\hat{\mathbf{g}}_2$ with $N_g - 1$ additional zeros. Pre-distortion filter $\mathbf{w}$ can be truncated to length N_w, where $1 \leq N_w \leq N_g$. Truncation has a significant effect on the maximum achievable performance of the I/Q imbalance mitigation algorithm. Moreover, illustration of the effect can be seen in Figure 3.14(a) when the frequency-selective I/Q imbalance model length has been fixed to three and pre-distorter lengths are from two to four. On the other hand, in Figure 3.14(b) shows how under- or over-determined estimation affects on the algorithm performance. This problem could be overcome with order-recursive adaptive filters but their computational complexity is considered too high, see e.g. [36]



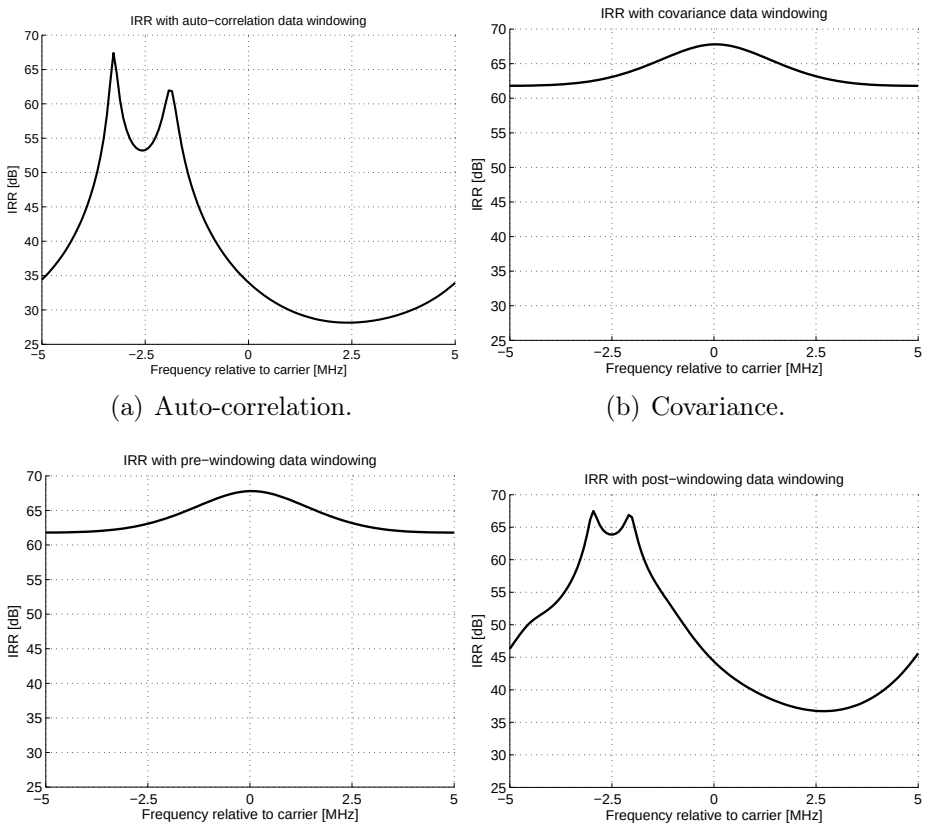
(a) Fixed I/Q imbalance model length, $N_T = 3$ and $N_w = N_g$.

(b) Fixed pre-distorter length, $N_w = N_g = 3$.

Figure 3.14: IRR as a function of normalized frequency when pre-distortion filter length and I/Q imbalance model lengths are not same. Low-IF signal with 16-QAM modulation, 35 % roll-off, and 5 times over-sampling. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees. 25,000 samples used for estimation process.

Recursive Least-Squares Approach

The RLS algorithm follows the LS principle but does not perform matrix inversions, which makes it computationally more efficient than plain LS. Recursion has been implemented through the *matrix inversion lemma*, which provides tools to create matrix inversions in a recursive manner [36, 77]. The following RLS formulation



(a) Auto-correlation. (b) Covariance.

(c) Pre-windowing. (d) Post-windowing.

Figure 3.15: Image Rejection Ratio for different data windowing methods as a function of normalized frequency. Low-IF signal with 16-QAM modulation, 5 times over-sampling and 30 % roll-off. Symbol rate was 2 MHz, sample rate 10 MHz and IF 2.55 MHz.

corresponds to the above block LS with covariance data windowing [36, 77].

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)} \tag{3.53}$$

$$e(n) = y(n) - \hat{\mathbf{g}}^H(n-1)\mathbf{u}(n) \tag{3.54}$$

$$\hat{\mathbf{g}}(n) = \hat{\mathbf{g}}(n-1)\mathbf{k}(n)e^*(n) \tag{3.55}$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1), \tag{3.56}$$

where $\mathbf{u}(n) = [z(n)\ z(n-1)\ \cdots\ z(n-N_g+1)\ z^*(n)\ z^*(n-1)\ \cdots\ z^*(n-N_g+1)]$, $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$ and $\hat{\mathbf{g}}(0) = \mathbf{0}$ with $2N_g$ zeros. Parameter δ is a small positive constant to ensure non-singularity of the covariance matrix of the original signal $\mathbf{z}(n)$. Likewise, it ensures non-singularity of $\mathbf{P}(n)$, which is in practice the inverse of the covariance matrix.

From estimated imbalance vectors $\hat{\mathbf{g}}$ we get $\hat{\mathbf{g}}_1 = [\hat{\mathbf{g}}(0)\ \cdots\ \hat{\mathbf{g}}(N_g-1)]^T$ and $\hat{\mathbf{g}}_2 = [\hat{\mathbf{g}}(N_g)\ \cdots\ \hat{\mathbf{g}}(2N_g-1)]^T$. Moreover, *zero-padded* versions $\hat{\mathbf{g}}_1^0$ and $\hat{\mathbf{g}}_2^0$ are $\hat{\mathbf{g}}_1^0 = [0\ 0\ \cdots\ \hat{\mathbf{g}}_1^T]^T$ and $\hat{\mathbf{g}}_2^0 = -[0\ 0\ \cdots\ \hat{\mathbf{g}}_2^T]^T$ both with $N_g - 1$ appended

zeros. After this pre-distrotion filter coefficients $\hat{\mathbf{w}}$ are again solved with recursive calculations comparable to equation (3.52), in basically a manner similar to the estimation of $\hat{\mathbf{g}}$. The only differences are that equation (3.54) is changed to

$$e(n) = \hat{\mathbf{g}}_2^0(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n), \tag{3.57}$$

where $\mathbf{u}(n) = [\hat{\mathbf{g}}_1^0(n)\ \hat{\mathbf{g}}_1^0(n-1)\ \cdots\ \hat{\mathbf{g}}_1^0(n-N_g+1)]$ and equation (3.55) receives the form

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1)\mathbf{k}(n)e^*(n). \tag{3.58}$$

Again pre-distortion filter $\hat{\mathbf{w}}$ can be truncated to length N_w, where $1 \leq N_w \leq N_g$.

Gauss-Newton Recursive Least-Squares Approach

The GN algorithm also follows the LS principle and does not perform matrix inversions, which makes it computationally more efficient than plain LS in a manner similar to the normal RLS algorithm. Recursion has been implemented similarly to the normal RLS algorithm with the additional parameter α, which controls recursion of the inverse of the covariance matrix $\mathbf{P}(n)$ of the original data. The following GN formulation also corresponds in principle to the above block LS with covariance data windowing.

$$\begin{aligned}
\mathbf{k}(n) &= \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{\frac{1-\alpha}{\alpha} + \lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)} && (3.59)\\
e(n) &= z(n) - \hat{\mathbf{g}}^H(n-1)\mathbf{u}(n) && (3.60)\\
\hat{\mathbf{g}}(n) &= \hat{\mathbf{g}}(n-1)\mathbf{k}(n)e^*(n) && (3.61)\\
\mathbf{P}(n) &= \frac{\lambda^{-1}}{1-\alpha}\left[\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)\right], && (3.62)
\end{aligned}$$

where $\mathbf{u}(n) = [z(n)\ z(n-1)\ \cdots\ z(n-N_g+1)\ z^*(n)\ z^*(n-1)\ \cdots\ z^*(n-N_g+1)]$, $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$ and $\hat{\mathbf{g}}(0) = \mathbf{0}$ with $2N_g$ zeros. Parameter α in equations (3.59) and (3.62) is a small positive constant which is greater than zero. Additionally, parameter δ is the same small positive constant to ensure non-singularity of the covariance matrix of the original signal $\mathbf{z}(n)$ as in normal RLS formulation.

From estimated imbalance vectors $\hat{\mathbf{g}}$ we get $\hat{\mathbf{g}}_1 = [\hat{\mathbf{g}}(0)\ \cdots\ \hat{\mathbf{g}}(N_g-1)]^T$ and $\hat{\mathbf{g}}_2 = [\hat{\mathbf{g}}(N_g)\ \cdots\ \hat{\mathbf{g}}(2N_g-1)]^T$. Furthermore, the *zero-padded* versions of $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$, namely $\hat{\mathbf{g}}_1^0$ and $\hat{\mathbf{g}}_2^0$, are $\hat{\mathbf{g}}_1^0 = [0\ 0\ \cdots\ \hat{\mathbf{g}}_1^T]^T$ and $\hat{\mathbf{g}}_2^0 = -[0\ 0\ \cdots\ \hat{\mathbf{g}}_2^T]^T$, both with $N_g - 1$ appended zeros. Thereafter pre-distrotion filter coefficients $\hat{\mathbf{w}}$ are solved again with recursive calculations exactly corresponding to the normal RLS

algorithm in equations (3.57)-(3.58) [77].

$$\mathbf{u}(n) = [\hat{\mathbf{g}}_1^0(n)\ \hat{\mathbf{g}}_1^0(n-1)\ \cdot\ \cdot\ \cdot\ \hat{\mathbf{g}}_1^0(n-N_g+1)] \tag{3.63}$$

$$e(n) = \hat{\mathbf{g}}_2^0(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n) \tag{3.64}$$

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1)\mathbf{k}(n)e^*(n). \tag{3.65}$$

Again pre-distortion filter $\hat{\mathbf{w}}$ can be truncated to length N_w, where $1 \leq N_w \leq N_g$.

Fast Approximate Recursive Least-Squares Approach

As is commonly known [32, 91], the main computational burden of the normal RLS algorithm comes from the recursion of $\mathbf{P}(n)$, which is the inverse of the covariance matrix of the original data sequence. The FARLS algorithm is an approximation to the regular RLS algorithm, being thus an approximation of the LS principle. The algorithm is actually more similar to the NLMS algorithm than to the normal RLS algorithm. The FARLS algorithm makes the assumption that the matrix $\mathbf{P}(n)$ can be replaced by a vector which includes only the diagonal elements of the matrix $\mathbf{P}(n)$.

Let the ith diagonal element of the diagonal matrix $\mathbf{D}(n)$ be defined with [20]

$$D_i(n) = \sum_{j=1}^{n-i+1} \|y(j)\|^2. \tag{3.66}$$

Next, denote the inverse of the diagonal elements of the matrix $\mathbf{D}(n)$ by $\mathbf{d}(n)$, and the ith element of the inverse is thus defined as [20]

$$d_i(n) = \frac{1}{\sum_{j=1}^{n-i+1} \|y(j)\|^2}. \tag{3.67}$$

The vector $d(n)$ can then be updated by recursion [20]

$$d_i(n+1) = d_{i-1}(n) \tag{3.68}$$

and

$$d_1(n+1) = \frac{1}{\sum_{j=1}^{n+1} \|y(j)\|^2}. \tag{3.69}$$

In practice, the above formulation means that updating the vector $d(n)$ requires only recalculation of $d_1(n)$. Using equations (3.68) and (3.69) the adaptation can

be performed with equations

$$d_1(n+1) = \frac{1}{\sum_{j=1}^{n+1} \|y(j)\|^2} \tag{3.70}$$

$$e(n) = y(n) - \hat{\mathbf{g}}^H(n-1)\mathbf{u}(n) \tag{3.71}$$

$$\hat{\mathbf{g}}(n) = \hat{\mathbf{g}}(n-1) + \mathbf{d}(n)e^*(n), \tag{3.72}$$

where $\mathbf{u}(n) = [z(n)\ z(n-1)\ \cdots\ z(n-N_g+1)\ z^*(n)\ z^*(n-1)\ \cdots\ z^*(n-N_g+1)]$. The algorithm is initialized with $\mathbf{d}(1) = [z(1)^{-2}\ 0\ 0\ \cdots\ 0]$ with $N_g - 1$ zeros and $\hat{\mathbf{g}}(0) = \mathbf{0}$ with $2N_g$ zeros.

The pre-distortion filter coefficients $\hat{\mathbf{w}}$ could be estimated with recursion based on equations (3.70)-(3.72), but, in practice, there are only N_g adaptation iterations and the FARLS algorithm will not have a sufficient number of iterations to converge. In consequence, it has been found that the estimation process for the pre-distortion filter coefficients $\hat{\mathbf{w}}$ should be performed with RLS recursion, which has been discussed in Section 3.3.2.

Approximate Recursive Least-Squares Approach

As already stated the main computational burden of the normal RLS algorithm comes from the recursion of $\mathbf{P}(n)$. The ARLS algorithm is another approximation to the regular RLS algorithm, thus being also an approximation of the LS principle. It is referred to in the literature as the stochastic gradient algorithm. The algorithm is similar to the FARLS algorithm and it makes the assumption that matrix $\mathbf{P}(n)$ can be replaced by a single coefficient which is the trace of the matrix $\mathbf{P}(n)$ [19]. Let

$$^{-1}(n) = Tr\left[\mathbf{P}^{-1}(n)\right] = Tr\left[\sum_{i=1}^{n} \mathbf{z}(i)\mathbf{z}^H(i)\right]. \tag{3.73}$$

From this it can be shown that the following formulation is equal to equation (3.73) [19]

$$^{-1}(n) = \sum_{i=1}^{n} \mathbf{z}^H(n)\mathbf{z}(n) = \sum_{i=1}^{n-1} \left[\mathbf{z}^H(i)\mathbf{z}(i)\right] + \mathbf{z}^H(n)\mathbf{z}(n). \tag{3.74}$$

This gives the recursion for $^{-1}(n)$, which becomes [19]

$$^{-1}(n) = \ ^{-1}(n-1) + \mathbf{z}^H(n)\mathbf{z}(n). \tag{3.75}$$

Based on the above formulation the algorithm uses the following equations for adaptation [19]

$$\begin{aligned} {}^{-1}(n) &= {}^{-1}(n-1) + \mathbf{u}^H(n)\mathbf{u}(n) && (3.76)\\ e(n) &= y(n) - \hat{\mathbf{g}}^H(n-1)\mathbf{u}(n) && (3.77)\\ \hat{\mathbf{g}}(n) &= \hat{\mathbf{g}}(n-1) + \quad (n)e^*(n), && (3.78) \end{aligned}$$

where $\mathbf{u}(n) = [z(n)\ z(n-1)\ \cdots\ z(n-N_g+1)\ z^*(n)\ z^*(n-1)\ \cdots\ z^*(n-N_g+1)]$. The algorithm is initialized with ${}^{-1}(0) = z(0)^2$ and $\hat{\mathbf{g}}(0) = \mathbf{0}$ with $2N_g$ zeros.

The pre-distortion filter coefficients $\hat{\mathbf{w}}$ could be estimated with recursion based on equations (3.76)-(3.78), but as already stated, in practice there are only N_g adaptation iterations and the ARLS algorithm will not have a sufficient number of iterations to converge. Consequently, RLS recursions should be used in estimating pre-distortion filters coefficients $\hat{\mathbf{w}}$.

Normalized Least Mean Squares Method

NLMS creates few assumptions on the LS principle to simplify recursion. As a result, its accuracy is not as good as that of the RLS algorithm. On the other hand, the computational complexity is markedly reduced. The NLMS algorithm executes equations (3.79) and (3.80) to obtain an estimate for $\hat{\mathbf{g}}$. The following formulations collect the NLMS algorithm [36, 77].

$$\begin{aligned} e(n) &= z(n) - \hat{\mathbf{g}}^H(n-1)\mathbf{u}(n) && (3.79)\\ \hat{\mathbf{g}}(n) &= \hat{\mathbf{g}}(n-1) + \frac{\mu}{\epsilon + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n). && (3.80) \end{aligned}$$

In equations (3.79) and (3.80) $\mathbf{u}(n) = [y(n)\ y(n-1)\ \cdots\ y(n-N_g+1)\ y^*(n)\ y^*(n-1)\ \cdots\ y^*(n-N_g+1)]$ and in equation (3.80) μ is the step-size parameter of the recursion and ϵ is an extremely small positive constant to overcome numerical difficulties when $\|\mathbf{u}(n)\|^2$ approaches zero. At first iteration $\hat{\mathbf{g}}(0) = \mathbf{0}$ with $2N_g$ zeros.

In a manner similar to the previous, from estimated imbalance vector $\hat{\mathbf{g}}$ we obtain $\hat{\mathbf{g}}_1 = [\hat{\mathbf{g}}(0)\ \cdots\ \hat{\mathbf{g}}(N_g-1)]^T$ and $\hat{\mathbf{g}}_2 = [\hat{\mathbf{g}}(N_g)\ \cdots\ \hat{\mathbf{g}}(2N_g-1)]^T$. Thus, the zero-padded versions $\mathbf{g}_1^0$ and $\mathbf{g}_2^0$ are $\hat{\mathbf{g}}_1^0 = [0\ 0\ \cdots\ \hat{\mathbf{g}}_1^T]^T$ and $\hat{\mathbf{g}}_2^0 = -[0\ 0\ \cdots\ \hat{\mathbf{g}}_2^T]^T$, both with $N_g - 1$ appended zeros. Finally, the estimate for the pre-distortion coefficients

$\hat{\mathbf{w}}$ is reached by the following equations

$$e(n) = \hat{\mathbf{g}}_2^0(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n) \quad (3.81)$$

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \frac{\mu}{\epsilon + \|\mathbf{u}(n)\|^2}\mathbf{u}(n)e^*(n), \quad (3.82)$$

where $\mathbf{u}(n) = [\hat{\mathbf{g}}_1^0(n)\ \hat{\mathbf{g}}_1^0(n-1)\ \cdots\ \hat{\mathbf{g}}_1^0(n-N+1)]$ and μ is the step size parameter of the recursion which is in practice different from the step-size used in equation (3.80). The small positive constant ϵ remains the same.

This section addressed different methods to solve the least-squares model fitting problem stated above. A number of algorithms based on LS and MSE approaches were considered and their recursions were given in the form of equations. All considered adaptive algorithms are able to solve the model fitting problem under ideal circumstances but their performance varies if signals are not ideal. Performance of the algorithms is compared in detail in Section 5.5.

4. DEVELOPMENT ENVIRONMENT FOR REAL-TIME IMPLEMENTATION

The development environment for real-time prototype implementation consists of a personal computer (PC) with a general purpose processor (GPP), an SDR evaluation board and an RF daughter board. In this thesis, the evaluation board is the Universal Software Radio Peripheral (USRP) or USRP2 [28]. The evaluation board together with the daughter board constitute an RF FE and digitalization unit of the transceiver [34]. Furthermore, if a computer is added to the preceding system, the components construct a fully functional transceiver unit [88, 58]. In addition, this transceiver unit is definable with computer software and capable of full-duplex operation.

The purpose in this chapter is to provide general information on the implementation environment and its capabilities. The chapter commences with an introduction to USRP and USRP2 platforms. Thereafter, different methods and software are discussed which can be used with USRP and USRP2. Particularly the GNU Radio and its graphical user interface, the GNU Radio Companion, are presented in Sections 4.2 and 4.2.3, respectively. The last section of the chapter briefly discusses different methods for the use of USRP and USRP2 in the Windows operating system.

4.1 USRP and USRP2

The USRP is an SDR platform developed by Ettus Research [28]. Nowadays, there exist two different platforms which are USRP, or USRP Original, and USRP2 [34, 28]. The original USRP has been on the market for a few years and USRP2 was launched at the beginning of 2009. Both platforms are field programmable gate array (FPGA)-based and they provide ADCs, DACs, decimating/interpolating low-pass filters, PC connectivity, multiple-input and multiple-output (MIMO) capability and advanced input/output (I/O) ports which enable automatic gain control (AGC) and received signal strength indication (RSSI) measurements [34]. In addition, both USRPs offer interfacing for different daughter boards, which will be further discussed in section 4.1.2.

Figure 4.1: Illustration of USRP1.

The original USRP has two transceiver paths, which enable it to function with two transceiver daughter boards or with two transmitter and two receiver daughter boards. The main components of the original USRP can be found in Table 4.1. All four ADCs inside the USRP are manufactured by Analog Devices and the exact model is AD9862 [34]. In fact AD9862 is capable of executing analog-to-digital and digital-to-analog conversions. As a result, there are no separate ADCs and DACs on the component level, only four ADC/DAC units. The full range of the ADCs is 2 V peak-to-peak voltage with 50 Ohm differential input [28], which yields a 16 dBm maximum input signal power level without clipping with 12-bit resolution. Likewise, DACs can provide 1 V peak-to-peak voltage, again with 50 Ohm differential load [28], which results in a maximum output power of 10 dBm with 14-bit accuracy. The programmable gain amplifier (PGA) can provide up to 17 dBm output power for the output signal. General block diagram of original USRP's FPGA board can be seen in Figure 4.2.

The only functionally significant parts of the FPGA not yet discussed are the digital down-converters (DDCs) and digital up-converters (DUCs). There is a DDC and a DUC for each signal path, four in total. Both DDCs and DUCs have two decimation and interpolation stages, respectively. The DDC consists of one cascade-integrator comb (CIC) filter stage with variable decimation from 4 to 256 and one half-band (HB) filter stage with a decimation factor 2. As a result, total decimation of the original USRP is from 8 to 512. In addition, CIC filters in all DDCs and DUCs have four cascaded stages. Similarly, DUCs consists of a CIC filter stage with variable interpolation from 2 to 256 and one HB filter stage with decimation factor 2. Consequently, total decimation of the original USRP is from 4 to 512. Figure 4.3 shows how the frequency response of the DDC behaves with different decimation factors. [34, 28, 14, 75]

USRP2 is topologically similar to the original USRP, its fundamental difference being that firmware is saved on a secure digital (SD) memory card and executed from

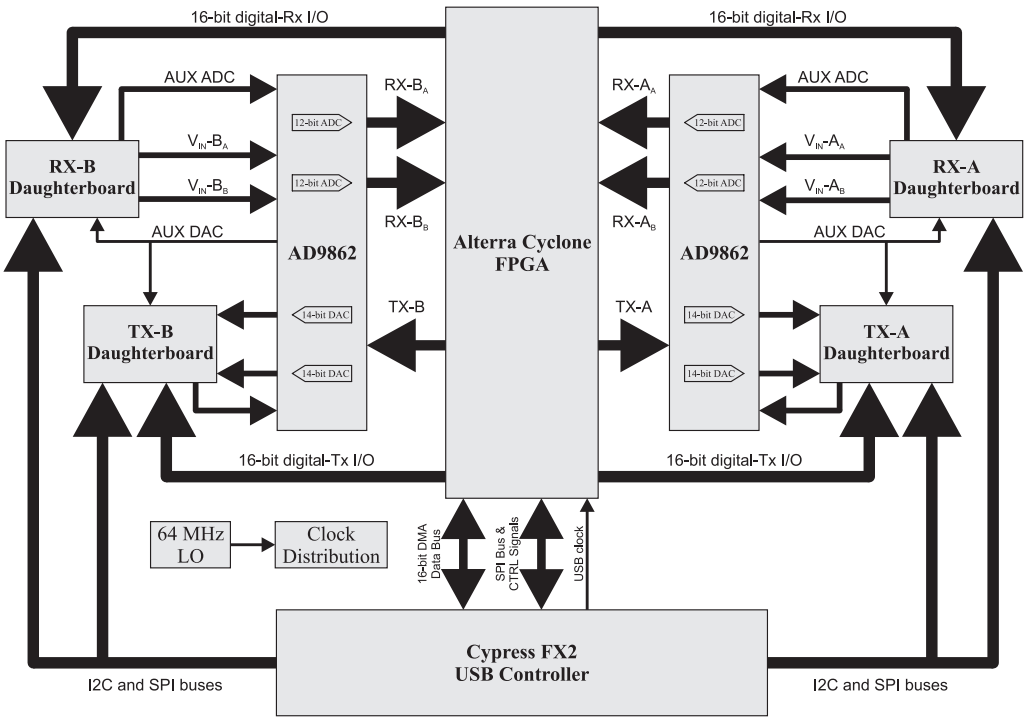
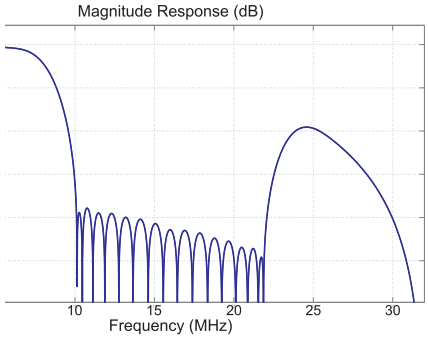


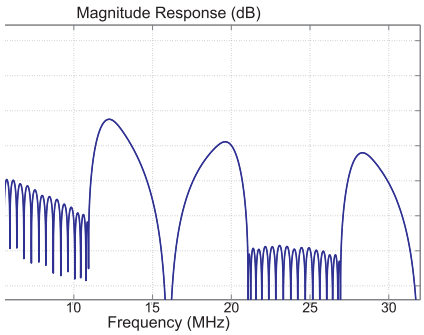
Figure 4.2: USRP1 FPGA block diagram.

there [28]. This difference gives an opportunity to write tailored FPGA implementations directly on the platform instead of writing signal processing blocks in GNU radio software. Moreover, this enables a stand-alone operation where no computer is needed. In general, USRP2 has better specifications throughout. The main components of the USRP2 are listed in Table 4.1 [28]. As will be noted from Table 4.1, the greatest structural difference between USRP and USRP2 is the number of transceiver paths, which also reduces the possible number of sub-channels in the transmitter or receiver on the side of the USRP2. On the other hand, this is the only drawback the new USRP2 has compared to the original USRP. USRP2 also has a serial expansion port for connecting multiple USRP2s together for extended MIMO capability [28].

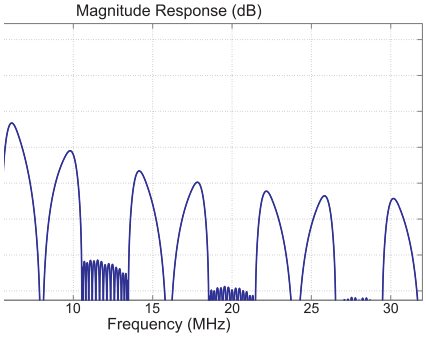
The useful bandwidth of both USRPs is in practice limited by the PC connectivity method. As discussed above, the original USRP has connectivity to the PC through the USB 2.0 interface, which has a theoretical transmission rate of 480 Mbps, though in practice the useful rate is around 320 Mbps. Although a 64 MHz sampling frequency enables the receiver, with reference to Nyquist criteria, to receive signal bandwidths as high as 32 MHz, this is in practice impossible. The useful transmission rate of the USB 2.0 interface becomes a limiting factor [58, 34]. Every complex sample is represented with two 16-bit signed integers, which yields in total 32 bits for every complex sample. If the maximum useful data rate of the USB 2.0 interface



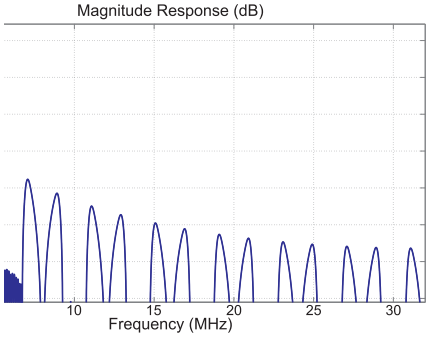
(a) Decimation factor 8



(b) Decimation factor 16



(c) Decimation factor 32



(d) Decimation factor 64

Figure 4.3: USRP DDC Responses with different total (CIC+HB) decimation factors.

is divided by the size of one complex sample we get

$$\frac{R_b}{N_{bits}} = \frac{320Mbps}{32bits} = 10MHz. \tag{4.1}$$

Additionally, the sampling frequency of original USRP must be an integer multiple of four. As a result, the smallest decimation factor is 8, which yields a maximum sampling frequency, using complex samples, of $64MHz/8 = 8MHz$ [34]. However, if a simultaneous transmission path is used, the derived maximum frequency span is divided between receiver and transmitter paths. Consequently, the maximum band width for both transmitter and receiver is narrowed to 4 MHz [34]. Similarly, the maximum bandwidth of the new USRP2 is 25 MHz and in transceiver state the maximum bandwidth is narrowed down to 12.5MHz. The only difference in calculation is that the theoretical transmission rate of the Gigabit Ethernet connection is 1000 Mbps [28].

4.1.1 VRT-49

There has been an enormous change in high-performance radio and signal processing architectures hitherto dominated by customized architectures which are mutually

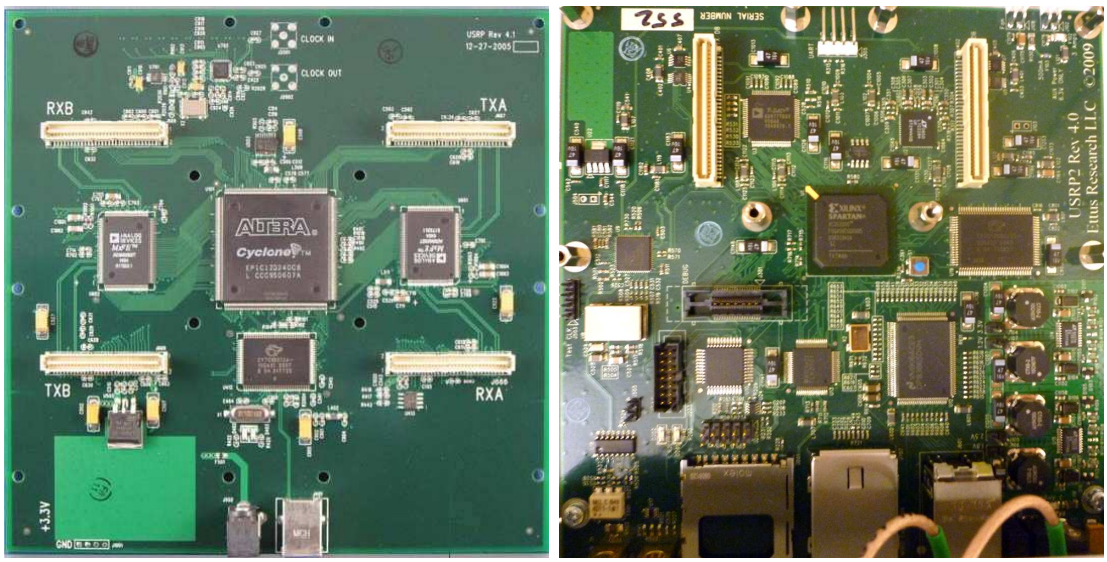
	USRP	**USRP2**
FPGA	Altera Cyclone EP1C12	Xilink Spartan 3-2000
ADCs	4 x 12-bit, $f_s = 64MHz$	2 x 14-bit, $f_s = 100MHz$
DACs	4 x 14-bit, $f_s = 128MHz$	2 x 16-bit, $f_s = 400MHz$
DDCs	4 x programmable DDCs	2 x programmable DDCs
DUCs	4 x programmable DUCs	2 x programmable DUCs
Memory	None	1 MByte SRAM
Connectivity	Cypress FX2 Universal Serial Bus (USB) 2.0 Interface (480 Mbps)	Gigabit Ethernet Interface (1000 Mbps) 2 Gbps serial expansion port Auxiliary reference clock 1 PPS connection for Global Navigation Satellite System (GNSS) purposes

Table 4.1: Main components of the original USRP and USRP2.

Figure 4.4: Picture of USRP2.

incompatible. The VITA radio transport (VRT) protocol V49.x is an American National Standards Institute (ANSI) norm which standardizes the framework for SDR hardware platforms. The objective in the process is to unify the hardware developed by different manufacturers. Further, the ultimate goal is that once written transceiver software can be executed on any SDR platform which is developed according to the VRT V49.x standard. Currently, the standard is in version 49.1. Unlike other emerging SDR standards, the VRT V49.x does not specify the architecture of the transceiver unit. [86]

The VRT V49.x protocol addresses upcoming requirements, a definition of a transport packet which includes unique signal data and signal context information. The signal data packet provides a variety of data formats and it conveys the desired data.



(a) USRP (b) USRP2

Figure 4.5: FPGA Motherboards of USRPs.

Further, the formats can be 1-32 bit real or complex either integer or floating point numbers and the signal data packet includes sample-wise time-stamp information. The context packets convey other important information such as frequency, bandwidth, gain, delay and inertial navigation parameters. Additionally, the packets include geo-location information on the transceiver. Context packets are linked to the signal data packets with time-stamp information. New stream identifiers give the possibility for multiple labeled data streams. [67]

4.1.2 Daughter Boards

Daughter boards provide the RF FE for both USRPs. Ettus Research manufactures a number of different daughter boards for different purposes [28, 34]. In addition, there are daughter boards which have only the capability to receive or transmit, but there are also transceiver daughter boards. Further, some of these latter are capable of full-duplex transmission, while some are only capable of half-duplex transmission. Receiver daughter boards perform bandpass filtering, amplifying with low noise amplifier (LNA), down-conversion from RF to IF and lowpass filtering. Likewise, transmitting daughter boards up-convert the desired signal from digital IF to RF and perform the required filtering and power amplification. The type of the up- and down-conversion depends on the selected daughter board, e.g. in RFX2400 they can be performed with either real or I/Q mixing. Transceiver daughter boards have, of course, the ability to perform both up- and down-conversion. All daughter boards and their key features can be found in Table 4.2.

There is also third-party daughter board for USRPs called Bitshark USRP RX (BURX). It is manufactured by Epiq Solutions. BURX employs direct-conversion architecture with RF tuner covering frequencies from 300 MHz to 4 GHz. In addition, it has configurable channel filter supporting RF bandwidths up to 50 MHz, integrated high-stability 26 MHz temperature compensated crystal oscillator (TCXO) and built-in MIMO capability. [27]

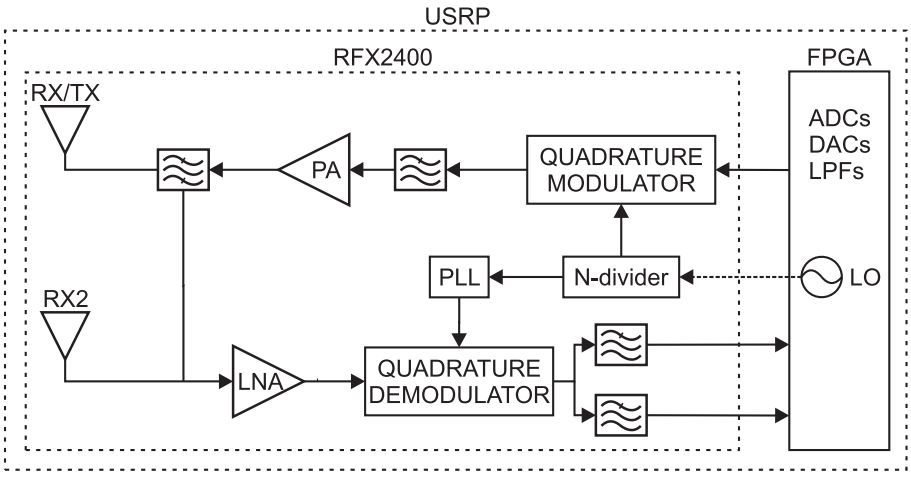


Figure 4.6: Block diagram of RFX2400 daughter board. Figure includes both receiver and transmitter paths.

Within the scope of this thesis the RFX2400 daughter board is used. This board has two RF antenna connections, another for half-duplex transmission and reception called RX/TX and another for receiving called RX2 [28]. If the RX/TX connection is used for transmitting and RX2 for receiving, the daughter board is capable of full-duplex data transmission. A block diagram of the RFX2400 board can be seen in Figure 4.6 [28, 34].

4.2 GNU Radio

GNU Radio is an open-source software running on the Linux operating system which can be used with multiple hardware platforms from a sound card to sophisticated SDR platforms such as USRP or USRP2. Although GNU Radio supports multiple platforms, it is mainly intended for use with the USRP product family. [58, 34]

In theory, GNU Radio can run on all Linux distributions, but there are a few distributions which are especially recommended, namely Arch, Debian, Fedora, Gentoo, Mandriva, SuSe and Ubuntu [34]. GNU Radio has also been built and installed on Mac OS X. Similarly, there have been attempts to install GNU Radio on Microsoft Windows, but there has been no reported success.

	Frequency range	Rx	Tx	Maximum Tx power
Basic Rx	1-250 MHz	Yes	No	N/A
Basic Tx	1-250 MHz	No	Yes	-
LFRX	0-30 MHz	Yes	No	N/A
LFTX	0-30 MHz	No	Yes	-
DBSRX	800-2400 MHz	Yes	No	N/A
TVRX	50-850 MHz	Yes	No	N/A
WBX0510	50-1000 MHz	Yes	Yes	20 dBm
RFX400 (Flex 400)	400-500 MHz	Yes	Yes	20 dBm
RFX900 (Flex 900)	800-1000 MHz	Yes	Yes	23 dBm
RFX1200 (Flex 1200)	1150-1450 MHz	Yes	Yes	23 dBm
RFX1800 (Flex 1800)	1500-2100 MHz	Yes	Yes	20 dBm
RFX2400 (Flex 2400)	2300-2900 MHz	Yes	Yes	17 dBm
XCVR2450	2400-2500 MHz, 4900-5900 MHz	Yes	Yes	20 dBm
WBX0510	50-2200 MHz	Yes	Yes	20 dBm

Table 4.2: Different daughter boards manufactured by Ettus Research and their main features.

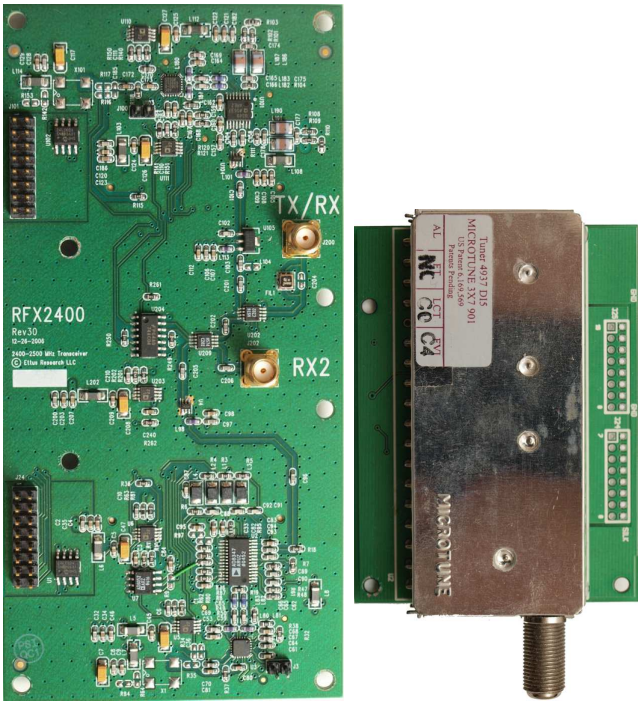
GNU Radio includes a wide variety of signal processing blocks, a core library for running signal processing blocks and a graphical user interface called the GNU Radio Companion. The GNU Radio Companion is further discussed in Section 4.2.3.

4.2.1 Python Flow Graphs

Python flow graphs act as a framework for GNU Radio applications. They consist of signal sources, signal processing blocks and signal sinks. As the name implies, a signal goes through different blocks as it were flowing. Nonetheless, signals in GNU Radio are discrete time signals, processed in block-wise manner. Each signal processing block is scheduled to process a certain number of samples and each block reports to the GNU Radio core scheduler the number of samples processed and the number produced to the output of the signal processing block. Signal sources only produce a signal, like the USRP and USRP2 source blocks which produce signals from USRP and USRP2. Similarly, signal sinks have only inputs. Examples of this are USRP and USRP2 sinks, which represent the transmission path of USRPs. [34, 58, 88]

4.2.2 Signal Processing Blocks

GNU Radio includes a wide variety of built-in signal processing blocks [34, 88], which usually work as intended. Due to the fact that GNU Radio comprises open-source software and there are a large number of separate developers, some signal processing blocks are realized better than others. Some blocks do not even work satisfactorily.



(a) RFX2400 (b) TVRX

Figure 4.7: Example pictures of daughter boards.

All signal processing blocks can be divided into two larger groups according to the programming language they are coded with [14], namely, C++ and Python signal processing blocks. The C++ programming language is considerably more efficient than the Python and it has much better ability to execute computationally complex calculations [14]. As a result, blocks with elaborate functionality are programmed with C++ language.

Signal processing blocks based on the Python programming language are simply Python flow graphs which do not have their own constructor. In addition, they always consist of other signal processing blocks.

4.2.3 GNU Radio Companion

The GNU Radio Companion is a graphical user interface for designing GNU Radio-based Python flow graphs. It enables the user to develop complicated flow graphs without an extensive knowledge of the Python programming language. The logic of user interface functionality is similar to the Matlab Simulink. The user drags-and-drops signal processing blocks to the working area, connects multiple blocks together and defines all the properties each signal processing block processes. A screen shot of the GNU Radio Companion is shown in Figure 4.8. [23]

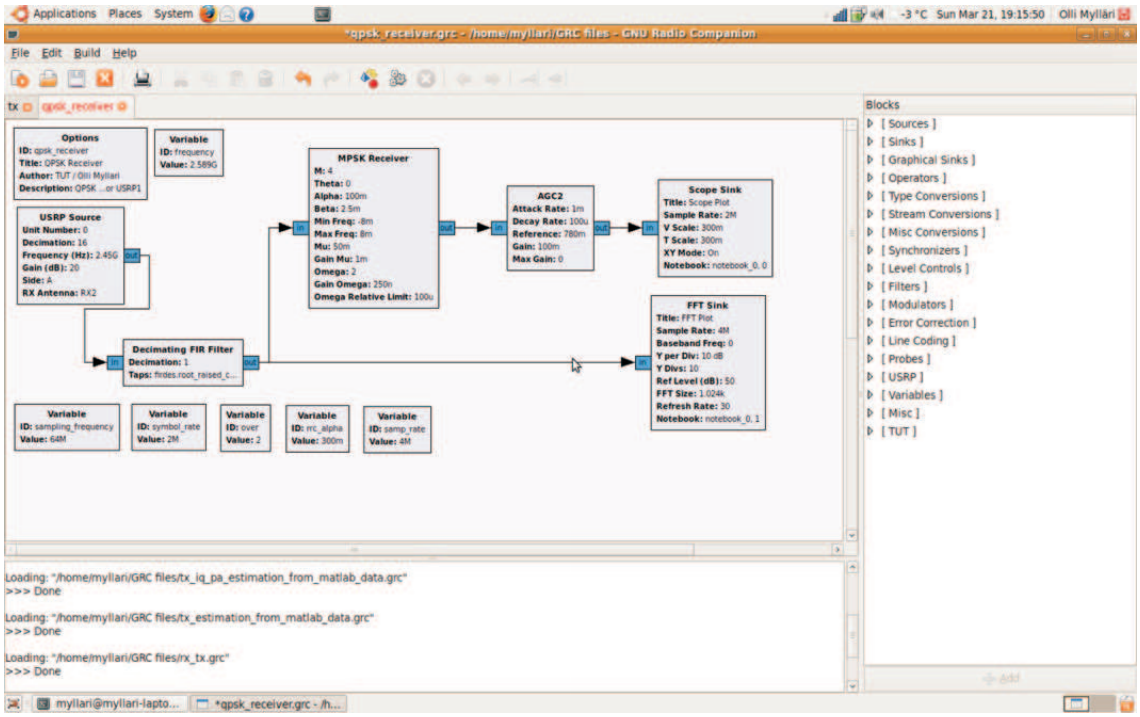


Figure 4.8: Screen shot of the GNU Radio Companion. The flow graph shown in the figure is a QPSK receiver.

4.3 Other Software for Use With USRPs

As already mentioned, the GNU Radio is not the only software to use with USRP and USRP2. There are multiple drivers and interface softwares for Linux, Windows and Mac OS X environments. The motivation is quite obvious, since most of the algorithm and simulation engineering work in the field of telecommunications is done with Matlab. As a result, there is a need to be able to use both USRPs in a Windows environment and in cooperation with Matlab. The goal in the following sections is to provide the reader with an understanding of the different means of controlling USRP and USRP2.

4.3.1 Windows Drivers

There are open-source windows drivers for the original USRP. These give a C++ interface to deliver control commands to USRP and daughter boards. Although the driver gives the opportunity to control USRP, it still has a limited control command set which restricts the use of the USRP. The driver also needs a Linux USB library for Windows called LibUSB32. Efforts are under way to develop a Windows driver for USRP2.

There is also on-going development towards universal hardware driver (UHD) drivers

by Ettus Research. UHD drivers should become compatible with all major operating systems and it should give support for both USRPs. [28]

4.3.2 Matlab and Simulink Interfacing

The Communications Engineering Lab at the Karlsruhe Institute of Technology has developed a Matlab Simulink interface called "Simulink-USRP: Universal Software Radio Peripheral Blockset". It includes normal Matlab Simulink blocks to control the original USRP. This approach should also be appropriate for real-time simulations. Another approach is a Matlab interface developed by the SDR4All group. This approach is used from the normal Matlab command line and in practice it captures a given time sequence of the received signal in a variable. Consequently, this strategy is not suitable for real-time applications but becomes useful in algorithm testing and simulation.

Currently, then, as there are no separate Windows drivers for USRP2, both approaches support only the original USRP, though, work is ongoing also to support USRP2.

5. IMPLEMENTATION DETAILS AND PRACTICAL ASPECTS

The implementation here was built on the USRP software radio platform and the RFX2400 daughter board was used as a RF FE. As mentioned earlier in Chapter 4, the RFX2400 has two different antenna connections, one for time division duplex (TDD)-based receiving/transmitting (RX/TX), called RX/TX, the other only for receiving, called RX2. If these connections are used in parallel they provide for full-duplex operation. In consequence, the RX/TX antenna connection of the RFX2400 was used as an RF output and the RX2 antenna connection as a feedback loop RF FE. Physical connections of the implementation can be seen more clearly in the block diagram in Figure 5.2.

The software part of the implementation was built on GNU Radio software by writing C++ signal processing blocks for direct current (DC) offset compensation, feedback loop signal synchronization, I/Q imbalance estimation and pre-distortion. Additionally, interfaces of its own were written for USRP to ensure better controllability of digital IF and to enable real down-conversion in the feedback loop receiver chain. Also GNU Radio Companion graphical user interface (GUI) blocks were written for all programmed signal processing blocks to obtain easier usability of the overall transmitter. GNU Radio Companion flow graph of the transmitter can be seen in Figure 5.1.

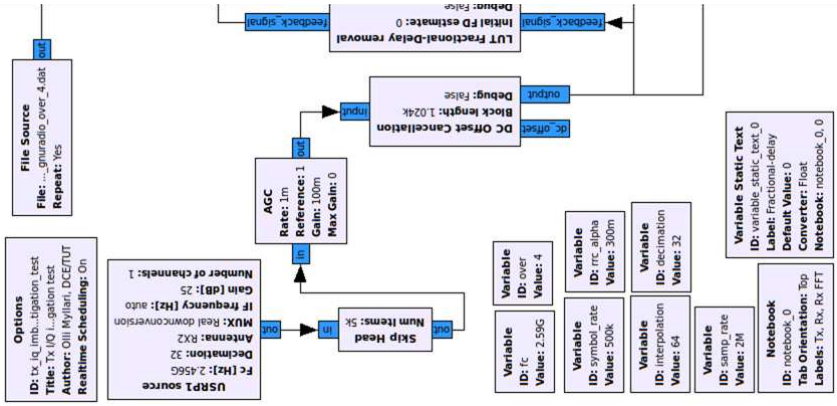


Figure 5.1: GNU Radio Companion flow graph realizing the transmitter with I/Q imbalance mitigation.

I/Q imbalance estimation and calibration algorithms have been discussed earlier in Section 3.3. The rest of this chapter will discuss in detail which supportive algorithms were used in the implementation. Here a supportive algorithm means and algorithm which does not make the estimation itself but has an important role in the overall process. First the basic algorithm for DC offset removal is addressed. Thereafter, two different algorithms for integer delay estimation are discussed. Similarly, two different iterative fractional-delay estimation algorithms are given. Then, the fractional-delay compensation structure used in the implementation is depicted and at the end of the chapter implementation related practical aspects are discussed.

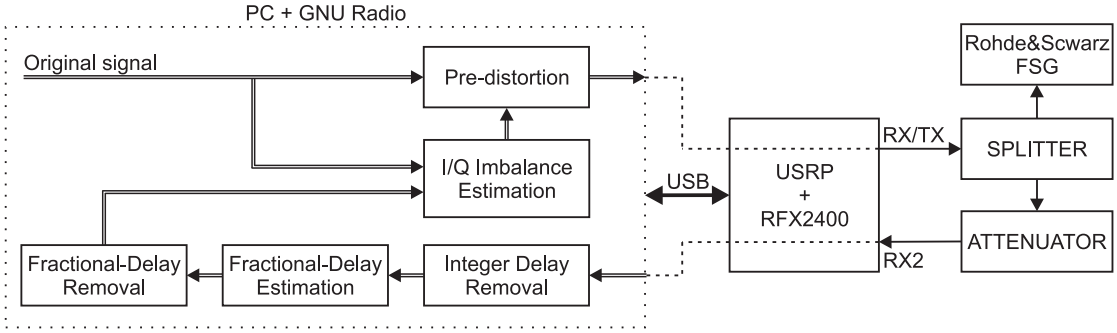


Figure 5.2: Block diagram of the measurement setup showing connections and signal flow.

In the implementation, the accuracy of transmitter and feedback loop receiver oscillator signals was very good (in the order of 100 Hz), and thus no excess carrier frequency offset (CFO) mitigation was here implemented. A block diagram of the main DSP functionalities of the implementation can be seen in Figure 5.2.

5.1 DC Offset Removal

DC offset has to be removed from the feedback loop signal before I/Q imbalance parameter estimation. If the DC offset is not properly compensated, the possibly strong DC component in the feedback loop signal may bias the estimation process or impair the performance of the algorithm.

In practice, DC offset means that the mean values of real and imaginary parts of the feedback loop signal are non-zero. This can be compensated by taking the mean value of the feedback loop signal in the following manner

$$y_{mean}(k) = \frac{1}{M} \sum_{i=1}^{M} y_k(i), \tag{5.1}$$

where $k = (1, 2, ..., r)$ and M the length of the signal block used for DC offset estimation. Moreover, by reason of the block-wise signal processing of GNU Radio, the DC offset estimates are averaged over all processed blocks in the following way.

$$DC_{est} = \frac{1}{r} \sum_{i=1}^{r} y_{mean}(i), \tag{5.2}$$

where r is the number of the current signal data block. Finally, the calculated mean values are subtracted from the real and imaginary parts of the feedback loop signal and the compensated feedback loop signal becomes then

$$\mathbf{y}_{comp} = \mathbf{y} - DC_{est}. \tag{5.3}$$

5.2 Integer Delay Estimation

Integer delay estimation is needed for estimating the integer number delay between the original signal and the feedback loop signal. In general, integer delay estimation and compensation is fairly trivial and easy to implement. In this work, two different algorithms were implemented and tested. The first of them was built on the fast Fourier transform (FFT)-based correlator and the other was based on time domain differential correlator.

Integer delay is compensated simply by changing the vector indexing in the digital sampled feedback loop signal.

5.2.1 Fourier Transform Fitting Approach

The approach addressed in this section retells the approach in [66]. Only difference is that in the implementation integer delay is estimated for shorter data blocks and the result is averaged over the time.

The delay between the original signal and the feedback loop signal appears as a linear phase shift in the frequency domain. Delay can thus be estimated by taking the FT of the original and the feedback loop signal. Then, integer delay between two signal can be estimated by multiplying the FTs, taking IFT of the product and finding maximum value of the IFT. Thereafter, to find fractional-delay the phase difference between the two signals over frequency is calculated, and the fractional-delay can be obtained from the slope of the phase. The major advantage of this approach is that the delay can be resolved with subsample accuracy without need for excessive amounts of oversampling.

If the original signal is denoted by $z(n)$ and the feedback loop signal by $y(n)$, the integer delay is simply estimated by multiplying the FTs of the original and the feedback loop signal as follows:

$$C(f) = Y(f)Z^*(f). \tag{5.4}$$

The time domain cross-correlation between the two signals can be determined with an IFT as follows.

$$c(n) = \|\mathcal{F}^{-1}\{C(f)\}\| = \| \int_{-\infty}^{\infty} C(f)e^{j2\pi fn}\, df \|. \tag{5.5}$$

The integer delay estimate $\hat{p}$ can then be defined simply by finding the maximum value of $c(n)$

$$\hat{p} = \arg\max_n c(n), n = 1..M. \tag{5.6}$$

After removal of the integer delay, the phase difference between the two signals is

$$\Delta\boldsymbol{\theta} = \arg(Y(f)Z^*(f)) \tag{5.7}$$

and weight vector as

$$\boldsymbol{\theta}_w = -2\pi\|C(f)\|. \tag{5.8}$$

The slope for fractional-delay can be determined by using a LS model-fitting to the phase of a unity delay as follows:

$$\hat{\tau} = (\boldsymbol{\theta}_w^H\boldsymbol{\theta}_w)^{-1}\boldsymbol{\theta}_w\Delta\boldsymbol{\theta}^H. \tag{5.9}$$

5.2.2 Iterative Digital Differential Approach

This algorithm was originally designed for use in FPGA-based systems, which makes it suitable for real-time prototype implementation. The integer delay between the original signal $z(n)$ and the feedback loop signal $y(n)$ is computed from an estimate of the cross-correlation between the two signals, as can be seen in equation (5.10) [82].

$$\hat{\mathbf{R}}_{zx}(m) = \begin{cases} \frac{1}{L}\sum_{n=0}^{L-m-1} z(n+m)y^*(n) & , m \geq 0 \\ \frac{1}{L}\sum_{n=0}^{L-m-1} z^*(n-m)y(n) & , m < 0 \end{cases}, \tag{5.10}$$

where $-L+1 \leq m \leq L+1$ and L is the number of samples used to calculate the cross-correlation. In the following calculations the absolute value of the cross-correlation is used, yielding

$$c(m) = \|\hat{\mathbf{R}}_{zx}(m)\|. \tag{5.11}$$

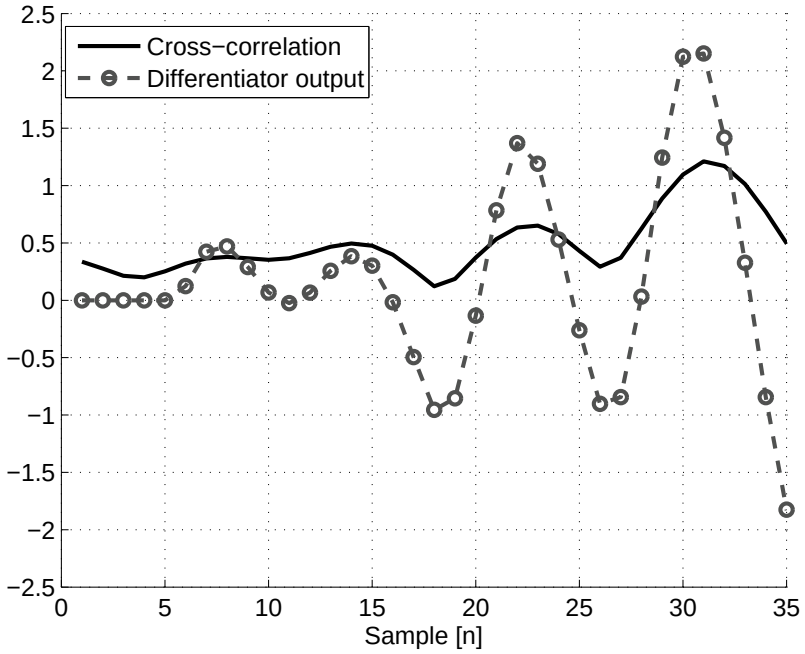


Figure 5.3: Example behavior of the iterative digital differential approach. Integer delay has been 31 samples, the signal a low-IF signal with 16-QAM modulation, 8 times oversampling, 30 % roll-off, and the SNR of the signal 20 dB. Frequency-selective I/Q imbalance model length of 3 with 4 per cent gain imbalance and 4 degree phase imbalance was used.

The second part of the iterative integer delay estimation process is a peak detector which continuously seeks the index of the peak value [82]. This peak detector is, in practice, implemented with a simple digital differentiator. The differentiator output is formulated as follows [82]:

$$d(m) = c(m+1) - c(m-1) + 2\left[c(m+2) - c(m-2)\right]. \tag{5.12}$$

The algorithm continuously observes the cross-correlation value and digital differentiator output. If cross-correlation $c(m_0) \geq T_h$, $d(m_0 - 1) > 0$, and $d(m_0) \leq 0$ the instant of the peak value is found [82]. When the peak value has been found the integer delay estimate $\hat{p}$ can be defined with the following equation [82]

$$\hat{p} = \begin{cases} m_0 - 1 & , if c(m_0) > c(m_0 - 1) \\ m_0 & , otherwise \end{cases} \tag{5.13}$$

An example case of the algorithm output can be seen in Figure 5.3 where the true integer delay has been 31 samples. The figure shows cross-correlation between the original signal and the feedback loop signal, as well as the corresponding digital differentiator output.

5.3 Fractional-Delay Estimation

Fractional-delay estimation is a crucial part of real-time prototype implementation in that the implemented algorithm has proved to be extremely sensitive to fractional-delay in the model-fitting-based parameter estimation stage. Also for fractional-delay estimation two totally different algorithms were implemented. Fractional-delay compensation schemes are further discussed in Section 5.4. Both of the algorithms perform reasonably well with signals distorted by I/Q imbalance. Interestingly, however, the maximum-likelihood non-data-aided (NDA) algorithm was shown to give the least biased result under I/Q imbalance.

5.3.1 Maximum-Likelihood Non-Data-Aided Approach

This maximum-likelihood NDA fractional-delay timing synchronization algorithm may be found in [60, 64]. Additionally, the performance of the maximum-likelihood NDA synchronization algorithm was improved with an interpolating polynomial approach.

As earlier, $z(n)$ and $y(n)$ denote original signal and observed feedback loop signal, respectively. Assume that any CFO between the two signals is set to zero. With this assumption, the log-likelihood function of the fractional-delay estimator takes the form [60]

$$\wedge\left(\mathbf{y}|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\tau}},\tilde{\mathbf{c}}\right)=exp\left\{\frac{1}{N_0}\Re\left(\int_0^{T_0}y(t)z^*(t)dt\right)-\frac{1}{2N_0}\int_0^{T_0}\|z(t)\|^2dt\right\}, \tag{5.14}$$

where $\tilde{\boldsymbol{\theta}}$ is likelihood parameter vector for phase, $\tilde{\boldsymbol{\tau}}$ for fractional-delay, and $\tilde{\mathbf{c}}$ for known complex symbols. The aim of the algorithm is to find the maximum value of $\wedge\left(\mathbf{y}|\tilde{\boldsymbol{\tau}}\right)$, which is the average of $\wedge\left(\mathbf{y}|\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\tau}},\tilde{\mathbf{c}}\right)$ with respect to $\tilde{\boldsymbol{\theta}}$ and $\tilde{\mathbf{c}}$. In order to find the maximum value of $\wedge\left(\mathbf{y}|\tilde{\boldsymbol{\tau}}\right)$ the following approximation is made [60]

$$\int_0^{T_0}y(t)z^*(t)dt\approx e^{-j\tilde{\theta}}\sum_{i=0}^{L_0-1}\tilde{c}_i^*y(iT+\tilde{\tau}) \tag{5.15}$$

and, additionally, preforming expectetion of (5.15) with $\tilde{\theta}$ uniformly distributed over the frequency band $[0,2\pi)$ yields the approximated log-likelihood function [60]

$$\wedge\left(\mathbf{y}|\tilde{\boldsymbol{\tau}}\right)\approx\sum_{i=0}^{L_0-1}\|y(iT+\tilde{\tau})\|^2. \tag{5.16}$$

In practice, the equation (5.16) is realized by calculating values for the approximated log-likelihood over different over-sampling lags which yields

$$\wedge(\mathbf{y}|i) = \sum_{n=0}^{M-1} \|y(nOSF + i)\|^2, i = 0, 1, .., OSF - 1, \tag{5.17}$$

where OSF denotes the over-sampling factor and M stands for data block length. An example of the log-likelihood function can be seen in Figure 5.4. From equation (5.17) the maximum value of the approximate log-likelihood function may be sought by

$$\gamma = \arg\max_i \wedge(\mathbf{y}|i)\,. \tag{5.18}$$

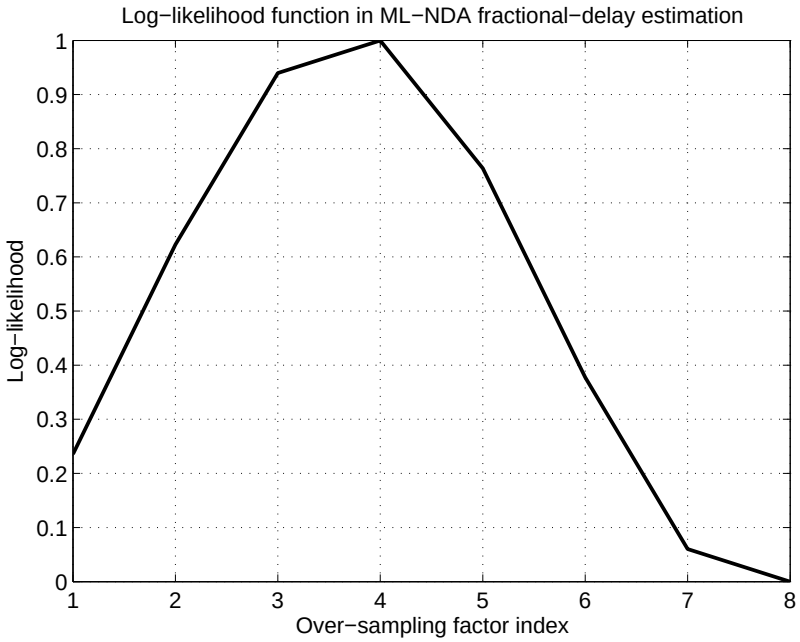


Figure 5.4: Example Log-Likelihood function of the maximum-likelihood NDA algorithm. Actual fractional-delay has been 32.1 % of the sample interval, the signal a low-IF signal with 16-QAM modulation, 8 times oversampling, 30 % roll-off, and the SNR of the signal 20 dB. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees.

The accuracy of the algorithm is improved by simple polynomial interpolation. For this purpose, three log-likelihood function values around the maximum value are chosen for further calculations in the following manner

$$\hat{p} = \begin{cases} S = \{OSF - 1, 1, 2\} & , \gamma = 0 \\ S = \{OSF - 2, OSF - 1, 1\} & , \gamma = OSF - 1 \\ S = \{\gamma - 1, \gamma, \gamma + 1\} & , otherwise \end{cases} \tag{5.19}$$

and the corresponding reorganized log-likelihood function is then

$$\mathbf{b} = \wedge\left(\mathbf{y}|i\right), i \in S. \tag{5.20}$$

Finally, from equations (5.19) and (5.20) the fractional-delay estimate $\hat{\tau}$ is derived with third order polynomial interpolation as follows

$$\hat{\tau} = \frac{3b(0) - 4b(1) + b(2)}{2\left(b(0) - 2b(1) + b(2)\right)}. \tag{5.21}$$

5.3.2 Recursive Maximum-Likelihood Data-Aided Approach

In this section, the maximum-likelihood data-aided (DA) fractional-delay timing synchronization algorithm as found in [54] is addressed. The algorithm was originally designed for use in FPGA-based systems, which makes it particularly suitable for real-time prototype implementation, even though it rests on computer-based DSP with GPPs.

The fractional-delay estimation involves four nearest neighboring samples with the following interpolation functions [54]

$$z_{I,in}(n) = \begin{cases} \sum_{i=-2}^{1} a_i(\tau) z_I(n+i) & ,\tau > 0 \\ \sum_{i=-2}^{1} a_i(1+\tau) z_I(n+1+i) & ,\tau < 0 \end{cases}, \tag{5.22}$$

where $z_I(n)$ represent either the real part of $z(n)$ and $z_{I,in}(n)$ denotes an interpolated version of $z_I(n)$. Interpolation weight vector is defined as [54]

$$\begin{aligned} a_1(\tau) &= \alpha\tau^2 - \alpha\tau \\ a_0(\tau) &= -\alpha\tau^2 + (\alpha - 1)\tau + 1 \\ a_{-1}(\tau) &= -\alpha\tau^2 + (\alpha + 1)\tau \\ a_{-2}(\tau) &= \alpha\tau^2 - \alpha\tau, \end{aligned}$$

where α is a parameter which controls the weight of the neighboring samples. It has to satisfy the requirement $0 < \alpha \leq 1$.

With the same interpolation functions, the real part of the delayed feedback signal can be expressed as [54]

$$y_I(n) = \sum_{i=-2}^{1} a_i(\tau) z_I(n+i). \tag{5.23}$$

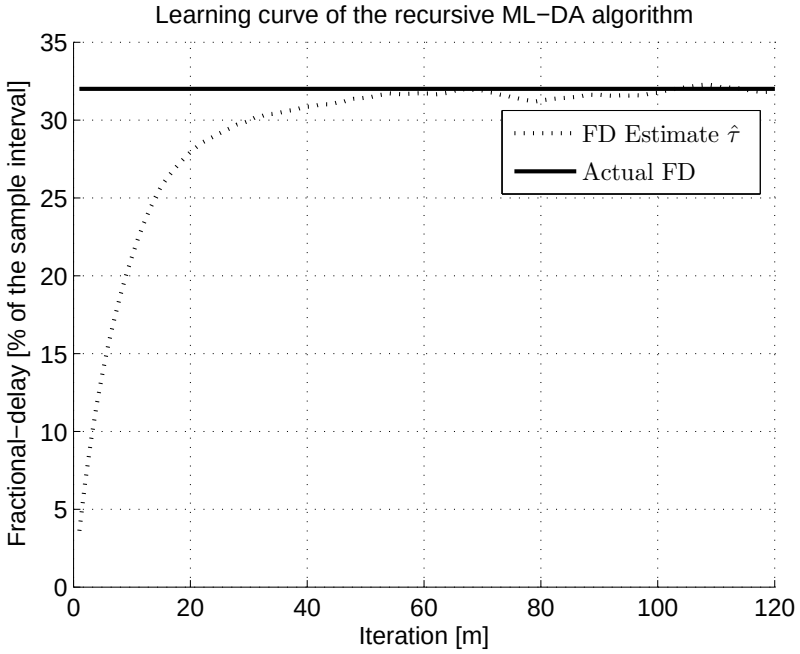


Figure 5.5: Example behavior of the recursive maximum-likelihood DA approach. Actual fractional-delay has been 32.1 % of the sample interval, signal a low-IF signal with 16-QAM modulation, 8 times oversampling, 30 % roll-off, and the SNR of the signal 20 dB. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees.

The error function is then defined to be

$$e(n) = z_I(n) - y_I(n) \tag{5.24}$$

and the fractional-delay estimate can be derived from [54]

$$\hat{\tau}_{m+1} = \hat{\tau}_m + \begin{cases} \frac{\zeta}{L} \sum_{j=0}^{L-1} sign\left[z_{I,in}(mL+j) - z_{I,in}(mL+j-1)\right] e(mL+j) & , \hat{\tau}_m > 0 \\ -\frac{\zeta}{L} \sum_{j=0}^{L-1} sign\left[z_{I,in}(mL+j) - z_{I,in}(mL+j+1)\right] e(mL+j) & , \hat{\tau}_m < 0 \end{cases} \tag{5.25}$$

In equation (5.25), L stands for the block length and ζ denotes the step size, and ζ must satisfy the requirement $0 < \zeta < 2$. An example plot of the adaptation can be seen in Figure 5.5, where $\alpha = 0.15$, $\zeta = 0.8$, and block length $L = 64$ samples.

5.4 Fractional-Delay Compensation

Fractional-delay was compensated with look-up table (LUT)-based fractional-delay filtering. The LUT includes 100 all-pass fractional-delay filters generated off-line with Matlab, thus giving an accuracy of 1 % of the sample interval for the compensation. In Figure 5.6 group delay responses of the 100 fractional-delay filters can be seen.

Fractional-delay filters are based on the use of all-pass type finite impulse response (FIR) filters with different group delays. Moreover, the group delay of the FIR filter can be a fraction of the sample interval. The design process in the fractional-delay FIR filters must be considered to lie outside the scope of this thesis, for which reason design process and requirements will not be further discussed. For more information on fractional-delay filters, see e.g [50].

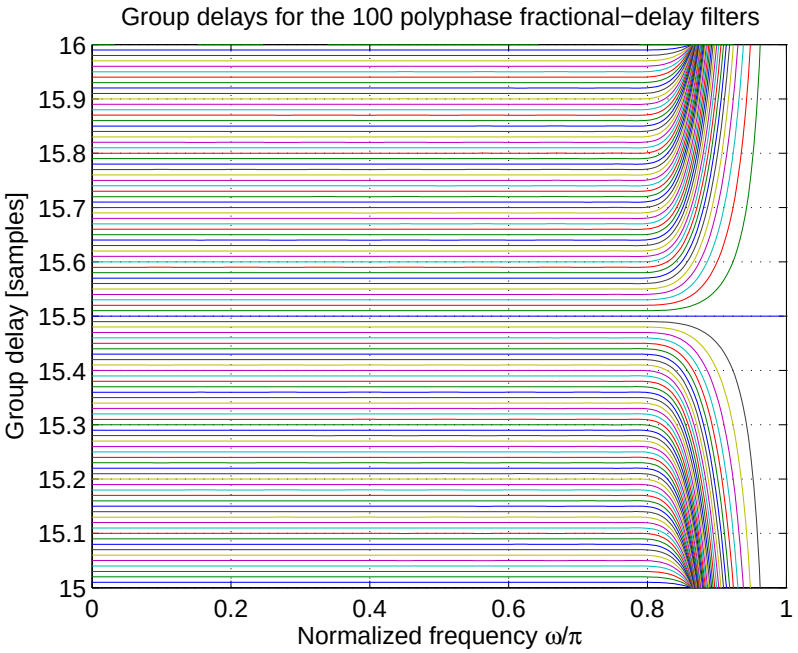


Figure 5.6: Group-delay responses of the 100 fractional-delay filters used in look-up table implementation. Order of the filters 32 and normalized cut-off frequency 0.8.

5.5 Implementation-Related Practical Aspects

There are a number of implementation-related issues which have to be taken into account to attain maximum performance for the WL-LS algorithm, namely CFO, delay between feedback loop signal and original signal, feedback loop SNR and the computational complexity of different adaptive algorithms. An IRR over 55 dB is considered sufficient performance for future wideband transmitters. In this section, other adaptive algorithms found in the literature are considered aside from the RLS algorithm, as this algorithm is computationally rather complex and the performance of the other algorithms has been compared to it.

5.5.1 Carrier Frequency Offset

First of all, the CFO of the feedback loop signal has to be within certain limits. This requirement should be easy to fulfill even with free-running LO found in USRP, since

the frequency source is usually common for up- and down-conversion. For additional information on frequency synchronization see e.g. [60, 64]. The basic impact of the way the feedback loop CFO affects the average (integrated) IRR on the whole transmission band can be seen in Figure 5.7. The figure shows that a CFO lower than 220Hz is sufficient for the desired performance. It is also worth noting that the GN algorithm outperforms other adaptive algorithms when considering CFO between the original and the feedback loop signal.

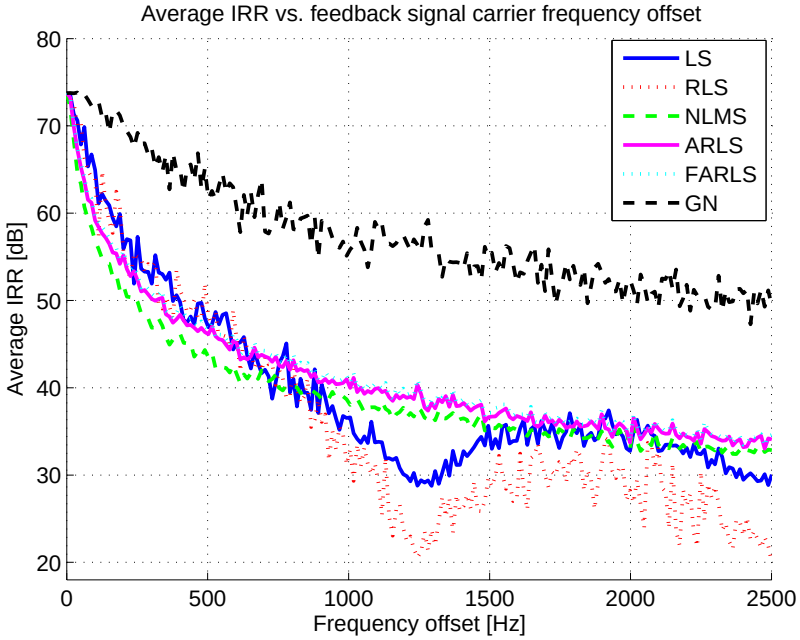


Figure 5.7: Average IRR as a function of carrier frequency offset. Low-IF signal with 16-QAM modulation, 35 % roll-off, and 5 times over-sampling. Symbol rate 1 MHz, sampling frequency 5 MHz, and IF has been 1.25 MHz. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees. Pre-distorter length of 3 and 25,000 samples used for estimation process.

5.5.2 Feedback Loop Signal Delay

Another more paramount requirement is delay estimation and compensation between the original digital transmitted signal and the digital feedback loop signal. Integer delay is easily estimated and compensated, whereas estimation and compensation of fractional-delay has been found to be more demanding. Moreover, I/Q imbalance tends to bias fractional-delay estimates, which renders compensation even more demanding. The effect of fractional-delay between original signal and feedback loop signal on the IRR is illustrated in Figure 5.8. It can be seen that the fractional-delay has a significant effect on the performance of the algorithm and

that fractional-delays less than 2 % give adequate performance. In addition, around 1 % accuracy should be achievable with the algorihtms discussed in Section 5.3.

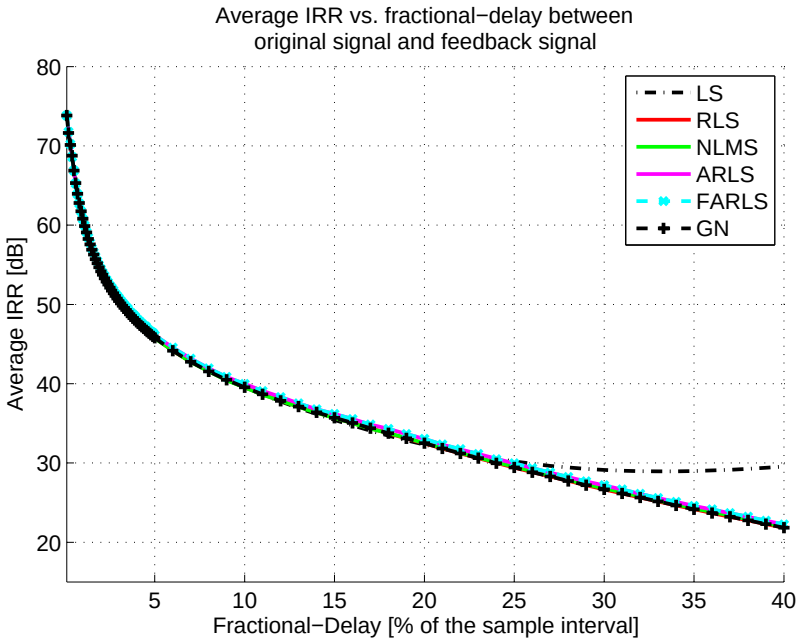
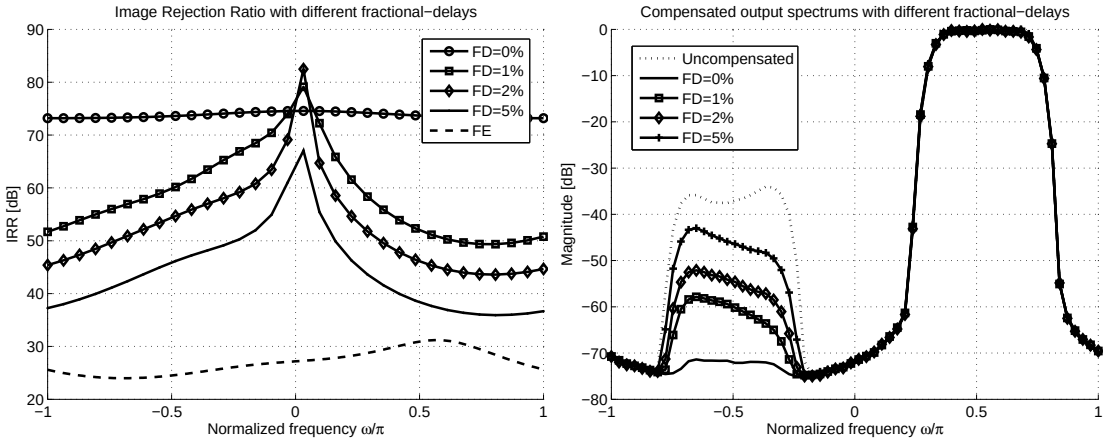


Figure 5.8: Average IRR as a function of fractional-delay between original signal and feedback loop signal. Low-IF signal with 16-QAM modulation, 35 % roll-off, and 5 times over-sampling. Symbol rate 1 MHz, sampling frequency 5 MHz, and IF 2.55 MHz. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees. Pre-distorter length of 3 and 25,000 samples used for estimation process.



(a) Image Rejection Ratio.

(b) Compensated spectra.

Figure 5.9: IRR graphs and compensated spectra with different fractional-delays between the original and the feedback loop signal as a function of normalized frequency. Low-IF signal with 16-QAM modulation, 35 % roll-off, and 5 times over-sampling. Symbol rate 1 MHz, sampling frequency 5 MHz, and IF 2.55 MHz. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees. Pre-distorter length length of 3 and 25,000 samples used for estimation process.

On the other hand, it is pointed out in [8] that with sufficiently long imbalance modeling filters $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$ the fractional-delay can be tolerated to some extent without separate fractional-delay estimation and compensation. In this thesis, the effect of longer imbalance modeling filters has been further simulated it has been shown that longer imbalance filters indeed relax fractional-delay compensation requirements. To achieve this, the assumption of imbalance filters $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$ being minimum-phase has to be relaxed and delay to them must be introduced. However, utilization of longer imbalance filters requires a higher feedback loop signal SNR. A comparison of IRR with different imbalance filter lengths, and corresponding delays, is presented in Figure 5.10. The optimum delay is always $D = \lfloor (N_g - 1)/2 \rfloor$. This yields that the observed feedback data sequence with covariance data windowing, in equation (3.50), becomes

$$\mathbf{y}(n) = [y(n - N_g - D + 1)\ y(n - N_g - D)\ \cdot\ \cdot\ \cdot\ y(n - L_b - D + 1)]^T \qquad (5.26)$$

From Figure 5.10, it is clear that algorithm performance between integer delay multiples is significantly improved, and it can tolerate fractional-delay to some extent without performance degradation.

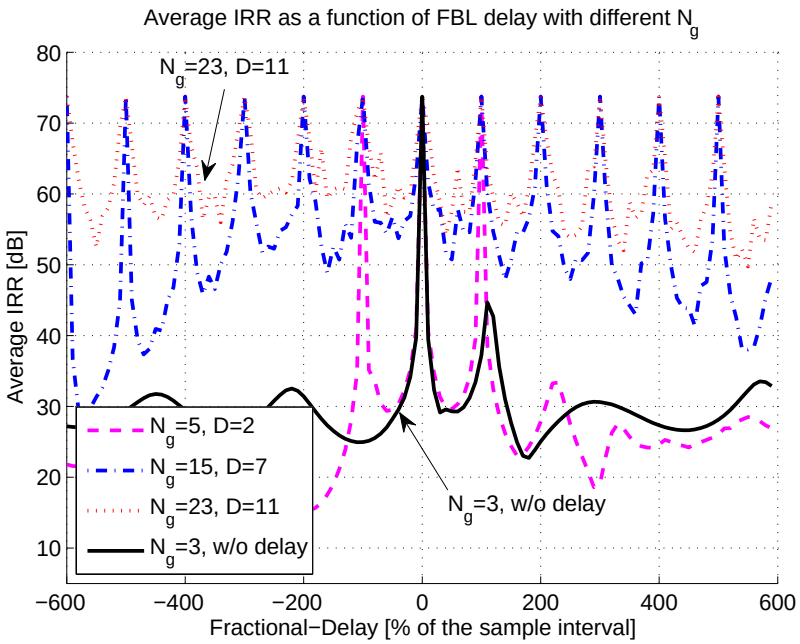


Figure 5.10: Average IRR with different filter lengths and delays as a function of delay between the feedback loop and the original signal. Low-IF signal with 16-QAM modulation, 35 % roll-off, and 5 times over-sampling. Symbol rate 1 MHz, sampling frequency 5 MHz, and IF 2.55 MHz. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees. Pre-distorter length fixed to 3 and 25,000 samples used for estimation process.

5.5.3 Feedback Loop SNR

In the real-time implementation in this thesis the SNR of the feedback loop signal is around 35-45 dB. With such a high feedback loop signal SNR this aspect should not become a limiting factor for the performance of the I/Q imbalance estimation algorithm if we consider 55dB as an adequate IRR value.

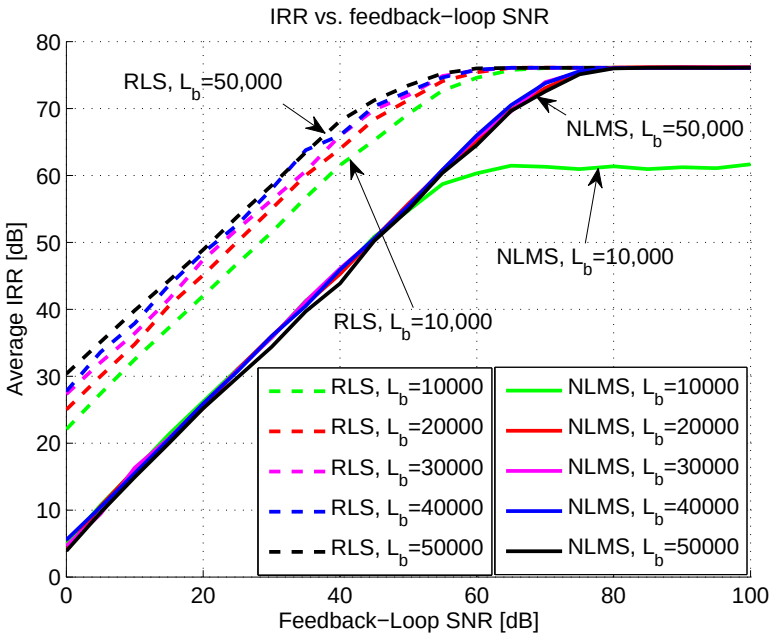
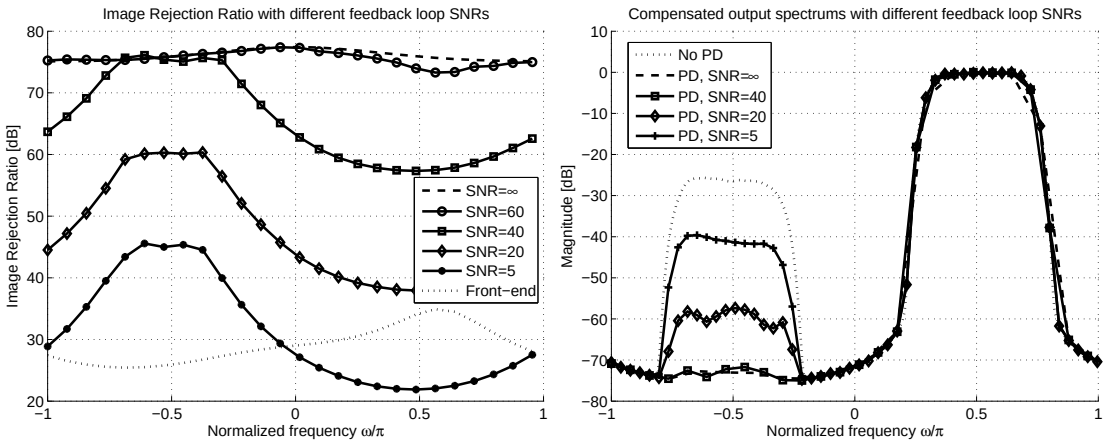


Figure 5.11: Average IRR with different estimation lengths L_b as a function of SNR of the feedback loop signal. Low-IF signal with 16-QAM modulation, 35 % roll-off, and 5 times over-sampling. Symbol rate 1 MHz, sampling frequency 5 MHz, and IF 2.55 MHz. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees. Pre-distorter length of 3 and 25,000 samples used for estimation process.

In general, it is known that adaptive algorithms need a higher number of samples for the estimation process if the input samples are distorted by noise. As a consequence, the number of samples used for the estimation also influences performance. It can be seen in Figure 5.11 that average IRR increases by about 1 dB for every additional 10,000 samples. It may also be seen that the SNR of the feedback loop signal should be around 30 dB to keep the performance degradation due to feedback loop noise acceptable. Fixed pre-distorter length results in a convergence of the IRR in Figure 5.11. Moreover, longer pre-distorter lengths would result in higher IRR.



(a) Image Rejection Ratio with different feedback loop SNRs.

(b) Simulated compensated spectra with different feedback loop SNRs.

Figure 5.12: IRR graphs and compensated spectra with different feedback loop SNRs as a function of normalized frequency. Low-IF signal with 16-QAM modulation, 35 % roll-off, and 5 times over-sampling. Symbol rate 1 MHz, sampling frequency 5 MHz, and IF 2.55 MHz. Frequency-selective I/Q imbalance model length of 3, where total amplitude response varies between 0.978 and 1.071, and total phase response varies between 1 and 7 degrees. Pre-distorter length of 3 and 25,000 samples used for estimation process.

5.5.4 Computational Complexity

The computational complexity of different adaptive algorithms is also a particularly important aspect of real-time implementation in that computation power is always limited and the computational burden should be kept as low as possible. In Figure 5.13 it will be seen that the RLS and GN algorithms are clearly the most complex algorithms considered. It should be noted that the curve in the figure assumes N_g to be the same length as N_w. The computational complexity of the RLS and GN algorithms grows exponentially as pre-distorter length increases. ARLS and FARLS algorithms are of a complexity close to the NLMS algorithm and their computational complexity grows linearly as pre-distorter length increases. Additionally, their performance is fairly good if the problems discussed above have been properly solved. On the other hand, RLS and GN algorithms outperform other algorithms under more demanding conditions.

5.5.5 Convergence Behavior of Different Adaptive Algorithms

This section briefly addresses the convergence behavior of the different adaptive algorithms. All the convergence plots show evolution of the average IRR values. In general, it will be noticed that RLS and GN algorithms have clearly the fastest

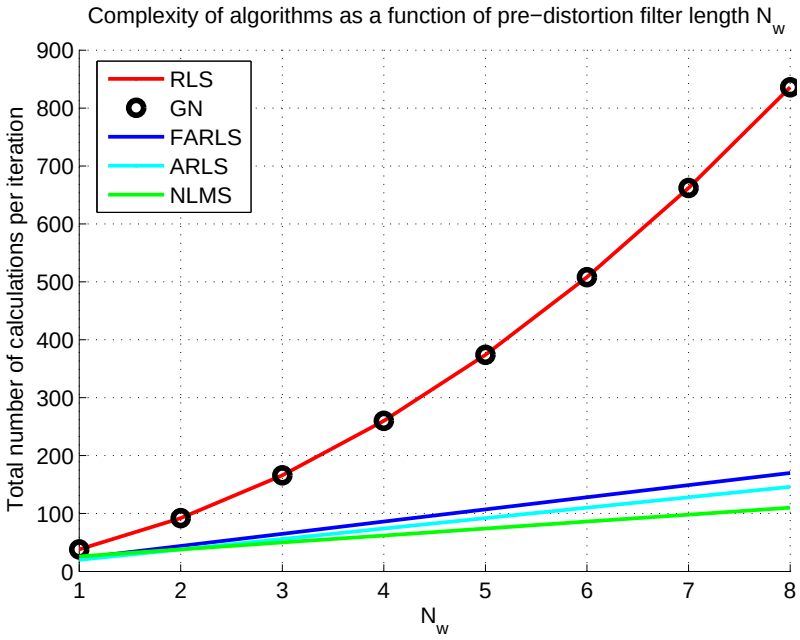


Figure 5.13: Computational complexity of adaptive algorithms as a function pre-distortion filter length. Total number of calculations means number of additions, subtractions, divisions and multiplications altogether during one iteration. N_g is assumed to be the length as N_w.

convergence speed and they have the highest steady-state IRR values in all cases. In practice, the limiting factor for the steady-state IRR of algorithms becomes the variance of the additive white Gaussian noise (AWGN) in the feedback loop signal and length of the pre-distortion filter. The IRR value can be considered good enough if it is in the order of 55 dB.

In Figure 5.14 the IRR evolution curves for the ideal case can be seen. Around 2,000 samples in fact suffice for RLS and GN algorithms to converge perfectly, while other algorithms would need a much higher number of samples to converge.

Similarly, in Figure 5.15 the IRR evolution graphs for the algorithms under different degrees of CFO are shown. Amount of the CFO clearly has an effect on the steady-state IRR of all the algorithms, but it is interesting to notice that RLS does not converge. After 1,000 iterations it starts to diverge strongly and average IRR value decreases significantly. It is also worth noting that the parameterless FARLS algorithm has the lowest steady-state IRR under CFO.

In Figure 5.16 the IRR evolution curves for the algorithms can be seen under fractional-delay. Around 5,000 samples is in fact a sufficient number for RLS and GN algorithms to converge perfectly, whereas other algorithms need about 15,000 samples to attain a performance which can be considered adequate. As will be

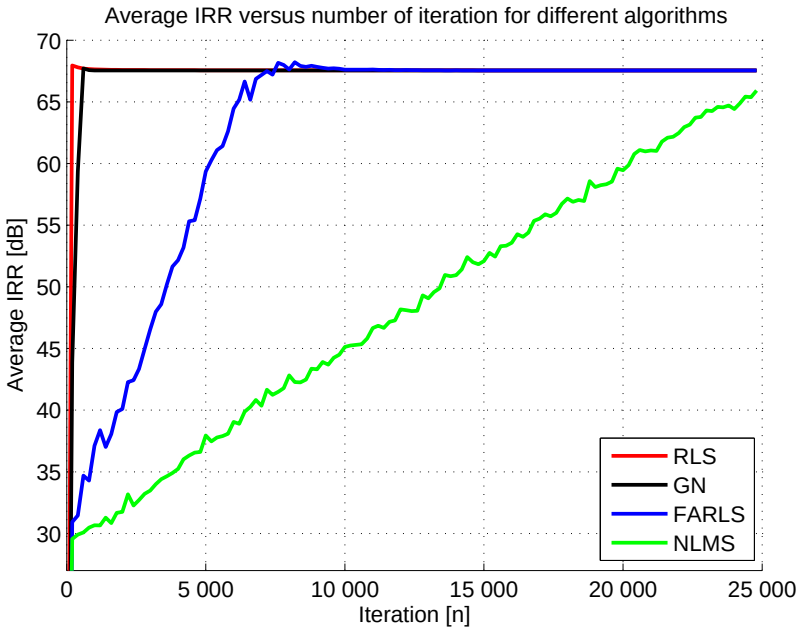


Figure 5.14: Comparison of the IRR evolution of different adaptive algorithms under ideal circumstances.

noted, the steady-state performance of different algorithms is fairly similar, the only difference being the convergence speed.

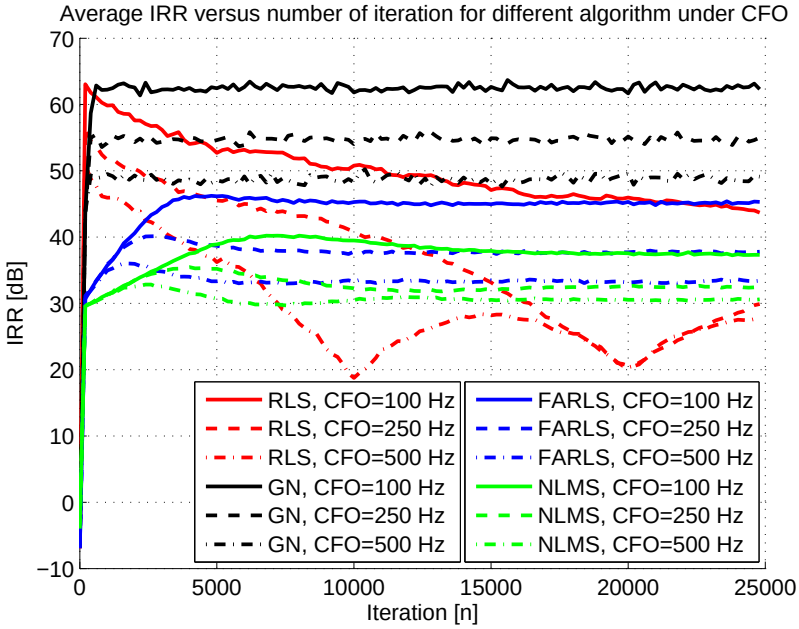


Figure 5.15: Comparison of the IRR evolution of different adaptive algorithms under carrier frequency offset between original and feedback loop signals. Sampling frequency was 5 MHz.

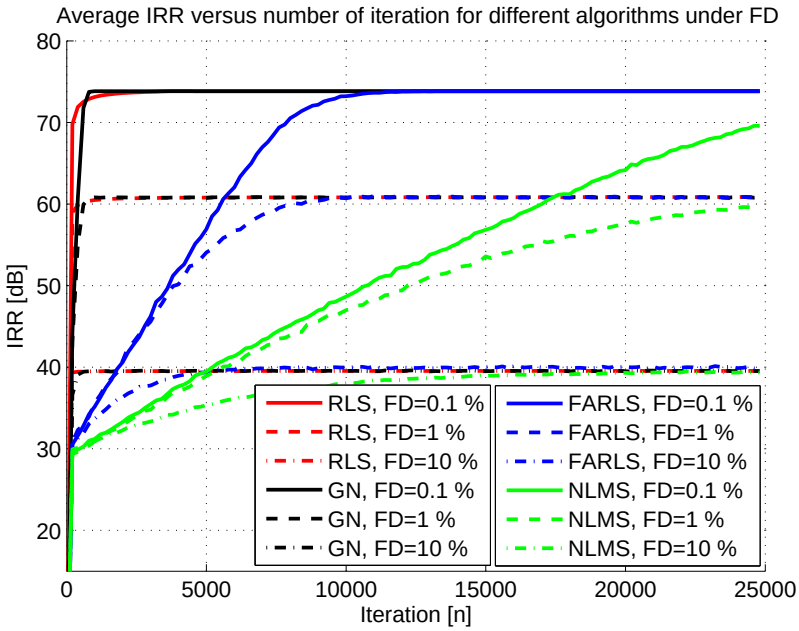


Figure 5.16: Comparison of the IRR evolution of different adaptive algorithms under different fractional-delays between original and feedback loop signals. Over-sampling factor was 5.

6. MEASUREMENTS AND RESULTS

This chapter aggregates the results which are based on the I/Q calibration concept discussed earlier in this thesis. It will discuss only the laboratory measurement results because Matlab simulation results have been given through out the thesis.

In the first section, an overview of the measurement setup is presented. Thereafter, the measurement results obtained using the developed real-time prototype implementation.

6.1 Measurement Setup

The measurement setup contained PC with an Intel Quadcore Q9550 processor, original USRP with RFX2400 daughter board, a Mini-Circuits splitter which had a frequency span from 1.5GHz to 8GHz, and an attenuator which was a variable RF attenuator from Mini-Circuits. Additionally, a number of shielded RF cables were used. All results were measured with a Rohde & Schwarz FSG 100A vector spectrum analyzer, which is also able to measure EVM figures for the measured signal. A general block diagram of the equipment connections can be seen in Figure 6.1.

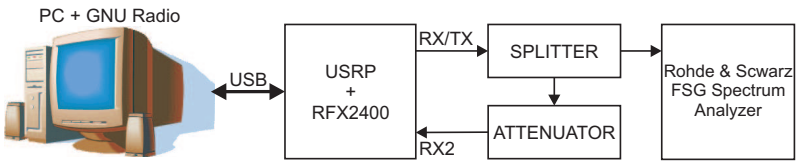


Figure 6.1: General block diagram of the used measurement setup.

6.2 Results from Real-Time Implementation

Although it has been verified with simulations that the pre-distortion algorithm presented is capable of suppressing image signals originating due to I/Q imbalance,

a practical real-time prototype implementation is indispensable to prove that the algorithm can also be used in practical real world radio transmitters.

The RLS parameters in all the measurements were $\lambda = 1$ and $\delta = 0.01$ and from 25,000 to 30,000 samples were used for I/Q imbalance parameter estimation.

6.2.1 Direct-Conversion Transmitter Mode Measurements

In Figure 6.2 are measured IRR curves for USRP RF FE with and without pre-distortion. The frequency-selective RF FE IRR was measured with two different methods. First, complex sinusoids were transmitted over the whole frequency band and IRR values were observed with spectrum analyzer. This can be seen as a solid red curve in Figure 6.2. Thereafter, it was measured with model-fitting-based approach. A data sequence of 200,000 samples was measured with spectrum analyzer and then the imbalance filter impulse responses $g_1(t)$ and $g_2(t)$ were estimated with I/Q imbalance estimation algorithm discussed earlier in Subsection 3.3.2. From $g_1(t)$ and $g_2(t)$, the RF FE IRR was calculated with (3.28). After this, estimated $g_1(t)$ and $g_2(t)$ were considered as correct imbalance parameters and RF FE IRRs for different pre-distortion filter lengths were calculated with (3.43). Lower performance of the pre-distorter with three and four taps could be explained with estimation noise due to the rather low feedback loop SNR or other impairments which have not been taken into account.

From Figure 6.3 it may be seen how the I/Q imbalance calibration algorithm affects the USRP output constellation when it is used in direct-conversion transmitter mode. The mean EVM value decreased from 10.15 % to 6.36 %. This means that signal quality increased by over 37 % if the mean EVM value is used as a measure.

Figure 6.4 shows that the calibration was really done in real time without any transmission interruptions. The EVM values were measured with a Rohde&Schwarz FSG 100A spectrum analyzer from the USRP RF output. The overall high level of the EVM value is a result of the rather high phase noise level in the USRP LO. In Figure 6.4(a) normal steady-state transmission is illustrated. Thereafter, in Figure 6.4(b) the corresponding EVM plots during the calibration can be seen for direct-conversion transmitter mode.

6.2.2 Low-IF Transmitter Mode Measurements

Figure 6.5 shows the USRP RF output spectrum with narrowband signal. The original digital signal was a low-IF signal with symbol rate of 250 kHz and 8 times

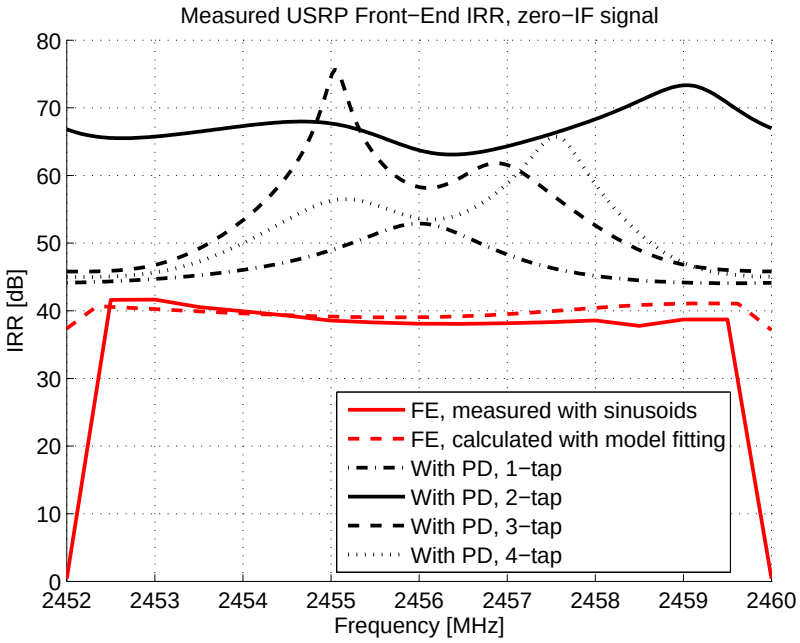


Figure 6.2: Measured USRP RF FE IRR graphs. FE was measured with Zero-IF signal with 16-QAM modulation, 4 MHz symbol rate, 2 times over-sampling, 30 % roll-off, and carrier frequency of 2.456 GHz. Pre-distorter coefficients were measured with Zero-IF signal with 16-QAM modulation, 2 MHz symbol rate, 2 times over-sampling, 30 % roll-off, and carrier frequency of 2.456 GHz. Pre-distorter length was from 1 to 4 and 30,000 samples used for the estimation.

over-sampling, which yields, with 25 % roll-off, a 312.5 kHz overall useful signal bandwidth and 2 MHz digital sampling frequency. Moreover, the digital IF was 0.5 kHz, analog IF for feedback loop 6MHz, carrier frequency 2.456 GHz, and analog sample rate 64 MHz. The pre-distorter length was 1 tap and 25,000 samples were used for estimation. As can be seen from the figure, the performance of the algorithm was very good. The image band signal was attenuated 67 dB, which is an eminently good result.

Figure 6.6 shows the USRP RF output spectrum with slightly wider band low-IF signal than in the previous spectrum in Figure 6.5. Digital sample rate was 4 MHz, analog sample rate 64 MHz, digital IF 1MHz, analog IF for feedback loop 6 MHz, and carrier frequency 2.456 GHz. The signal symbol rate was 1 MHz, which yields, with 30 % roll-off, a 1.3 MHz overall useful signal bandwidth. The pre-distorter length was 1, 2, and 3 taps. Moreover, 25,000 samples were used for estimation. As can be seen from the figure, the performance of the algorithm was particularly good. The image band signal was attenuated with 1-tap pre-distorter 59.4 dB, with 2-tap pre-distorter 55.7 dB, and with 3-tap pre-distorter 55.8 dB. Linear distortion in the spectrum comes from the overall response of the digital interpolation filter stage of the USRP transmission path.

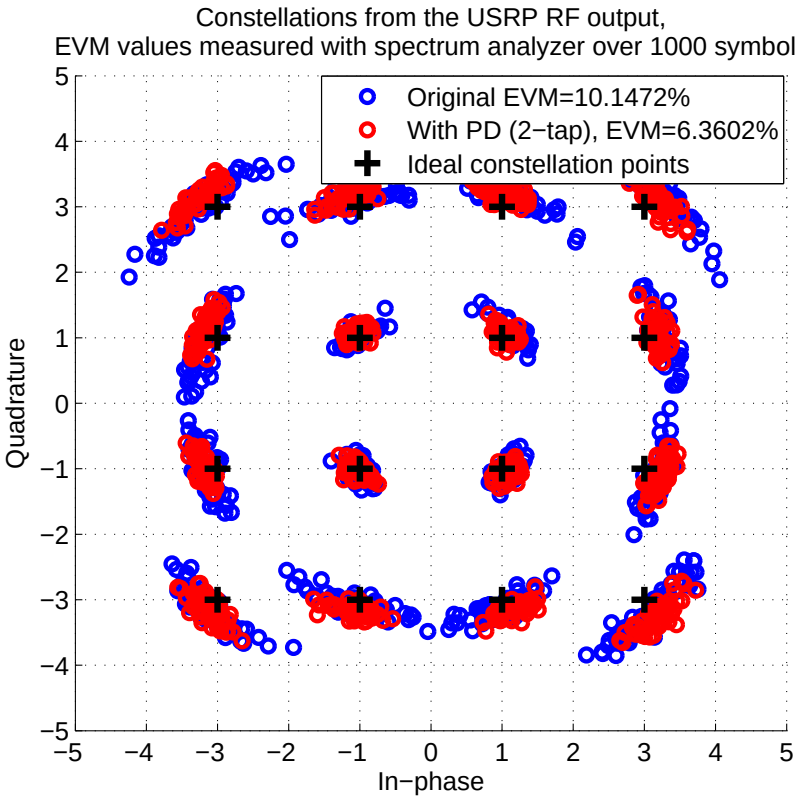
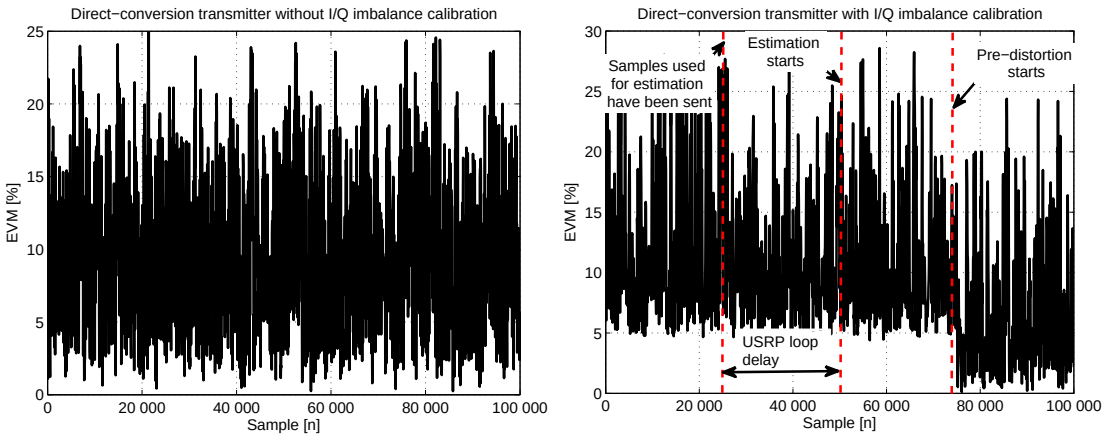


Figure 6.3: Measured constellation from the USRP RF output signal. Zero-IF signal with 16-QAM modulation, 1 MHz symbol rate, 4 times over-sampling, 30 % roll-off, IF of 1 MHz, and carrier frequency of 2.456 GHz. Pre-distorter length 2 and 25,000 samples used for the estimation.

Figure 6.7 illustrates USRP RF output spectra and measured constellations in low-IF mirror image interference demonstration. We have a reasonably weak desired signal at center frequency of 2455.5 MHz generated with vector signal generator. Desired signal is 16-QAM modulated signal with 500 kHz symbol rate and 30 % roll-off. In addition, we have band-limited noise transmitted from the USRP at center frequency of 2457 MHz with 1.4 MHz bandwidth. Due to the finite IRR of the USRP RF FE without I/Q calibration, mirror image of the band-limited noise falls over the desired signal which can be seen in Figure 6.7(a). In the measured constellation this is seen as additional noise, see Figure 6.7(b). If the WL-LS I/Q imbalance calibration scheme is used with 2-tap pre-distorter, the mirror image disappers almost entirely and measured constellation is clearly less noisy. In addition, 30,000 samples were used for the I/Q imbalance parameter estimation. The effect of the digital pre-distortion-based I/Q calibration can also be seen in Figures 6.7(a) and 6.7(b).

All the results in the Section 6.2 show that the algorithm gives reasonably good performance in real-time operation with real world signals. Multi-tap pre-distorters perform equally as well as a 1-tap pre-distorter because the measurement signal can



(a) Normal steady operation without PD. (b) During the calibration, direct-conversion.

Figure 6.4: In-band EVM figures from the output of the USRP unit during the calibration. Zero-IF signal with 16-QAM modulation, 4 times over-sampling, 30 % roll-off, and carrier frequency 2.456 GHz.

be considered narrowband and frequency-selectivity of the USRP RF FE is mild.

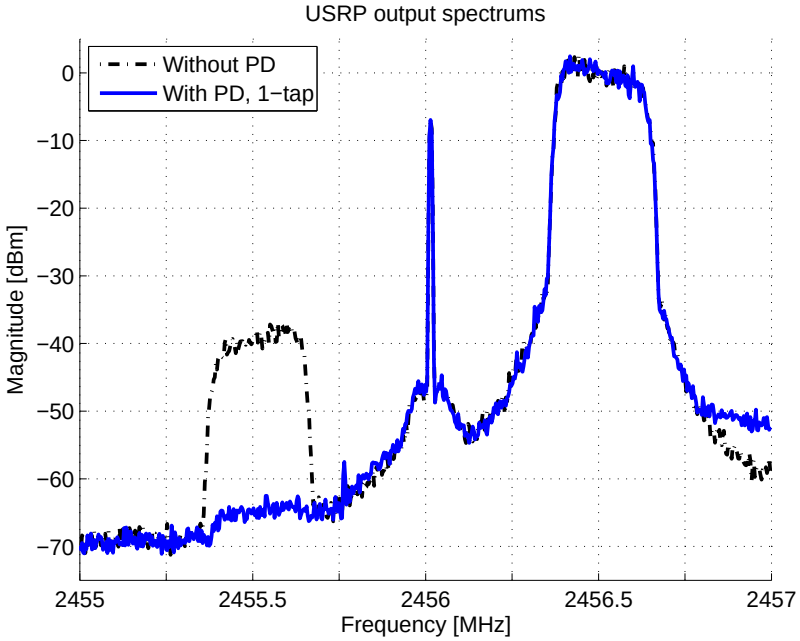


Figure 6.5: USRP RF output spectrum without and with I/Q imbalance mitigation algorithm. Low-IF signal with 16-QAM modulation, 250 kHz symbol rate, 8 times over-sampling, 25 % roll-off, IF of 500 kHz, and carrier frequency of 2.456 GHz. Pre-distorter length was 1 and 25,000 samples used for the estimation.

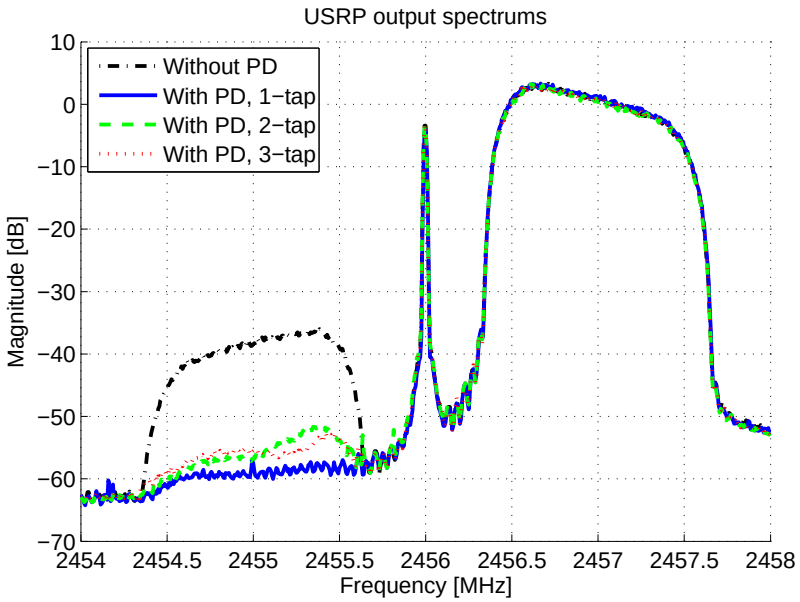
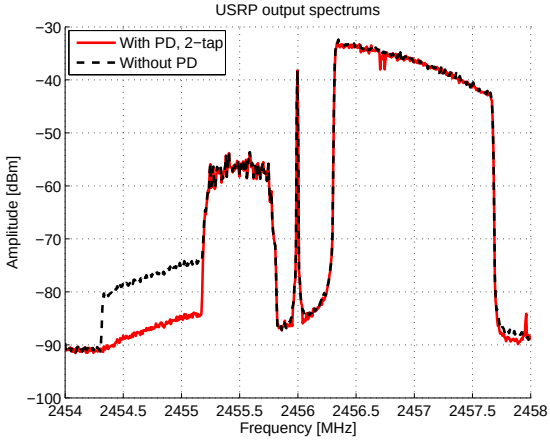
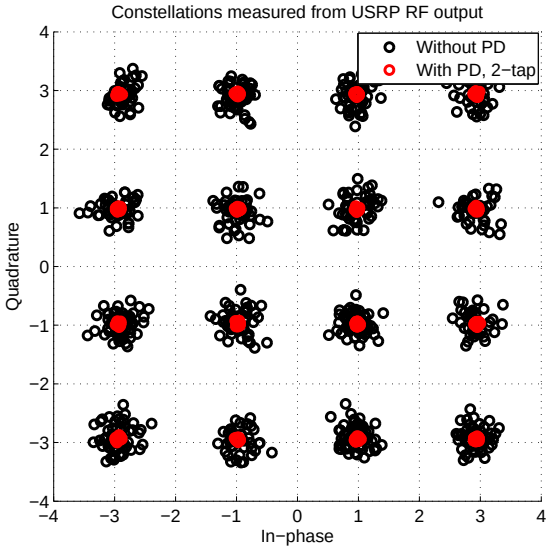


Figure 6.6: USRP RF output spectrum without and with I/Q imbalance mitigation algorithm. Low-IF signal with 16-QAM modulation, 1 MHz symbol rate, 4 times over-sampling, 30 % roll-off, IF of 1 MHz, and carrier frequency of 2.456 GHz. Pre-distorter length 1, 2, and 3. 25,000 samples used for the estimation.



(a) USRP RF output spectra.



(b) Constellations measured from USRP RF output.

Figure 6.7: Measurement results from low-IF mirror image interference demonstration.

7. CONCLUSIONS

The goal in this thesis project was to study all the implementation-related aspects of the digital pre-distortion-based transmitter I/Q imbalance calibration algorithm utilizing a WL-LS estimation approach. First, transmitter architectures and their prospects and downsides were discussed. Thereafter, signal models for the frequency-selective I/Q imbalance calibration algorithm were addressed. Then, an efficient digital pre-distortion-based imbalance calibration method was presented and the required equations defined. Next, the development platform and the development environment were introduced and the performance figures briefly discussed. The implementation details were also extensively discussed and the practical aspects of the algorithm were addressed, with closest focus on the prototype implementation-related aspects. Finally, the results from the Matlab simulations and from the laboratory measurements were presented.

As mentioned in Chapter 3, the state of the art transmitters can achieve about 35-40 dB mirror image attenuation without additional calibration. In simulations the WL-LS estimation and mitigation approach was shown to improve IRR from the current level to about 55-70 dB, which is a very substantial advance and step towards the realizability of a wideband multi-standard transceiver. In addition, the real-time prototype implementation was shown to give results congruent with the Matlab simulations when practical aspects were taken into account. In practice, the reliability of the estimation algorithm is mostly dependent on the exact and consistent delay compensation, which proved to be a demanding task. The study also showed that recursive implementation of the WL-LS compensation approach is feasible with the current SDR platforms and it was shown to be implementable with quite limited computational resources.

This real-time prototype implementation gives a good basis for future research and work to realize possible FPGA-based solutions. Future work will also include real-time prototype implementation of the joint PA linearization and I/Q imbalance mitigation algorithm proposed by the authors in [9, 10].

REFERENCES

[1] A.A. Abidi, "CMOS wireless transceivers: the new wave," *IEEE Commun. Mag.*, vol. 39, iss. 8, pp. 119-124, Aug. 1999.

[2] A.A. Abidi, "Direct-conversion radio transceivers for digital communications," *IEEE J. Solid-State Circuits*, vol. 30, iss. 12, pp. 1399-1410, Dec. 1995.

[3] A.A. Abidi, "Low-power radio-frequency ICs for portable communications," *IEEE Proceedings*, vol. 83, iss. 4, pp. 544-569, 1995.

[4] F. Agnelli, G. Albasini, I. Bietti, A. Gnudi, A. Lacaita, D. Manstretta, R. Rovatti, E. Sacchi, P. Savazzi, F. Svelto, E. Temporiti, S. Vitali, and R. Castello, "Wireless multi-standard terminals: system analysis and design of a reconfigurable RF front-end," *IEEE Circuits and Systems Mag.*, vol. 6, pp. 38-59, Jun. 2006.

[5] L. Angrisani, A. Napolitano, and M. Vadursi, "Measuring I/Q impairments in WiMAX transmitters," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, iss. 5, pp. 1299-1306, 2009.

[6] L. Angrisani, and R. Colella, "Detection and evaluation of I/Q impairments in RF digital transmitters," *IEEE International SoC Design Conference (ISOCC)*, vol. 151, iss. 1, pp. 39-45, 2004.

[7] L. Angrisani, M. D'Arco, and M. Vadursi, "Clustering-based method for detecting and evaluating I/Q impairments in radio-frequency digital transmitters," *IEEE Transactions on Instrumentation and Measurement*, vol. 56, iss. 6, pp. 2139-2146, 2007.

[8] L. Anttila, M. Valkama, and M. Renfors, "Frequency-selective I/Q mismatch calibration for wideband direct conversion transmitters," *IEEE Trans. Circuits Syst.*, vol. 55, pp. 359-363, Apr. 2008.

[9] L. Anttila, P. Händel, and M. Valkama, "Joint mitigation of power amplifier and I/Q modulator impairments in broadband direct-conversion transmitters," *IEEE Trans. Microw. Theory Tech.*, vol. 58, iss. 4, pp. 730-739, Apr. 2010.

[10] L. Anttila, P. Händel, O. Mylläri, and M. Valkama, "Recursive learning based joint digital predistorter for power amplifier and I/Q modulator impairments," *EuMA Internationa Journal on Microwave Theory, Special Issue*, To appear in Apr. 2010.

[11] V. Arkesteijn, "Analog front-ends for software-defined radio receivers," Ph.D. dissertation, University of Twente, Twente, Neatherlands, 2007.

[12] A. Baschirotto, R. Castello, F. Campi, G. Cesura, M. Toma, R. Guerrieri, R. Lodi, L. Lavagno, and P. Malcovati, "Baseband analog front-end and digital back-end for reconfigurable multi-standard terminals," *IEEE Circuits Systems Mag.*, vol. 6, pp. 8-28, Jun. 2006.

[13] D.M. Bland, and A. Tarczynski, "The effect of sampling jitter in a digitized signal," *Proceedings of 1997 IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 2685-2688, 1997.

[14] E. Blossom, "How to write a signal processing block," [online], accessible: http://www.gnu.org/software/gnuradio/doc/howto-write-a-block.html, referred on 13.11.2009.

[15] M. Bruno, M. Murdy, P. Perrault, A.M. Wyglinski, and J.A. McNeill, "Widely tunable RF transceiver front end for software-defined radio," *IEEE Military Communications Conference MILCOM 2009*, 2009.

[16] A.B. Carlson, Paul B.C., and J.C. Rutledge, *Communication Systems*, 4^{th} ed., McGraw-Hill, 2001.

[17] J.K. Cavers, and M.W. Liao, "Adaptive compensation for imbalance and offset losses in direct conversion transceivers," *IEEE Trans. Veh. Commun.*, vol. 42, pp. 581-588, Nov. 1993.

[18] J.K. Cavers, "The effect of quadrature modulator and demodulator errors on adaptive digital predistorters for amplifier linearization," *IEEE Trans. Veh. Commun.*, vol. 46, iss. 2, pp. 456-466, May 1997.

[19] M.M. Chansarkar, U.B. Desai, and B.V. Rao, "A comparative study of the approximate RLS with LMS and RLS algorithms," *Fourth IEEE Region 10 Int. Conf., TENCON '89.* pp. 255-258, Nov. 1989.

[20] M.M. Chansarkar, and U.B. Desai, "A fast approximate RLS algorithm," *IEEE Region 10 Conf. on Computer, Communication, Control and Power Engineering, TENCON '93*, vol.3, pp. 532-536, Oct. 1993.

[21] H-H. Chen, J-T. Chen, and P-C. Huang, "Adaptive I/Q imbalance compensation for RF transceivers," *GLOBECOM '04. IEEE Global Telecommunications Conference*, vol. 2, pp. 818-822, Nov. 2004.

[22] S-J. Chen, and Y-H. Sieh, *IQ Calibration Techniques for CMOS Radio Transceivers*, Springer, 2006.

[23] A.B. Cooper, P. Mulligan, W.R. Kenan, and P. Strasser, "GNU Radio Companion," [online], accessible: http://gnuradio.org/trac/wiki/GNURadioCompanio, referred on 13.11.2009.

[24] J. Crols, and M. Steyaert, "A fully integrated 900 MHz CMOS double quadrature down converter," *Proceedings IEEE International solid-State Circuits Conference*, pp. 136-137, Feb. 1995.

[25] I. Dagres, A. Zalonis, N. Dimitriou, K. Nikitopoulos, and A. Polydoros, "Flexible radio: a framework for optimized multimodal operation via dynamic signal design," *EURASIP Journal on Wireless Communications and Networking*, 2005.

[26] L. Ding, Z. Ma, D.R. Morgan, M. Zierdt, and G.T. Zhou, "Frequency-dependent modulator imbalance in predistortion linearization systems: modeling and compensation," *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 1, 2003.

[27] Epiq Solutions, [online], accessible: http://www.epiq-solutions.com, referred on 5.5.2010.

[28] Ettus Reseach, [online], accessible: http://www.ettus.com, referred on 24.3.2010.

[29] R. Farrell, M. Sanchez, and G. Corley, "Software-defined radio demonstrators: an example and future trends," *Hindawi International Journal of Digital Multimedia Broadcasting*, 2009.

[30] G. Fettweis, M. Löhning, D. Petrovic, M. Windisch, P. Zillman, and W. Rave, "Dirty RF: a new paradigm," *IEEE Personal Indoor and Mobile Radio Communications*, vol. 4, pp. 2347-2355, Sept. 2005.

[31] G. Fettweis, E. Zimmermann, V. Jungnickel, and E.A. Jorswieck, "Challenges in future short range wireless systems," *IEEE Veh. Commun. Mag.*, vol. 1, iss. 2, pp. 24-31, 2006.

[32] K. Fujii, and J. Ohga, "Sub-RLS algorithm with an extremely simple update equation," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 2325-2328, 1997.

[33] J. Glyn, and M.Moeneclaey, *Advanced Modern Engineering Mathematics*, Prentice Hall, 2003.

[34] GNU Radio, [online], accessible: http://www.gnuradio.org, referred on 24.3.2010.

[35] Q. Gu, *RF system design of transceivers for wireless communications*, Springer, 2005.

[36] S. Haykin, *Adaptive Filter Theory*, 3^{rd} ed., Prentice Hall, 1996.

[37] S. Haykin, *Communication systems*, 4^{th} ed., John Wiley & Sons, Inc., 2001.

[38] D.S. Hilborn, S.P. Stapleton, and J.K. Cavers, "An adaptive direct conversion transmitter," *IEEE Trans. Veh. Commun.*, vol. 34, iss. 2, pp. 223-233, May 1994.

[39] G. Hueber, Y. Zou, K. Dufrene, R. Stuhlberger, and M. Valkama, "Smart front-end signal processing for advanced wireless receivers," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, iss. 3, pp. 472-487, 2009.

[40] M. Isaksson, D. Wisell, and D. Ronnow, "A comparative analysis of behavioral models for RF power amplifiers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, iss. 1, pp. 348-359, 2006.

[41] M. Ismail, *Radio Design in Nanometer Technologies*, Springer, 2006.

[42] International Telecommunication Union, [online], accessible: http://www.itu.int/ITU-D/ict/newslog/ITU+Sees+5+Billion+Mobile +Subscriptions+Globally+In+2010.aspx, referred on 23.3.2010.

[43] M. Iqbal, J. Lee, and K. Kiseon, "Performance comparison of digital modulation schemes with respect to phase noise spectral shape," *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 856-860, 2000.

[44] A. Katz, "Linearization: reducing distortion in power amplifiers," *IEEE Microwave Magazine*, vol. 2, iss. 4, pp. 37-49, 2001.

[45] P.B. Kenington, "Linearized transmitters: an enabling technology for software defined radio," *IEEE Communications Magazine*, vol. 40, iss. 2, pp. 156-162, 2002.

[46] P. B. Kenington, *High Linearity RF Amplifier Design*, Artech House, 2000.

[47] W. Kester, *The Data Conversion Handbook*, Newnes, 2005.

[48] A.Q. Kiayani, "DSP based transmitter I/Q imbalance calibration - implementation and performance measurements," M.Sc. Thesis, Tampere University of Technology, Tampere, Finland, 2009.

[49] Y. Kim, and C-S. Lee, "A novel method to remove the mismatch of the up-conversion mixer for a low IF transmitter," *MWSCAS '04. The 2004 47th Midwest Symposium on Circuits and Systems*, vol. 1, pp. 25-28, Jul. 2004.

[50] T.I. Laakso, V. Välimäki, M. Karjalainen, and U.K. Laine, "Splitting the unit delay," *IEEE Signal Process. Mag.*, vol. 13, iss. 1, pp. 30-60, Jan. 1996.

[51] M. Laddomada, F. Daneshgaran, M. Mondin, and R.M. Hickling, "A PC-based software receiver using a novel front-end technology," *IEEE Commun. Mag.*, vol. 39, pp. 136-145, Aug. 2001.

[52] K-Y. Lee, S-W. Lee, Y. Koo, H-K. Huh, H-Y. Nam, J-W. Lee, J. Park, K. Lee, D-K. Jeong, and W. Kim, "Full-CMOS 2-GHz WCDMA direct conversion transmitter and receiver," *IEEE Journal of Solid-State Circuits*, vol. 38, iss. 1, pp. 45-53, 2003.

[53] T.H. Lee, and A. Hajimiri, "Oscillator phase noise: a tutorial," *IEEE Journal of Solid-State Circuits*, vol. 35, iss. 3, pp. 326-336, 2000.

[54] H. Li, D.H. Kwon, D. Chen, and Y. Chiu, "A fast digital predistortion algorithm for radio-frequency power amplifier linearization with loop delay compensation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, iss. 3, pp. 374-383, 2009.

[55] A. Luzzatto and G. G. Shirazi, *Wireless Transceiver Design*, John Wiley & Sons, 2007.

[56] P-I. Mak, S-P. U, and R.P. Martins, "Transceiver architecture selection: review, state-of-the-art survey and case study," *IEEE Trans. Circuits Syst.*, vol. 7, pp. 6-25, April 2007.

[57] F. Maloberti, *Data Converters*, Springer, 2007.

[58] N. Manicka, "GNU Radio testbed," M.Sc. Thesis, University of Delaware, Delaware, USA, 2007.

[59] C. Masse, "A 2.4GHz direct conversion transmitter for Wimax applications," *IEEE Radio Frequency Integrated Circuits (RFIC) Symposium*, pp. 400-404, Jun. 2006.

[60] U. Mengali, and A.N. D'Andrea, *Synchronization Techniques for Digital Receivers*, Plenum Press, 1997.

[61] S. Mirabbasi, and K. Martin, "Classical and modern receiver architectures," *IEEE Commun. Mag.*, vol. 38, pp. 132-139, Nov. 2000.

[62] J. Mitola, "The software radio architecture," *IEEE Commun. Mag.*, vol. 33, pp. 26-38, May 1995.

[63] J. Mitola, "Software radio architecture: a mathematical perspective," *IEEE Journal on Selected Areas in Communications*, vol. 17, iss. 4, pp. 514-538, Apr. 1999.

[64] H. Meyr, and M. Moeneclaey, *Digital communication Receivers: Synchronization, Channel Estimation and Signal Processing*, Wiley Interscience, 1998.

[65] K. Muhammad, and R.B. Staszewski, "Direct RF sampling mixer with recursive filtering in charge domain," *Proceedings of the 2004 International Symposium on Circuits and Systems*, vol. 1, pp. 577-580, May 2004.

[66] M. Nentwig, "Delay estimation by FFT," [online], accessible: http://www.dsprelated.com/showarticle/26.php, referred on 24.3.2010.

[67] R. Normoyle, and P. Mesibov, "The VITA radio transport as a framework for software definable radio architectures," *Proceeding of the SDR 08 Technical Conference and Production Exposition*, 2008.

[68] J.B. Simoneau, and L.W. Pearson, "Digital augmentation of RF component performance in software-defined radio," *IEEE Trans. Microw. Theory Tech.*, vol. 57, pp. 573-581, Mar. 2009.

[69] P.Z. Peebles, *Digital Communication Systems*, Prentice-Hall, 1987.

[70] C. Peng, and Q. Wei, "A local oscillator leakage signal eliminating circuit for direct quadrature up-conversion transmitters," *IEEE International Conference on Communication Technology*, pp. 1-3, 2006.

[71] J. Proakis, *Digital Communications*, 4^{th} ed., McGraw-Hill, 2000.

[72] X. Qian, "Transceiver for Future Wireless Communications," *The 3rd Conference on Mobile Technology, Applications and Systems - Mobility*, 2006.

[73] B. Razavi, "Challenges in portable RF transceiver design," *IEEE Circuits Devices Mag.*, vol. 12, pp. 12-25, 1996.

[74] J. H. Reed, *Software Radio*, Prentice Hall, 2002.

[75] E. Rikkonen, "Waveform development for software defined radio," M.Sc. Thesis, Tampere University of Technology, Tampere, Finland, 2008.

[76] H-G. Ryu, and Y-S. Lee, "Phase noise analysis of the OFDM communication system by the standard frequency deviation," *IEEE Transactions on Consumer Electronics*, vol. 49, iss. 1, pp. 41-47, 2003.

[77] Sayed, A.H. *Adaptive Filters*, IEEE Press, Wiley-Interscience, 2008.

[78] J.B. Simoneau, and L.W. Pearson, "Multitone feedback through demodulating log detector for detection of spurious emissions in software radio," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, iss. 10, pp. 2222-2228, Oct. 2007.

[79] J. Stevenhans, D. Haspeslagh, A. Delarbre, L. Kiss, Z. Chang, and J. F. Kukielka, "An analog radio front end chip set for a 1.9 GHz mobile radio telephone application," *Proceedings IEEE International Solid-State Circuits Conference*, pp. 44-45, Feb. 1994.

[80] M. Streifinger, T. Müller, J.-F. Luy, and E.A. Biebl, "A software-radio front-end for microwave applications," *Digest of Papers IEEE Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*, pp. 53-56, April 2003.

[81] A. Swaminathan, M. Snelgrove, S. Jantzi, and S. Bazarjani, "A monolithic complex sigma-delta modulator for digital radio," *IEEE-CAS Region 8 Workshop on Analog and Mixed IC Design*, pp. 83-86, Sept. 1996.

[82] S. Tang, K. Gong, J. Wang, K. Peng, C. Pan, and Z. Yang, "Loop delay correction for adaptive digital linearization of power amplifiers," *IEEE Wireless Communications and Networking Conference*, pp. 1987-1990, 2007.

[83] M. Valkama, "Advanced I/Q signal processing for wideband receivers: models and algorithms," Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, 2001.

[84] N. Vasudev, and O.M. Collins, "Near-ideal RF upconverters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, iss. 11, pp. 2569-2575, 2002.

[85] J. Venkataraman, and O. Collins, "An all-digital transmitter with a 1-Bit DAC," *IEEE Transactions on Communications*, vol. 55, iss. 10, pp. 1951-1962, 2007.

[86] VITA, "Radio transport for SDR," [ónline], accessible: http://www.digitalif.org, referred on 25.11.2009.

[87] M. Windisch, and G. Fettweis, "Adaptive I/Q imbalance compensation in low-IF transmitter architectures," *VTC2004-Fall IEEE 60th Vehicular Technology Conference*, vol. 3, pp. 2096-2100, Sept. 2004.

[88] A.M. Wyglinski, M. Nekovee, and Y.T.Hou, *Cognitive Radio Communications and Networks*. Elsevier Inc., 2010.

[89] Y. Xiao, G. He, and J. Ma, "A novel method for estimation and compensation of transmitter I/Q imbalance," *IEEE International SoC Design Conference (ISOCC)*, pp. 261-265, 2009.

[90] S-R. Yoon, and S-C. Park, "All-digital transmitter architecture based on bandpass delta-sigma modulator," *IEEE 9th International Symposium on Communications and Information Technology*, pp. 703-706, 2009.

[91] Y.V. Zakharov, G.P. White, and J. Liu, "Low-complexity RLS algorithms using dichotomous coordinate descent iterations," *IEEE Transactions on Signal Processing*, vol. 56, iss. 7, pp. 3150-3161, 2008.

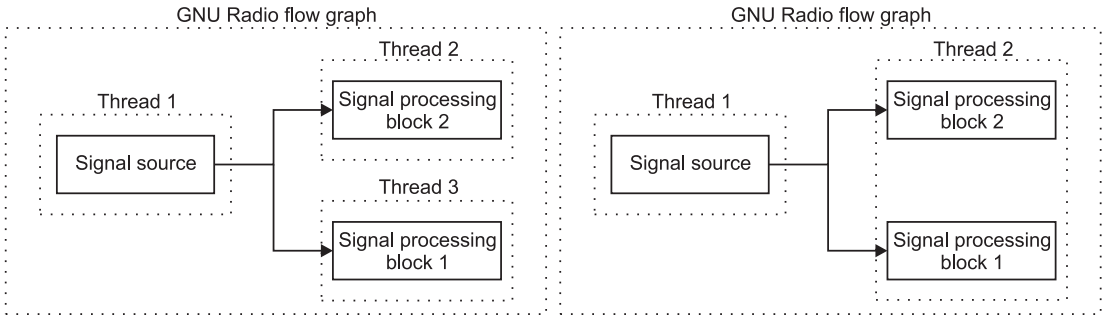
[92] Y. Zou, "Analysis and mitigation of I/Q imbalances in multi-antenna transmission systems," Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, 2009.

Appendix A. GOOD PRACTICES IN GNU RADIO ENVIRONMENT

This chapter will discuss some practices which have shown to be useful with GNU Radio in general and while writing new signal processing blocks. The chapter will not address any detailed information about the development of a new signal processing block because there are reasonably good instruction on the Internet for that, see for example [A1,A2].

A.1 Sample Timing Through Parallel Flow Graph Paths

If flow graph consists of parallel structures the programmer must take into account that samples on the parallel paths may not be perfectly synchronized. Of course, this may depend on various reasons but two main reasons are structure of signal processing blocks and thread-based scheduling. Different delay between parallel paths is obvious because different signal processing blocks may need varying number of input samples to get desired output. On the other hand, the thread-based scheduling may generate delay to the parallel sample streams which is not that obvious. If parallel signal processing blocks are in the same thread the sample streams remain synchronized, but if they are not, there will, or may, be some integer sample offset in the parallel sample streams. This may become a major problem because the programmer cannot know which signal processing blocks are in which threads. This is why parallel signal processing blocks have to utilize some kind of synchronization signal which keeps both blocks operating on same sample intervals. See Figure A.1 for conceptual illustration how parallel signal processing blocks can be divided to different threads. In Figure A.1(a) parallel signal processing blocks are inside separate threads and symbol timing between parallel paths will be compromised. On the other hand, Figure A.1(b) illustrates a case where symbols on parallel paths will remain synchronized.



(a) Inside different threads. (b) Inside same thread.

Figure A.1: Illustration of how parallel signal processing blocks can be divided to different threads.

A.2 Data Types in GNU Radio

There are special data types defined in GNU Radio and they can be found in more detail from header files gr_types.h and gr_complex.h. As is seen in Table A.1, the data types with the prefix 'gr_' are nothing but new names for the C++ built-in data types. They are just represented by using a consistent and convenient way.

GNU Radio data type	True C++ data type
gr_complex	std::complex<float>
gr_complexd	std::complex<double>
gr_vector_int	std::vector<int>
gr_vector_float	std::vector<float>
gr_vector_double	std::vector<double>
gr_vector_void_star	std::vector<void *>
gr_vector_const_void_star	std::vector<const void *>
gr_int16	short
gr_int32	int
gr_uint16	unsigned short
gr_uint32	unsigned int

Table A.1: GNU Radio data types.

A.3 Callback Methods

Certain blocks have callback methods that allow their parameters to be changed while executing flow graph. These callback methods can be used in normal Python flow graphs or in GNU Radio Companion, albeit their usability in GNU Radio

Companion is little bit limited. Of course, variables can be passed from one block to another with help of signal streams but this kind of procedure loads the flow graph unacceptably much. As a consequence, the advantage of callback methods is that the variable passing is done only on demand.

For example, in transmitter I/Q imbalance mitigation algorithm, callback methods are used to pass pre-distortion filter coefficients from I/Q imbalance estimation block to pre-distortion block and to pass permission to start estimation from synchronization block to I/Q imbalance estimation block.

A.4 Real-Time Scheduling

Real-time scheduling gives the flow graph a low-level access to the operating system kernel enabling faster and more reliable Gruel scheduling for the application. This significantly improves the performance of the application.

In Python programming language real-time scheduling is enabled with command "gr.enable_realtime_scheduling()" and in GNU Radio Companion selecting it from document properties. In addition, the flow graph has to to be ran with super-user privileges, otherwise the flow graph will be ran without real-time scheduling.

A.5 Useful Methods in All Signal Processing Blocks

There are a few useful methods related to all signal processing blocks. These methods are given and briefly discussed in this section.

A.5.1 The Method forecast()

The method forecast() is used to estimate the required number of input samples given the requested number of output samples. The first parameter noutput_items is the number of the required number of output samples for each output stream and it has been introduced in general_work(). The second parameter ninput_items_required is a vector of integer values, saving the number of input samples required from each input stream.

```
1  void tut_tx_iq_imb_mitig_cc::forecast(int noutput_items, gr_vector_int &
       ninput_items_required) {
2    for (unsigned int n = 0; n < ninput_items_required.size(); n++) {
3      ninput_items_required[n] = 2048;
4    }
```

```
5     }
```

Listing A.1: Example code of using method forecast()

Listing A.1 gives an example how method forecast() can be overridden. It requires 2048 samples from each input. When the forecast() method is override, the number of data items required on each input stream has to be estimated, given the request to produce noutput_items samples for each output stream. The estimate does not have to be exact, but should be close. The argument ninput_items_required is passed by reference, so that the calculated estimates can be saved directly into it. The program itself should not call method forecast() because GNU Radio scheduler calls it when needed.

A.5.2 The Method set_history()

The method set_history() is used for telling the scheduler required history length. The effect of this is that GNU Radio frame work will automatically overlap consecutive signal blocks. This is particularly useful method with signal processing blocks which include filtering and need given number of samples from previous signal block because of the filter delays. Command "set_history(N)" sets the history to be N samples. In practice, in the example, $N-1$ samples are preserved from the previous signal block.

A.5.3 The Method set_output_multiple()

Now let's introduce the fourth member variable defined in gr_block: d_output_multiple. It is used to constrain the noutput_items argument passed to forecast() and general_work().

The scheduler will ensure that the noutput_items argument passed to forecast() and general_work() will be an integer multiple of d_output_multiple. The default value of d_output_multiple is 1.

Here is a critical point worth emphasizing. Suppose we're going to design a block class, after we override the general_work() or forecast() methods, who will use or call these methods, and how? When we implement general_work() or forecast(), we always assume noutput_items and other arguments have been provided conceptually. In fact, we never call these methods and set the arguments explicitly. The scheduler will organize everything and call the methods according to the higher level

policy and buffer allocation strategy. The tricks behind the scene involve too many details and are beyond the necessity.

We do have some level of control to the argument noutput_items. The variable d_output_multiple tells the scheduler noutput_items must be an integer multiple of d_output_multiple. We can set the value of d_output_multiple using the method set_output_multiple() and get its value using the method output_multiple().

```
1  void gr_block::set_output_multiple (int multiple) {
2    if (multiple < 1) {
3      throw std::invalid_argument ("gr_block::set_output_multiple");
4    }
5    d_output_multiple = multiple;
6  }
```

Listing A.2: Method set_output_multiple()

A.5.4 The Method set_relative_rate()

The fifth member variable d_relative_rate gives the approximate information on the relative data rate, i.e. the approximate output rate / input rate. This information provides a hint to the buffer allocator and scheduler, so that they can arrange the buffer allocation and adjust parameters accordingly. d_relative_rate is 1.0 by default, which is true for most signal processing blocks. Obviously, the decimators' d_relative_rates should be less than 1.0, while the interpolators' d_relative_rates is larger than 1.0. We can set the value of d_relative_rate using the method set_relative_rate() and get its value using the method relative_rate().

```
1  void gr_block::set_relative_rate (double relative_rate) {
2    if (relative_rate < 0.0) {
3      throw std::invalid_argument ("gr_block::set_relative_rate");
4    }
5    d_relative_rate = relative_rate;
6  }
```

Listing A.3: Method set_relative_rate()

References

[A1] Radioware, "Writing a signal processing block for GNU Radio," [online], accessible: https://radioware.nd.edu/documentation/advanced-gnuradio.

[A2] E. Blossom, "How to write a signal processing block," [online], accessible: http://www.gnu.org/software/gnuradio/doc/howto-write-a-block.html.

Appendix B. GNU RADIO EXAMPLES

This chapter gives a few basic examples of the use of GNU Radio with different programming languages and GNU Radio Companion. First, simple FM radio receiver and QPSK receiver without GUI are given with Python programming language. Thereafter, GNU Radio Companion flow graphs are given for FM radio receiver, QPSK receiver and basic spectrum analyzer. These flow graphs include GUIs. At the end of this chapter, examples of C++ signal processing source codes are given.

B.1 Python Code Examples

In this section an example of a simple FM radio receiver is addressed. The receiver receives 320 kHz bandwidth around given center frequency (e.g. 99.9 MHz). After this, it filters, demodulates and decimates the signal and outputs it through sound card. In the following, almost line-by-line comments are given for the source code. More information about Python programming language can be found from [B1].

In the following, Listing B.1 is commented line-by-line. Line 1 tells to the GNU Radio environment that this is a Python file and lines 3-11 simply import built-in signal processing blocks and functions to be used in the implementation. Thereafter, lines 13-14 define a method which finds and chooses suitable daughter board for the receiver. Lines 17-20 start the class definition of the receiver. Lines 24-26 initialize receiver parameters. Line 32 initializes USRP source with complex-valued samples. Line 35 determines the sampling frequency of the USRP which should be 64 MHz. After this, USRP DDC decimation rate is set accordingly on lines 38-39 to get 320 kHz sample rate. The sample rate is calculated on line 42. Thereafter, needed decimation rates for sound card audio sample rate are calculated on lines 45-48. Line 51 selects proper daughter board from the USRP. On line 54 USRP receiver chain gain factor is set. Line 57 takes care of selecting USRP MUX value. In other words, it selects which I/Q modulator outputs are connected to which ADC inputs. Line 60 determines information on selected daughter board and on line 63 USRP and daughter board are asked to tune to desired center frequency. Line 66 designs lowpass filter for given parameters and line 67 generates corresponding lowpass filter

block. Line 70 initializes built-in FM demodulator block. Constant which is used as a volume control is initialized on line 73. Output to sound card is initialized on line 76. On line 79 all initialized signal processing blocks are connected together in a right order. Lines 82-87 just run the flow graph and exit if any key is pressed.

```
1  #!/usr/bin/env python
2
3  from gnuradio import gr, gru, eng_notation, optfir
4  from gnuradio import audio
5  from gnuradio import usrp
6  from gnuradio import blks2
7  from gnuradio.eng_option import eng_option
8  from optparse import OptionParser
9  from usrpm import usrp_dbid
10 import sys
11 import math
12
13 def pick_subdevice(u):
14   return usrp.pick_subdev(u, (usrp_dbid.TV_RX, usrp_dbid.TV_RX_REV_2, usrp_dbid.TV_RX_REV_3,
         usrp_dbid.TV_RX_MIMO, usrp_dbid.TV_RX_REV_2_MIMO, usrp_dbid.TV_RX_REV_3_MIMO,
         usrp_dbid.BASIC_RX))
15
16 class wfm_rx_block (gr.top_block):
17
18   def __init__(self):
19     gr.top_block.__init__(self)
20
21       ######################################################################
22       # Initialize parameters
23       ######################################################################
24       self.vol = .1
25       self.freq = 99.9e6
26       self.gain = 10
27
28       ######################################################################
29       # Build flow graph
30       ######################################################################
31       # Initialize USRP
32       self.u = usrp.source_c()                    # usrp data source
33
34       # Determine USRP sampling frequency
35       sampling_frequency = self.u.adc_rate()                # 64 MHz
36
37       # Set USRP DDC decimation rate
38       usrp_decim = 200
39       self.u.set_decim_rate(usrp_decim)
40
41       # Calculate sample rate
42       sample_rate = sampling_frequency / usrp_decim          # 320 kHz
43
44       # Calculate final audio signal sample rate
45       chanfilt_decim = 1
46       demod_rate = sample_rate / chanfilt_decim
47       audio_decimation = 10
48       audio_rate = demod_rate / audio_decimation  # 32 kHz
49
50       # Select sub-device (daughter board)
51       self.rx_subdev_spec = pick_subdevice(self.u)
52
53       # Set USRP gain
54       self.subdev.set_gain(self.gain)
55
56       # Set USRP MUX value
57       self.u.set_mux(usrp.determine_rx_mux_value(self.u, self.rx_subdev_spec))
58
59       # Determine sub-device (daughter board)
60       self.subdev = usrp.selected_subdev(self.u, options.rx_subdev_spec)
61
62       # Tune USRP to given center frequency
63       self.u.tune(0, self.subdev, target_freq)
64
65       # Lowpass filter as channel filter
66       chan_filt_coeffs = optfir.low_pass(1, sample_rate, 80e3, 115e3, 0.1, 60)
67       chan_filt = gr.fir_filter_ccf (chanfilt_decim, chan_filt_coeffs)
```

```
68
69          # Built-in FM receiver block as FM demodulator
70          self.fm_demod = blks2.wfm_rcv(demod_rate, audio_decimation)
71
72          # Simple multiplication as volume control
73          self.volume_control = gr.multiply_const_ff(self.vol)
74
75          # Output audio to sound card
76          audio_sink = audio.sink(int(audio_rate), options.audio_output, False)
77
78          # Connect different blocks together
79          self.connect(self.u, chan_filt, self.fm_demod, self.volume_control, audio_sink)
80
81
82  if __name__ == '__main__':
83      tb = wfm_rx_block()
84      try:
85          tb.run()
86      except KeyboardInterrupt:
87          pass
```

Listing B.1: Example Python code of FM receiver without GUI.

B.2 GNU Radio Companion Examples

In this section a few example flow graphs for GNU Radio Companion are given. First, FM radio receiver similar to the above Python example is presented. Thereafter, a simple OFDM transmitter is given. Then, a QPSK receiver is given without and with receiver I/Q imbalance mitigation.

First example is a FM radio receiver with GUI. The flow graph for the receiver can be seen in Figure B.1. The flow graph consists of three variable sliders, one variable, USRP source, WBFM receiver, constant multiplication, audio sink and FFT sink. Variable sliders give an ability to modify program parameters during the execution. In this example we have variable slider for coarse frequency, fine frequency tuning and volume control. Moreover, we have variable for decimation with decimation factor 200 which yields 320 kHz sample rate for received signal. "USRP Source" block is an interface to the physical device and it control USRP behavior. "WBFM Receive" block is a quadrature FM demodulator block which takes received complex samples as an input and outputs corresponding received audio signal. "Multiply Const" block is used as a volume control for the received audio stream. Finally, the "Audio Sink" block outputs the audio signal to a sound card installed to a PC. "FFT Sink" block simply plots frequency spectrum of the received signal.

The example found in Figure B.2 is a flow graph for a simple OFDM transmitter. The flow graph consists of random data source, OFDM modulator, gain controller, FFT plotter and USRP sink. As is known, the DAC sampling frequency of the USRP is 128 MHz and with interpolation factor 32 we get 4 MHz output bandwidth. The "Random Source" block generates random byte sequence for the "OFDM Mod" block. In this example "OFDM Mod" block uses QPSK as a sub-carrier modulation

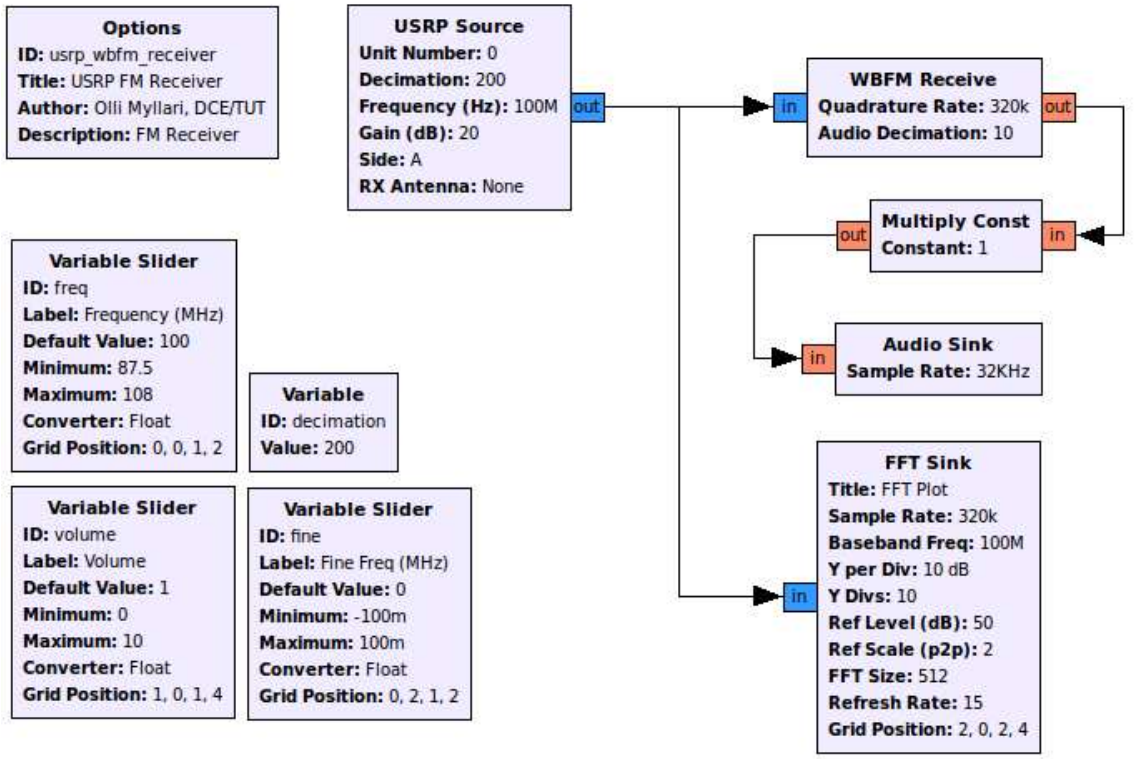


Figure B.1: GNU Radio Companion flow graph of FM radio receiver.

and FFT length is 1024 with 800 occupied sub-carriers. Cyclic prefix length is 128. Pad for USRP field means that if input data is too short, the output USB packages are padded with random data to maintain continuous data flow. "AGC2" block is a output gain controller block which scales digital amplitude of the signal. This is only output power control with RFX2400 daughter board because it does not have separate analog variable gain controller. "USRP Sink" acts as a interface between GNU Radio and USRP transmitter chain. "FFT Sink" just shows digital output spectrum of the signal.

Next example found in Figure B.3 is a basic QPSK receiver for original USRP. This example has a number of variables with and without tuning sliders. The receiver is designed for QPSK signal with 2 MHz symbol rate and 2 times over-sampling is used, thus sampling frequency of the received signal is 4 MHz. RRC receiver filter with 30 % roll-off factor is used. The "Decimating FIR Filter" block realizes the RRC filter. "MPSK Receiver" block consists of a Costas loop for frequency synchronization and a Müller & Muller algorithm for symbol timing recovery. Multiplication with a complex constant is used to make the constellation look like $\pi/4$-QPSK. "AGC2" block scales symbols to original amplitude and "Scope Sink" block shows the received constelaltion. For constellation plot the "XY Mode" of the "Scope Sink" has to be turned on.

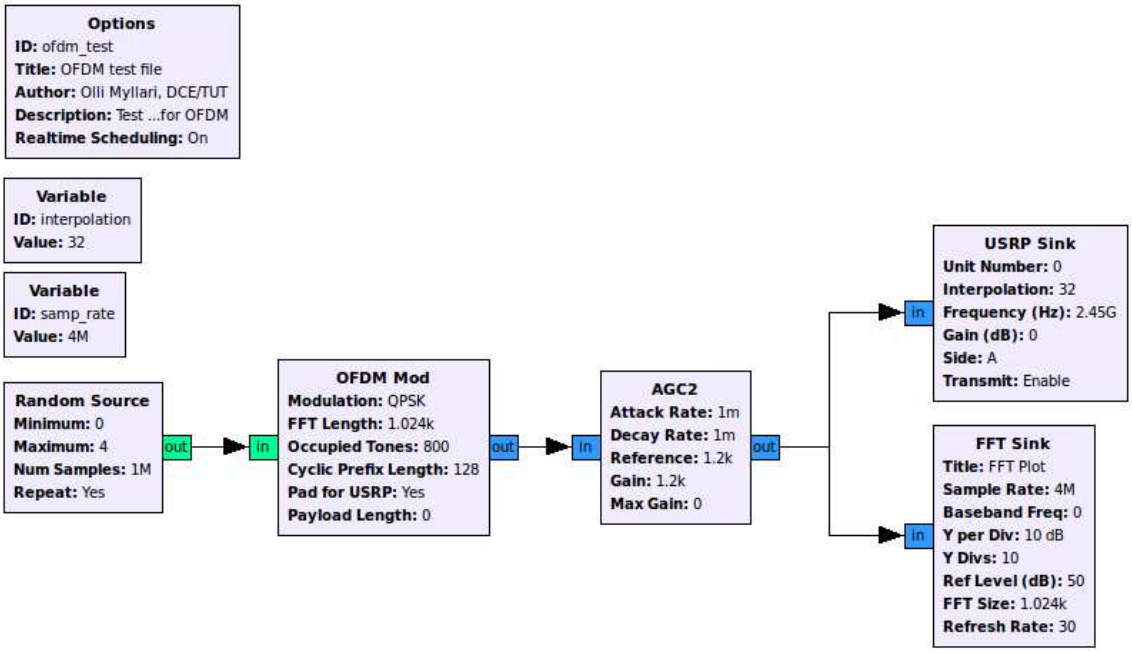


Figure B.2: Flow graph of simple OFDM transmitter for USRP.

Next example found in Figure B.4 is similar to the above QPSK receiver example. Difference is that this example flow graph is utilizing receiver I/Q imbalance mitigation algorithm found in [B2]. In the following, only blocks different from above QPSK receiver example are considered. "Rx I/Q Imbalance mitigation" block is programmed following the paper [B2]. It mitigates I/Q imbalance with circularity-based algorithm which is realized with LMS-style recursion. "Frequency Xlating FIR Filter" block converts the desired complex signal to the baseband, low-pass filters and decimates it. All remaining blocks function just like in above QPSK receiver example.

Last GNU Radio Companion example is a simple spectrum analyzer. This example does not have any new signal processing blocks, thereby they will not be separately examined. The flow graph can be found in Figure B.5 and screen shot of the running spectrum analyzer program is in Figure B.6.

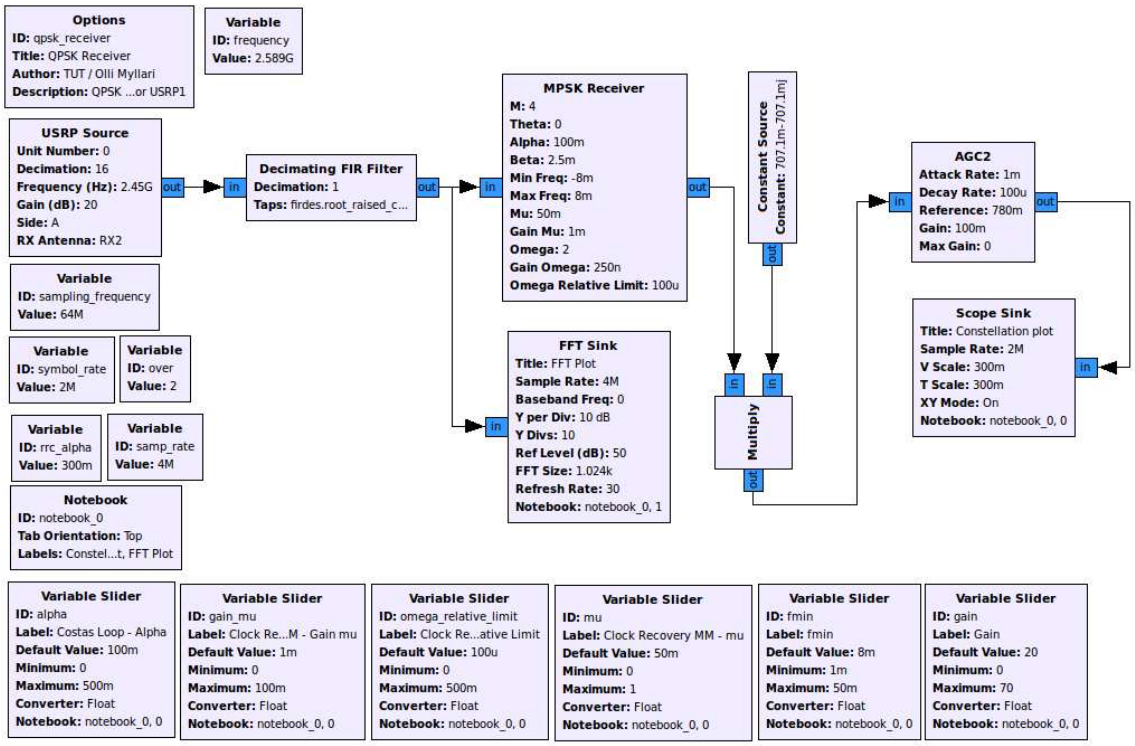


Figure B.3: Flow graph of QPSK receiver for USRP.

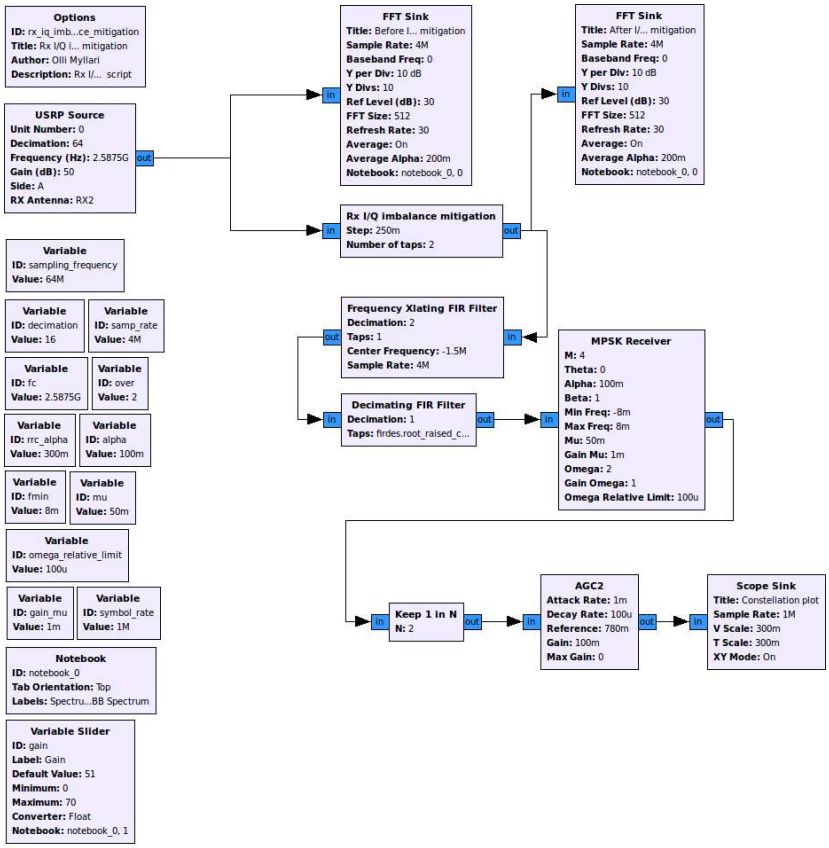


Figure B.4: Flow graph of QPSK receiver for USRP with I/Q imbalance mitigation block.

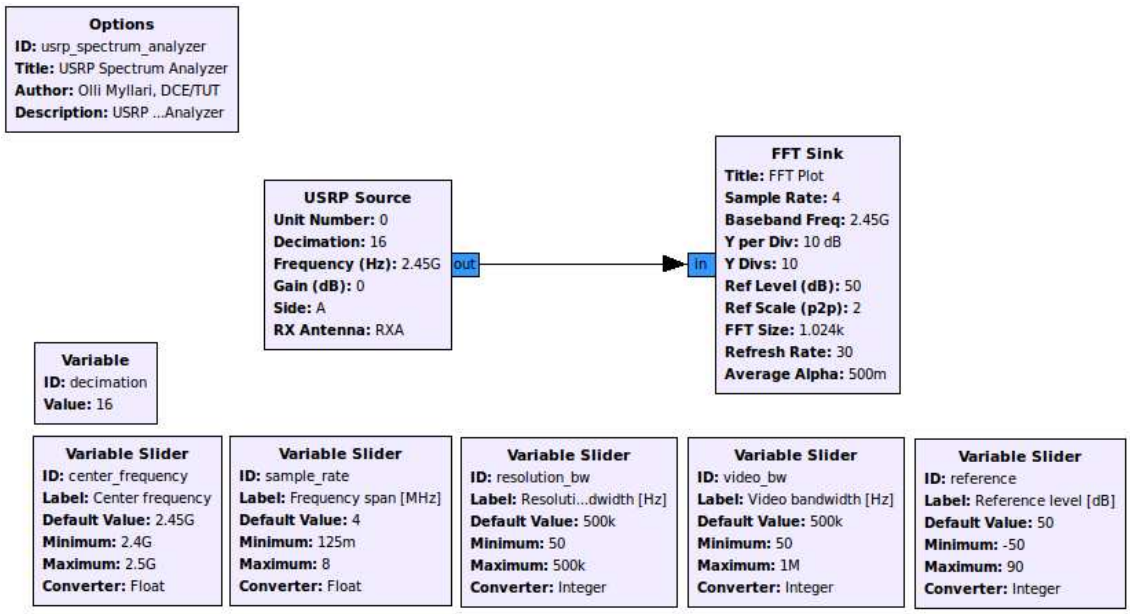


Figure B.5: GNU Radio Companion flow graph of a simple spectrum analyzer.

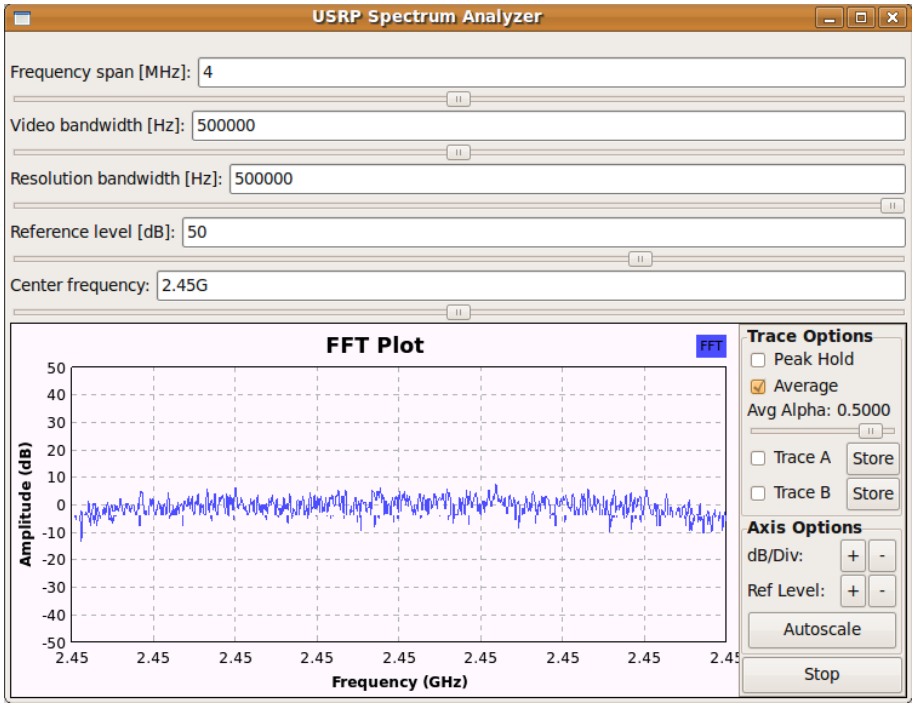


Figure B.6: GNU Radio GUI when the spectrum analyzer is running.

B.3 C++ Code Examples

In this section few C++ source files and corresponding supporting files are examined. When new GNU Radio signal processing block is developed it always needs a header file, a source file and references to them in "Makefile.am". In addition, interface to the signal processing block has to be introduced in a SWIG file which is in our case "tut.i". It should be noted, that only algorithm specific code lines are commented because C++ programming language specific issues are outside the scope of the thesis. More information about C++ programming language and object-oriented programming can be found e.g. in [B3, B4].

An example of SWIG interface file is found in Listing B.2. It introduces an interface to the signal processing block which is used for pre-distortion with transmitter I/Q imbalance mitigation. Most of the lines from 1 to 10 are common for all signal processing blocks. Line 8 is an exception because it points out the header file name for the interface file. Line 13 is a GNU Radio function which makes the current signal processing block to be a part of "tut" -class. In other words, the signal processing block becomes accessible from Python application with command tut.tx_iq_imb_mitig_predist(unsigned int n_taps, std::vector<gr_complex> taps, bool debug). Lines 15-23 simply tell what kind of interface the signal processing block has. It is completely same as the actual header file, thus it will be examined later in this section.

```
1  /* -*- c++ -*- */
2
3  %include "exception.i"
4  %import "gnuradio.i"
5
6  %{
7  #include "gnuradio_swig_bug_workaround.h"       // mandatory bug fix
8  #include "tut_tx_iq_imb_mitig_predist_cc.h"
9  #include <stdexcept>
10 %}
11
12 // Tx I/Q imbalance pre-distortion
13 GR_SWIG_BLOCK_MAGIC(tut,tx_iq_imb_mitig_predist_cc);
14
15 tut_tx_iq_imb_mitig_predist_cc_sptr tut_make_tx_iq_imb_mitig_predist_cc (unsigned int n_taps,
       std::vector<gr_complex> taps, bool debug);
16
17 class tut_tx_iq_imb_mitig_predist_cc : public gr_block {
18   private:
19     tut_tx_iq_imb_mitig_predist_cc (unsigned int n_taps, std::vector<gr_complex> taps, bool
         debug);
20
21   public:
22     void set_taps(std::vector<gr_complex> taps);
23 };
```

Listing B.2: Example interface file of signal processing block which performs pre-distortion filtering.

Next we go through and example of "Makefile.am" file found in Listing B.3. This file is used as an input for Linux automake which is used when signal processing

blocks are installed to a computer. Practically on lines 4, 12 and 23 have to be edited. Line 4 gives names of the header files to be included. The names can be separated on multiple lines with back-slash. Line 12 is for the name of the SWIG interface file introduced above. And line 23 similar to line 4 but it is used to declare C++ source files.

```
1  include $(top_srcdir)/Makefile.common
2
3  # C/C++ headers get installed in ${prefix}/include/gnuradio
4  grinclude_HEADERS =                \
5    tut_tx_iq_imb_mitig_predist_cc.h
6
7  ######################################
8  # SWIG Python interface and library
9  ######################################
10
11 TOP_SWIG_IFILES =          \
12   tut.i
13
14
15 # Install so that they end up available as: import gnuradio.tut
16 # This ends up at:
17 # ${prefix}/lib/python${python_version}/site-packages/gnuradio
18 tut_pythondir_category =       \
19   gnuradio
20
21
22 # additional sources for the SWIG-generated library
23 tut_la_swig_sources =              \
24   tut_tx_iq_imb_mitig_predist_cc.cc
25
26 # additional LD flags for linking the SWIG-generated library
27 tut_la_swig_ldflags = \
28   $(AVCODEC_LIBS) \
29   $(AVFORMAT_LIBS) \
30   -outdir $(builddir)
31
32 include $(top_srcdir)/Makefile.swig
33
34 # add some of the variables generated inside the Makefile.swig.gen
35 BUILT_SOURCES = $(swig_built_sources)
36
37 # Do not distribute the output of SWIG
38 no_dist_files = $(swig_built_sources)
```

Listing B.3: Example Makefile.am file of signal processing block which performs pre-distortion filtering.

In Listing B.4 we have a basic example of header file for GNU Radio signal processing block. The header file is programmed for transmitter I/Q imbalance pre-distortion. Lines 1-2 define are standard class definitions. Line 4 includes GNU Radio library for synchronous signal processing block. This means that the block assumes number of input samples to be same as number of output samples. Line 5 includes GNU Radio library for input/output functionality. In practice, this library controls the signal flow through the block. Line 6 includes GNU Radio library for complex exponential function. Thereafter, lines 6 and 7 include GNU Radio libraries for mathematical functions and complex arithmetics. Line 9 includes C++ standard library for complex arithmetics. Line 11 includes GNU Radio library for FIR filter with complex input and output, and complex filter coefficients. Then, line 23 includes GNU Radio library for FIR filter design tools. Lines 17 and 19 define interface to boost dynamic pointer to the class. It is used while interfacing the C++ signal processing block to

Python flow graphs. Lines 23-25 define private class variables for the block. Thereafter, on line 27 is a interface to overridden method forecast(). This is used because we want to control the number of needed input samples. Line 28 is directly related to lines 17 and 19. It defines that the boost dynamic pointer is safe in private side of the class. Line 29 defines the interface for the class initialization. Line 32 defines interface for the class destructor and line 33 for a public callback method set_taps() which is used to set the pre-distortion filter coefficients. Line 34 defines interface to the general_work method of the signal processing block. All signal processing is performed in the general_work method.

```
1  #ifndef INCLUDED_TUT_TX_IQ_IMB_MITIG_PREDIST_CC_H
2  #define INCLUDED_TUT_TX_IQ_IMB_MITIG_PREDIST_CC_H
3
4  #include <gr_sync_block.h>
5  #include <gr_io_signature.h>
6  #include <gr_expj.h>
7  #include <gr_math.h>
8  #include <gr_complex.h>
9  #include <complex>
10 #include <vector>
11 #include <gr_fir_ccc.h>
12 #include <gr_fir_util.h>
13
14 class tut_tx_iq_imb_mitig_cc;
15
16 // Shared boost pointer
17 typedef boost::shared_ptr<tut_tx_iq_imb_mitig_cc> tut_tx_iq_imb_mitig_cc_sptr;
18
19 tut_tx_iq_imb_mitig_cc_sptr tut_make_tx_iq_imb_mitig_cc(unsigned int n_taps,
       std::vector<gr_complex> taps, bool debug);
20
21 class tut_tx_iq_imb_mitig_cc : public gr_block {
22   private:
23     unsigned int d_n_taps;
24     std::vector<gr_complex> d_taps;
25     gr_fir_ccc *d_composite_fir;
26
27     void forecast(int noutput_items, gr_vector_int &input_items_required);
28     friend tut_tx_iq_imb_mitig_cc_sptr tut_make_tx_iq_imb_mitig_cc(unsigned int n_taps,
           std::vector<gr_complex> taps, bool debug);
29     tut_tx_iq_imb_mitig_cc(unsigned int n_taps, std::vector<gr_complex> taps, bool debug);
30
31   public:
32     ~tut_tx_iq_imb_mitig_cc();
33     void set_taps(std::vector<gr_complex> start_estimation);
34     int general_work(int noutput_items, gr_vector_int &ninput_items, gr_vector_const_void_star
           &input_items, gr_vector_void_star &output_items);
35 };
36
37 #endif
```

Listing B.4: Example header file of signal processing block which performs pre-distortion filtering.

In Listing B.5 we have a basic example of C++ source file corresponding to the header file found in Listing B.4. Line 5 includes the header file from Listing B.4 to the source file. Line 7 includes C++ standard library for printing to command prompt and line 8 to be able to use string variables. Lines 12-14 realize the method defined on line 19 in Listing B.4. It just returns the boost pointer to the signal processing block. On lines 18-24 is the private class constructor of the block. On the line 18 "gr_make_io_signature(2, 2, sizeof(gr_complex))" defines minimum

and maximum number of input streams, and size of on sample in the stream. Similarly, "gr_make_io_signature(1, 1, sizeof(gr_complex))" defines numbers of output streams. Line 21 initializes FIR filter used for the pre-distortion. Line 23 sets length of the history for the block. Lines 26-28 realize the class destructor which releases the memory used for the FIR filter. Lines 30-34 realize the overridden method forecast() which was discussed in Section A.5.1. Lines 36-39 realize the callback method used to set pre-distortion filter coefficients. Lines 41-53 realize the general_work method of the block. The actual signal processing is done here. Lines 42-44 define pointers to input and output streams. Thereafter, lines 47-49 perform pre-distortion for current data block. Line 51 defines how many samples will be consumed from each input streams. Finally, line 52 reports to the scheduler how many samples were produced to the output stream.

```
1  #ifdef HAVE_CONFIG_H
2  #include "config.h"
3  #endif
4
5  #include <tut_tx_iq_imb_mitig_predist_cc.h>
6  #include <gr_io_signature.h>
7  #include <iostream>
8  #include <string.h>
9  #include <vector>
10 #include <gr_math.h>
11
12 tut_tx_iq_imb_mitig_predist_cc_sptr tut_make_tx_iq_imb_mitig_predist_cc(unsigned int n_taps,
       std::vector<gr_complex> taps, bool debug) {
13   return tut_tx_iq_imb_mitig_predist_cc_sptr(new tut_tx_iq_imb_mitig_predist_cc(n_taps, taps,
         debug));
14 }
15
16
17 // Private constructor
18 tut_tx_iq_imb_mitig_predist_cc::tut_tx_iq_imb_mitig_predist_cc(unsigned int n_taps,
       std::vector<gr_complex> taps, bool debug): gr_block ("tx_iq_imb_mitig_predist_cc",
       gr_make_io_signature(2, 2, sizeof(gr_complex)), gr_make_io_signature(1, 1,
       sizeof(gr_complex))){
19   d_n_taps = n_taps;
20   d_taps = taps;
21   d_composite_fir = gr_fir_util::create_gr_fir_ccc(d_taps);
22
23   set_history(d_n_taps);
24 }
25
26 tut_tx_iq_imb_mitig_predist_cc::~tut_tx_iq_imb_mitig_predist_cc() {
27   delete d_composite_fir;
28 }
29
30 void tut_tx_iq_imb_mitig_predist_cc::forecast(int noutput_items, gr_vector_int
       &ninput_items_required) {
31   for (unsigned int n = 0; n < ninput_items_required.size(); n++) {
32     ninput_items_required[n] = 1024;
33   }
34 }
35
36 void tut_tx_iq_imb_mitig_predist_cc::set_taps(std::vector<gr_complex> taps) {
37    d_taps = taps;
38    d_composite_fir->set_taps(d_taps);
39 }
40
41 int tut_tx_iq_imb_mitig_predist_cc::general_work(int noutput_items, gr_vector_int
       &ninput_items, gr_vector_const_void_star &input_items, gr_vector_void_star &output_items) {
42   const gr_complex *d_if_signal = (const gr_complex *) input_items[0];
43   const gr_complex *d_conj_if_signal = (const gr_complex *) input_items[1];
44   gr_complex *out = (gr_complex *) output_items[0];
45
46   // Perform pre-distortion filtering
47   for (int n = 0; n < noutput_items; n++) {
48     out[n] = d_if_signal[n] + d_composite_fir->filter(&d_conj_if_signal[n]);
49   }
```

```
50
51     consume_each(noutput_items);
52     return noutput_items;
53   }
```

Listing B.5: Example source file of signal processing block which performs pre-distortion filtering.

References

[*B*1] J. Kasurinen, *Python - ohjelmointiopas*, Lappeenrannan teknillinen yliopisto, 2007.

[*B*2] L. Anttila, M. Valkama, M. Renfors, "Circularity-based I/Q imbalance compensation in wideband direct-conversion receivers ," *IEEE Transactions on Vehicular Technology*, vol. 57, iss. 4, pp. 2099-2113, 2008.

[*B*3] B. Eckel, *Thinking in C++: Introduction to Standard C++*, Prentice Hall Inc., 2000.

[*B*4] R. Lafore, *Object-Oriented Programming in C++*, Sams Publishing, 2002.

Appendix C. USRP2 FIRMWARE UPDATE PROCEDURE

USRP2 firmware update procedure is quite tricky. First of all, the SD card does NOT contain a file system, and it CANNOT be read from or wrote to using normal tools. To perform firmware update an SD card programmer is needed. Any available SD memory card reader is suitable for the procedure. In the following, the steps to upgrade the firmware are briefly described. All firmware versions of USRP2 can be found from http://gnuradio.org/releases/usrp2-bin/trunk/.

C.1 Building tools

First the firmware and u2_flash_tool have to be build by running make in the gnuradio/usrp2 directory. The directory is found from GNU Radio installation directory. After the build is completed USB SD memory card reader is connected to the computer. After this the mounting point of the SD memory card reader has to be determined. To perform this, additional software called sg3-utils is installed. This is extremely important because if the mounting point used in the flashing phase is wrong, the system disk can be overwritten.

C.2 Installing sg3-utils

To install sg3-utils following commands have to be run in the terminal window

sudo apt-get update
sudo apt-get install sg3-utils

After this, command

sudo sg_scan -i

runs additional software to generate listing of all raw SCSI devices on the current system. From the listing needs to be determined which SCSI device is most likely the USB SD memory card reader attached to the computer. Finally, following command has to be run in the terminal windows to discover which SCSI device is associated with USB SD memory card reader

sg_map

This device could for example be /dev/sdb but it changes from time to time.

C.3 Flashing firmware and FPGA code

After determining SCSI device you have to go to gnuradio/usrp2 directory where you have to run command

sudo u2_flash_tool –dev=/dev/XXXX -t s/w usrp2/firmware/txrx.bin -w

in the terminal, where XXXX is the mounting point of the memory card reader. After this also the FPGA code has to be updated. This happens by running command

sudo u2_flash_tool –dev=/dev/XXXX -t fpga u2_rev3.bin -w

in the terminal, where, again, XXXX is the mounting point of the memory card reader. Now the firmware has been updated and the SD memory card can be inserted to USRP2 memory card slot.