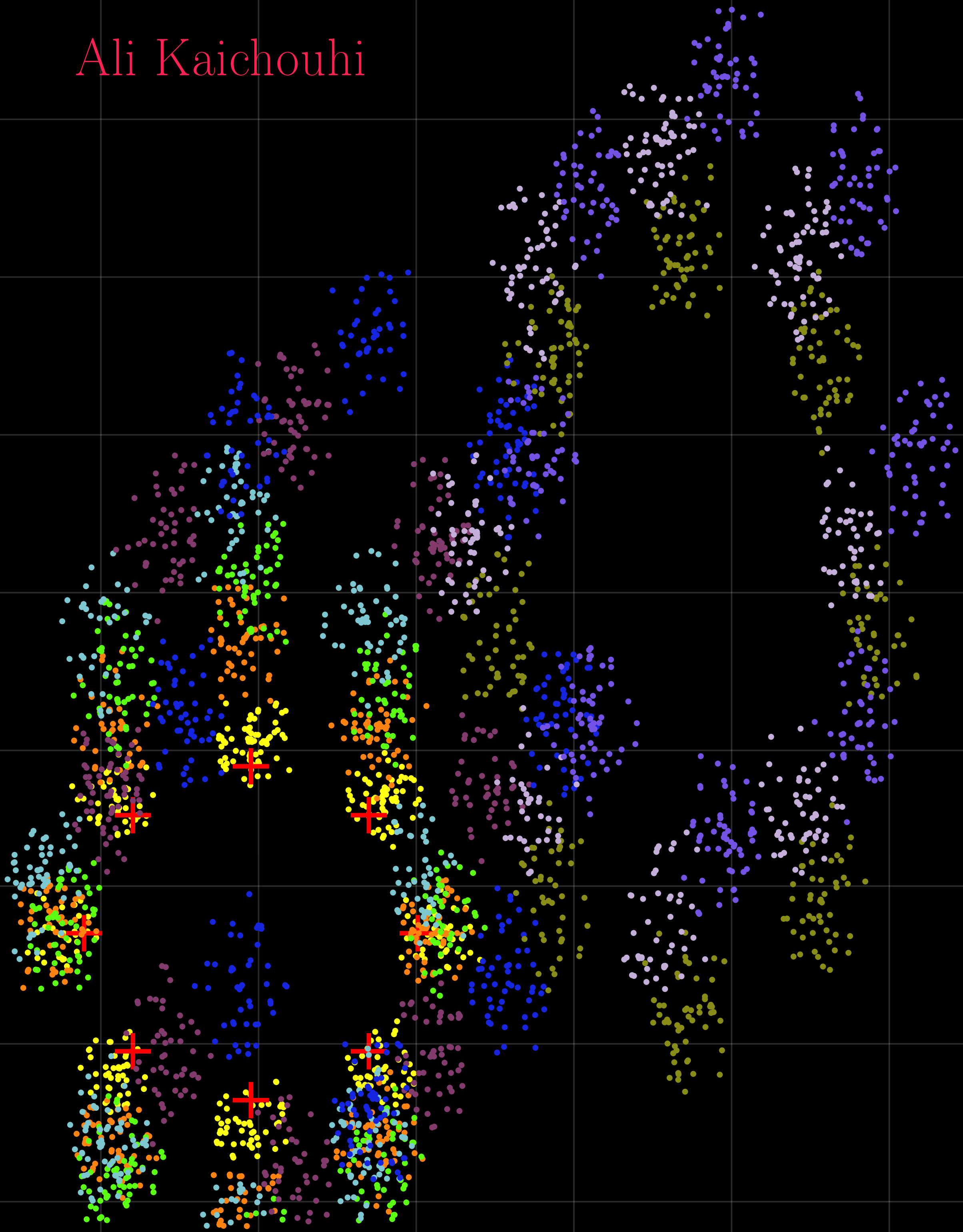
IQ offset and gain mismatch compensation for phase-domain ADC

Ali Kaichouhi



Delft University of Technology

IQ offset and gain mismatch compensation for phase-domain ADC

by

Ali Kaichouhi

A thesis Submitted to
The Faculty of Electrical Engineering, Mathematics and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Electrical Engineering
Track Microelectronics

Supervisor: Prof. Dr. ir. Wouter A. Serdijn

Thesis committee

Prof.dr.ir. W.A. Serdijn
Prof.dr. E. McCune
Dr.ir. G.J.M. Janssen
Dr. Y. Liu

TU Delft

Preface

The purpose of this study was to design and implement the IQ offset and gain mismatch compensation for a phase domain ADC (PhADC) on a chip and make the whole suitable for Bluetooth Low Energy.
This report is limited in scope to show the concept of IQ offset and gain mismatch compensation for PhADC and to make a circuit level design. I believe that this work will be found complete and self-sufficient by the student who undertakes the actual implementation on a chip. Most of the circuit parts are fully combinatorial logic which are modeled in an analog behavior description language VerilogAMS. The models can be simply converted into a *Gate Level Netlist* and synthesized to schematic level with the *Cadence Encounter RTL (register-transfer level) compiler*. The cells can be placed&routed in *Cadence Innovus* to make a layout. Cadence IC designkits nowadays provide complete design flows from behavior modeling to layout synthesis. The PhADC receiver has been designed in the AMS hitkit CMOS H18 technologie. The circuit parts of the IQ offset and gain mismatch control loop have been designed and analyzed in the same technology on a schematic circuit level.
First of all, I want to thank my supervisor, Prof.dr.ir. W.A. Serdijn, for his professional and intelligent guidance. Always a good approach, it is nice to work with him. As a person I thank him for his patience and listening ear. I would also like to thank, Dr. Yao Liu, for our many meetings and support that contributed a considerable part to the realization of this report. Then also Prof.dr. E. McCune, the expert, a person who can bring the theory and practice together. I want to thank him for the fresh understanding of wireless telecommunication techniques. And Dr.ir. G.J.M. Janssen for his contribution that helped me on the right track to do this research properly.
And not to mention, I would like to thank my family for their patience and support during my graduation. And finally, I want to thank the students of the bio-electronics group for their tips and explanations.

Ali Kaichouhi
Delft, May 2018

List of Figures

List of Tables

List of Symbols

ω_{RF}	Information signal RF - radians per second	1
ω_{LO}	Information signal RF - radians per second	1
Δ_{φ}	Phase quantization noise - degrees	2
Δi	Amplitiude quantization noise in I signal - volt	2
Δq	Amplitiude quantization noise in Q signal - volt	2
I	The in-phase baseband signal - volt	2
Q	The quadrature baseband signal - volt	3
$QPSK(t)$	Quadrature modulated signal - volt	4
φ	The phase of signal vector - degrees	6
lsb	Least significant bit	6
v_{os}	Voltage offset - volt	7
FS_I	Full scale range of I - volt	8
FS_Q	Full scale range of Q - volt	8
INL_{max}	Maximum Integral Nonlinearity	8
α	Mismatch factor between fullscale range FS_I and FS_Q	8
$v_{os,I}$	I offset - volt	10
$v_{os,Q}$	Q offset - volt	10
RHP	Number of signal symbols in the right half of IQ complex plane	11
LHP	Number of signal symbols in the left half of IQ complex plane	11
N	Total number of symbols	11
$IMB_{os,I}$	Vector distribution imbalance in case of an I offset	11
THP	Number of signal symbols in the top half of IQ complex plane	11
BHP	Number of signal symbols in the bottom half of IQ complex plane	11
$IMB_{os,Q}$	Vector distribution imbalance in case of an Q offset	11
v_{os,I_f}	Feedback I offset signal - volt	12
$v_{osd,I}$	IQ offset detector output for I signal - volt	15
IMB_I	Phase vector distribution imbalance in I	18
v_{osd,I_f}	IQ offset detector output for I - volt	18
IMB_Q	Phase vector distribution imbalance in Q	18
$v_{osd,Q}$	IQ offset detector output for Q - volt	18
$v_{tqd,I}$	Two-quadrant Detector output for I - volt	23
$v_{tqd,Q}$	Two-quadrant Detector output for Q - volt	23
BER	Bit-error rate	24
$\lvert u \rvert$	Magnitude of signal vector u - volt	27
$\angle u$	Phase of signal vector u - degrees	27
EVM	Error vector magnitude	31
e_k	Error vector - volt	31
$\tilde{I}_k$	In-phase measurement of the kth symbol in the burst - volt	31
$\tilde{Q}_k$	Quadrature phase measurement of the kth symbol in the burst - volt	31
P_{avg}	The value for *Average constellation power* - watt	31
EVM_{rms}	RMS magnitude of error vectors normalized to the average signal power	31
MER	Modulation Error Ratio	31
SNR	Signal-to-noise power ratio	33
E_b	Energy per bit - joule per bit	34
E_s	Energy per symbol - joule per symbol	34
N_o	Noise power spectral density - watt per Hertz	34
P_s	Total signal Power - watt	34
P_n	Noise power in the signal bandwidth - watt	34
f_b	Bit rate - bits per second	34

f_s	Symbol rate - baud	34
T_b	Bit time - seconds	34
T_s	Symbol time - seconds	34
M	The number of states available to the signal	34
T_{sym}	Signal symbol period - seconds	35
T_{samp}	Signal sampling period - seconds	35
k	The number of information bits per symbol	35
$V_{Q(rms)}$	Phase quantization noise signal - volt	39
$P_{w,1\Omega}$	Signal power normalized to 1 Ω - watt	41
$IQ_{\_offset}$	Added IQ offset in Simulink model	43
$SNR_{\varphi,ph}$	Phase SNR of an PhADC	46
$SNR_{am,ph}$	Amplitude SNR of an IQ ADC	46
SNR_{prec}	Preceding SNR without considering the noise of PhADC	46
$p(f)$	Frequency spectrum of a rectangular pulse	49
$p(t)$	Time domain signal of a rectangular pulse	49
s	Complex frequency	55
ξ	The damping ratio	60
ω_n	Undamped natural frequency - hertz	60
H_{ss}	State-space filter model	67
H_z	Discrete filter model	67
zoh	Zero-order hold	67
IMB_{mis}	Phase vector distribution imbalance due to IQ gain mismatch	75
(1)	Region between $\frac{\pi}{4}$ and $\frac{3}{4}\pi$ in the IQ complex plane	76
(3)	Region between $-\frac{\pi}{4}$ and $-\frac{3}{4}\pi$ in the IQ complex plane	76
(2)	Region between $\pm\frac{3}{4}\pi$ in the IQ complex plane	76
(4)	Region between $\pm\frac{\pi}{4}$ in the IQ complex plane	76
A_I	Gain of I signal	76
A_Q	Gain of Q signal	76
α_f	IQ gain feedback mismatch factor	76
$v_{trd_{12}}$	Two Regions Detector output for phase vector distribution imbalance signal between Regions (1) and (2) - volt	87
$v_{trd_{14}}$	Two Regions Detector output for phase vector distribution imbalance signal between Regions (1) and (4) - volt	87
$v_{trd_{32}}$	Two Regions Detector output for phase vector distribution imbalance signal between Regions (3) and (2) - volt	87
$v_{trd_{34}}$	Two Regions Detector output for phase vector distribution imbalance signal between Regions (3) and (4) - volt	87
C_{pump}	Charge pump capacitor - Farad	103
G_m	Transconductance amplifier - ampere per volt	104
I_{comp}	Compensating current source - ampere	106
I_{nom}	Nominal current source - ampere	106
v_{in,I_p}	Positive half of input signal I - volt	106
V_{in,I_n}	Negative half of input signal I - volt	106
v_{out,I_p}	Positive half of output signal I - volt	106
v_{out,I_n}	Negative half of output signal I - volt	107
σ_{PFE+BB}	DC offset standard deviation of passive front-end, base-band LNA and analog baseband - volt	108
σ_{ph}	DC offset standard deviation of PhADC - volt	109
$\Delta V_{S,TH_{max}}$	Maximum DC offset at the T&H transistor source node - volt	109
$g_{m,TH}$	Small signal transconductance of the track-and-hold pMOSFET transistor - ampere per volt	110
$V_{S,TH}$	Track-and-hold transistor source voltage - volt	110
I_{MSB}	Most significant current from current-steering DAC - ampere	110
MSB	Most significant bit	110
R_{on}	PMOS switch on resistance - ohm	112
R_{off}	PMOS switch off resistance - ohm	112

DNL	Differential nonlinearity	113
DNL_n	Differential nonlinearity at n digital input transistion	113
INL	Integral nonlinearity	115
INL_n	Integral nonlinearity at n output transition	115
Dynamic Range (DR)	Dynamic range	115
SQNR	Signal-to-quantization noise ratio	115
$AVDD$	Voltage source for analog circuits - volt	117
R_d	PGA degeneration resistor - ohm	122
R_o	PGA output resistor - ohm	122
$R_{o_{fixed}}$	Fixed part of programmable PGA output resistor - ohm	123
$R_{o_{ctrl}}$	Programmable part of PGA output resistor	123
W	Binary to decimal converted value	123
R_{step}	PGA programmable output resistor step value - ohm	123
$sel < N - i >$	(N-i)th bit of programmable PGA output resistor	123
N	Number of bits of the Up/Down counter	123
R_f	LNA feedback resistor - ohm	126
C_F	LNA feedback capacitor - ohm	126
f_{HP}	The high pass 3dB point of bandpass LNA - Hertz	126
$DVDD$	Supply voltage of digital circuits - volt	127
$VPWL$	Votage piece-wise linear source - volt	127
g_m	Transconductance of T$H transistor - ampere per volt	132
r_{ds}	Drain-source mosfet impedance - ohm	132
W/L	Mosfet width and length dimensions - meter	133
V_{SG}	Source-gate voltage - volt	134
V_{SD}	PMOS source-drain voltage - volt	134
t_r	Input rise time - seconds	134
t_f	Input fall time - seconds	134
t_{LH}	Output rise time - seconds	134
t_{HL}	Output fall time - seconds	134
t_{PLH}	Propagation delay when output changes from low to high - seconds	134
V_{OH}	Output high threshold voltage - volt	134
V_{OL}	Output low threshold voltage - volt	134
C_{ox}	Oxide capacitance - Farad	135
C_{inn}	Gate-source capacitance of simple digital MOSFET model - Farad	135
C_{outp}	Output capacitance of digital MOSFET model - Farad	135
R_p	Effective switching resistance of the short-channel PMOS device - ohm	135
T_{sw_0}	High active PMOS current-steering DAC switch transistor for LSB current source	135
$T_{sw_{2^{N-1}-2}}$	High active PMOS current-steering DAC switch transistor for $2^{N-1} - 2$ current source	135
$T_{\overline{sw}_0}$	Low active PMOS current-steering DAC switch transistor for LSB current source	135
$T_{\overline{sw}_{2^{N-1}-2}}$	Low active PMOS current-steering DAC switch transistor for $2^{N-1} - 2$ current source	135
$C_{ox}^{'}$	Gate oxide capacitance per area - Farad per square meter	135
R_n	Effective switching resistance of the short-channel NMOS device - ohm	135
$C_{ox,p}$	Oxide capacitance for a W/L PMOS transistor - Farad	135
$C_{ox,n}$	Oxide capacitance for a W/L NMOS transistor - Farad	135
τ_p	Intrinsic switching speed of the PMOS transistor - seconds	136
τ_n	Intrinsic switching speed of the NMOS transistor - seconds	136
t_{delay}	Delay time of a simple RC circuit - seconds	136
t_{rise}	Rise or fall time of a simple RC circuit - seconds	136
C_{tot}	The total capacitance, including the load capacitance of a digital MOSFET switch model - Farad	136
C_L	Capacitance from drain of the MOSFET to ground - Farad	136
V_{SP}	Switching point of an inverter - volt	138
W_n	NMOS transistor width - meter	138
W_p	PMOS transistor width - meter	138

C_{in}	Effective inverter input capacitance - Farad	138
C_{out}	Effective inverter output capacitance - Farad	138

Acronyms

μC Microcontroller.

ADC Analog to digital converter.

ASK Amplitude shift keying.

atan inverse tangent function.

AWGN Added white Gaussian noise.

BB Base band.

BER Bit error rate.

BERTool Bit error rate tool.

Bip-NRZ Bipolar nonreturn-to-zero.

biquad biquadratic.

DFILT Discrete-time filter.

DNL Differential non-linearity.

DQPSK Quadrature phase shift keying.

DWC Digital wireless communication.

EE Electrical engineering.

ENOB Effective number of bits.

EVM Error vector magnitude.

FIR Finite impulse response filter.

FSK Frequency shift keying.

IF Intermediate frequency.

IIR Infinte impulse response filter.

INL Integral non-linearity.

IQ In-phase and quadrature.

ISI Intersymbol interference.

LNA Low noise amplifier.

LO Local oscillator.

LSB Least significant bit.

LTI Linear time invariant.

MIMO	Multi-input multi-output.
MSB	Most significant bit.
MUX	Multiplexer.
PFE	Passive front-end.
PGA	Programmable gain amplifier.
PhADC	Phase domain analog-to-digital converter.
PSK	Phase shift keying.
QPSK	Quadrature phase shift keying.
RC	Raised cosine.
REM	Modulation error ration.
RF	Radio frequency.
RMS	Root mean square.
RTL	Register-transfer level.
SNDR	Signal-to-noise and distortion ratio.
SNR	Signal-to-noise power ratio.
SQNR	Signal-to-quantization noise ratio.
SRRC	Square root raised cosine.
T_IFS	Inter Framespace.
ZOH	Zero-order-hold.
zpk	zero-pole-gain model.

Abstract

A phase-domain analog-to-digital converter (PhADC) is a promising alternative to a pair of amplitude-domain in-phase and quadrature (IQ) ADCs for low power FSK/PSK demodulation, but due to the nonlinear amplitude-to-phase conversion, IQ offsets and gain mismatch can produce nonlinear phase distortions which that can lead to phase errors and an increase in bit-error rate (BER).

An IQ offset and gain mismatch detection technique is developed for PhADCs and verified. The offset and gain mismatch is estimated by detecting the phase vector distribution imbalance among the four phase quadrants after the analog-to-phase conversion. A feedback path has been constructed between the PhADC output and the offset/mismatch compensation interface. The closed loop will help the compensation interface to settle to the proper compensation values of the offset and mismatch. Incorporating this cancellation loop in a receiver system can improve its sensitivity and robustness.

In a conventional IQ ADC receiver we have two quantizations, i.e., one for I and one for Q, and then process the information to extract the phase. By contrast, the PhADC, due to its embedded demodulation attribute, performs only phase quantization. So only one quantization is needed. Because of its compactness, the hardware is simpler and thereby consumes less energy. Moreover the PhADC is immune to magnitude variations, because in an IQ ADC, amplitude quantization noise produces larger phase quantization noise at small vector magnitudes, while in a PhADC a larger vector has the same phase quantization error as a small vector.

Because of these advantages the PhADC is a very good candidate for low power communication. A detection algorithm has been developed that detects the phase vector distribution imbalance between the left and right IQ complex half plane in case of I offset and between the top and bottom of IQ complex half plane in case of Q offset. Using this imbalance we can determine the sign and size of IQ offset. A similar approach has been done for IQ gain mismatch, where the phase vector distribution imbalance is detected between the four phase quadrants in the IQ complex plane. A mixed-signal approach, i.e., detecting the offset and mismatch in the digital domain and compensate in the analog domain.

It is well known that in the IQ ADC receiver the offset and mismatch can be detected and calibrated if necessary before the amplitude is converted into phase in the digital domain, but in the PhADC receiver the offset and mismatch cannot be determined directly due to the absence of amplitude information. Here, we use phase information rather than amplitude information for detection and compensation. For this reason, we have created a direct mismatch and offset detection technique using the output phase signal of the PhADC.

The relation between signal-to-noise power ratio (SNR) and its digital counter parts bit-energy-to-noise density (E_b/N_o) and symbol-energy-to-noise density (E_s/N_o) is established and is used to show the effect of IQ offset and gain mismatch on the BER as a function of channel and phase quantization noise from the PhADC. It is shown that E_b/N_o and or E_s/N_o are not the same as SNR and less understood ratios that are often confused with SNR. The relation between SNR and E_b/N_o, or for higher dimensions (i.e., multiple bits per symbol), the E_s/N_o depends on the modulation parameters. For $\pi/4$ DQPSK modulation, the following relationship holds:E_b/N_o = SNR + 0.97 dB.

An ideal PhADC receiver for $\pi/4$ DQPSK demodulation, with a noise-free channel, can tolerate a maximum IQ offset of 28 mV and a gain mismatch of ± 25 dB for a required BER of 10^{-5} according to the IEEE802.15.6 standard. But with a channel noise of SNR= 21dB these become 22 mV and ± 6 dB, respectively. A practical PhADC receiver with the same channel noise can tolerate a maximum IQ offset of 14 mV and a gain mismatch of ± 3 dB.

An approach is presented to convert the PhADC receiver analog channel select filter to a digital one for discrete modeling purposes in Matlab Simulink. First, the Laplace transfer functions of the filter stages and from those the overall Laplace transfer function of the total filter are derived. A mapping procedure is proposed to convert the continous time domain Laplace transfer function to a discrete one.

Contents

1 Introduction

1.1. Phase-Domain ADC in direct-conversion receivers for FSK/PSK demodulation

In short-range wireless telecommunication usually a Direct-Conversion Receiver is used because it has some advantages over the Heterodyne Receivers. In a Direct-Conversion, *zero-IF*, or *homodyne receiver*[*], the information signal RF is multiplied by a carrier LO with a frequency equal to that of signal RF, $\omega_{RF} = \omega_{LO}$. In *downconversion mixing* the information signal is downconverted to an intermediate frequency (IF), $\omega_{RF} - \omega_{LO}$, of zero, hence the name *zero-IF*. In a Heterodyne receiver[†], the information signal RF is multiplied by a carrier LO with a frequency unequal to that of signal RF, $\omega_{RF} \neq \omega_{LO}$. In *downconversion mixing* the information signal is downconverted to an intermediate frequency (IF), $\omega_{RF} - \omega_{LO} \neq 0$. Three aspects of direct conversion make it a superior choice with respect to heterodyning. First, the absence of an image (like the signal component $\omega_{RF} + \omega_{LO}$ in a downconversion mixing heterodyne receiver) greatly simplifies the design process. Second, RF channel selection is performed by *low-pass filters*, which can be realized on-chip as active circuit topologies with relatively sharp cut-off characteristics. Third, spurs that are produced by mixing the RF input with all the LO harmonics are considerably reduced in number and hence simpler to handle.

These receivers lend themselves perfectly for widely adopted FSK and PSK modulation schemes in low power short range wireless standards, e.g., BLE, Zigbee, IEEE 802.15.6 [2]. In the bio-electronics these modulation schemes are widely used in short range wireless personal and body-area networks due to their power efficient transmitter hardware and lower susceptibility to interference than ASK. About these digital modulation schemes, demystifying information can be found in appendices A to D.
Downconverted FSK/PSK signals are usually digitized by a pair of I and Q magnitude analog-to-digital converters (ADCs) before subsequent phase demodulation in the digital domain. See Figure 1.1(a).

[*]The term *homodyne* orginates from *homo* (same) and *dyne* (mixing).
[†]The term *heterodyne* orginates from *hetero* (different) and *dyne* (mixing).

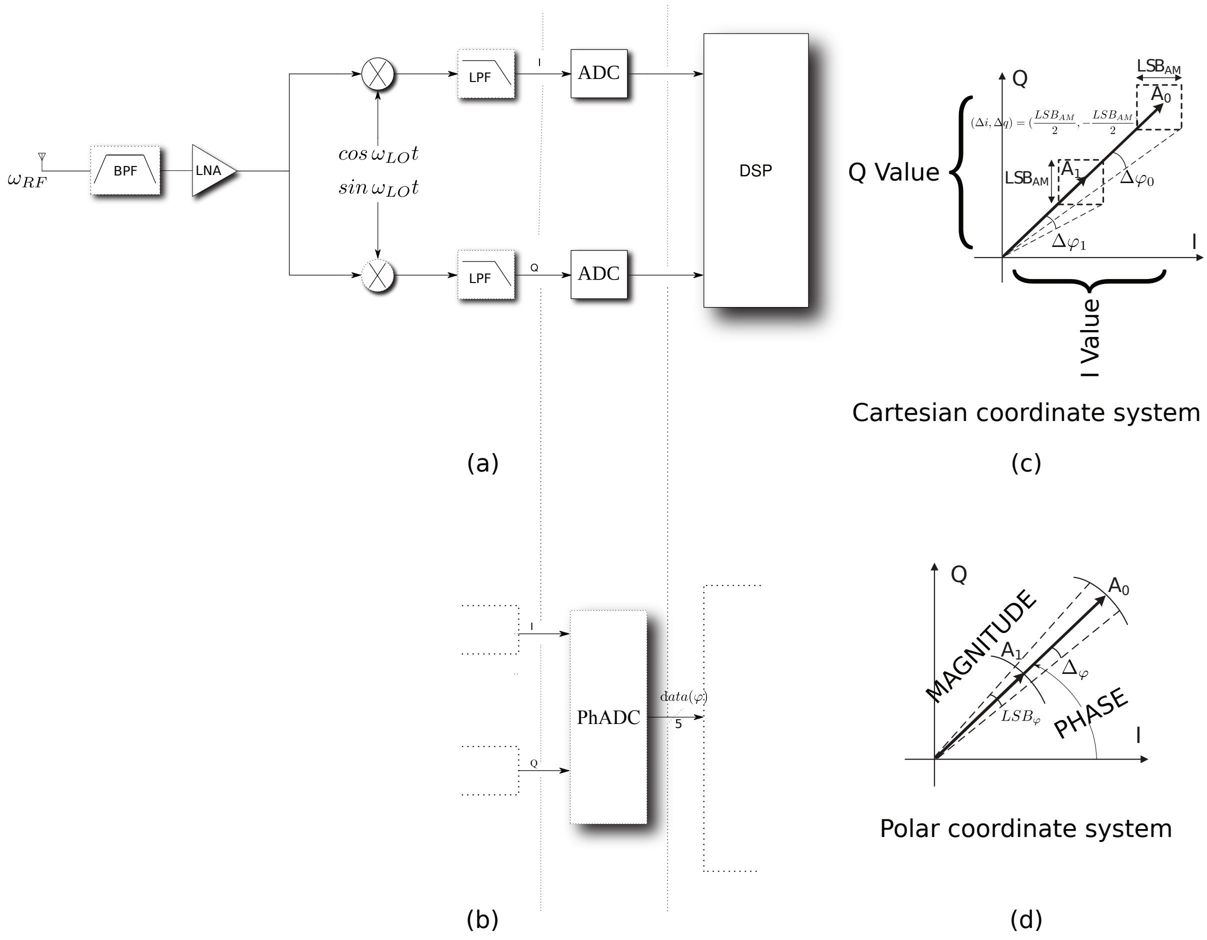


Figure 1.1: Concept of a direct-conversion receiver with magnitude ADC (a) and its Cartesian coordinate system (c), phase-domain ADC receiver (b) with its polar coordinate system (d). The two coordinate systems show the effect respectively of amplitude quantization noise (Δ_i, Δ_q) and phase quantization noise (Δ_φ) on the magnitude vector.

Yet the data information is encoded in the signal phase alone rather than in the magnitude. In such cases a phase-domain analog-to-digital converter (PhADC) is a promising alternative, in which I and Q signals are directly digitized in the phase domain, resulting in compact and power efficient receiver systems [11]. In comparison with the IQ ADC, the PhADC, due to its embedded demodulation attribute, is a more compact quantization and demodulation solution [11]. Phase only quantization gives simplified hardware with respect to two IQ ADC's and therefore can save power consumption. Another important property is that it is immune to magnitude variations. See Figure 1.1(d). But due to offsets and gain mismatches in I and Q signals it will produce nonlinear distortions in phase. This thesis is about detecting and compensating IQ offset and IQ gain mismatch in a phase-domain ADC.

Figures 1.1(c) and 1.1(d) show the phase quantization noise effects on an IQ ADC and a Phase-Domain ADC, respectively, when the input vector has different magnitudes that might be caused due to time-varying transmission power and communication channel attenuation in a practical receiver system. In Figure 1.1(c) amplitude quantization noise ($\Delta i, \Delta q$) produces a larger phase quantization noise ($\Delta\varphi_1$) at a small vector magnitude (A_1) than it does at a larger magnitude, i.e., $\Delta\varphi_0$ associated with A_0. Thus, the total phase quantization noise power associated with a non-constant vector magnitude is greater than that associated with a constant vector magnitude. Since the Phase-Domain ADC is a direct linear quantizer in the phase domain, the phase quantization noise is independent of the vector magnitude. This independence is also conceptually illustrated in Figure 1.1(d), showing that a larger vector A_0 has the same phase quantization error as a small vector A_1.

The concept of I and Q signals

We can write real signals as a vector sum of two signals in quadrature called I and Q. We can think of I and

Q as the x, y axis projection of a signal.

Figure 1.2 shows two views of a signal. One shows a signal in cartesian and the other in its polar form.

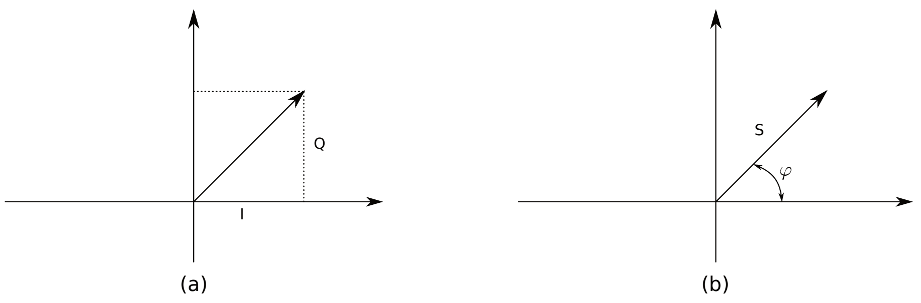


Figure 1.2: Signal vector plotted on IQ space (a) and the other in its polar form (b)

We can think of I and Q as the x, y axis projection of a signal as shown in Figure 1.2(a). These represent the amplitudes and give the signal vector. Figure 1.2(b) shows the same signal in polar form, with its length equal to its amplitude and the angle is equal to its phase. The angle the signal vector makes with the x-axis is the phase of this signal.

Magintude of signal $S = \sqrt{I^2 + Q^2}$
Phase of the signal = $arctan\frac{Q}{I}$

The concept of a symbol

A symbol is quite apart from a bit in concept although both can be represented by sinusoidal or wave functions. Where bit is the unit of information, the symbol is a unit of transmission energy. The symbol contain little piece of signal information. A symbol is a just a symbol. It can stand for any number of bits, not just one bit. It is a representation of the bit that the medium transmits to convey the information. The bits that it stands for are not being transmitted, what is transmitted are the signal states. Imagine bits as widgets, and symbols as boxes in which the widgets travel on a truck. We can have one widget per box or we can have more. Packing of widgets (bits) per box (symbols) is what modulation is all about.

In communications, the analog signal shape, by pre-agreed convention, stands for a certain number of bits and is called a symbol. See Figure 1.3

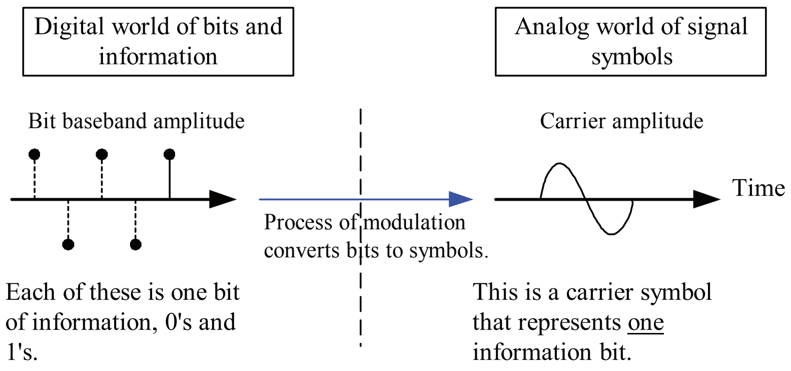

Digital world of bits and information
Analog world of signal symbols
Bit baseband amplitude
Carrier amplitude
Time
Process of modulation converts bits to symbols.
Each of these is one bit of information, 0's and 1's.
This is a carrier symbol that represents one information bit.

Figure 1.3: Digital information travels on analog carrier.

As an example for clarifying what exactly a state is, we take a QPSK signal that is defined as:

$$QPSK(t) = a_1 Acos\omega t + a_2 Asin\omega t \tag{1.1}$$

In "quadrature phase shift keying (QPSK)", we change the phase of the sinusoidal carrier to indicate information. See Equation (1.2). To obtain QPSK constellation, we assume bits a_1 and a_2 are pulses with height ±1.

$$QPSK(t) = \begin{cases} cos\omega t + sin\omega t & for \quad bits \quad 11 \quad (a_1 = 1 \quad a_2 = 1) \\ -cos\omega t + sin\omega t & for \quad bits \quad 01 \quad (a_1 = -1 \quad a_2 = 1) \\ -cos\omega t - sin\omega t & for \quad bits \quad 00 \quad (a_1 = -1 \quad a_2 = -1) \\ cos\omega t - sin\omega t & for \quad bits \quad 10 \quad (a_1 = 1 \quad a_2 = -1) \end{cases} \tag{1.2}$$

Figure 1.4 shows principle of a QPSK modulator. Data comes in and a serial-to-parallel converter (demultiplexer) separates the even- and odd numbered bits, applying one group to the upper arm and the other to the lower arm, hence, the symbol rate of QPSK is half of its bit rate. The two groups are then multiplied by the quadrature components of the carrier and summed at the output. See Figure 1.5 showing the QPSK modulated signal with the symbols indicated. The data on A and B in Figure 1.4 are the symbols and to obtain the QPSK constellation, the bits on A and B need to be converted to pulses with height ±1 or any other voltage $\pm V$ using a so called Bipolar nonreturn-to-zero (Bip-NRZ) device. For instance, if we take voltage levels of $\pm1V$ then a logical '1' becomes +1 V and a logical '0' becomes -1 V. This is a principal of line coding.

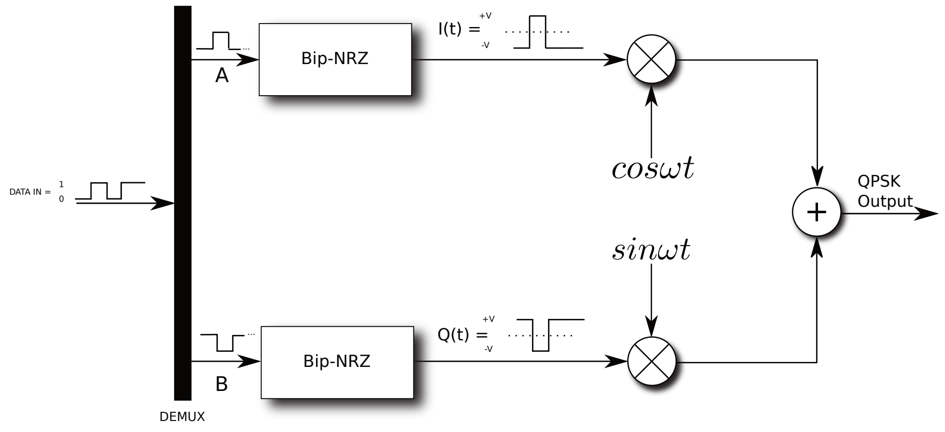


Figure 1.4: QPSK modulator.

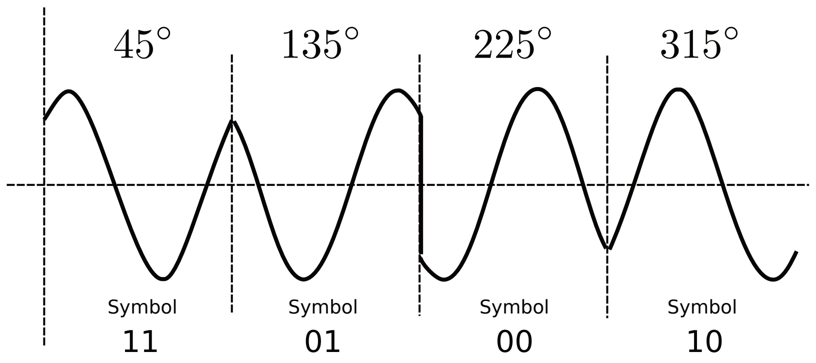


Figure 1.5: QPSK output waveform showing the signal states and the corresponding symbols.

The concept of a constellation

In a signal analyzer, we obtain the constellation diagram by sampling both I and Q channels at the same time instant and then plotting I value against Q value on a x-y diagram. The time axis can be imagined as coming out of the analyzer so new points lie on top of the old ones at the transmitter. A constellation is a plot of the I channel amplitude against the Q channel amplitude when sampled at the symbol rate. If the symbol rate is .1 seconds, then we would sample the time-domain signal every .1 seconds at the best possible moment and then plot the measured I and Q values. Each point is a pair of (I,Q) values of I and Q amplitudes. See Figure 1.6 showing the constellations of the previously discussed QPSK signal.

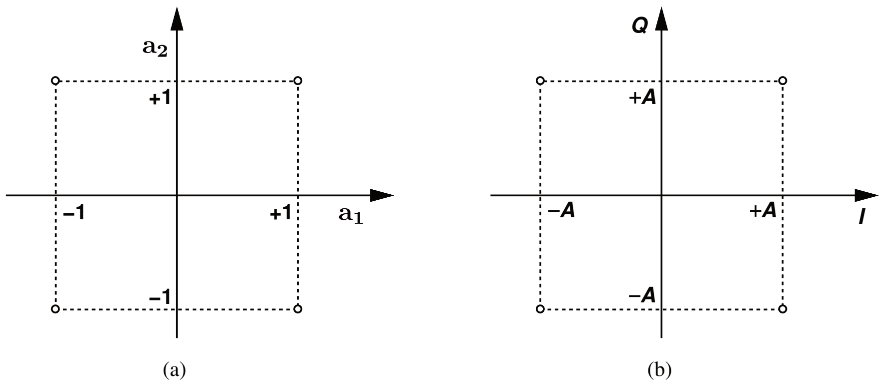


Figure 1.6: QPSK signal constellation in terms of a_1 and a_2 (a), and quadrature phases of carrier (b).

These I and Q values are computed by multiplying signal expression (its amplitude) by sine and cosine of the phase angle. This constellation diagram is created just prior to combing both I and Q into a composite QPSK signal. Constellation diagrams are always at baseband, that is at carrier frequency equal to zero. For instance, Eye diagrams similarly are also at baseband and show the same information but in time-domain. In practice when a constellation is created at the receiver, the picture is not perfect square and tell us the type of distortions experienced by the signal. To clarify even more, Figure 1.7 shows a picture of a constellation diagram with the analog signals that form the symbols.

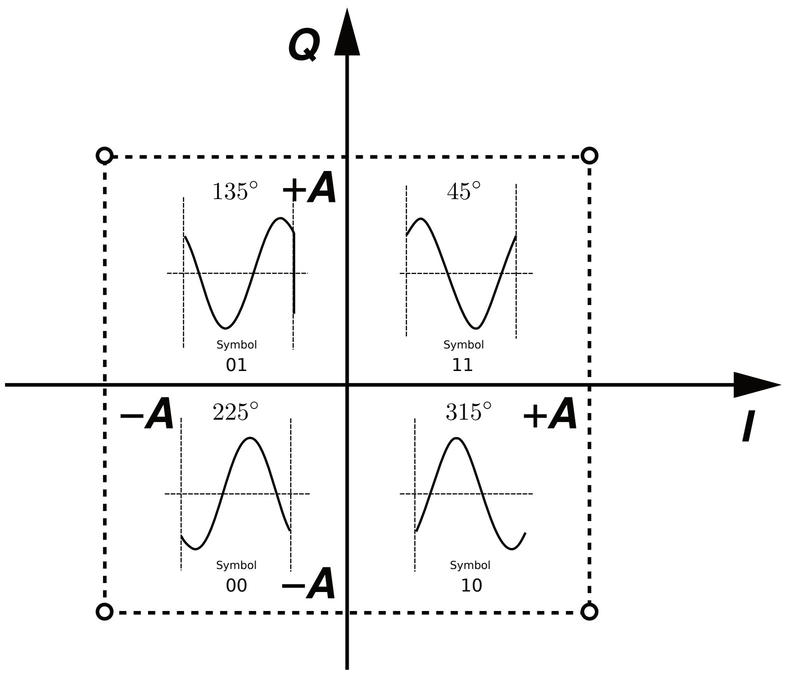


Figure 1.7: QPSK signal constellation including the analog signals that constitute the symbols.

Model of the Phase-Domain ADC (PhADC)

The model for the phase-domain ADC is implemented as a Four-quadrant inverse tangent function (*atan(Q,I)*) for the phase detection as shown in Equation (1.3) and converted to digital code according to the coding scheme as shown in Figure 1.8. The actual phase-domain ADC implementation concerns 5 bit, but for better visual purposes a 4 bit representation has been chosen in Figure 1.8. The idea of the coding scheme is the same in both cases.

$$\varphi = \begin{cases} arctan\frac{Q}{I} & I > 0 \\ \pi + arctan\frac{Q}{I} & Q \geqslant 0,\ I < 0 \\ -\pi + arctan\frac{Q}{I} & Q < 0,\ I < 0 \\ \frac{\pi}{2} & Q > 0,\ I = 0 \\ -\frac{\pi}{2} & Q < 0,\ I = 0 \\ 0 & Q = 0,\ I = 0 \end{cases} ,\ \varphi \in [-\pi, \pi] \tag{1.3}$$

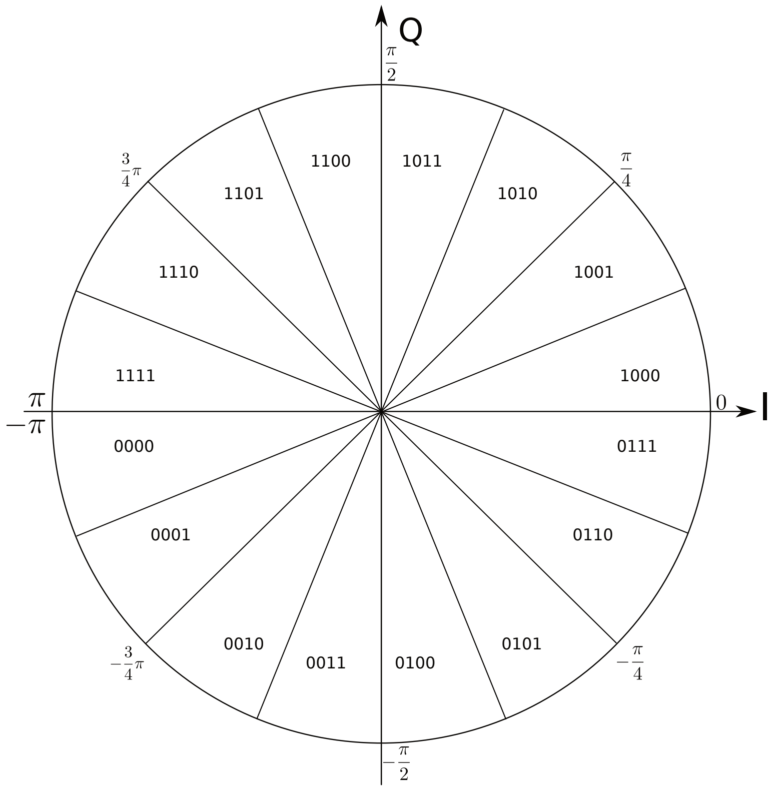
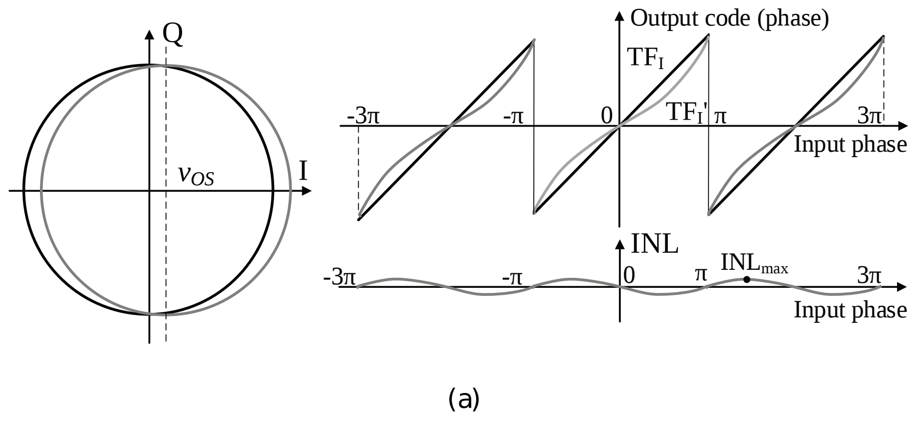


Figure 1.8: The phase between $-\pi$ and π is coded and split into 16 uniform sections by a 4 bit Phase-Domain ADC.

As shown in Figure 1.8 the complex IQ plane is split in 16 uniform sectors by a 4 bit PhADC. Each sector is $360°/16 = 22.5°$ and is mapped to a 4 bit digital word. According to definition (1.3) the PhADC returns phase values between $-\pi$ and $+\pi$. The phase quantization error of the PhADC is equal to 1 lsb = $\frac{\pi}{8}$ and so ideally if the quantization error were centered about zero the maximum phase quantization error is equal to $\pm\frac{LSB}{2} = \pm\frac{\pi}{16}$.

1.2. IQ offset and gain mismatch problems in a Phase-Domain ADC

The conversion of IQ magnitude to phase is not linear (i.e. arctan function) and so an IQ offset and gain mismatch will lead to non-linear phase distortions. In theory, both the IQ ADC and PhADC suffer from this effect in the same way [10], but it is well known that in practice it is possible that the mismatch and offset can be detected and calibrated before an amplitude to phase conversion takes place for the IQ ADC, but this direct amplitude and offset detection can not be applied for the PhADC due to the lack of amplitude information. Ideally, if the PhADC does its job perfectly well, then it can even extract the correct phase from a signal that is infinitely small, as the PhADC only responds to the phase and not to the magnitude of the signal at its input. However, due to unwanted offset and IQ mismatch, the IQ constellation will move to the left or right (I offset), or up or down (Q offset), or becomes an ellipse rather than a circle (IQ mismatch). This leads to incorrect detection of the phase. Next, these two effects will be discussed in detail.



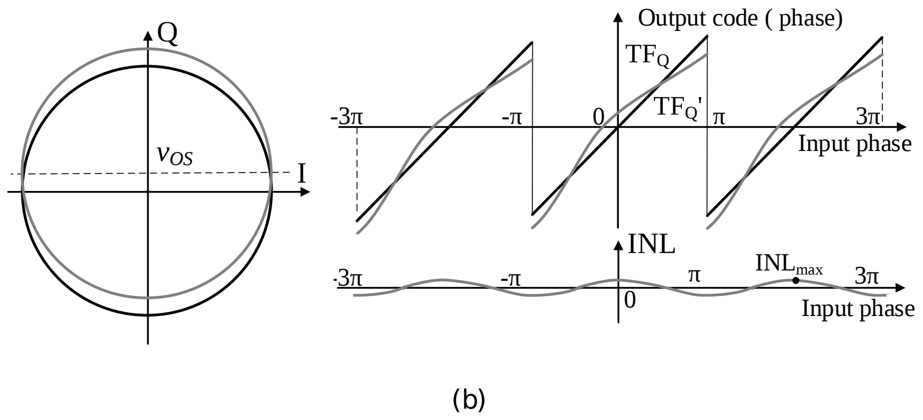


Figure 1.9: Vector trajectory shifting and nonlinear phase transfer functions due to (a) I offset, and (b) Q offset. The linear phase shape is indicated as TF_I and TF_Q respectively and the non-linear phase with its Integral Nonlinearity (INL) by $TF_{I'}$ and $TF_{Q'}$. (Pictures courtesy of Yao Liu [10].)

An ideal PhADC has a linear transfer function such as the curves TF_I and TF_Q shown in Figure 1.9(a) and 1.9(b) respectively, resulting in a circular output trajectory centered extacly at the origin of the IQ plane for a constant-magnitude input vector, as shown on the left side of these Figures. This can be realized when we select a sinusoidal signal with constant and equal amplitude for I and Q but with a 90° phase difference (i.e., in quadrature). The symbols in constellation diagram then lie on a circle. When one sine cycle is completed then the phase symbols would describe exactly one circle.
When a positive offset v_{os} is added to the I amplitude, the trajectory is shifted to the right and the corresponding phase transfer function changes to the non-linear curve $TF_I^{'}$. Similarly, an Q offset with a value of v_{os} causes a nonlinear phase transfer function $TF_Q^{'}$ as seen in Figure 1.9(b). $TF_I^{'}$ is an odd-order nonlinear function [10] with a maximum integral nonlinearity (INL) [3] of INL_{max}, whereas $TF_Q^{'}$ has both even- and odd order nonlinear terms even though with the same INL_{max}. The difference in $TF_I^{'}$ and $TF_Q^{'}$ shows that I and Q offset have different effects on the phase distortion when the two-dimensional vector is mapped onto the one-dimensional phase.

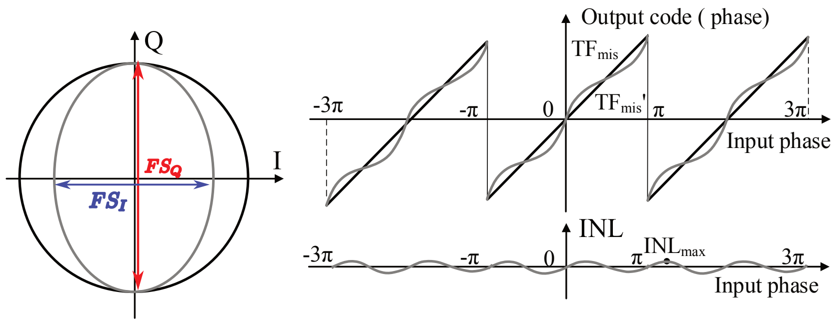


Figure 1.10: Vectory trajectory shifting and phase nonlinearities due to IQ gain mismatch. The linear phase shape is indicated as TF_{mis} and the non-linear phase with its Integral Nonlinearity (INL) by $TF_{mis'}$. (Pictures courtesy of Yao Liu [10].)

The same analysis can also be applied to investigate the effect of IQ gain mismatch. As shown in Figure 1.10, when the I and Q full scale range FS_I and FS_Q have a mismatch factor of

$$\alpha = \frac{FS_Q - FSI}{FS_Q} = 1 - \frac{FS_I}{FS_Q}, \tag{1.4}$$

the vectory trajectory is elliptical instead of circular, and the corresponding phase transfer function is an odd-order nonlinear curve TF'_{mis} with INL_{max} increasing with mismatch factor α [10]. IQ gain mismatch detection and compensation will be discussed in Chapter 3.

From this section we have learned that during the conversion of a two-dimensional vector to a one-dimensional phase, both offsets and mismatch of the amplitude can be transferred into non-linearities of the phase.

1.3. The limits of this study

This study only concerns solving the problems with IQ offset and mismatch in a Phase Domain ADC receiver [10]. Effects such as interference were not included in the study since interference was not considered in the design of the Phase-Domain ADC receiver [10]. The control system is based on the IEEE 802.15.6 standard using $\pi/4$ DQPSK modulation for the band 402-405MHz. And is not adapted for example to any communication protocol e.g. blue-tooth low energy or any other kind.

1.4. Assumptions

For modulation and demodulation only $\pi/4$ DQPSK is used and no other scheme. We also assume that the communication signal has already been amplified and converted to base-band.
There is no antenna mismatch at the receiver side, so no LO leakage can occur and no offset is generated after conversion. If an LO leakage appears at the LNA input along with a desired signal, this component is amplified and mixed with the LO. Called "LO self-mixing", this effect produces a dc component in the baseband because multiplying a sinusoid by itself results in a dc term. This dc component will cause trouble because it makes the processing of baseband signal difficult. The casecade of RF and baseband stages in a receiver must amplify the antenna signal by a large gain typically ranging from 70 to 100 dB. The received signal and the LO leakage are amplified and processed alongside each other. In the phase-domain receiver [10] the RF signal level is -73.1 dBm at the antenna, the receiver provides a maximum gain of 77.1 dBm, providing a maximum signal of +4 dBm at the analog baseband. Applying this gain to an LO leakage of, say, -53dBm, yields a very large dc offset in the baseband stages. Such an offset saturates the baseband circuits simply prohibiting signal detection. Direct conversion receivers must therefore incorporate some means of offset mitigation technique in each of the baseband I and Q paths.
Also effects of channel switching and gain setting are not included in the study. We also assume that the communication hardware is optimized for noise. The IQ offset compensation strategy is based on first calibrating the system for low offset and then activating the communication.

1.5. Approach and Presentation

In Chapter 1, it is shown what the effects of IQ offset and mismatch are in a Phase-Domain ADC. In Chapter 2, it is be shown how to correct the IQ offset by detecting the phase vector distribution imbalance and from that determine the offset. Next, the control system is verified and the performance simulated. Then a VerilogAMS system-level design is designed that is used as a starting point towards a transistor-level design in Chapter 4. In Chapter 3 the same is done but then for the effects of IQ gain mismatch in a Phase-Domain ADC. As mentioned in 4, the transistor-level design is done. It will be a mix of analog and digital. The logic blocks are synthesized in Cadence Encounter to transistor level and the analogue part is dealt with in a traditional analog design approach.In Chapter 4 some simulation test benches are designed and the performance of transistor level based control system design is simulated and verified. Finally, in Chapter 5 conclusions are drawn and recommendations made.

2 IQ offset detection and compensation in a Phase-Domain ADC

The above analysis shows that non-linearities introduced by offset and gain mismatch may significantly degrade the phase signal to noise ratio of a PhADC [10], thereby dictating proper techniques to detect and calibrate the offset and the mismatch. A mixed-signal approach, i.e., detecting the offset and the mismatch in the digital domain and calibrate them in the analog domain is usually incorporated in a system with an IQ ADC. Similarly, a digital phase-domain detection principle is proposed here, offering a possibility for calibration in the analog domain.

This detection principle will be described with the help of Figure 2.1 and Figure 2.2. Suppose that a positive I offset $v_{os,I}$ is added to the amplitude of I, then the circular vector trajectory will be shifted to the right, see Figure 2.1(a). In this case there are more vectors in the right half plane than in the left half plane assuming all vectors are evenly distributed along the circle. See the red vectors in Figure 2.2(a). We say that there is an imbalance in the distribution of vectors in the right and left half plane. By detecting the number of vectors (i.e., the symbols containing the phase information) in these half planes we can detect the imbalance and thus the sign and the amplitude of the I offset. Exactly the same detection principle can also be applied in case of a positive Q offset $v_{os,Q}$. In this case we look for the vector distribution imbalance in the top and bottom half plane of the circular vector trajectory, see Figure 2.1(b) and Figure2.2(b). For clarity, in this example a positive offset has been taken but can just as well be a negative offset at which the circular vector trajectory shifts to the left in case of I offset and in the case of Q offset shifts downwards. Since we receive the phase digitally from the PhADC, it is very easy to realize a density analysis process in the digital domain.

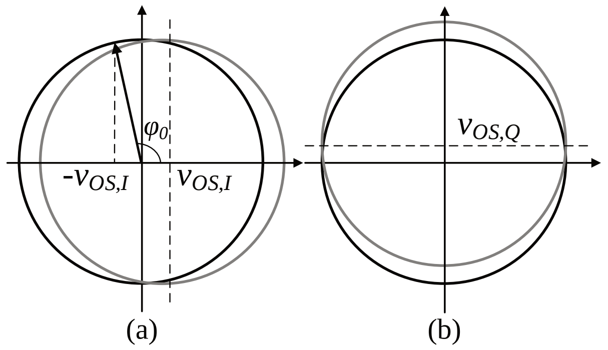


Figure 2.1: Vector distribution imbalance caused by (a) I offset, (b) Q offset. (Pictures courtesy of Yao Liu.)

Now we are going to define the distribution imbalance by taking the I offset, shown in Figure 2.1(a), as example. Due to the offset $v_{os,I}$, phase φ_0 is now shifted to $\frac{\pi}{2}$, indicating that all vectors between $[-\varphi_0, \varphi_0]$ are now in the right half of the shifted vector circle. Hence, the distribution imbalance between the right and left half plane is:

$$IMB_{OS} = \frac{2\varphi_0}{2\pi} - \frac{2(\pi-\varphi_0))}{2\pi} = \frac{2\varphi_0}{\pi} - 1 \tag{2.1}$$

Substituting $\varphi_0 = \frac{\pi}{2} + arcsin\frac{v_{os,I}}{FS/2}$ into Equation (2.1) and using small angle approximation, we get:

$$IMB_{OS} = \frac{2arcsin\frac{v_{OS,I}}{FS/2}}{\pi} \approx \frac{4v_{OS,I}}{\pi \cdot FS} \tag{2.2}$$

This equation holds also true for the distribution imbalance between the top and bottom half planes if $v_{os,I}$ is replaced by Q offset $v_{os,Q}$.

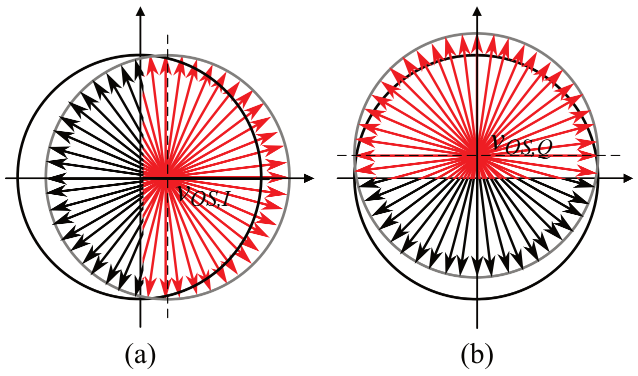


Figure 2.2: Shows in red higher distribution density of vectors in the right half plane (a) and top half plane (b) due to positive I and Q offset, indicated by $v_{os,I}$ and $v_{os,Q}$ respectively.

In order to calculate the absolute value of the offset, one of the full scale values FS_I and FS_Q should be available. The other one can be calculated from the known α and Equation (1.4) used for the IQ gain mismatch compensation. This can be accomplished by employing one auxiliary amplitude ADC in the calibration procedure. For the Phase-Domain ADC receiver [10] the signal level set at the PhADC input is $1V_{pp}$ full scale (equivalent to +4 dBm in a 50Ω system) for the proposed 1.2V system.
The distribution imbalance can be found by taking the difference of number of symbols in the right- and left half plane and dividing it by the total number of symbols. We define the number of symbols in the right- and left half plane as RHP and LHP and the total number of symbols as N. For simplification we call the vector distribution imbalance in case of an I offset $IMB_{os,I}$. The distribution imbalance becomes:

$$IMB_{os,I} = \frac{RHP - LHP}{RHP + LHP} = \frac{RHP - LHP}{N} \tag{2.3}$$

In the same way we can also do this for a Q offset $v_{os,Q}$. Lets call the number of symbols in the top- and bottom-half plane as THP and BHP. The distribution imbalance is defined as being $IMB_{os,Q}$ and becomes:

$$IMB_{os,Q} = \frac{THP - BHP}{THP + BHP} = \frac{THP - BHP}{N} \tag{2.4}$$

The above analysis provides an effective approach to detect the IQ offset in the digital domain, enabling the possibility to calibrate them in the analog domain, which is usually necessary in a receiver system.

We know that for sinusoidal IQ signals the symbols containing the phase information are sequentially distributed in a circle in the constellation diagram. See Figure 2.3. The input phase for the PhADC is linear and after just one sinus cycle we should then have enough symbols to detect the vector distribution imbalance accurately. But what does this mean for the determination of the vector distribution imbalance if the IQ signals have a random pattern, see Figure 2.4. In this case the symbols are randomly distributed in the constellation diagram and do not follow a circle in sequence. We do not know in advance how the symbols will spread over the constellation diagram and thus we need to take more data samples to make sure that enough symbols are distributed over the constellation diagram to accurately detect the vector distribution imbalance. This means that we will probably need far more symbols than if we were to take sinusoidal IQ signals to detect the vector distribution imbalance.

But we will see later that testing the performance of the IQ offset compensation system with sinusoidal IQ signals is fortunately a worst-case scenario. The system will perform even better at random signals and so it is also a good starting point to test the system for sinusoidal IQ signals for reference. This has to do with the sequence of the phase transitions for the different signal types and compensation in the analog domain.

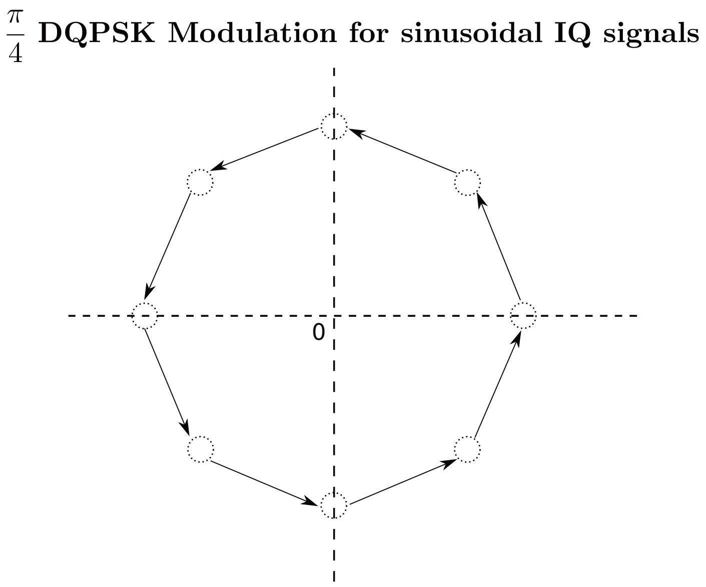


Figure 2.3: Possible phase transistions for pure sinusoidal IQ signals with no mismatch and no offset. Half of all symbols have a 100% chance of ending in one half of the constellation diagram and the other half 100% chance in the other half of the constellation diagram.

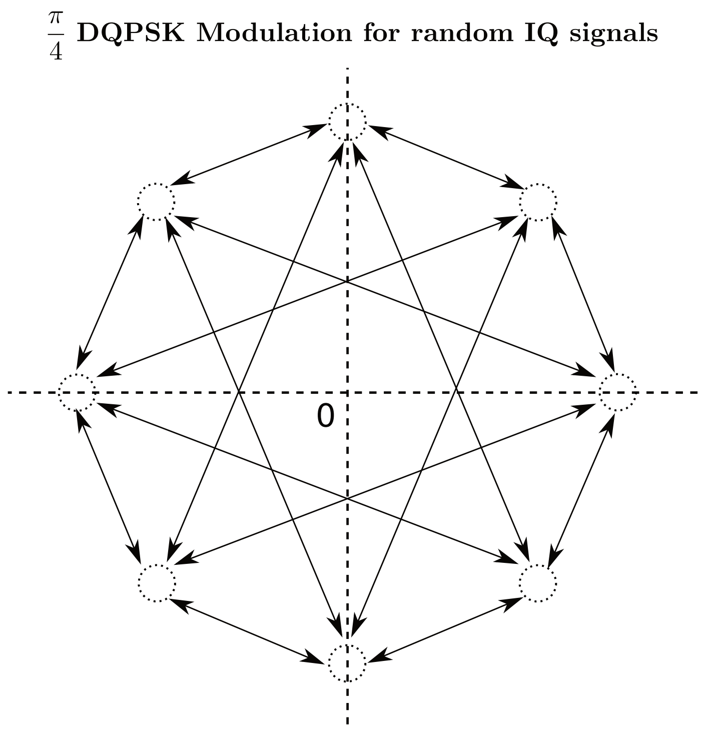


Figure 2.4: All possible phase transistions for random IQ signals with no mismatch and offset. All phase transitions have a 25% chance to occur.

Suppose we have an IQ offset compensation system like shown in Figure 2.5, where we omit for now the operation of the two-quadrant detector, and an IQ dc voltage offset of 0.2 V (i.e., $v_{os,I}$=0.2 V and $v_{os,Q}$ = 0.2 V) has been detected and compensated. Figure 2.6 shows one of the feedback signals (i.e. v_{os,I_f}) in case of random IQ input signals and Figure 2.7 for sinusoidal IQ input signals. The simulation time has been taken long enough such that the feedback signal settles around the desired offset of $v_{os,I} = 0.2$ V.

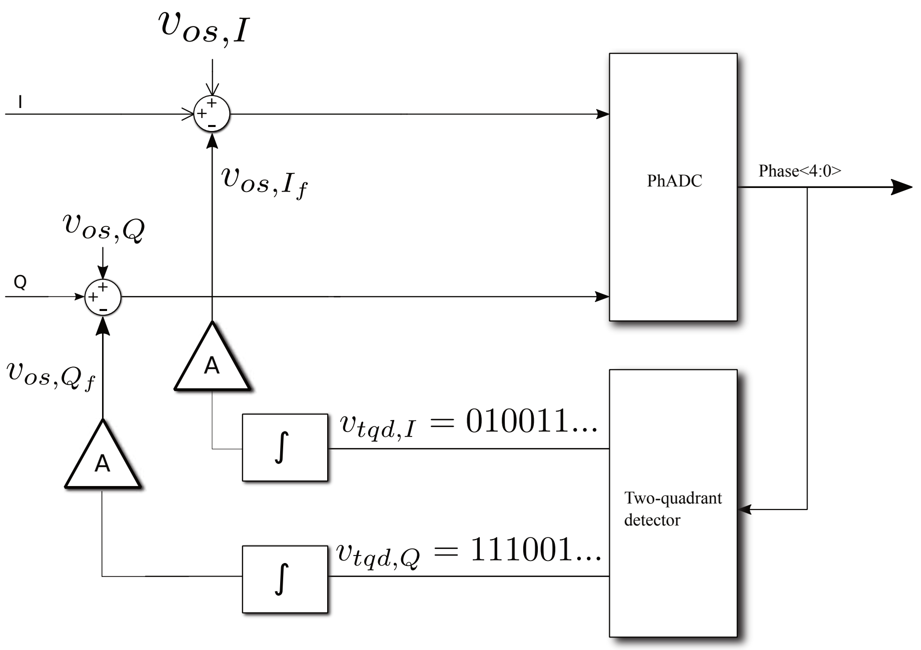


Figure 2.5: System architecture of the IQ offset compensation model based on the two-quadrant detector.

In Figure 2.6, the random behavior of phase transitions for random IQ signals, the feedback offset signal steps randomly around the desired end value of 0.2 V. While in Figure 2.7, the phase transitions go through a circle in sequence as is the case for sinusoidal IQ signals, the feedback offset signal steps up and down close to the desired end value of 0.2 V with a frequency equal to the information signal frequency. We take 8 samples every one complete cylce in the constellation at a rate equal to the information signal sampling rate and that is why we see a pattern of 4 steps up and 4 steps down. One complete phase cycle in the constellation is equal to one complete phase information signal cycle.

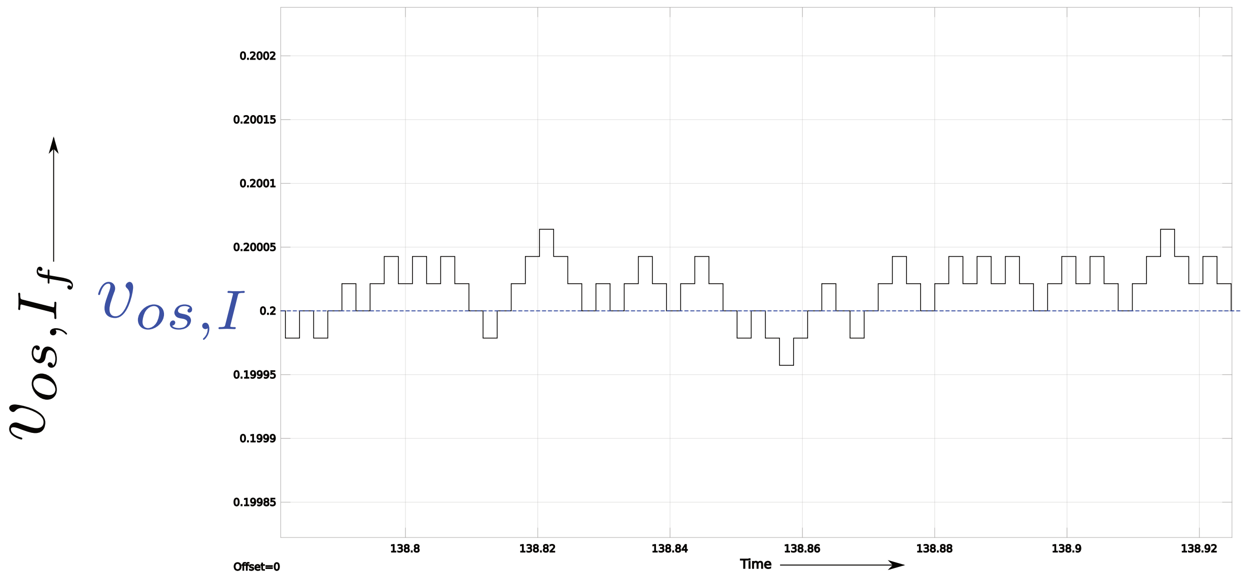


Figure 2.6: Thanks to the random phase transitions the feedback offset averages very close to the desired value of $v_{os} = 0.2$V.

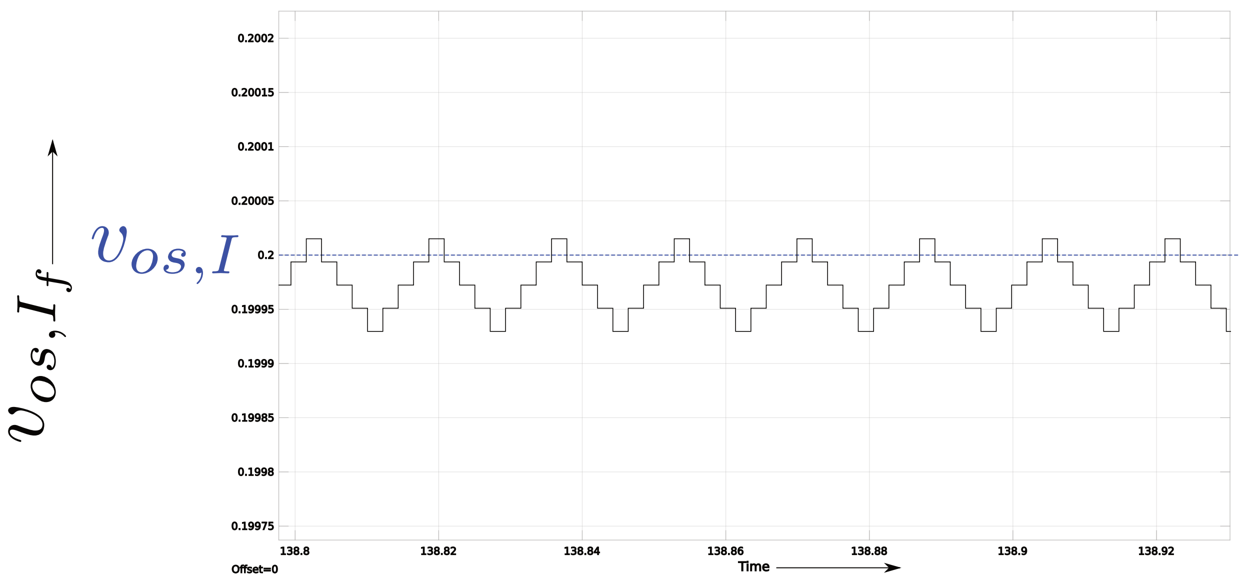


Figure 2.7: For offset compensated $\frac{\pi}{4}$ DQPSK signal the feedback offset averages close to the desired value of v_{os} = 0.2V. It always varies by 4 steps up and 4 steps down almost below the desired value of 0.2V.

2.1. System architecture

Figure 2.8 shows the system architecture of the IQ offset compensation model based on input-output frame-based data processing. The system processes a frame of N phase samples at a rate that is $1/N$ times the phase sample rate of the original phase sample-based signal. In Section 2.2 the sample and frame-based processing are discussed in more detail.
The system has two feedback loops, namely one for I and one for Q offset compensation. Basically the PhADC does an $atan(Q, I)$ as shown in Equation (1.3) and presents the 5 bit digital phase to the IQ offset detector. In case of I offset compensation, the number of symbols are detected in the right- and left half of the IQ complex plane. The vector distribution imbalance is then detected and therefrom the detected I offset $v_{osd,I}$ is passed to the integrator. Same thing holds also for the Q offset compensation where the number of symbols are detected in the top- and bottom half of the IQ complex plane.
The integrators are needed for error tracking to eliminate the steady state error between the actual- and desired output phase, i.e., eliminate the offsets $v_{os,I}$ and $v_{os,Q}$. To enhance the speed of control a gain block A is added in the feedback loop. But increasing the gain also lowers the systems *damping ratio* that makes the system slower. The gain is inversely proportional to the damping ratio. Damping describes how rapidly the signal response of a system decays with respect to time. *Under Damped* system, i.e., damping ratio is less then 1, the signal response oscillates and comes to rest gradually with decaying amplitude. A damping ratio less then zero, the system unstable and the response grows in time without bound. *Over damped* system, i.e., damping ratio greater then 1, system response doesn't vibrate at all. *Critically Damped* system, i.e., damping ratio is equal to one, system response doesn't vibrate either but soon settles and the system is quick to adjust to any change in the input. So a system is usually best with a damping ratio close to 1. This means that a compromise has to be found between the amount of gain and damping ratio so that the system response is quick and stable.

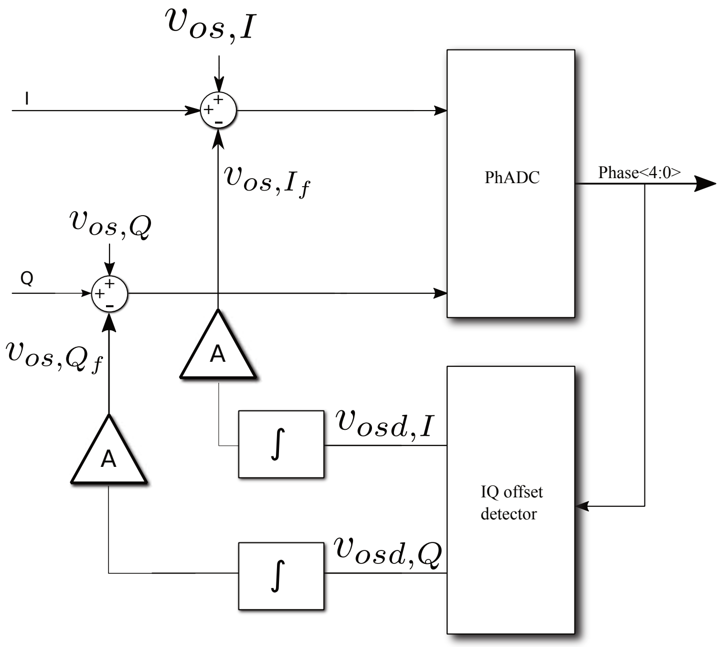


Figure 2.8: System architecture of the IQ offset detector based compensation model.

2.2. Sample-based and Frame-based input-output data processing

Sample-based signals are the most basic type of signal and are the easiest to construct from a real-world (physical) signal. We can create a sample-based signal at a given sample rate, and output each individual sample as it is received. In general, most Digital-to-Analog converters output sample-based signals.
We can create a frame-based signal from a sampled signal. When we buffer a batch of N samples, we create a frame of data. We can then output sequential frames of data at a rate that is $1/N$ times the sample rate of the original sample-based signal. The rate at which we output the frames of data is also known as the *frame rate* of the signal.
Frame-based data is a common format in real-time systems. Data acquisition hardware often operates by accumulating a large number of signal samples at a high rate. The hardware then propagates those samples to the real-time system as a block of data. Doing so maximizes the efficiency of the system by distributing the fixed process overhead across many samples. The faster data acquisition is suspended by slower interrupt processes after each frame is acquired, rather than after each individual sample.
The efficiency is defined as the ratio between the time needed to process the N samples of a data frame available at any given time and the time needed to process the N samples of a data frame of which only one sample is available at any given time. The lower this ratio, the higher the efficiency. The system frame-based signal processing is only interrupted at the end of a frame rather than at every sample. Having N samples of data available, it saves time lost due to overhead processing such as memory access, data latency, and the system response to each interrupt at every sample.

Model Sample- and Frame-Based Signals in MATLAB and Simulink

When we process signals using Simulink Communications System Toolbox software [17], we can do so in either sample- or frame-based manner. When we are working with blocks in Simulink, we can specify, on a block-by-block basis, which type of processing the block performs. In most cases, we specify the processing mode by setting the *Input processing* parameter. When we are using System objects in MATLAB, only frame-based processing is available. Table 2.1 shows the common parameter settings we can use for the blocks used in MATLAB Simulink to perform sample- and frame-based processing.

	Sample-Based Processing	**Frame-Based Processing**
Simulink – Blocks	Input processing = Elements as channels	Input-processing = Columns as channels

Table 2.1: Common parameters settings for sample- and frame-based processing in MATLAB Simulink.

What is Sample-Based Processing?

In sample-based processing, blocks process signals one sample at a time. Each element of the input signal represents one sample in a distinct communication channel. For example, from a sample-based processing perspective, the 3-by-2 matrix contains the first sample in each of six independent communication channels as shown in Figure 2.9.

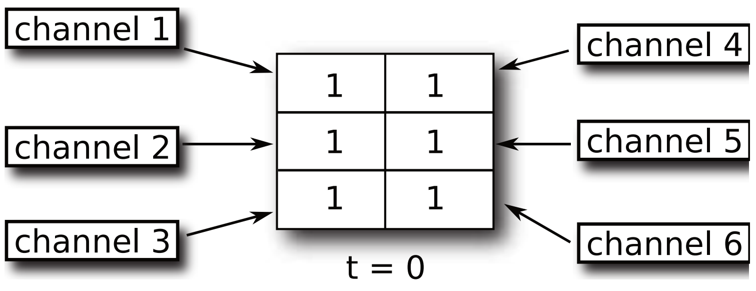


Figure 2.9: Example sample-based processing, each channel contains the first sample in a 3-by-2 matrix.

When we configure a block to perform sample-based processing, the block interprets scalar input as a single-channel signal, an M-by-N matrix as multichannel signal with $M * N$ independent channels. For example, in sample-based processing, blocks interpret the sequence of 3-by-2 matrices as six-channel signal as shown in Figure 2.10.

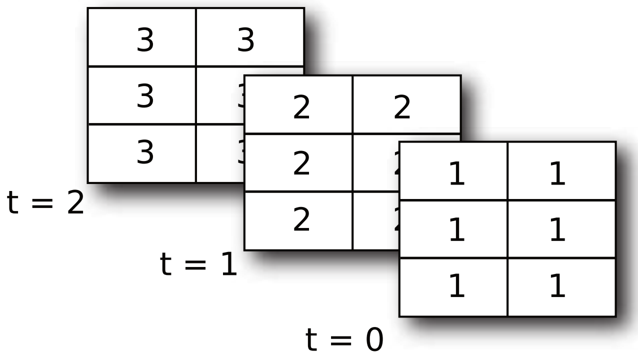


Figure 2.10: Example of sample-based processing. Blocks interpret this sequence of 3-by-2 matrices as a six channel signal.

What is Frame-Based Processing?

In frame-based processing, blocks process data one frame at a time. Each frame of data contains sequential samples from an independent communication channel that is represented by a column of the input signal. For example, from a frame-based processing perspective, the 3-by-2 matrix has two channels, each of which contains three samples as shown in Figure 2.11.

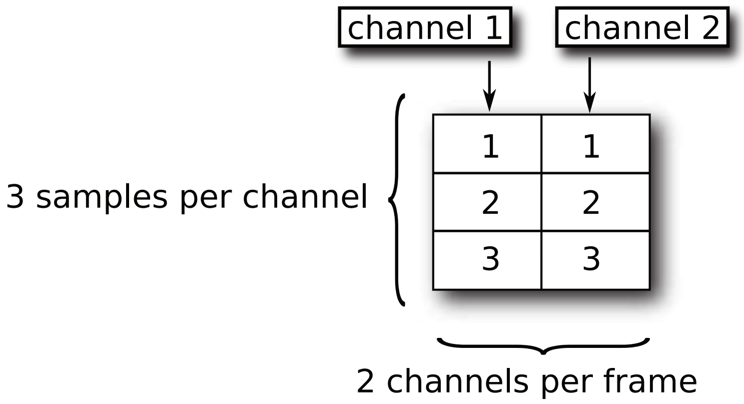


Figure 2.11: Example of frame-based processing, The 3-by-2 matrix has two channels, Each channel contains three samples.

When we configure a block to perform frame-based processing, the block interprets an M-by-1 vector as a single-channel signal containing M samples per frame. Similarly, the block interprets an M-by-N matrix as a multichannel signal with N independent channels and M samples per channel. For example, in frame-based processing, blocks interpret the sequence of 3-by-2 matrices as a two-channel signal with a frame size of 3 as shown in Figure 2.12.

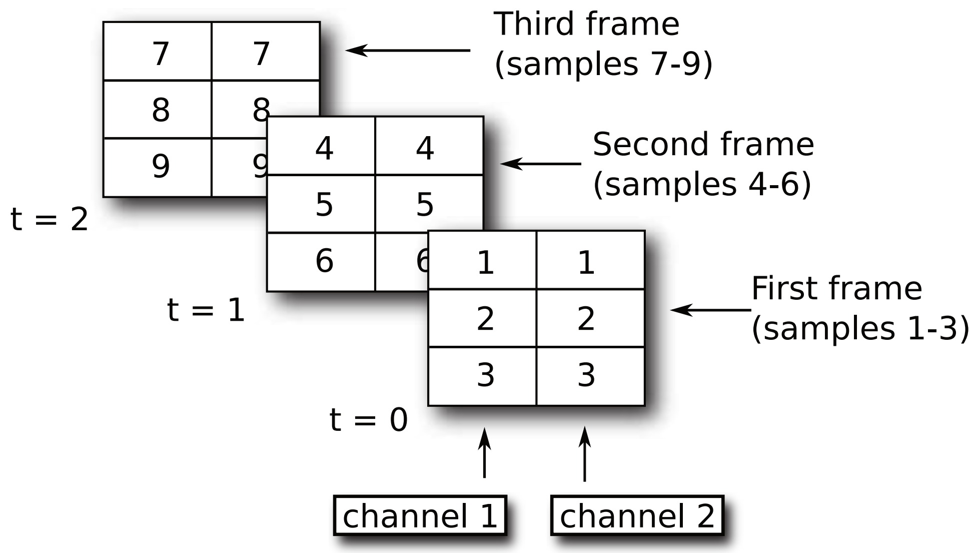


Figure 2.12: Example of frame-based processing. Blocks interpret this sequence of 3-by-2 matrices as a two-channel signal with a frame size of 3.

2.2.1. IQ offset detection based on IQ signal input-output data frames

Figure 2.13 shows a flowchart of the detection algorithm of the IQ offset detector block shown in Figure 2.8. This algorithm detects the number of symbols in the right- and left half of IQ complex plane (i.e., RHP and LHP) in case of I offset compensation and calculates the phase vector distribution imbalance (i.e., IMB_I) and feedback I offset (i.e., v_{osd,I_f}) from that. The same holds for Q offset in which the number of symbols are detected in the top- and bottom (i.e., THP and BHP) half of the IQ complex plane and calculates the phase vector distribution imbalance (i.e., IMB_Q) and feedback Q offset (i.e., $v_{osd,Q}$). Frame-based operation refers to the fact that phase vector distribution imbalance and from that the offset is determined on the basis of a frame of IQ data signal symbols. The algorithm first reads in one N sized frame of phase data symbols, detects its phase vector distribution in the IQ complex plane and then determines the imbalance and IQ offset. This process is iterative.

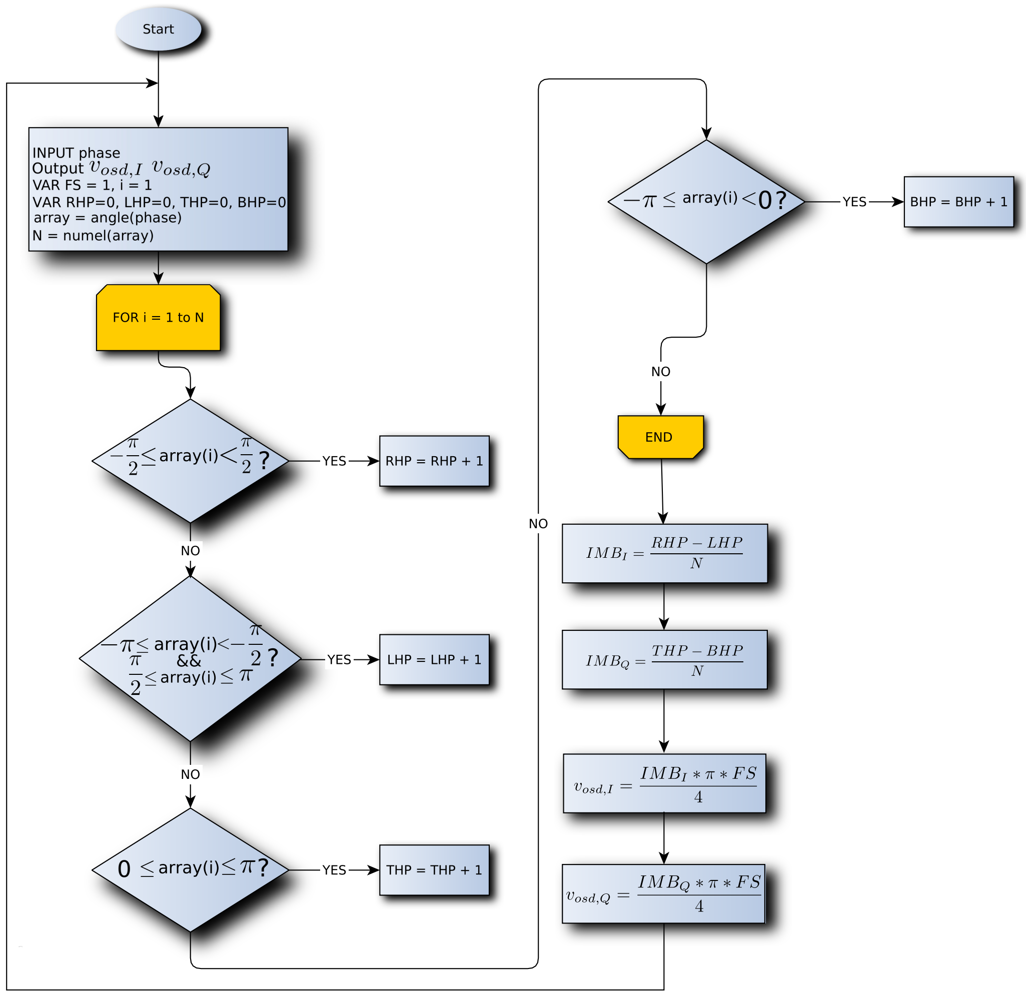

Start
INPUT phase
Output $v_{osd,I}$ $v_{osd,Q}$
VAR FS = 1, i = 1
VAR RHP=0, LHP=0, THP=0, BHP=0
array = angle(phase)
N = numel(array)
FOR i = 1 to N
$-\frac{\pi}{2} \leq$ array(i) $< \frac{\pi}{2}$?
YES
RHP = RHP + 1
NO
$-\pi \leq$ array(i) $< -\frac{\pi}{2}$ && $\frac{\pi}{2} \leq$ array(i) $\leq \pi$?
YES
LHP = LHP + 1
NO
$0 \leq$ array(i) $\leq \pi$?
YES
THP = THP + 1
NO
$-\pi \leq$ array(i) < 0 ?
YES
BHP = BHP + 1
NO
END
$IMB_I = \frac{RHP - LHP}{N}$
$IMB_Q = \frac{THP - BHP}{N}$
$v_{osd,I} = \frac{IMB_I * \pi * FS}{4}$
$v_{osd,Q} = \frac{IMB_Q * \pi * FS}{4}$

Figure 2.13: Flowchart of offset detection algorithm for the *IQ offset Detector* implementation.

Using Equation (2.2) we write for the I offset:

$$v_{os,I} \approx \frac{\pi.FS}{4} IMB_{os,I} \tag{2.5}$$

And the same goes for the Q offset:

$$v_{os,Q} \approx \frac{\pi.FS}{4} IMB_{os,Q} \tag{2.6}$$

The phase vector distribution imbalance for I and Q is determined by (2.3) and (2.4) respectively. By taking a sum of a limited positive number of symbols in the right and negative number in left half IQ complex plane in case of I offset and a sum of a positive number of symbols in top and negative number in bottom IQ half plane in case of Q offset can be recognized as integration.
So, in frame-based operation the IQ offset detector does integral action but at the same there is also an integrator in the feedback path. The system thus performs a double integration compared to the single integration in sample-based operation.
If we look again at (2.3) and (2.4) then we see that the detector also takes an average of the difference between number of symbols in both IQ half planes. Since it averages out the differences between number of symbols in left and right IQ half planes, it can be recognized as low pass filtering. The conclusion is that the IQ offset detector is a low pass filter that behaves with integral action and has a pole not in the origin of complex s-plane.
Because the phase is offered digitized by the PhADC, we can precisely determine the phase vector distribution imbalance, $IMB_{os,I}$ and $IMB_{os,Q}$, so the offset can be more or less detected accurately. The size of control

steps outputted by the *IQ offset Detector + integrator* is dynamically adjusted according to the imbalance between the number of symbols in left and right IQ half planes. The frame-based IQ offset compensation system will then, as a result, converge better to the desired final value of the feedback offset. See Figure 2.14.

In summary:

1. Using small angle approximation, IQ offset is linearly related to the phase vector distribution imbalance.
2. Phase is digitally presented by the PhADC and thus the phase vector distribution imbalance can be detected accurately.
3. Because of point 2, the IQ offset detection is also accurate and thus the system will do better controlling since the size of feedback offset control steps from *IQ offset Detector + integrator* are dynamically adjusted according to the imbalance of phase vector distribution in both IQ complex half planes.
4. The IQ offset detector is a low pass filter with integral action and thus does have a pole not in the origin of complex s-plane.

If we now use the system shown in Figure 2.8 with this detection algorithm implemented in the IQ offset detector block then the feedback signal v_{os,I_f} looks like shown in Figure 2.14. Comparing this result with that of Figures 2.6 and 2.7 we see that the feedback offset signal fluctuates better around the desired end value. Not only does the feedback signal fluctuate closer to the desired value of 0.2 V, but the size of control steps also change dynamically. Both the frame-based and sample-based compensation system converge very well since the transient signal v_{os,I_f} decays to small value so that it is almost in the steady state and is far within 1% variation around the desired end value, in this case 0.2 V.

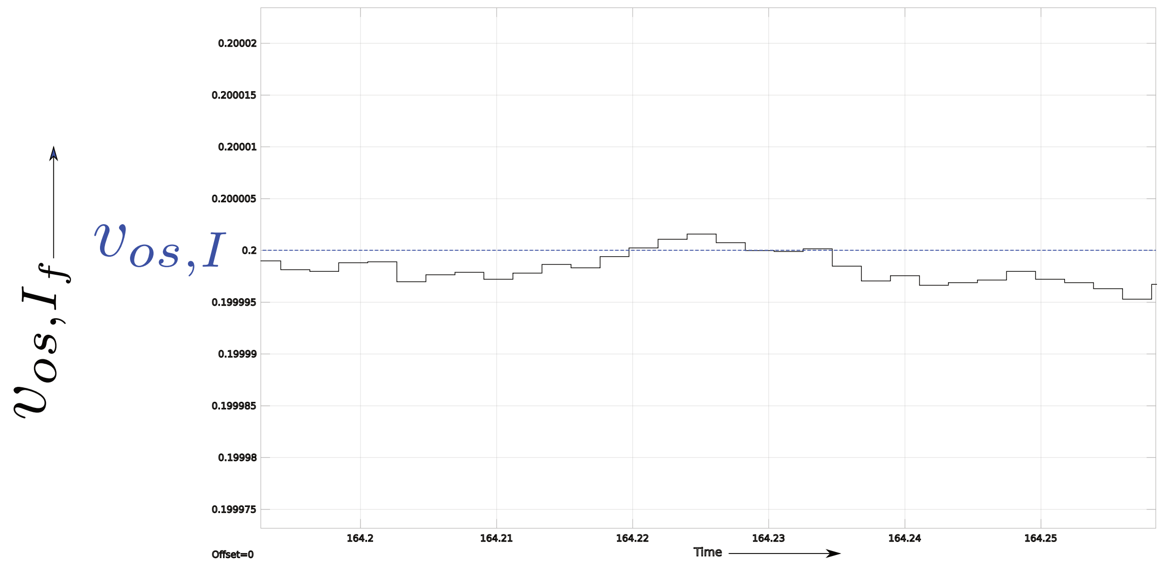


Figure 2.14: Thanks to the random phase transitions and better detection algorithm the feedback offset average path is closer to the desired value of $v_{os,I}$ = 0.2 V with dynamically changing the size of control steps.

But the frame-based operation method also has its disadvantages. Firstly, arithmetics are performed and data needs to be stored, and for that we need a micro-controller (μC) and memory. Secondly, this will cost more power and circuit area on chip. Especially in the case of wireless body electronics, ultra low power consumption and size is very important. So, we have to look for an even simpler implementation of this offset compensation principle in which we do not need a μC and memory.

2.2.2. IQ offset detection based on IQ signal input-output data samples

The detection algorithm for IQ offset compensation can even be made simpler. The method of detecting symbols in the IQ complex plane is the same as in the previous system but the difference between sample-based approach is that it works on an input-output sample-based processing principle. This means that the system tries to compensate for each passing symbol individually instead of on every frame of symbols as in frame-based processing.

In case of I offset compensation, a symbol detected in right half of the IQ complex plane will output a logical '1' and one in the left half of the IQ complex plane will output a logical '0', as shown in Figure 2.15.

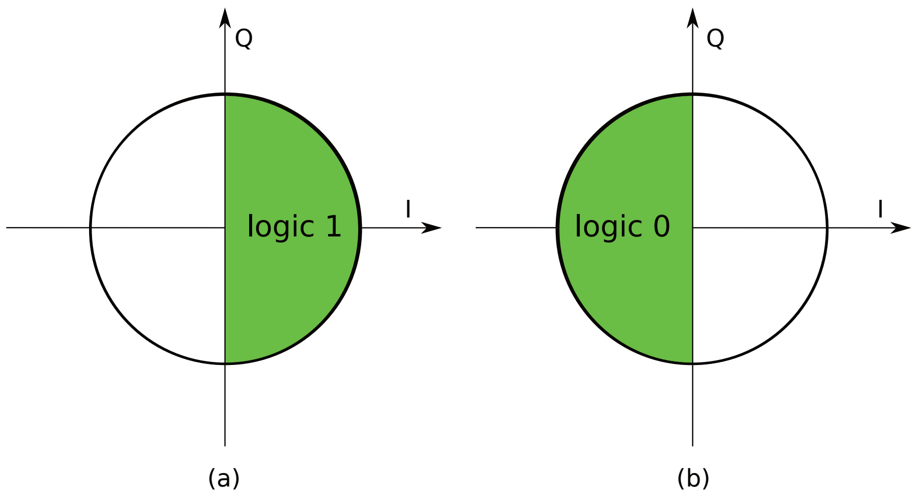


Figure 2.15: two-quadrant detection for I offset compensation. If a symbol is in right half plane the output is a logic "1" (a) and if a symbol is in the left half plane the output is a logic "0" (b)

The previous applies exactly to the case of Q offset. Here a symbol in the top half of IQ complex will output a logical '1' and a symbol in the bottom half of IQ complex plane will output a logical '0' as shown in Figure 2.16. This algorithm is implemented in the *two-quadrant Detector* shown in Figure 2.5. This algorithm tries to detect the symbols in one of the IQ complex half planes and based on the detection it outputs a logical "1" or "0". The IQ complex half plane consist of two-quadrants, hence the name *two-quadrant Detector.*

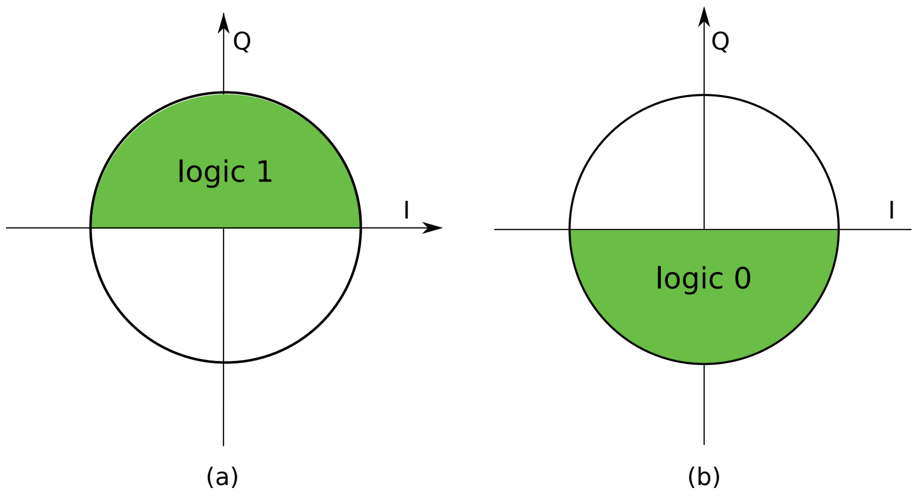
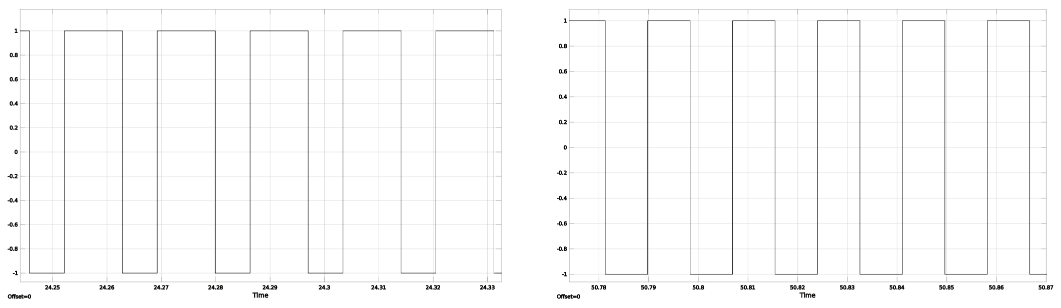


Figure 2.16: two-quadrant detection for Q offset compensation. If a symbol is in the top half plane the output is a logic "1" (a) and if a symbol is in the bottom half plane the output is a logic zero "0"(b)

Based on this detection principle the *Two-Quadrant Detector* will produce a digital signal of random ones and zeros, in case of random phase transitions, to the input of integrators as shown in Figure 2.5. This in turn will produce a stept up or stept down signal which will eventually settle at a certain value when the offset is compensated. But unlike the previous system, the size of the control steps do not change due to the integration of logical signals only.



(a) (b)

Figure 2.17: The vector distribution unbalance translates into a non-50% duty cycle at the output of the *Two-Quadrant Detector*: vector distribution not in balance, duty-cycle $\neq$ 50% (a), vector distribution is in balance and perfect duty cycle of 50% (b).

It is interesting to see that in case of sinusoidal IQ input signals the unbalance of phase vector distribution in the IQ complex plane is translated into a duty cycle at the output of *Two-Quadrant Detector*, see Figure 2.17. Due to the linear input phase, the phase vector trajectory is circular, the detector will output a group of logical ones and zeros. A duty cycle of less than 50% suggests a negative remaining offset and for a duty-cycle of more than 50% a positive one. Figure 2.17(a) shows, for example, that there is still a small positive offset remaining. On the other hand, in Figure 2.17(b), the duty-cycle is exactly 50% which means that the phase vector distribution is perfectly in balance and the offset is eliminated.

But of course this is not surprising that we see this, as previously discussed, the phase transitions for IQ sinusoidal input signals go through a circle in sequence in a constellation and therefore the *Two-Quadrant Detector* will output the phase vector distribution unbalance as a group of logic "1" and logic "0" pulses and therefore we also see this in the duty cycle of the signal.

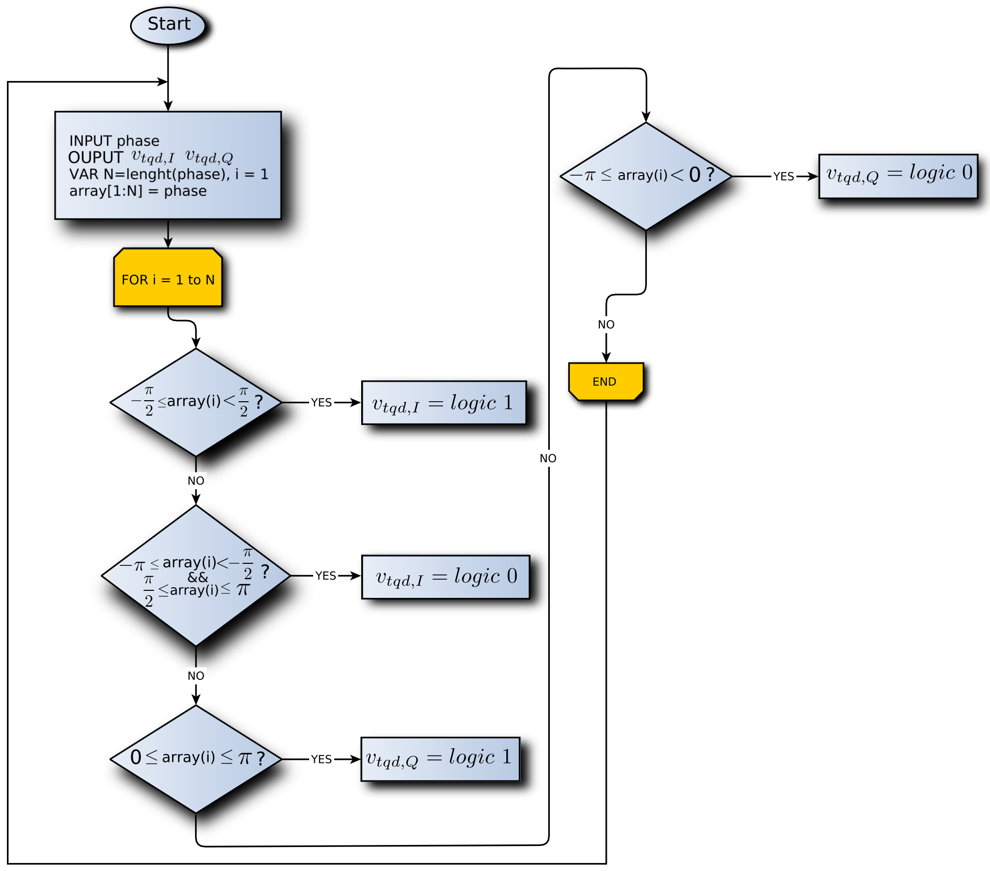

Start
INPUT phase
OUPUT $v_{tqd,I}$ $v_{tqd,Q}$
VAR N=lenght(phase), i = 1
array[1:N] = phase
FOR i = 1 to N
$-\frac{\pi}{2} \leq$ array(i) $< \frac{\pi}{2}$?
YES
$v_{tqd,I} = logic\ 1$
NO
$-\pi \leq$ array(i) $< -\frac{\pi}{2}$ && $\frac{\pi}{2} \leq$ array(i) $\leq \pi$?
YES
$v_{tqd,I} = logic\ 0$
NO
$0 \leq$ array(i) $\leq \pi$?
YES
$v_{tqd,Q} = logic\ 1$
NO
$-\pi \leq$ array(i) < 0 ?
YES
$v_{tqd,Q} = logic\ 0$
NO
END

Figure 2.18: Flowchart of two-quadrant detection algorithm for the *Two- Quadrant Detector* implementation.

Figure 2.18 shows a flowchart of the two-quadrant detection algorithm of the *Two-Quadrant Detector* shown in Figure 2.5.
If we compare this flowchart with the one shown in Figure 2.13 we see that the outputs $v_{tqd,I}$ and $v_{tqd,Q}$ are directly controlled within the *For loop*. The control algorithm shown in Figure 2.13 will output $v_{osd,I}$ and $v_{osd,Q}$ after its *For loop* has been finished. This clearly shows the difference between frame- and sample based processing. The two-quadrant detection algorithm reacts on each symbol individually while the IQ offset detection algorithm first processes the whole frame of phase data symbols and then outputs $v_{osd,I}$ and $v_{osd,Q}$. And in contrast, the sample-based operation using the *Two-Quadrant Detector* has no integral action but outputs a logical "0" or a logical "1" based on the position of a symbol in the IQ complex plane. It is not detecting the IQ offset but only the position of a phase symbol in one of the two quadrants. So then we can assume that the *Two-Quadrant Detector*, unlike the *IQ offset Detector*, is not a filter that has integral action and therefore has no pole but has a gain that is highly non-linear.

In summary:

1. The *Two-Quadrant Detector* does not detect IQ offset but outputs a logical "0" or a logical "1" based on the position of a symbol in the IQ complex plane.
2. So, the *Two-Quadrant Detector* is not a filter that has integral action but has a gain that is highly non-linear.

The IQ offset compensation system based on the *IQ offset Detector* has dual integral behavior and the IQ offset compensation system based on the *Two-Quadrant Detector* has a single integral behavior. And that is why we can say that the *IQ offset Detector* based compensation system is a second order system and the compensation system based on *Two-Quadrant Detector* is a first order system. The response of both systems should therefore be different.

Bit Error Rate BER

An important performance parameter that is used for the comparison of both systems is the *bit error rate* (BER).

In a digital transmission, BER is the number of bits with errors divided by the total number of bits that have been transmitted, received or processed over a given time period. That is

$$BER = \frac{Number\ of\ bits\ with\ errors}{total\ number\ of\ bits}$$

Bit error rate is a key parameter that is used in assessing the system's performance that transmits digital data from one location to another. When data is transmitted over a data link, there is a possibility of errors being introduced into the system. As a result, it is necessary to assess the system performance, and BER provides an ideal way in which this can be achieved. BER assesses performance of a system including the transmitter, receiver and the medium between both.

There are three factors that are important for determining the performance of the system, namely;

- bandwidth or the frequency of the system, depending on whether it is bandpass or low pass. In the digital domain, we can associate this with the amount of bits processed per second.
- Linearity. Manifests itself as a clipping or weak distortion. In the digital domain we can associate this with the full scale range of the PhADC.
- Noise, determines the dynamic range of the system, namely the ratio between signal power and noise power in dB. In digital domain we can associate this with the quantization noise (LSB). Determines the number of bits of the PhADC also called the ENOB (effective Number of bits). So the actual resolution of the PhADC.

At this moment we have enough information to test one of these, namely, the speed of both systems using a Simulink simulation system setup like shown in Figures F.1 and F.3 in Appendix F Sections F.1 and F.2.

The simulations for speed consists of two runs, the first run is to detect the moment when the number of bit errors remains constant in the system. We note the associated parameter values; bit error rate, number of bit errors and total number of bits. And on the second we run again a simulation for this total number of bits + 1.10^5 bits to simulate for the minimum required BER of 1.10^{-5} for a $\frac{\pi}{4}$ DQPSK modulation. Again, we note the three parameters mentioned above. Important is to see if the number of bit errors has been increased compared to the first run. In order to achieve the required BER of 1.10^{-5}, only 1 bit error would then be allowed in the last 1.10^5 bits.

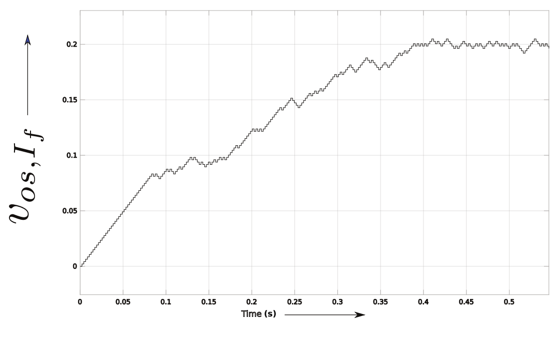
Just for the sake of clarity, both systems have been simulated in Simulink in frame-based data input-output processing mode, but the way in which both algorithms process the frames of data is different. A distinction must therefore be made between sample and frame-based channel processing in Simulink and that of the detection algorithms.

Tabel 2.2 shows the simulation results. It is important to note that in the second simulation run the number of bit errors has remained the same with respect to the first run. This means that the system performs better by having a lower BER value than 1.10^{-5}. And when we look at the total simulation time of both systems, we see that the system with *IQ offset detector* is slightly slower than the system with the *Two-qudrant detector*. So we can conclude that both systems are about the same in terms of speed. We could ask ourselves what happens if we would not choose a first order integrator ($1/s$), but a second order ($1/s^2$) integrator in the feedback path of the sample-based system just like the frame-based which is based on second order compensation. Perhaps then the compensation would be more accurate and even faster.

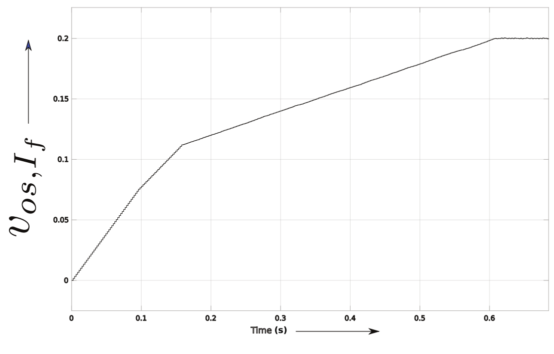
IQ offset compensation system based on	**BER Measurement**	**First run**	**Second run**	**Total simulation time (s)**
Two-quadrant detection	Bit Error rate: Number of error bits: Total number of bits:	0.3111 $3.259.10^4$ $1.048.10^5$	0.1585 $3.259.10^4$ $2.056.10^5$	0.546
IQ offset detection	Bit error rate: Number of error bits: Total number of bits:	0.2679 $4.201.10^4$ $1.568.10^4$	0.1632 $4.205.10^4$ $2.576.10^5$	0.685

Table 2.2: BER measurements of the IQ offset compensation system based on IQ offset- and Two-quadrant detection.

Figure 2.19 shows the response of feedback offset signal v_{os,I_f} of both systems.



(a)



(b)

Figure 2.19: System simulation: response of feedback offset signal v_{os,I_f} after no bit errors plus 1.10^5 bits based on *Two-quadrant detection* (a) and *IQ offset detection* (b).

The sample-based model has some issues to be considered:

- the sample-based compensation is not smooth since it does only a *Two-quadrant detection* and no *offset detection.* It is a classic 1^{st} order type 1 loop.
- So it does not converge as well as the frame-based compensation.
- The gain in the feedback path has to be chosen properly since it is sensitive to large gains and tends to make the system unstable.

But has also very attractive advantages:

- the control system can be a simple digital implementation.
- no need for memory and μC, just simple combinatorial logic is sufficient. As a consequence, less energy consumption due to simple architecture.
- it can be implemented in a mixed signal design, in contrast to the frame-based model where we are forced to do the implementation completely digitally

So given the advantages, despite the disadvantages mentioned, a sample-based model is a better choice.

2.3. Functional performance Simulink system model

In this subsection we will test the functionality of the IQ offset compensation system. Both the sample-based and the frame-based have been tested and for this we use the Simulink models from Appendix F, Sections F.1 and F.2. The settings of each block in the models are indicated in boxes in the relevant model. Detailed information about the blocks used in model can be found in Appendix G.
Most of the settings in the model are straightforward but three blocks in the model deserve extra attention; the square-root raised-cosine (SRRC) transmit and receive filter and the phase-domain ADC (PhADC).

The SRRC Transmit and Receive filter

The settings for the SRRC transmit filter and receive filter are shown in Figures G.24 and G.27 in appendix G.
One of the most important parameters is the *roll-off factor*, also known as the *excess bandwidth factor* α of the SRRC filter. The value for this parameter can be between 0 and 1. A value of $\alpha = 0$ would mean that the filter spectrum is perfectly rectangular (i.e., brickwall). The occupied bandwidth would be the same as the symbol rate (i.e., half of bit rate for $\frac{\pi}{4}$QPSK modulation), but this is not practical. An alpha of zero is impossible to implement. We would need infinite number of impulse response coefficients which is not practical. But fortunately, it does not have to be. The *rectangular baseband pulses* used in the binary sequence contain sharp transitions between ONEs and ZEROs which leads to unnecessarily wide bandwidth and hence to an increase in integrated noise. For this reason, the baseband pulses in communication systems are usually *shaped* to reduce their bandwidth using these square root raised filters and that's why the excess bandwidth factor is in practice between 0 and 1. More on the topic of filtering will be discussed in Subsection 2.4.4.

The exact value of the excess bandwidth factor α and the duration of the filter pulse shape is implementation dependent. Typical values are in the range of 0.3 to 0.5. In our models we have chosen this factor to be 0.5, the average of 0 and 1. There are good reasons for this. Our phase-domain ADC receiver is very low power and is used for very short range distances, so the picked up disturbances should not be that big compared to the ones picked up over a large communication distance. Hence, we don't need to do sharp filtering to filter out these disturbances and so we do not need to have low α (i.e., near 0.2) values to define sharp filter edges. For this purpose we can use a relatively broadband than a narrow band filter. Especially in a digital implementation, in this case a Finite Impulse Response filter (FIR), for a narrow band spectrum you need a high number of *FIR taps*(i.e., filter order - 1) than for a narrow band spectrum. In other words, the number of coefficients of the filter impulse response $h[n]$ becomes large. A *FIR tap* is simply a coefficient/delay pair. The number of *FIR taps* (often designated as *N*) is one of the terms that is used to describe FIR filters and an indication of:

1. the amount of memory required to implement the filter. The more FIR taps, the more memory locations needed to store previous inputs for computing each output value.

2. number of calculations. The more FIR taps, the more multiplications and additions needed for computing each output value.

3. the amount of filtering can do; in effect, more taps means increasing stop-band attenuation, less ripple, narrower filter, etc.

So a relatively simple filter also means lower costs, because a narrow-band in this case, due to the many *FIR taps*, requires much more digital gates and therefore also means more power consumption.

We would like a filter response like $P(f)$ shown in Figure 2.20 But this filter has a response like shown in Figure G.25 on Appendix G section G.4. It has *side lobes*, this is because the filter response is *truncated* to a finite number of these side lobes. This can be set in the *Filter span in symbols* field of the *Block Parameters: Raised Cosine Transmit Filter* dialog box shown in Figure G.24. There is a reason for that. In practice, it is impossible to realize a filter with the ideal characteristic of a rectangular spectrum or like the less ideal $P(f)$ in Figure 2.43(b). We would need infinite number of impulse response coefficients which is not practical. In order to develop stable and realizable filter, the frequency response is relaxed and the sharp edges are removed and replaced with a transition band, and small ripple is allowed in the pass-band and stop band. We have chosen 12 for the Filter span in symbols. This choice ensures that the first side lobe of the filter impulse response is 40 dB lower then the pass band (i.e., -40 dB at 0 dB passband). And this is acceptable value. Filter span is simply the number of $\frac{1}{T_s}$ spaced side lobes in the filter response.

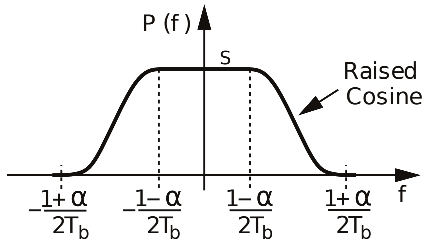


Figure 2.20: Raised-cosine pulse shape spectrum. (Pictures courtesy of Behzad Razavi.)

The last parameter setting that still needs attention is the *Linear amplitude filter gain*. By default it is set with the function *rcfiltgaincompat(gcbh)* or simply value 1. This setting makes sure that the filter coefficients are normalized to provide unit energy gain. However, the filter would then have a gain of 9 dB (i.e., power gain of 8) while we need a unity gain filter. That is why the filter gain of both the transmit and receive filter are divided by $\sqrt{8}$.

$$\sqrt{\frac{F}{8}} * \sqrt{\frac{F}{8}} = \frac{F}{8}$$

That is also why in both transmitter and receiver a square root(hence, $\sqrt{F}$) raised cosine filter is used to implement the $P(f)$ spectrum filter response with unity gain. Another reason is that we will replace the SSRC filter in the receiver with an analog one with a unity gain. And in order to make a fair comparison of this transceiver model with that including SSRC filters, it is sensible if both filter-gain paths are set to unity gain. More on matched filtering will be discussed in subsection 2.4.4.

The parameter settings for the raised cosine receive filter are set accordingly, taking into account that the *Decimation factor* value must be equal to that of the *Output samples per symbol* of the transmit filter.

Simulink model of the Phase-Domain ADC

In Figure 2.21 we don't see a separate I (in-phase) and Q (quadrature) inputs. That is correct, because in Simulink the signals are complex. The in-phase I is the real part and the quadrature Q is the imaginary part of the complex signal. In the model we see that the input is first converted into a magnitude $|u|$ and phase $\angle u$ in the *Complex to Magnitude-Angle* block. Then the phase vector magnitude $|u|$ is limited to an amplitude of 1 by a *Saturation* block, because, ideally, the changes in phase magnitude $|u|$ will not affect the phase. Hence, the symbols in constellation diagram will be on a unity circle. And due to the absence of phase quantization in the phase signal path, the phase $\angle u$ is actually passed unchanged, as shown in Figure. So there's no phase quantization and this can be seen as having a Phase-domain ADC with an infinite bit resolution and thus no phase quantization noise is added. Then the signal is again converted back to complex in the *Magnitude-Angle to complex* block. Since our PhADC is a 5 bit we will of course add phase quantization in this block in a later stage of our simulations.

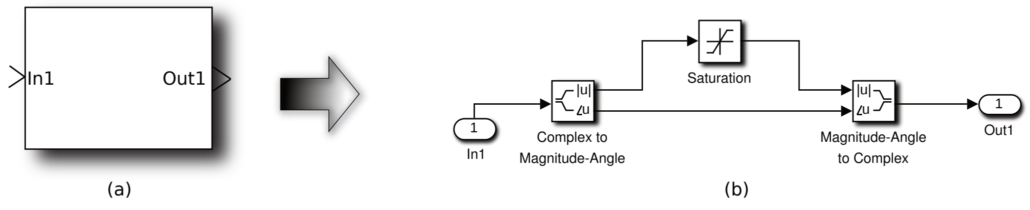
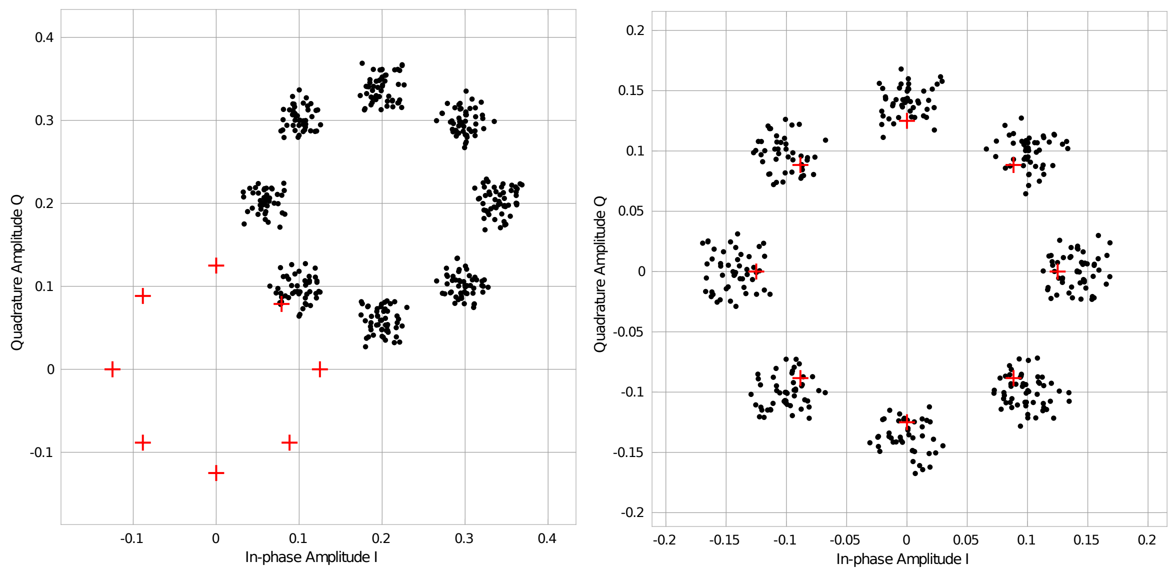


Figure 2.21: Simulink model of an infinite resolution ideal Phase Domain ADC with its symbol (a) and structure (b).

2.3.1. Simulation of IQ offset compensation

Figure 2.22(a) shows the constellation of receiver input signal in black after offset is added of I = 0.2 V and Q = 0.2 V. Incidentally, this is a pretty large value and will in practice be more likely to be tens of mV, but this has been done to clarify the concept of IQ offset compensation. As we can see in the Figure, the constellation points are moved 0.2 V to the right and 0.2 V to the top while they should have been around the origin. This reference location is indicated by red crosses. It defines the exact locations of the constellation points for $\pi/4$ QPSK modulation with no disturbances. But after IQ offset compensation we can see that the constellation points are again at the reference locations. See Figure 2.22(b). So we can conclude that the system is working!

The cloud of points around each reference (i.e., ideal) constellation point is an inevitable consequence of the SRRC action. An SRRC alone does not meet the Nyquist filtering criteria. More on this is discussed in Subsection 2.4.4.



(a) (b)

Figure 2.22: Shows the reference constellation in red and receiver input signal in black for uncompensated IQ offset (a) and compensated IQ offset (b).

Figure 2.23(a) shows the input signal of the PhADC. The clouds around the reference points in the constellation are now much smaller due to complete raised cosine filtering and the symbols will be closer to the reference locations when the system is compensated. After offset compensation it looks like Figure 2.23(b).

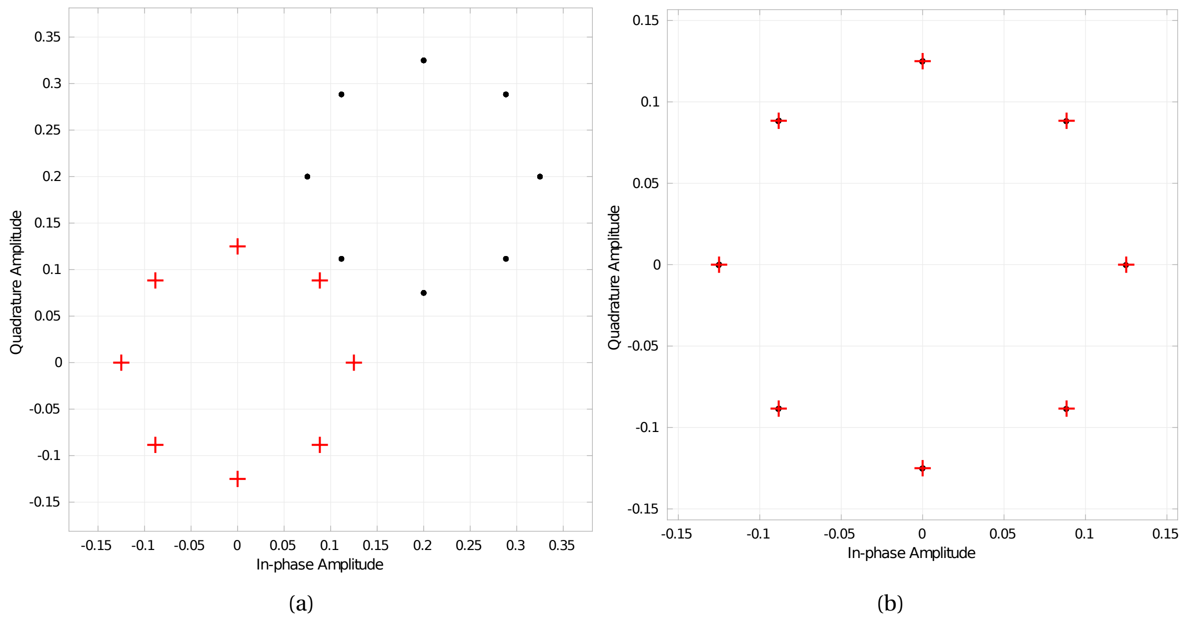


Figure 2.23: Shows the reference constellation in red and Phase-Domain ADC input signal in black for uncompensated IQ offset (a) and compensated IQ offset (b).

It is also interesting to see what happens at the output of the PhADC, see Figure 2.24(a). At the beginning of offset compensation, all the symbols are concentrated in the upper right corner of the constellation diagram as a result of the offset and then during compensation they move gradually to their reference points. See Figures 2.25(a) and 2.25(b). The concentration of symbols in the upper right corner are spread over the constellation circle in the direction of the reference locations. It is also important to see that the phase vector magnitudes, the tip of these vectors indicate the location of a symbol in the constellation, are saturated to a magnitude of one. Hence, the symbols are located on a unity circle. The ideal PhADC is immune to phase vector magnitudes variations and that is the reason why it is saturated to a magnitude of one. Figure 2.24(b) shows a zoomed-in small part of the symbols concentration at the beginning of compensation. This clearly shows the effect of IQ offset on the phase vector distribution. Finally, after compensation completion, all the symbols are at their reference locations. See Figure 2.25(b).

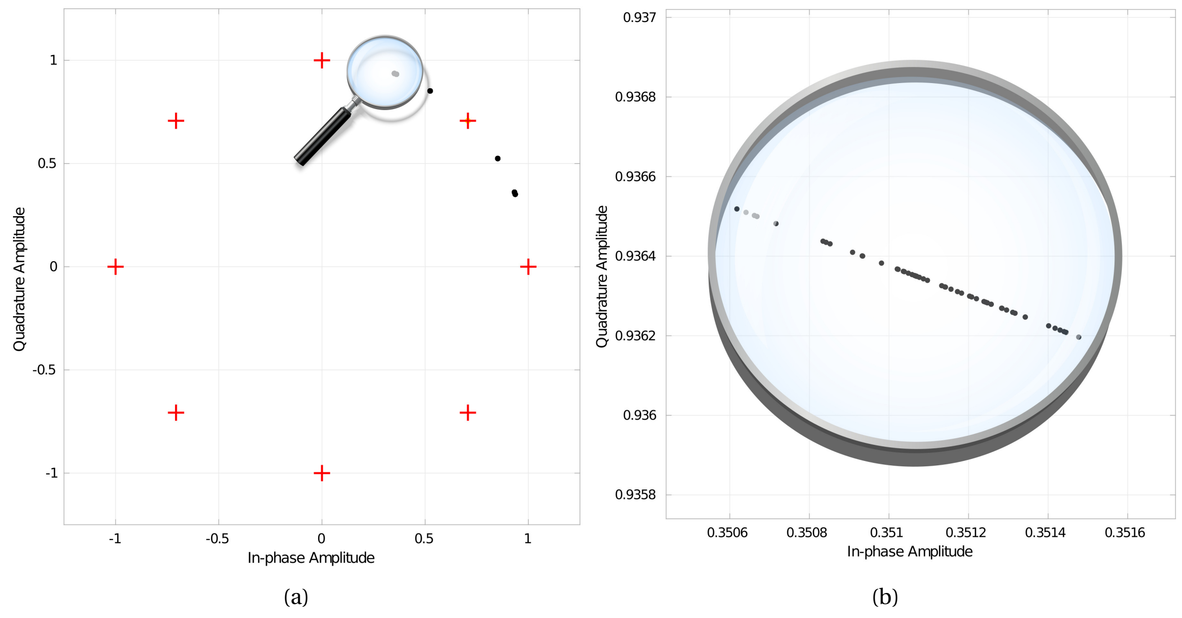


Figure 2.24: Shows the reference constellation in red and DQPSK demodulator input signal in black for uncompensated IQ offset (a) and zoomed in part of a black spot in (a).

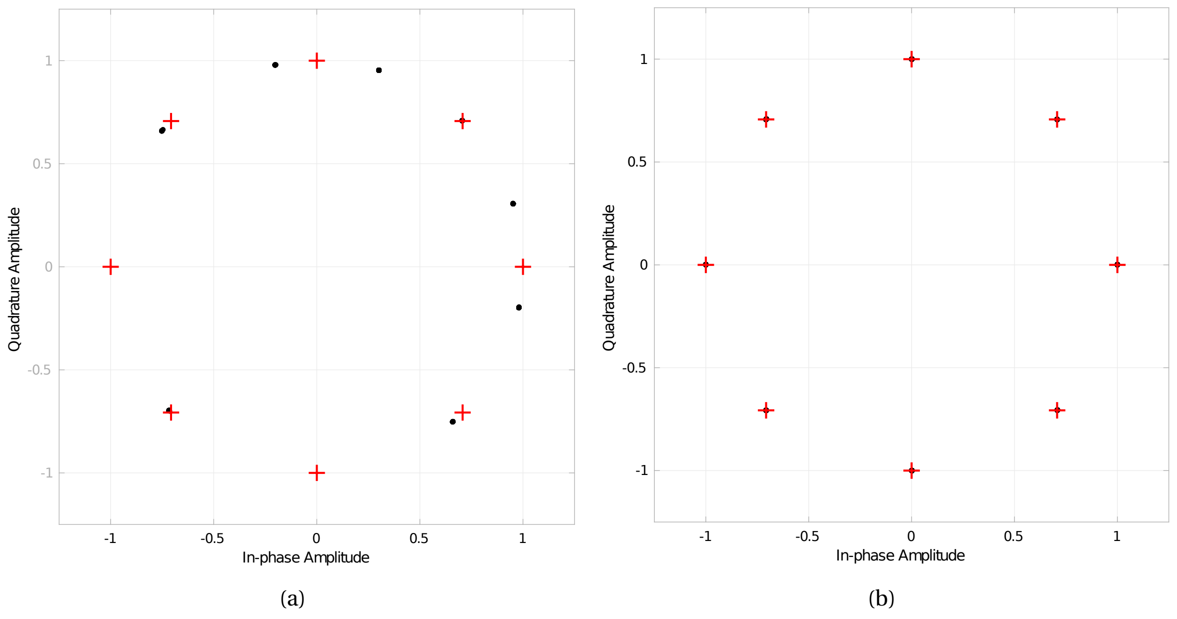


Figure 2.25: Shows reference constellation in red and partially compensated DQPSK demodulator input signal in black (a) and fully compensated IQ offset.

Figure 2.26 shows the phase vector trajectory of the transmitter and receiver signal. As we can see in Figure 2.26(a), due to the pulse shaping by the SRRC filter, the phase symbols are scattered around the ideal points and the phase transitions are gradual and not abrupt. At the end of receiver side we see perfect $\pi/4$ DQPSK phase transitions, see Figure 2.26(b).

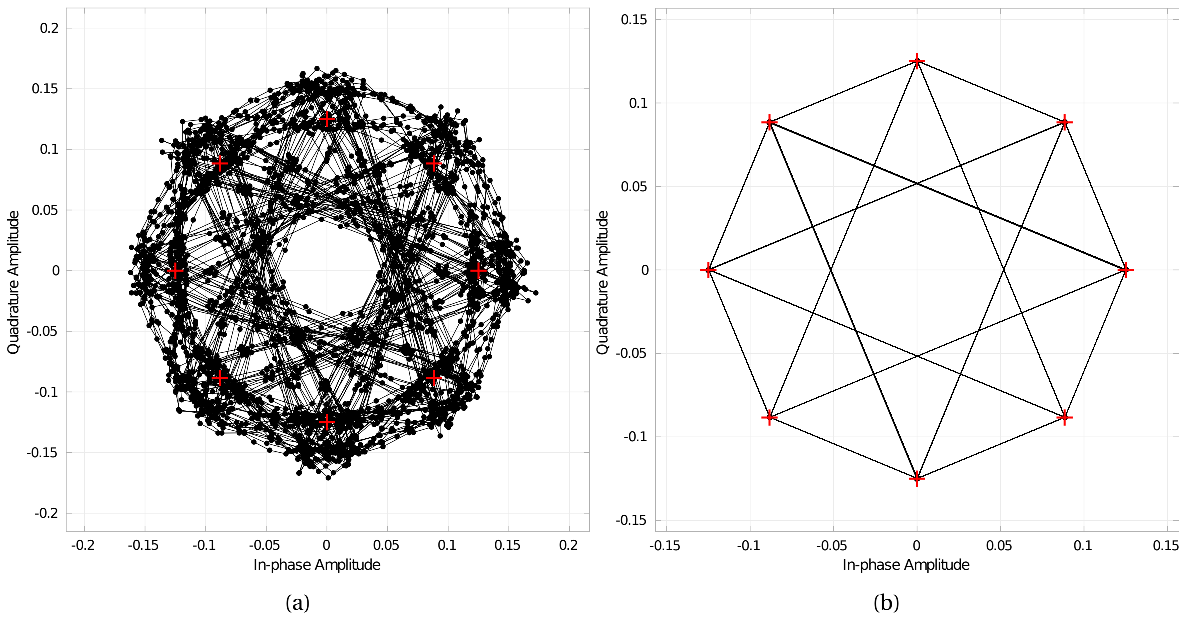


Figure 2.26: Shows signal trajectory of transmitted Tx signal (a) and received Rx signal(b).

A list of the steady-state signals at several locations of the transceiver chain can be found in Appendix H.

2.3.2. Signal quality of received signal

The constellation can also provide a *quantitative* measure of the impairments that corrupt the signal. Representing the deviation of the constellation points from their ideal positions, the *error vector magnitude* (EVM) is such a measure. To obtain the EVM, a constellation based on a large number of detected samples is constructed and a vector is drawn between each measured point and its ideal position, see Figure 2.27.

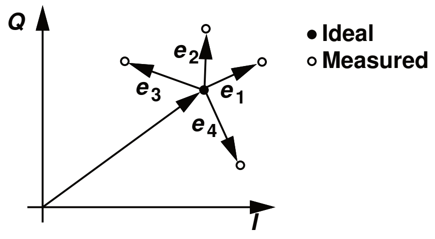


Figure 2.27: Illustration of EVM.

The EVM is defined as the RMS magnitude of these error vectors normalized to the average signal power [20]. See Equation (2.7).

$$EVM_{rms} = \sqrt{\frac{\frac{1}{N}\sum_{k=1}^{N} e_k^2}{P_{avg}}} \tag{2.7}$$

Where:

- $e_k = \sqrt{(I_k - \tilde{I}_k)^2 + (Q_k - \tilde{Q}_k)^2}$
- $\tilde{I}_k$ = In-phase measurement of the kth symbol in the burst
- $\tilde{Q}_k$ = Quadrature phase measurement of the kth symbol in the burst
- N = input vector length
- P_{avg} = The value for *Average constellation power*

I_k and Q_k represent the ideal (reference) values. $\tilde{I}_k$ and $\tilde{Q}_k$ the measured (received) symbols.

Equations (2.8) and (2.9) represent the EVM in dB and percentage respectively.

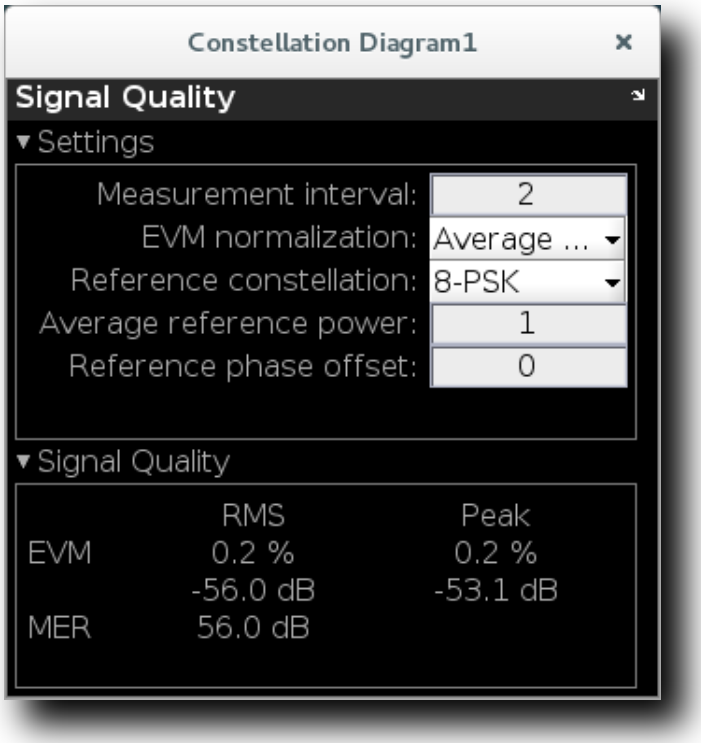
$$20logEVM_{rms} \ (dB) \tag{2.8}$$

$$EVM_{rms} * 100 \ (\%) \tag{2.9}$$

Another system performance parameter is the *Modulation Error Ratio* (MER), which is a measure of the signal-to-distortion ratio (SNR) in a digital modulation application. This type of measurement is useful for determining system performance in communication applications and is expressed in dB. See Equation (2.10).

$$REM = 10log\frac{\sum_{k=1}^{N}(I_k^2 + Q_k^2)}{\sum_{k=1}^{N} e_k^2} \tag{2.10}$$

Figure 2.28 shows a quality measurement of the received signal from our transceiver system.



Measurement interval: 2
EVM normalization: Average ...
Reference constellation: 8-PSK
Average reference power: 1
Reference phase offset: 0

Signal Quality

	RMS	Peak
EVM	0.2 %	0.2 %
	-56.0 dB	-53.1 dB
MER	56.0 dB	

Figure 2.28: Signal quality dialog box.

As we can see the quality is pretty good compared to the permissible EVM numbers for $\pi/4$ DQPSK modulation shown in Figure 2.29. But this is actually a non-fair comparison because we have not yet added any noise to the communication channel and this what we will do in the next section.

Constellation	EVM error (EVM_{dB})
π/2-DBPSK	–11 dB
π/4-DQPSK	–15 dB
π/8-D8PSK	–20 dB

Figure 2.29: Permissible EVM numbers as a function of constellation size.

2.4. System performance analysis

Communication channel model

A signal through any form of medium, a channel, is corrupted by disturbances, whether it is through a physical medium such as electronic circuitry, a wire, optical fibers, or through space. The different types of disturbances can originate from random vibrations of electrons (i.e., thermal noise) in electronic devices such as resistors and other active components, shot noise or environmental disturbances such as radiation from earth, etc. Ideally, if there are no disturbances then we could receive a signal of any size and over any distance. But that is of course not possible in a real world.

Wireless communication channels are modeled in various kinds of ways but the net effect can be described by two phenomena:

1. *modification of the transmitted signal* (characterized by the channel-impulse response). The transmitted waveforms across a channel, may experience effects like reflection for instance against buildings, absorption, attenuation (scaling of amplitude), dispersion (spreading) in time, diffraction (bending the waveform around edges of obstacles whose size is comparable to the transmitted wavelength).

2. *addition of noise.*

In this study the channel is assumed to corrupt the signal by *Additive White Gaussian noise* (AWGN) and all other channel effects are excluded. The AWGN channel thus serves as a reasonable model under these conditions, and the Hartley-Shannon law [13] gives the maximum rate for reliable communication.

In this model, the channel noise is assumed to have a Gaussian nature and is additive, meaning the noise adds to the signal. Compared to other equivalent channels, the AWGN channel does the maximum bit corruption and the systems designed to provide reliability in AWGN channel is assumed to give best performance results in other real-world channels. This model is available in Simulink and is described in Appendix G section G.1.

The effect of Noise on BER

An $\pi/4$ DQPSK receiver must decide whether the received phase is 45°, 135°, 225°, 315° or 0°, 90°, 180°, 270°. In the presence of noise, a cloud forms around each point in the constellation , see Figure 2.30, causing an error if a particular sample crosses the decision boundary.

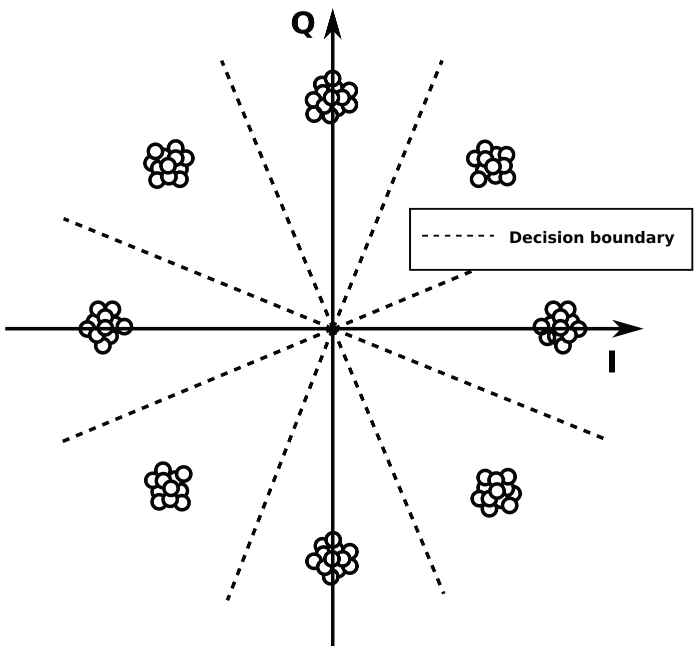


Figure 2.30: Noisy $\pi/4$ DQPSK signal.

Of course, the more noise, the wider the cloud and thus the more frequent phase errors.

The relation between signal-to-noise power ratio (SNR) and the counterparts bit-energy-to-noise density E_b/N_o and symbol-energy-to-noise density E_s/N_o from the digital domain

In digital wireless communication (DWC), bits are transmitted through the communication channel from one location to another. The bits are energy packets and so it is logical if there is a certain performance metric that is related to this fundamental objective property.
For analog communications, the *SNR* is clearly a good way of describing the quality of the received signal. The numerator represents the useful signal power, and the denominator represents the noise power that degrades the signal. For digital communications, however, a few adjustments must be made.

First, the *energy per bit* (E_b), or for higher dimensions, the *energy per symbol*(E_s) is a much more natural measure for the "strength" of the signal than the power. In itself, signal power is not very enlightening in digital communications, because the useful information is now divided into bits instead of being continuous. This means that the same average power can represent wildly different signal qualities for different bitrates, making it quite uninformative. Hence, it is better to work with the energy per bit, which can be computed by integrating the power over the lenght of a bit.
And certainly, we will show later that snr and E_b/N_o or E_s/N_o are not the same thing.

The electrical engineering (EE) community, especially the analog engineers, are very familiar with *snr* or *SNR* (i.e., *snr* in dB) but not really with E_b/N_o and E_s/N_o. In fact, the literature often refers to snr, or even the words signal-to-noise ratio, while in fact E_b/N_o is used. Fortunately, the reverse is much less common.
The E_b/N_o and E_s/N_o cannot be measured with the existing equipment, which is built for *snr* measurements. So we have to derive them from the *snr* measurements. And that's why it is important to understand the relationship between *snr*, E_b/N_o and the E_s/N_o. And also because of the fact that we need to measure BER as a function of SNR. The Matlab bit-error-rate tool (BERTool) gives a graph of the BER as a function of E_b/N_o, but not SNR.

Fortunately, we do not have to make all the derivations here, as it has already been done in Appendix I and the found relationships are shown here again.

$$\frac{E_b}{N_o} = \frac{P_s T_b}{P_n/B} = \frac{snr}{f_b B} \text{ (for uncoded per-bit based input signals)} \tag{2.11}$$

$$\frac{E_s}{N_o} = \frac{P_s T_s}{P_n/B} = \frac{snr}{f_s B} \text{ (for uncoded per-symbol based input signals)} \tag{2.12}$$

$$\frac{E_s}{N_o} = log_2 M \frac{E_b}{N_o} \text{ (relation between uncoded per-symbol and per-bit based input signals)} \tag{2.13}$$

Where,

E_b = Energy per bit in joule/bit.
E_s = Energy per symbol in joule/symbol.
N_o = Noise power spectral density in watt/Hz.
P_s = Total signal Power in watt.
P_n = Noise power in the signal bandwidth in watt.
snr = Signal-to-noise power ratio.
B = Bandwidth required for the signal in Hz.
f_b = Bit rate: the reciprocal of the bit time, the number of binary bits transmitted per second: unit is bps (bits per second).
f_s = Symbol rate: the reciprocal of the symbol time, equal to the number of signal states transmitted per second of time. Unit is baud [*].
T_b = Bit time: the time duration of an input binary bit; unit is seconds (usually microseconds).
T_s = Symbol time: the duration that an information symbol is mapped onto the signal, which equals the time duration of a signal state; unit is seconds (usually microseconds). The signal mapper converts the symbols, the desired information to be transferred, to specific signal states. For instance, the Gray constellation ordering for a QPSK signal.
M = The number of states available to the signal. On a constellation diagram, it represents the number of points.

[*] The unit *baud* (Bd) is an offical SI unit for symbol rate. Baud is named in honor of J.M. Emile Baudot (1845-1903) who established a five-bits-per-character code for telegraph use which became an international standard (commonly called the Baudet code)

Comparison between the derived relations and the relations from the Simulink AWGN channel noise model

The Simulink AWGN is a very useful block to add noise in the communication channel. But applying this block into the Simulink $\pi/4$ QPSK transceiver system was also the reason during this research to investigate the relations between the snr, E_b/N_o and E_s/N_o. We can use one of these modes of the model to inject noise into the channel and it creates confusion for people who are more familiar with snr but not with the others, simply because these units are not the same thing.

The relations between snr, E_b/N_o and E_s/N_o have already been derived in Appendix I and presented above, but to increase confidence in these relations, a final check has to be done, namely the comparison of derivative relations between E_b/E_s, E_s/N_o and snr with those of the relations from AWGN channel noise model. If the relationships match then we know exactly what and how much noise we inject into the channel. The model of the AWGN block can be described by the equations below:

$$\frac{E_s}{N_o} = \frac{T_{sym}}{T_{samp}} \cdot snr \text{ (for uncoded complex inputs)} \tag{2.14}$$

$$\left.\frac{E_s}{N_o}\right|_{dB} = \left.\frac{E_b}{N_o}\right|_{dB} + 10 \cdot \log_{10} k \ \ (dB) \text{ (for uncoded complex inputs)} \tag{2.15}$$

$$\frac{E_s}{N_o} = 0.5(T_{sym}/T_{samp}) \cdot snr \text{ (for real input signals)} \tag{2.16}$$

Where,

T_{sym} = the signal's symbol period.
T_{samp} = the signal's sampling period.
k = the number of information bits per symbol.

Because in Simulink the input and output signals of the AWGN model are complex, we only have to verify Equations 2.14 and 2.15.

Let us first re-write Formulas (2.11) and (2.12) in terms of snr.

$$snr = \frac{E_b}{N_o} \cdot \frac{f_b}{B} \tag{2.17}$$

$$snr = \frac{E_s}{N_o} \cdot \frac{f_s}{B} \tag{2.18}$$

If we equate (2.17) and (2.18) we get

$\frac{E_b}{N_o} f_b = \frac{E_s}{N_o} f_s$

and so

$$\frac{E_s}{N_o} = \frac{E_b}{N_o} \cdot \frac{f_b}{f_s} \tag{2.19}$$

in which f_b/f_s is the theoretical bandwidth efficiency and equal to the number of bits per symbol. More information about the Spectral (Bandwidth) efficiency can be found in Appendix I section I.4. Expressing formula (2.19) in dB we get

$$\left.\frac{E_s}{N_o}\right|_{dB} = \left.\frac{E_b}{N_o}\right|_{dB} + 10 \cdot log\frac{f_b}{f_s} \text{ (dB)} \tag{2.20}$$

if we compare (2.20) with (2.15) it's exactly what we expected, that they are equal, since k in the AWGN model is defined as the number of information bits per input symbol and thus $k = \frac{f_b}{f_s}$. Rewriting (2.19 in terms of symbol- and bit time we get

$$\frac{E_s}{N_o} = \frac{E_b}{N_o} \cdot \frac{T_s}{T_b} \tag{2.21}$$

from (2.17) we can write for $E_b/N_o = T_b * B * snr$ and if we fill this in (2.21) we find

$$\frac{E_s}{N_o} = T_s \cdot B \cdot snr \tag{2.22}$$

and since the AWGN block has a perfect brick wall noise power spectral density, see Figure G.2 in Appendix G section G.1, the noise bandwidth $B = 1/T_b$. Then (2.22) becomes

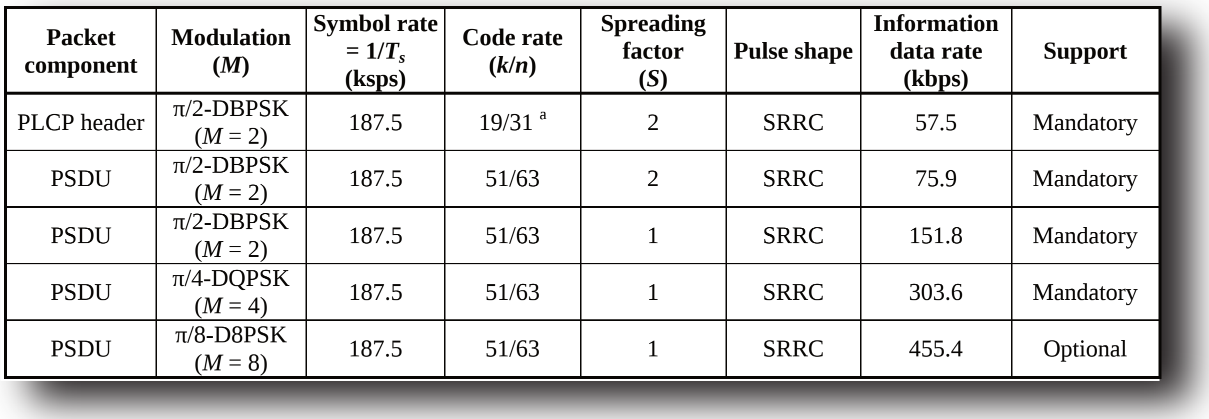
$$\frac{E_s}{N_o} = \frac{T_s}{T_b} \cdot snr \tag{2.23}$$

which is exactly the same as (2.14) because T_{sym} in AWGN model is defined as being the signal's symbol period which is equal to T_s and T_{samp} as the signal's sampling period and is equal to T_b so $T_s/T_b = T_{sym}/T_{samp}$.

So we can conclude now that the Matlab AWGN noise model relations between snr, E_b/N_o and E_s/N_o match the relations that have been derived in this subsection.

Completing the relation between snr and E_b/N_o using $\frac{\pi}{4}$ DQPSK modulation parameters

The implementation of the Phase-domain ADC receiver [10] is aiming for the IEEE 802.15.6 standard [2] using the $\pi/4$ DQPSK modulation parameters for the band 402-405MHz. A table that shows these parameters has been taken from the standard [2] and is shown in Figure 2.31.



Packet component	Modulation (M)	Symbol rate = $1/T_s$ (ksps)	Code rate (k/n)	Spreading factor (S)	Pulse shape	Information data rate (kbps)	Support
PLCP header	π/2-DBPSK (M = 2)	187.5	19/31 [a]	2	SRRC	57.5	Mandatory
PSDU	π/2-DBPSK (M = 2)	187.5	51/63	2	SRRC	75.9	Mandatory
PSDU	π/2-DBPSK (M = 2)	187.5	51/63	1	SRRC	151.8	Mandatory
PSDU	π/4-DQPSK (M = 4)	187.5	51/63	1	SRRC	303.6	Mandatory
PSDU	π/8-D8PSK (M = 8)	187.5	51/63	1	SRRC	455.4	Optional

Figure 2.31: IEEE Std 802.15.6. modulation parameters for the band 402MHz to 405MHz. [2]

So far we have found out how the snr, E_b/N_o and E_s/N_o relate to each other and to complete this relation the $\pi/4$ DQPSK modulation parameters have to be filled in these to get a precisely defined relations for the $\pi/4$ DQPSK modulation. But of course we are mainly interested in finding the relationship between snr and E_b/N_o. Lets recall Equation (2.17), we only have to fill in the values for the bit rate f_b and the channel bandwidth B to find this relation.
The bit rate f_b is equal to the *coded bit rate* and in this case is being defined as

f_b = 1/Code rate * Information data rate

The k/n from table 2.31 is being defined as the *Code Rate Factor.* In telecommunication and information theory, the *code rate* (or *information rate*) of a forward error correction code is the proportion of data-stream that is useful (non-redundant). That is, if the *code rate* is k/n, for every k bits of useful information, the coder generates totally n bits of data, of which $n-k$ are redundant. From table 2.31 the code rate for $\pi/4$ DQPSK modulation is 51/63 and it means that from every 63 bits, 51 are useful information. However, for the transmitter hardware there's no difference between whether a bit used in the modulation is actual overload due to coding. Incoming bits for the DWC transmitter are just that —bits. What the bits represent, including whether they are coded or not from the underlying information, is *immaterial to the DWC hardware.* From the table we see that the *Information data rate*, which is the *uncoded bit rate*, is 303.6 kbps. Then the coded bit rate becomes

$f_b = 63/51 * 303.6 = 375\ kb/s$

The standard states that the channel bandwidth B from the transmitter specifications is 300kHz. Now equation (2.17) becomes

$$snr = \frac{E_b}{N_o}\cdot\frac{f_b}{B} = \frac{E_b}{N_o}\cdot\frac{\frac{1}{CodeRate}\cdot Information\ data\ rate}{channel\ bandwidth} = \frac{E_b}{N_o}\cdot\frac{\frac{63}{51}\cdot 303.6Kbps}{300kHz} \approx \frac{E_b}{N_o}\cdot 1.25 \tag{2.24}$$

In dB

$$SNR = \left.\frac{Eb}{No}\right|_{dB} + 0.9691\ \ \text{(dB)} \tag{2.25}$$

This result reveals that in case of $\pi/4$ DQPSK communication the difference between E_b/N_o and SNR is almost 1dB.

Figure 2.32 shows a graphical representation of the $\pi/4$ DQPSK modulation parameters in a transceiver chain. The data rate is shown at different stages in the transceiver chain. At the input of the transmitter we see the uncoded information signal with data rate of 303.6kb/s which has to be coded by the coder with some redundant bits to increase the reliability of communication across the channel. In general, a wireless channel is of low to medium reliability [13]. The probability of making an error of a decision of what was actually send at the transmitter is rather high, ranging from 0.1% to 1% [13]. But we require any communication across this channel to be of high reliability with probability of error well below one part of a million (0.0001%) [13]. With the use of coding we can achieve this goal because although the signal is deteriorating due to all sorts of disruptions on its way through the communication channel, we can still decipher the signal into the original with the help of coding. For the $\pi/4$ DQPSK modulation scheme a coded data rate of 375 kb/s is required. Since this coding scheme uses 2 bits per symbol, the modulator will lower the data rate to 187.5 kb/s. According to the standard [2] the $\pi/4$ DQPSK modulation operates in the band of 402MHz to 405MHz and since the information signal bandwidth is 300kHz we can have up to 10 channels operating in this frequency band. A matched SRRC filtering is required to allow Nyquist signaling and in the receiver the reverse operation of the transmitter will occur.

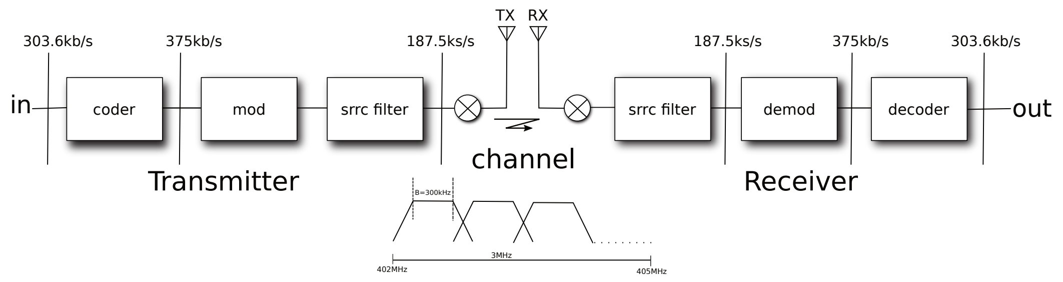


Figure 2.32: Transceiver system representing the IEEE Std 802.15.6 $\pi/4$ DQPSK modulation parameters for the band 402MHz to 405MHz.

Table 2.3 shows a series of E_b/N_o and SNR values that were used for the BER simulations. We consciously chose this range in the vicinity of the E_b/N_o of 12 dB, where the bit error rate is about 1.10^{-5}, to see the effect of the noise on the bit error rate. The amount of noise injected into the communication channel is determined by the E_b/N_o value entered in the Simulink AWGN block. We start with the highest E_b/N_o and then go down in steps of 1 dB. The corresponding SNR has also been calculated from the E_b/N_o to indicate the BER as a function of the SNR, because the SNR is the most familiar performance metric used as the measure of the quality of the signal.

To verify the derived SNR from the E_b/N_o, a Simulink model for simulating the SNR in a channel bandwidth of 300kHz has been used. See Figure M.1 on Appendix M section M.1. The simulated values of SNR can be found in the most right column of Tabel 2.3. These simulated values have been compared with the calculated SNR shown in the second column of the Tabel and as we see the values are quite similar. For the determination of simulated SNR, noise in terms of E_b/N_o is injected into the channel and signal power before the noise injection is measured by means of an RMS power meter and noise power is measured by subtracting the signals after and before the noise injection from each other. Noise is measured in a 300 kHz band with the help of a spectrum analyzer. See Figure M.1. The ratio between measured signal power and the noise power is then the SNR.

Calculated values $SNR = \frac{E_b}{N_o}\Big\|_{dB} + 0.9691\ \ (dB)$		Simulated values Simulated Time = 10 sec. Number of spectral averages = 10000 Input signal power = 10.49 dBm@1Ω Channel bandwidth = 300 kHz SNR = Signal-Noise (dB)	
E_b/N_o (dB)	**SNR (dB)**	**Channel noise power@1Ω (dBm)**	**SNR (dB)**
23	23.9691	-13.472	23.962
22	22.9691	-12.472	22.962
21	21.9691	-11.472	21.962
20	20.9691	-10.472	20.962
19	19.9691	-9.472	19.962
18	18.9691	-8.472	18.962
17	17.9691	-7.472	17.962
16	16.9691	-6.472	16.962
15	15.9691	-5.472	15.962
14	14.9691	-4.472	14.962
13	13.9691	-3.472	13.962
12	12.9691	-2.472	12.962
11	11.9691	-1.472	11.962
10	10.9691	-0.472	10.962
9	9.9691	0.528	11.018

Table 2.3: Conversion Table E_b/N_o to SNR.

Including the effect of phase quantization noise

In addition to the noise in communication channel, we also have to take into account phase quantization noise from the PhADC. White noise, or thermal noise, of the PhADC is negligibly small compared to the phase quantization noise [10]. However, which is not the case if the resolution of the PhADC is high such that the phase quantization noise is much lower than thermal noise of the receiver. But white noise level of this 5 bit PhADC is so small that it has no effect at all on the BER. It is much smaller than the rms value of the quantization noise signal given by

$$V_{Q_{(rms)}} = \frac{V_{LSB}}{\sqrt{12}} \tag{2.26}$$

assuming the quantization noise signal is uniformly distributed over the interval $\pm V_{LSB}/2$.

This quantization noise adds up to the channel noise and because the PhADC has a rather low resolution of 5 bit, therefore relatively large quantization noise, it will contribute to a higher bit error rate. A low resolution PhADC has considerably larger quantization noise then a PhADC with much higher resolution. The noise power decreases by 6 dB for each additional bit in the PhADC converter.

So we can also conclude here that our Simulink IQ offset compensation model is modeled correctly, because there is channel noise, IQ offset and the phase quantization noise of the PhADC.

The model of the PhADC shown in Figure 2.21 is now being expanded with phase quantization modeling. See Figure 2.33.

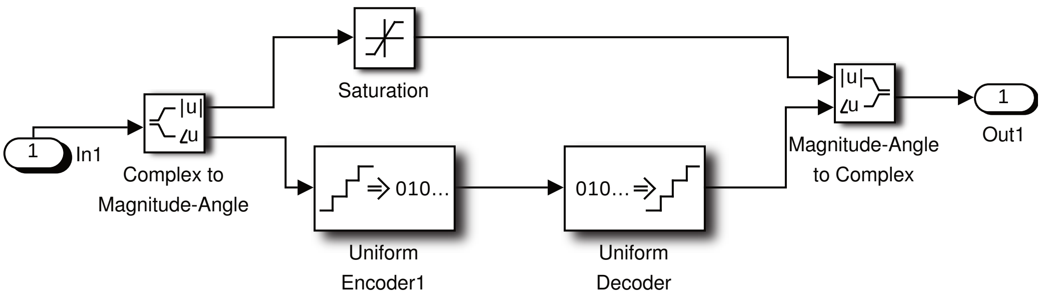
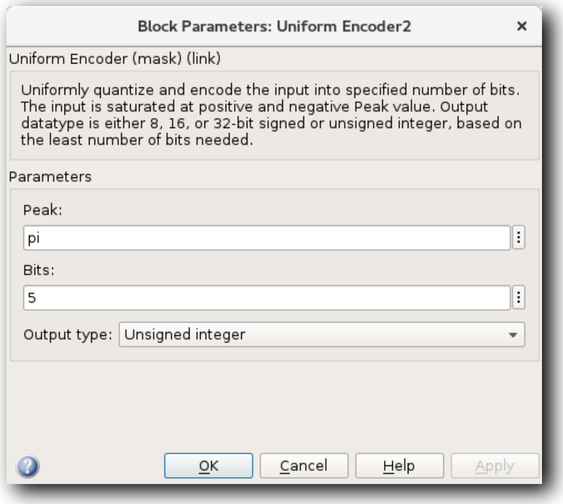


Figure 2.33: Simulink structure model of an Ideal 5 bit Phase-Domain ADC.

The phase quantization is done by the *Uniform Encoder* block, so it converts the real discrete phase signal to an integer one. Then in turn it is converted from digital back to real discrete again. The most important objective is to add the effect of phase quantization noise in the model. The phase signal $\angle u$ goes from $-\pi$ to $+\pi$ and the PhADC has a resolution of 5 bits. Figure 2.34 shows the settings of the *Uniform Encoder* and *Decoder* block. It seems confusing that both blocks have the same *negative- and positive input peak values* since at least the decoder input is an unsigned integer from -16 to +15 that represents the digital phase signal while the encoder input is a double from $-\pi$ to $+\pi$, which is the discrete real value of the phase. The reason is that to correctly decode values encoded by the *Uniform Encoder* block, the *Bits* and *Peak* parameters of the *Uniform Decoder* block should be set to the same values as the *Bits* and *Peak* parameters of the *Uniform Encoder* block.


Block Parameters: Uniform Encoder2

Uniform Encoder (mask) (link)

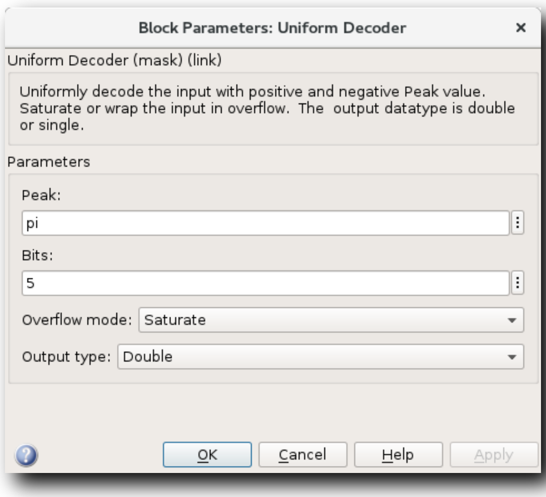
Uniformly quantize and encode the input into specified number of bits. The input is saturated at positive and negative Peak value. Output datatype is either 8, 16, or 32-bit signed or unsigned integer, based on the least number of bits needed.

Parameters

Peak: pi

Bits: 5

Output type: Unsigned integer


Block Parameters: Uniform Decoder

Uniform Decoder (mask) (link)

Uniformly decode the input with positive and negative Peak value. Saturate or wrap the input in overflow. The output datatype is double or single.

Parameters

Peak: pi

Bits: 5

Overflow mode: Saturate

Output type: Double

(a) (b)

Figure 2.34: Function block parameters dialog box of Encoder (a) and Decoder (b).

2.4.1. Bit Error Rate (BER) measurement as a function of channel- and quantization noise

For the simulation of BER as a function of channel noise and quantization noise we use the Simulink model shown in Figure J.1 from Appendix J. As we can see from the model, the feedback loop for the IQ offset compensation, as shown in Figures F.1 and F.3 in Appendix F, has been omitted because for this objective we only want to see the effect of channel noise on the BER and not on the offset so we don't need it. And Figure J.2 shows a Matlab script to generate the data for the curves shown in Figures 2.36 and 2.37. But it is still advisable to apply the *Matlab BERTool* for these simple simulations because it will collect data and generate these curves automatically, if set properly. It can be started with the command *bertool* in the Matlab command window.

The PhADC receiver [10] system performance evaluation is shown in Figure 2.35. It is important that the BER simulations are performed at worst case conditions and therefore at minimum information input signal level of PhADC. In this case, according to the system performance chart from the Phase-Domain ADC receiver [10], this is -6.5 dBm ($0.3V_{pp}$) at 50Ω load. See Figure 2.35(b).
If the signal level at the PhADC input is set to full scale, which is 1 Vpp (equivalent to +4 dBm in a 50Ω system) this corresponds to a antenna input level, for the highest gain setting of 77.1 dB (36.6 dB of PFE + BB LNA and 40.5 dB of analog BB), of -73.1 dBm. See Figure 2.35(a). The phase SNDR (signal-to-noise + distortion ratio) is 30.8 dB [10] which is equivalent to that of an ideal PhADC with an input signal of +4 dBm and an input-referred amplitude noise of -16.8 dBm, as indicated by Equation 2.28. The noise level at the output of the analog BB is -27.5 dBm, and the total noise level at the interface of analog BB and the PhADC is -16.4 dBm (the sum of -16.8 dBm and -27.5 dBm). Thus, the overall SNR at the interface is 20.4 dB (= +4 dBm + 16.4 dBm). See Figure 2.35(a).
If the signal level at the PhADC input is set to the lowest of the measured range, i.e., -6.5 dBm (0.3 V_{pp}), as shown in Figure 2.35(b) this corresponds to an antenna input level, for the highest gain setting of 77.1 dB, of -83.6 dBm. The resulting phase SNDR is 26.6 dB [10] and the equivalent amplitude noise of the PhADC is -23.1 dBm, as indicated by Equation 2.28. Comparing this equivalent input noise with that shown in Figure 2.35(a) (-16.8 dBm), it is important to note that the noise level decreases with the signal level, which is a favorable effect introduced by the PhADC's immunity to amplitude variations. Accordingly, the overall noise at the interface of analog BB and the PhADC is -21.8 dBm (the sum of -23.1 dBm and -27.2 dBm). Thus the overall SNR at the interface is 15.3 dB (=-6.5 dBm + 21.8 dBm). See Figure 2.35(b).
The overall amplitude SNR would be reduced from 20.4 dB to 9.9 dB for a system using an amplitude ADC when the antenna input is reduced from -73.1 dBm to -83.6 dBm. While it is 15.3 dB for the system using a PhADC. However, the phase SNR of the PhADC still has dropped by 4.2 dB (30.8 dB - 26.6 dB) rather than remaining constant as the input amplitude reduces from +4 dBm to -6.5 dBm. This is because the amplitude nonidealities, such as noise and offset, become more pronounced as the input amplitude decreases, thereby degrading the output phase quality. And that is why we can say that, for testing the system performance in terms of BER, setting a signal level at the input of PhADC to the lowest of the measured range is a worst case condition.

The signal powers in Simulink are referred to a load of 1Ω and so to make a good comparison we need to convert the signal input power of the PhADC, which is normalized to a 50Ω load, to a signal power normalized to a load of 1Ω. Using Equation 2.27 for a input signal level of -6.5 dBm (223.8 μ W) normalized to a load of 50Ω equals a power of 10.49 dBm (11.2 mW) normalized to a load of 1Ω.

$$P_{w,1\Omega} = 50 \times 10^{\frac{P_{dBm,50\Omega} - 30dB}{10}} \tag{2.27}$$

The signal power from the transmitter output to the receiver input in the Simulink model is however 11.94 dBm and so it has to be attenuated with 1.45 dB by simply putting a gain block at transmitter output with a gain of -1.45 dB.

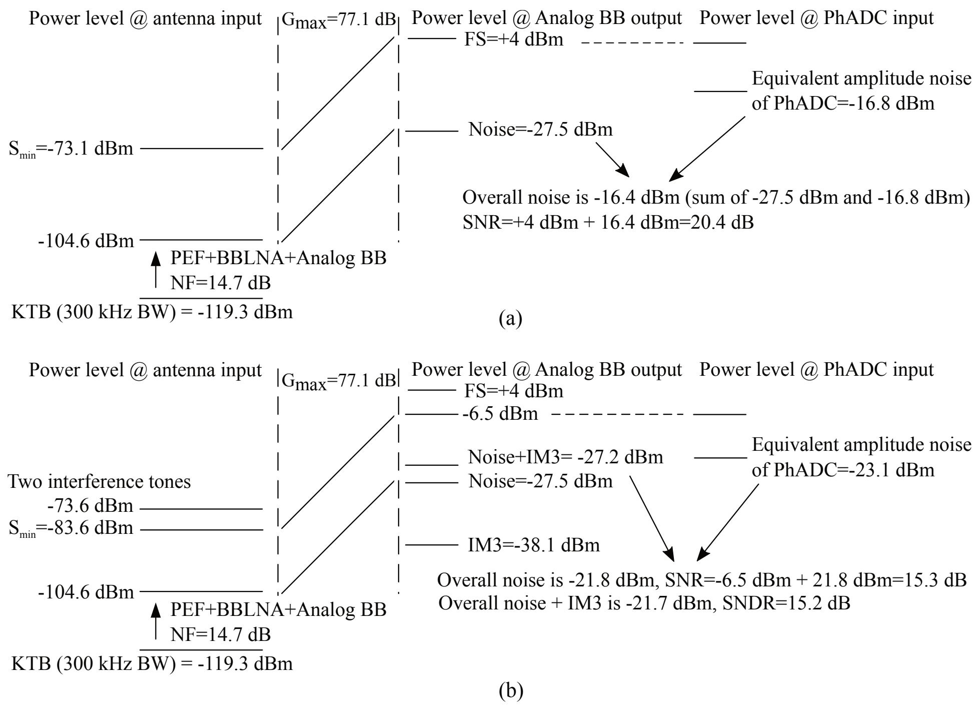

Power level @ antenna input
Gmax=77.1 dB
Power level @ Analog BB output
Power level @ PhADC input
FS=+4 dBm
Equivalent amplitude noise of PhADC=-16.8 dBm
Noise=-27.5 dBm
Smin=-73.1 dBm
Overall noise is -16.4 dBm (sum of -27.5 dBm and -16.8 dBm)
SNR=+4 dBm + 16.4 dBm=20.4 dB
-104.6 dBm
PEF+BBLNA+Analog BB NF=14.7 dB
KTB (300 kHz BW) = -119.3 dBm
(a)
Power level @ antenna input
Gmax=77.1 dB
Power level @ Analog BB output
Power level @ PhADC input
FS=+4 dBm
-6.5 dBm
Noise+IM3= -27.2 dBm
Equivalent amplitude noise of PhADC=-23.1 dBm
Two interference tones -73.6 dBm
Noise=-27.5 dBm
Smin=-83.6 dBm
IM3=-38.1 dBm
-104.6 dBm
Overall noise is -21.8 dBm, SNR=-6.5 dBm + 21.8 dBm=15.3 dB
Overall noise + IM3 is -21.7 dBm, SNDR=15.2 dB
PEF+BBLNA+Analog BB NF=14.7 dB
KTB (300 kHz BW) = -119.3 dBm
(b)

Figure 2.35: PhADC receiver system performance evaluation for an input level of -73.1 dBm (a) and -83.6 dBm (b). [10]

The very first thing that has to be done is to verify the Simulink transceiver model for $\pi/4$ DQPSK communication. And this can be done using the Matlab BERTool [19] to simulate the systems BER using only channel noise and comparing this against the theoretical BER for a $\pi/4$ DQPSK modulation. These two curves are indicated in blue and red in Figure 2.36 en 2.37. As can be seen, the curves are almost perfectly aligned and this means that the model is correctly set up and works very well. A small detail is that the blue curve does not go all the way to the bottom of the graph, because due to long simulations, the time is shortened by limiting the number of test bits to 10^7.

The IEEE 802.15.6 standard requires an E_b/N_o of 12 dB for $\pi/4$ DQPSK demodulation with a BER of 10^{-5} which can also be read from the red curve in Figure 2.36. And in 2.37 we see the same graphs but then as a function of SNR. At a BER of 10^{-5} we read a value for the SNR of 13 dB. So, to achieve a BER of 1.10^{-5} an SNR of 13 dB is required.

The third curve, in yellow, is the BER simulation including channel- and phase quantization noise. The curve has shifted a bit more upwards showing that phase quantization does have an influence on the BER. As a consequence, to achieve a BER of 10^{-5} we need an Eb/No of at least 13 dB or an SNR of at least 14 dB.

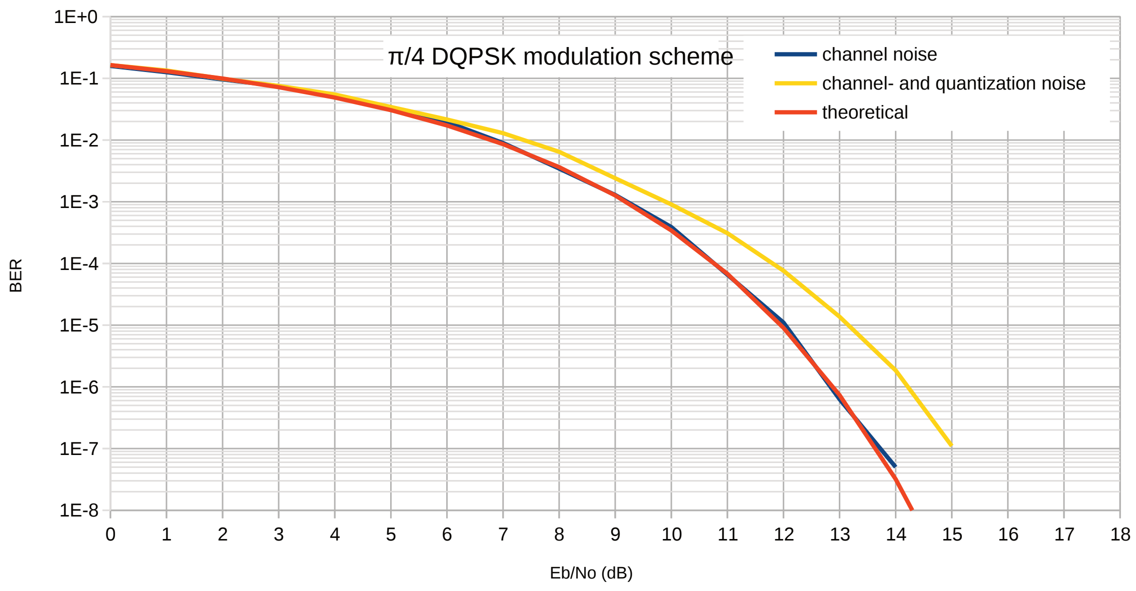


Figure 2.36: The simulated and theoretical BER as a function of channel- and Phase-Domain ADC quantization noise in terms of E_b/N_o for a $\pi/4$ DQPSK modulation scheme.

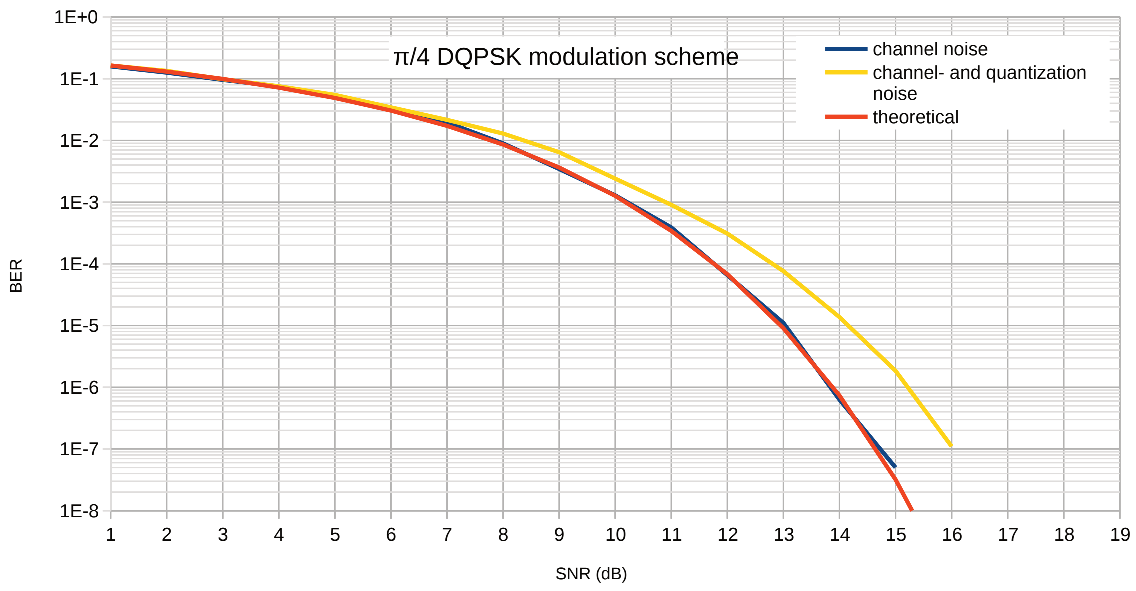


Figure 2.37: The simulated and theoretical BER as a function of channel- and Phase-Domain ADC quantization noise in terms of SNR for a $\pi/4$ DQPSK modulation scheme.

2.4.2. Bit Error Rate (BER) measurement as a function of IQ offset

For this simulation we only look at the effect of IQ offset on the BER and noise is not included. For this simulation the model is applied as shown in Figure K.1 of Appendix K section K.1. In the model we see that the AWGN channel noise block has been removed and instead an offset $IQ_{\_offset}$ in I and Q channel is added.

For the following curves of BER as a function of IQ offset, the offset for I and Q is set as $I = IQ\_{offset}$ and $Q = IQ\_{offset}$. So the I and Q have the same value at every point in the x-axis. This is deliberately done because simulations showed a higher BER when both both I and Q get a certain offset than when only I or Q have an offset. We assume that the phase vector distribution imbalance is highest when both I and Q move away from the origin.
Without noise and IQ offset, there should be no bit errors as the number of bits used in $\pi/4$ DQPSK is 2 and the resolution of the PhADC is infinite, which of course is infinitely more than needed. In other words, there is no error mechanism that can cause the bits to be incorrectly detected. Hence, the BER will be 0.
Let's take a look at the results shown in Figure 2.38. This test was done at 10^7 bits and as we can see up to an offset of about 30mV there is just one bit error. It is expected to be 0, but we do see 1 bit error (i.e., BER=10^{-7}) and this might arise from one single artefact in the beginning of 10^7 bits. Of course, this result is still much better than the required BER of 10^{-5}. So this part of the curve tells us that an IQ offset up to a maximum of 30mV is tolerable and still gives virtually no bit errors. If the offset is increased above this value then we see bit errors. This region shows a quite sharp transition, a threshold. Below 30mV we hardly see errors and between 30 and 35mV we will have a BER in the order of 2.10^{-7} to 0.06, i.e., from 2.10^{-5}% to 6%. Since DC offset is a constant DC value rather than a noise-like disturbance, i.e., constant deterministic interference, the relationship with a sharp transition makes more sense suggesting that the BER doesn't increase smoothly with DC offset as is in the case of noise like shown in Figures 2.36 and 2.37. Noise in the signal chain prior to the PhADC also blurs the constellation (the points become clouds) and this every now and then leads to incorrect detection of the phase. Of course. the more noise, the wider the cloud and thus the more frequent the phase errors. For IQ offset it is different, slightly below a IQ offset of 30 mV you have no errors at all and a little above you will have a BER in the order of 2.10^{-5}% to 6%. Continuing with Figure 2.38, beyond an IQ offset of 35mV, the BER increases even further, appearing as a staircase, like an ADC output and so it grows further and further to a BER of 1.

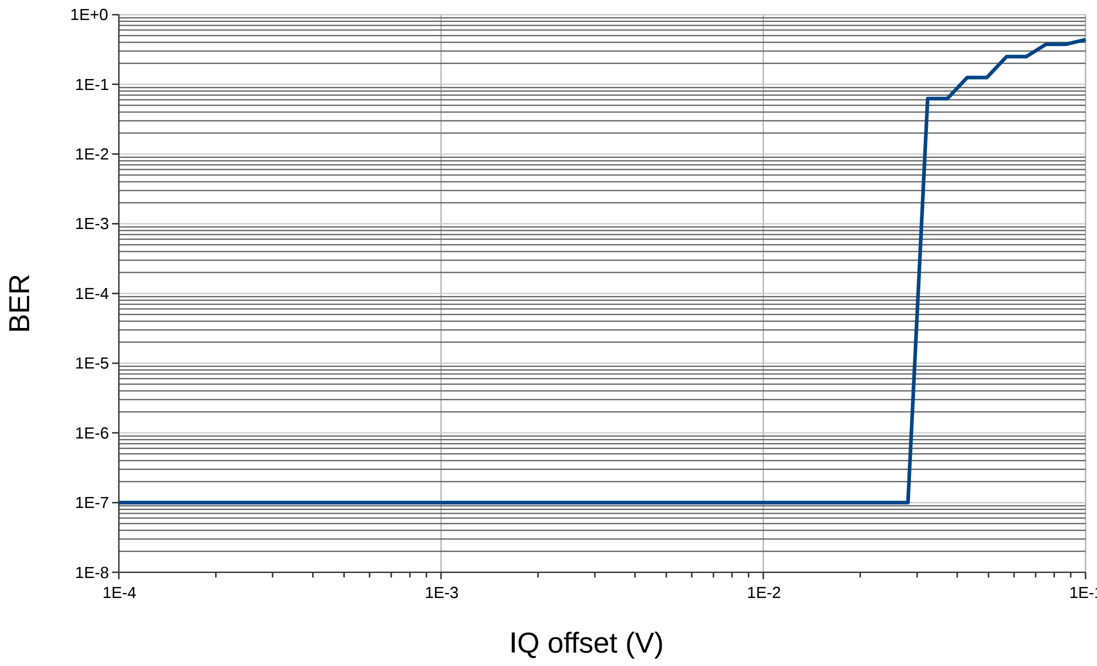


Figure 2.38: Plot of BER as a function of IQ offset of the Simulink transceiver system with matched SRRC filters and Ideal PhADC using $\pi/4$ DQPSK communication parameters.

2.4.3. Bit Error Rate (BER) measurement as a function of both IQ offset and noise

Previously we have tested the effect of IQ offset on the BER, but now noise is included to the offset. See Figure L.1 in Appendix L Section L.1. At each injected channel noise value, the offset is swept from $100\mu V$ to $100mV$ and a new curve of BER as a function of this offset is included in the graphs below. Noise in the signal chain prior the PhADC will blur the constellation (i.e., the points become clouds) and this will lead to an increase of phase errors, hence, BER. The more noise, the wider the cloud and thus the more frequent phase errors.

BER as a function of only IQ offset and channel noise

The Figures 2.39 and 2.40 show BER as a function of IQ offset and channel noise, no quantization noise. That means the PhADC is ideal in sense of having no phase quantization, i.e., has an infinite resolution. The Figures show no doubt the same curves but in Figure 2.39 the injected channel noise is in terms of E_b/N_o and in Figure 2.40 in terms of the SNR.

If we continue to compare, the Figures 2.41 and 2.42 are basically the same as the ones previously mentioned, but the curves in latter are all shifted upwards due to the added phase quantization noise of the PhADC. And since we want to implement a 5 bit PhADC in the Simulink model, we go directly to the last two Figures to interpret the simulation data there. It has no added value to treat both 2.39 and 2.40 here. The transceiver model with the 5 bit PhADC implementation gives a more realistic picture of the real Phase-Domain ADC receiver [10] receiver. Using the Ideal PhADC, i.e., with no phase quantization, is just a tool to show the effect of only IQ offset and channel noise on the BER without phase quantization. Hence, we can see how much the BER has increased when we add phase quantization to the model.

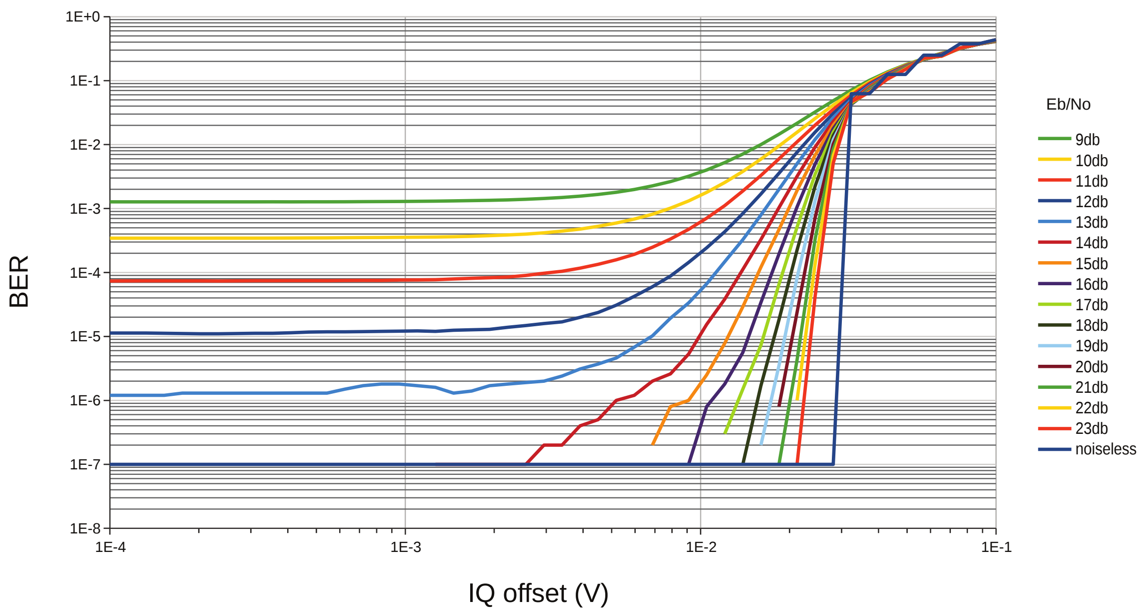


Figure 2.39: Plot of BER as a function of IQ offset and channel noise E_b/N_o of the Simulink transceiver system with matched SRRC filters and Ideal PhADC using $\pi/4$ DQPSK communication parameters.

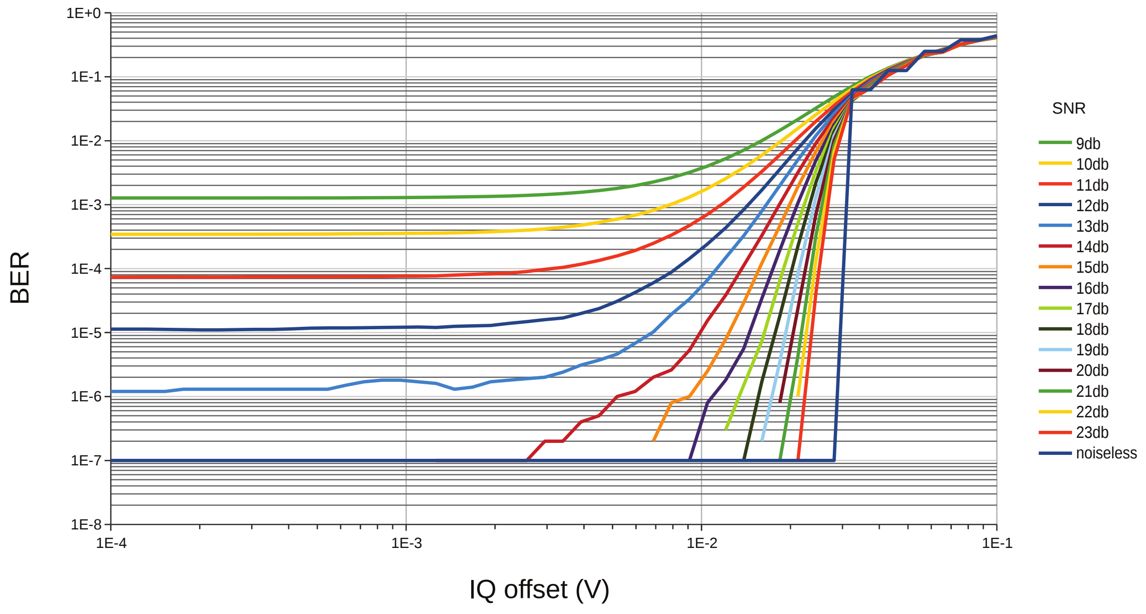


Figure 2.40: Plot of BER as a function of IQ offset and channel noise SNR of the Simulink transceiver system with matched SRRC filters and Ideal PhADC using $\pi/4$ DQPSK communication parameters.

BER as a function of only IQ offset, channel noise and phase quantization noise

Let we neglect the IQ offsets at the moment (i.e., looking at the very left axis of the results of Figure 2.42. First of all, the ADC model used in Simulink is an ideal 5 bit PhADC, rather than the real implemented PhADC (i.e., Phase Domain ADC with lots of non idealities [10]) where the performance is shown in Figure 2.35. The Phase SNR of the PhADC ($SNR_{\varphi,ph}$) is related to the amplitude SNR of the IQ ADC ($SNR_{am,ph}$) as

$$SNR_{\varphi,ph} = SNR_{am,ph} + 10 \ \text{dB} \tag{2.28}$$

[8]

and

$$SNR_{\varphi,ph} = 6.02 * 5 + 1.76 = 31.76 \ dB \tag{2.29}$$

[8]

This means that the input equivalent amplitude SNR of the ideal 5-bits PhADC, is equal to

$SNR_{am,ph} = 6.02 * 5 + 1.76dB - 10 = 21.76$ dB.

Given this number, we should expect that when the preceding SNR (i.e., without considering the noise of the PhADC)

$SNR_{prec} = P_{s_{min}} - Injected\ Noise = -6.5 - Injected\ Noise$ dB

is > 22 dB, $SNR_{\varphi,ph}$ is dominant and hence BER should be more or less independently on SNR_{prec}. We can somehow see it from Figure 2.42, since BER stays around 10^{-7} from noiseless to $SNR = 17$ dB. When $SNR_{prec} < 22$ dB, SNR_{prec} becomes more dominant and hence BER should increase as SNR_{prec} decreases. However, we don't see it in 2.42 for the curves of SNR_{pred}=17, 18, 19, 20, 21, 22, 23, 24 dB, since BER stay around 10^{-7}. This might because, for these good SNRs, the BER is way better than 10^{-7}, but the number of bits in simulations is limited to 10^7, and the simulated BER of 10^{-7} as said before, might arise from one single artefact in the beginning of the 10^7 bits. When $SNR_{prec} < 16dB$, we do see the expected increasing of the

BER. In order to see the increase in BER for the curves SNR_{prec} = 17, 18, 19, 20, 21, 22, 23, 24 dB at very low IQ offsets we should do a test at higher number of bits. For example, at 10^{10} bits. But the simulations will take so long that is almost impossible to do simulations for a whole batch of different values for noise and IQ offset. In summary, considering a 5-bit Ideal PhADC, Figure 2.42 makes sense.
How should we use Figure 2.42 to determine the maximally allowed offset for a BER of 10^{-5}? We need to see it actually as follows. When the offset is negligible (very left of the X-axis), BER stays around 10^{-7} from noiseless to approximately 17 dB, suggesting that the quantization noise of the PhADC is dominated. From 17 dB on, increases as SNR_{prec} decreases, suggesting preceding noise becomes dominated. Considering the SNR of Phase Domain ADC [10], using Figure 2.35(b)

$$SNR_{am,ph} = -6.5 + 23.1 = 16.6 \ dB$$

then somehow 17 dB is indeed the moment where both noise sources are in the same level. Following text is a more detailed explanation of the results from Figure 2.42. At this moment, the targeted bit rate and channel bandwidth comes into play. Suppose, for a given SNR, if we increase the bandwidth efficiency (i.e., f_s/B), BER will increase because now data becomes more susceptible to the noise. If SNR_{prec} = 21 dB (which actually is the SNR_{prec} shown in the performance chart Figure 2.35(a)), finding the curve corresponding to SNR = 21 dB, then it should be the brown curve for SNR = 20.97. Reading the X-axis location at the intersection between this curve and that of BER = 10^{-5}, the maximally allowed offset is approximately 17 mV.

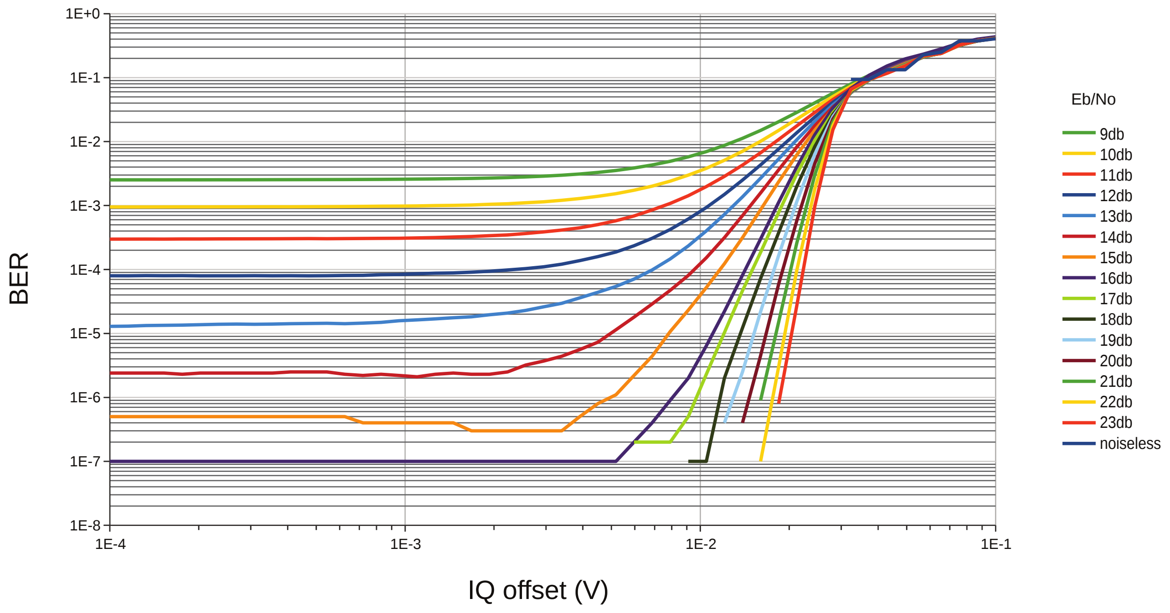


Figure 2.41: Plot of BER as a function of IQ offset and channel noise E_b/N_o including quantization noise E_b/QN_o of the Simulink transceiver system with matched SRRC filters and 5 bit ideal PhADC using $\pi/4$ DQPSK communication parameters.

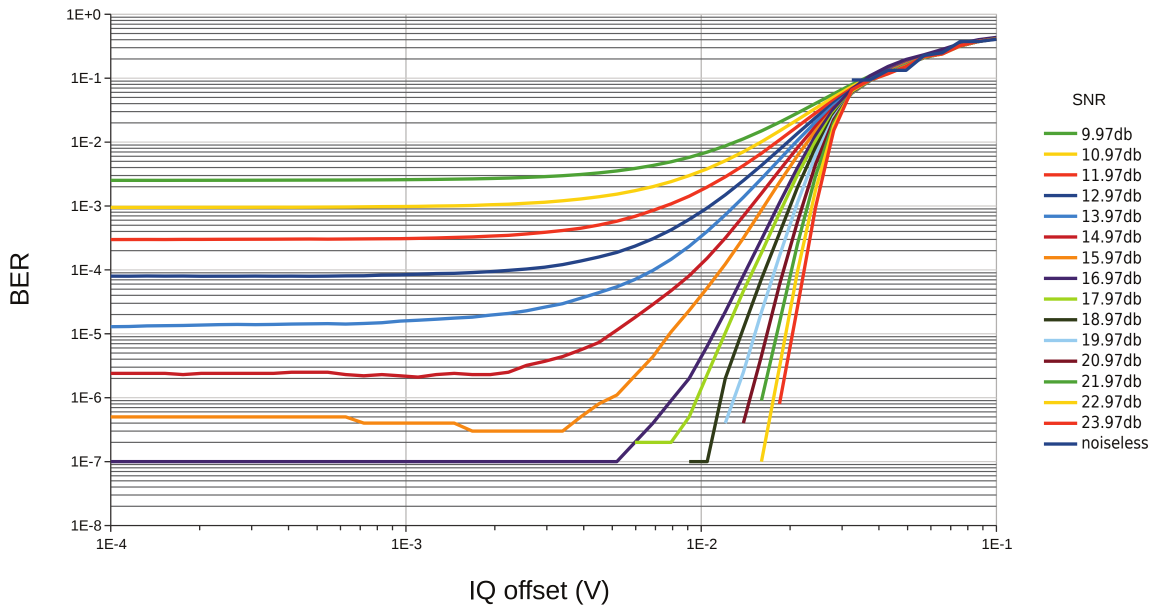


Figure 2.42: Plot of BER as a function of IQ offset and channel noise SNR including quantization noise SQNR of the Simulink transceiver system with matched SRRC filters and 5 bit ideal PhADC using $\pi/4$ DQPSK communication parameters.

However, if we now reduce the bandwidth efficiency (f_s/B), the required SNR_{prec} for a given BER will decrease and all the curves will be lower. Now using SNR_{prec} =21 dB again, we will get a higher IQ offset number. This make intuitive sense since bandwidth efficiency decreases, data is more robust and hence more offset is allowed. Because according to this data means that BER depends on the bit rate and bandwidth of the communication channel, this means once again that these simulations had to be performed together with the $\pi/4$ DQPSK modulation parameters.

2.4.4. Modeling an analog channel select receive filter

Introduction

In the "ideal" descriptions of digital modulated signals, transitions between successive states are usually taken to be instantaneous. While this is mathematically convenient, the bandwidth occupied by such a signal is impractical in modern applications. In effectively every instance the frequency allocated to any transmitted signal is a finite range, which requires bandlimiting of the DWC (Digital Wireless Communication) signal. How much filtering to use, and of what kind, is a major decision in DWC system design.

Intersymbol Interference (ISI)

ISI is a necessary consequence of partial response filtering, i.e. linear time-invariant systems can "distort" a signal if they do not provide sufficient bandwidth (like a poor designed low pass filter can do). It is the effect of any input symbol that is spread across several output symbol times. Each bit level is corrupted by decaying tails created by previous bits. Thus in the output signal there are effects from several input symbols present at any particular time. This phenomenon leads to a higher error rate because it brings the peak levels of ONEs and ZEROs closer to the detection threshold. In general, any system that removes part of the spectrum of a signal introduces ISI.

Nyquist filters

In 1928 Harry Nyquist worked out the characteristics that a filter (or equivalently, a channel) must have to eliminate ISI at specific and regular sampling instants. Though he was working on ways to improve the telegraph service then in wide use, his results are equally valid for DWC. These filters are known as Nyquist filters, in honor of his monumental achievement. Nyquist filters are a special case of partial response filtering where the ISI is designed to be zero at the exact sampling time. The rules needed to generate a Nyquist filter are rather simple. They are:

1. Define the filter bandwidth as the 6 dB bandwidth
2. Set this bandwidth at one-half the symbol rate (the Nyquist frequency)
3. Provide odd frequency-domain symmetry (in linear magnitude) about the -6dB point

The response of a Nyquist filter exceeds the bandwidth of the ideal brickwall filter by a factor of alpha (α) both above and below the ideal cutoff frequencies. See Figure 2.43(b). The exact value for the excess bandwidth factor α and the duration of the Nyquist filter pulse shape is implementation dependent. Typical values of α are in the range of 0.3 to 0.5. Also, α is always defined normalized to the Nyquist frequency $1/(2T_s)$. Clearly, α cannot exceed 1. Because of the odd-symmetry of the added response, just as the response bandwidth increases by factor α, the unit magnitude pass-band bandwidth shrinks by the same factor α.

By very large margin, the most well known Nyquist filter is the raised cosine filter. This just means that the shape of the spectral response in linear units, seen in Figure 2.43(b), follows the cosine shape. What pulse shape yields the tightest spectrum? Ideally, for $\alpha = 0$, this would be a spectrum of a rectangular pulse and a sinc pulse in the time domain. For a rectangular pulse of width T_b (and unity height),

$$p(f) = T_b \frac{sin \pi f T_b}{\pi f T_b} \tag{2.30}$$

In practice, sinc pulses are difficult to generate and approximations are used instead. A common pulse shape is shown in Figure 2.43(a) and expressed as

$$p(t) = \frac{sinc(\pi / T_s)}{\pi t / T_s} \frac{cos(\pi \alpha t / T_s)}{1 - 4\alpha^2 t^2 / T_s^2} \tag{2.31}$$

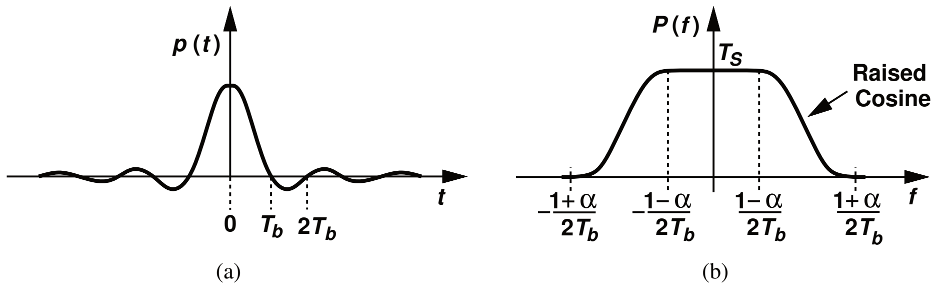


Figure 2.43: Raised-cosine pulse shaping: basic pulse (a) and corresponding spectrum (b). (Pictures courtesy of Behzad Razavi.)

Matched filtering

In principle, for a minimum-bandwidth transmitted signal all of the signal bandlimiting should be performed by the filter at the transmitter. While this may be a practical method for DWC applications which are not noise-limited, nearly all DWC applications are noise-limited. This unfortunate reality causes DWC engineers to rethink the actual approach needed in our actual hardware. In any noise-limited application the additive noise from the transmission channel must be removed in the receiver, outside of the signal bandwidth, before demodulation. The action of this receive-noise bandlimiting filter also further bandlimits the incoming signal. This changes the shape of the transmitted signal before it gets to the receive demodulator, likely destroying any carefully crafted signal characteristics present at the transmitter output. At the receiver demodulator, decision errors are minimized when the signal to noise ratio into the demodulator is as high as possible. The receiver filter has a dominant impact on this performance requirement, for two reasons. First, because it is restricting the input noise power due to its bandlimiting function – this is good. Second, this same bandlimiting operation also reduces the bandwidth into the demodulator of the input signal – this may be bad. By definition, a matched filter is the one filter design which provides the best tradeoff here, measured by providing a maximum signal to noise ratio at its output. See Figure 2.44.

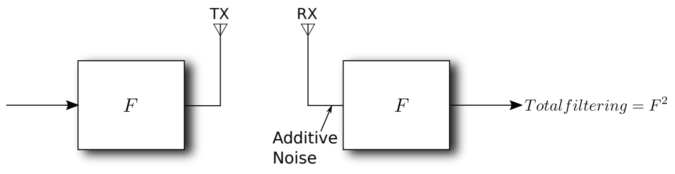


Figure 2.44: Series filtering at both the TX and RX. If the same filter F is used in both the transmitter and the receiver, double filtering results.

Using this same filter in the receiver removes the added noise to exactly the same bandwidth as the transmitted signal. However, at the same time the DWC signal is now filtered twice. Following this receive filter the zero-ISI characteristic of the transmitter Nyquist filter is destroyed. But we know that the design of Figure 2.44 is matched, so this provides a certain SNR. But is this the best SNR that can be done? No – significant improvements can be made while still using matched filtering. Consider the design of Figure 2.45. Dividing the full signal filtering equally between the two sides is an option. Since compound filtering is multiplicative, we allocate the square-root of the filter response F to both the transmitter and the receiver.

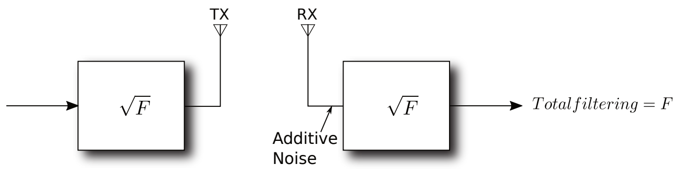


Figure 2.45: Equally dividing the desired total filtering F between the transmitter and the receiver.

So, ideally, in a digital transceiver system, the square-root raised cosine filter is decomposed into two sections, one placed in the transmitter and the other in the receiver [15], see Figure 2.46. Using a transfer function equal to the square root of $p(f)$ shown in Figure 2.43(b), the two sections together allow Nyquist signaling while the section used in the receiver also operates as a matched filter. This technique is described in [7].

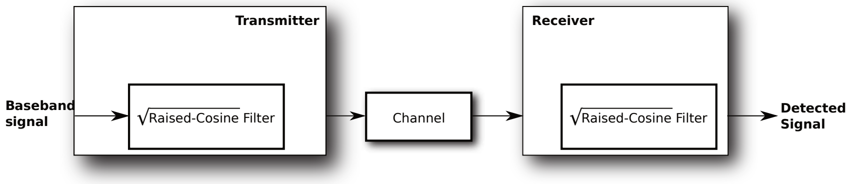


Figure 2.46: Decomposition of a raised-cosine filter into two sections (matched filtering).

Channel Select Receive Filter Design

So previous section suggests that we must implement a SRRC (square-root raised cosine) filter in our receiver to allow the zero-ISI Nyquist filtering. However, on the reception side usually the filter is analog and the bandwidth is always slightly more then the bandwidth of the SRRC. That means that there is more noise added and also adjacent frequency bands can be coupled in the band of interest by intermodulation. Thus, in our Simulink simulations of the BER (bit-error rate) as a function of channel noise and IQ offsets, the BER will be higher than if a SRRC receive filter is applied.

This gives us a more realistic picture of the performing communication system in terms of the BER so that we can determine the maximum allowable IQ offset more accurately for a given BER of 1.10^{-5} ($\pi/4$ DQPSK modulation). Figure 2.47 shows a transceiver system including the rates for a $\pi/4$ DQPSK modulation scheme in the 402 MHz to 405 MHz band.

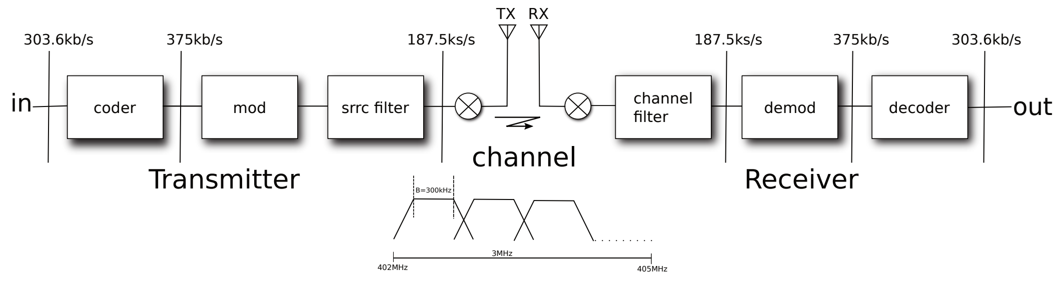


Figure 2.47: Transceiver system representing the IEEE Std 802.15.6 $\pi/4$ DQPSK modulation parameters for the band 402MHz to 405MHz.

The idea is to place a channel select filter instead of a SRRC filter in the receiver as shown in Figure 2.47. Now we have created a mismatch due to inequality between a SRRC in the transmitter and a channel select filter in the receiver. These filters are not the same. An SRRC filter only in the transmitter does not posses the zero-ISI property. Of course we do not expect this, because a root-Nyquist filter does not meet all the three criteria for a Nyquist filter. So it, by itself, is not a Nyquist filter. The impulse response of the SRRC filter, has shifted zero-crossing times compared to the RC (Raised Cosine) filter. According to the IEEE 15.6 Wireless Body Area Network standard [2], an SRRC pulse shape filter is used in the transmitter and according to this standard we can also select the excess bandwidth factor, i.e., *Rolloff factor* α for the receive filter according to the application. This means that we can tweak the α of the Biquad Channel Select Filter on the receiver side to get such a good match with the SRRC transmit filter to allow zero-ISI Nyquist filtering.

The theme of this thesis research is among other (i.e. IQ gain mismatch compensation) IQ offset compensation in the Phase-domain ADC Receiver [10]. So it is a logical consequence to implement the filter chain of this receiver system in our Simulink receiver model, see Figure 2.48. If we take a closer look at the model we see that channel noise is modeled by an AWGN (averaged white Gaussian noise) block in the communication channel and a biquad Filter on the receiving side. The IQ offset is positioned at the input of the PhADC because the biquad filter has a high-pass 3 dB point at 10 Hz, therefore, pure DC signals are not fully passed. The IQ offset would otherwise be filtered out and therefore we can not model the effect of IQ offset properly. The intention is that the biquad filter should resemble the analog filter consisting of the passive front-end (PFE), baseband LNA (BB LNA), analog baseband programmable amplifier (PGA) and Sallen&Key filter [10], but on which I shall discuss in subsequent sections. Figure 2.49 shows a block diagram view of the Phase-domain ADC receiver [10].

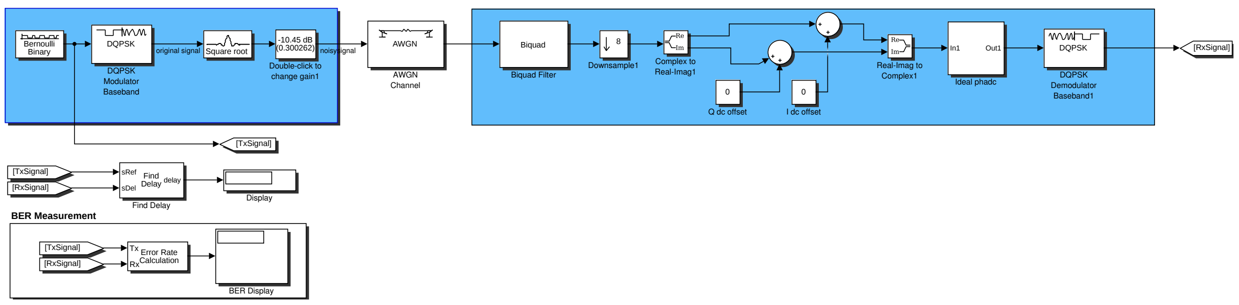


Figure 2.48: Simulink model for simulating the BER as a function of IQ offset and channel noise.

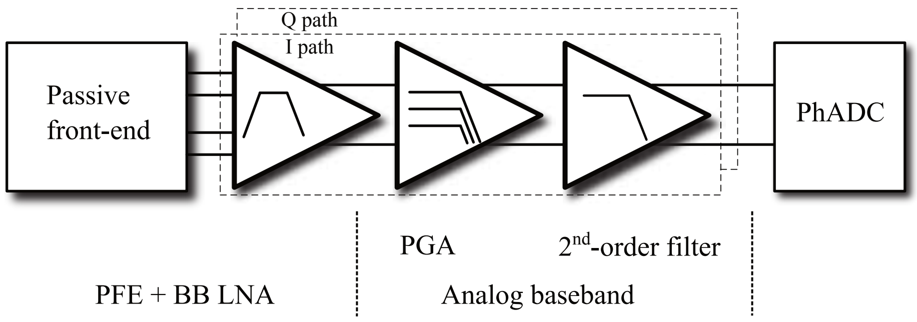


Figure 2.49: A receiver system with a PFE and PhADC.

Between the Passive Front End (PFE) and the PhADC shows a fifth order filter system. In order from left to right: baseband low-noise amplifier (BB LNA), programmable-gain amplifier (PGA) and a 2^{nd} -order filter. Figure 2.50 shows the filter chain in a form of the filter transfer characteristics of the individual filters. And Figure 2.51 shows the poles and zeros in the s-plane of the overall filter system. The stability margins are given in the following Figure 2.52. As you can see, the filter system is stable and you may have noticed, the overall gain of the filter chain is 0 dB while the receiver system [10] has a maximum gain of 77.1 dB (36.6 dB of PFE + BB LNA and 40.5 dB of analog BB). At this gain the minimum signal level is -6.5 dBm. We need to measure the BER as a function of IQ offset and channel noise at this minimum signal level. So, in the Simulink model shown in Figure 2.48 the information signal is set at this level and used the filter chain with an overall gain of 0 dB. Both configurations will have the same effect on the BER. The Simulink simulation model works only with discrete signals with the result that we can not directly implement herein the analog filter. The analog filter must therefore be converted to a discrete filter with the same specifications as the analog filter. In other words, we need to digitize the analog filter. Analog filtering is performed on continuous-time signals and yields continuous-time signals. It is implemented using passive and active electronic components like opamps, resistors, and capacitors. Digital filtering is performed on discrete-time signals and yields discrete-time signals. It is usually implemented on a computer, using operations such as addition, multiplication, and data movement. So, because our Simulink simulation model deals only with discrete signals, it is therefore obvious that we must discretize the analog filter. One way to do this is to describe the analog filter in the s-domain (i.e., Laplace domain), and mapping it with a particular procedure to the z-domain (i.e., discrete domain). The filter in the S-domain describes a time-continuous (i.e. analog) system and the filter in the z-domain describes a discrete (i.e. digital) system.

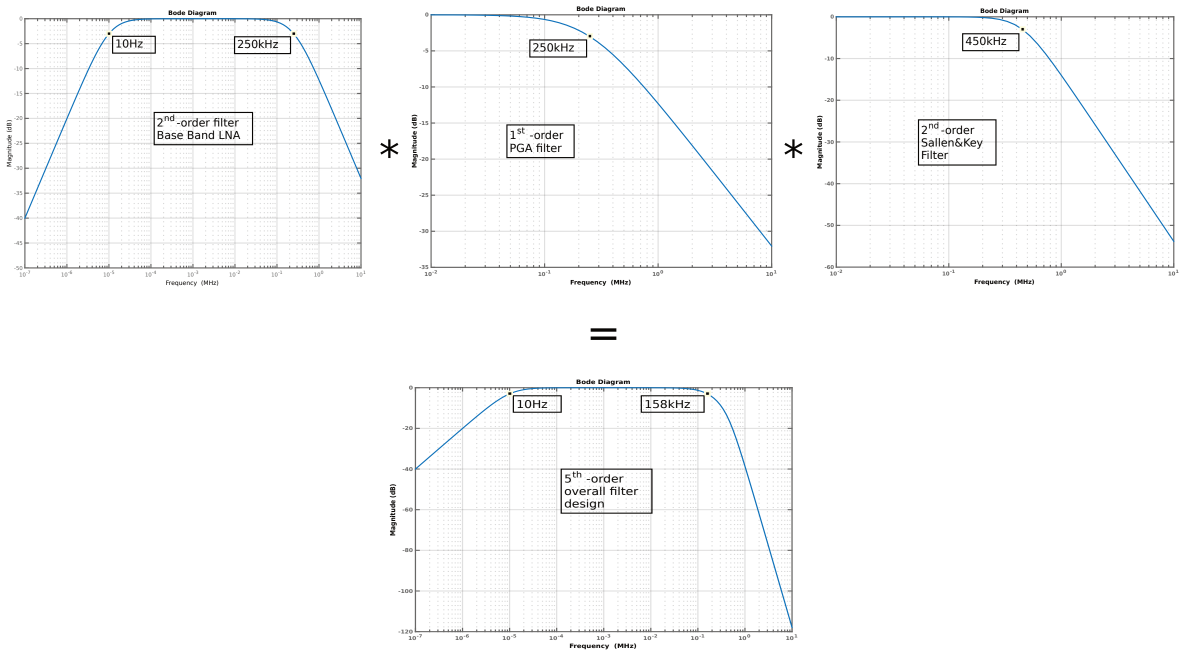


Figure 2.50: Filter chain in a form of the filter transfer characteristics of the individual filters.

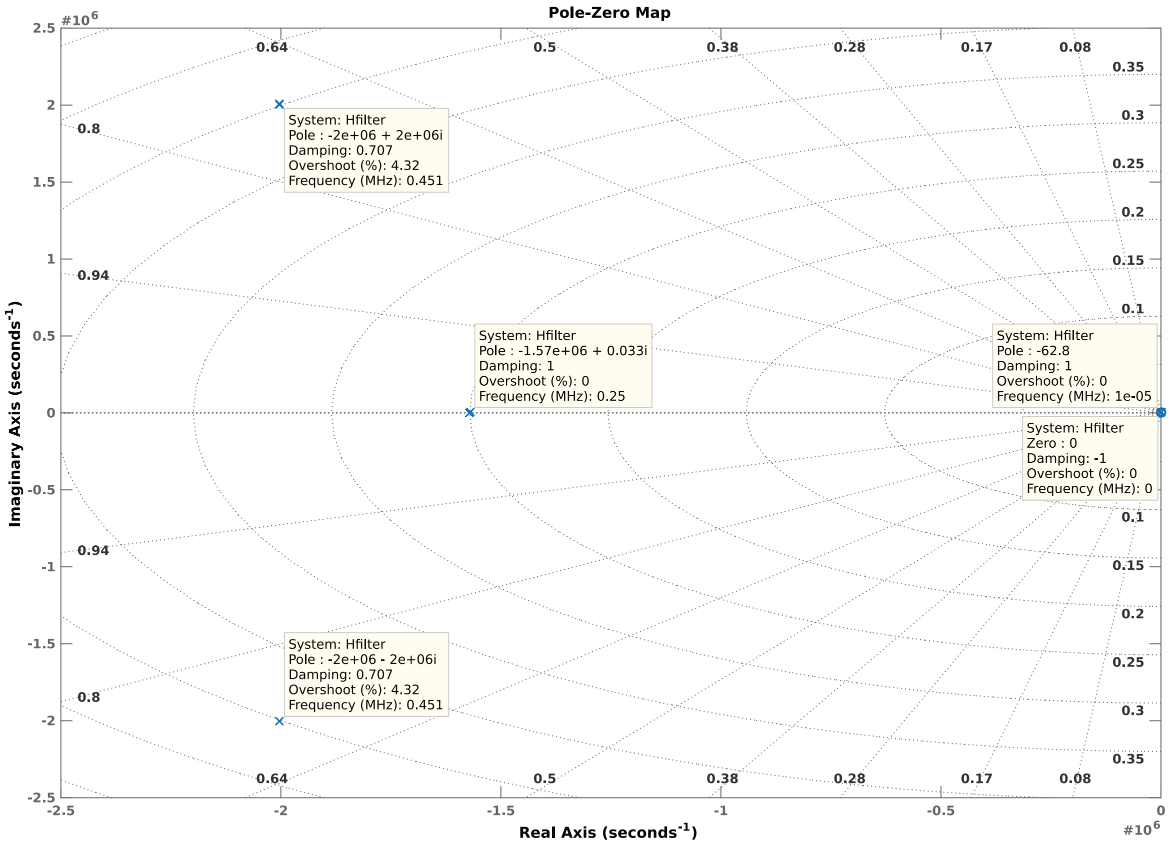


Figure 2.51: Pole-Zero Map of the overall filter transfer function in the S-plane.

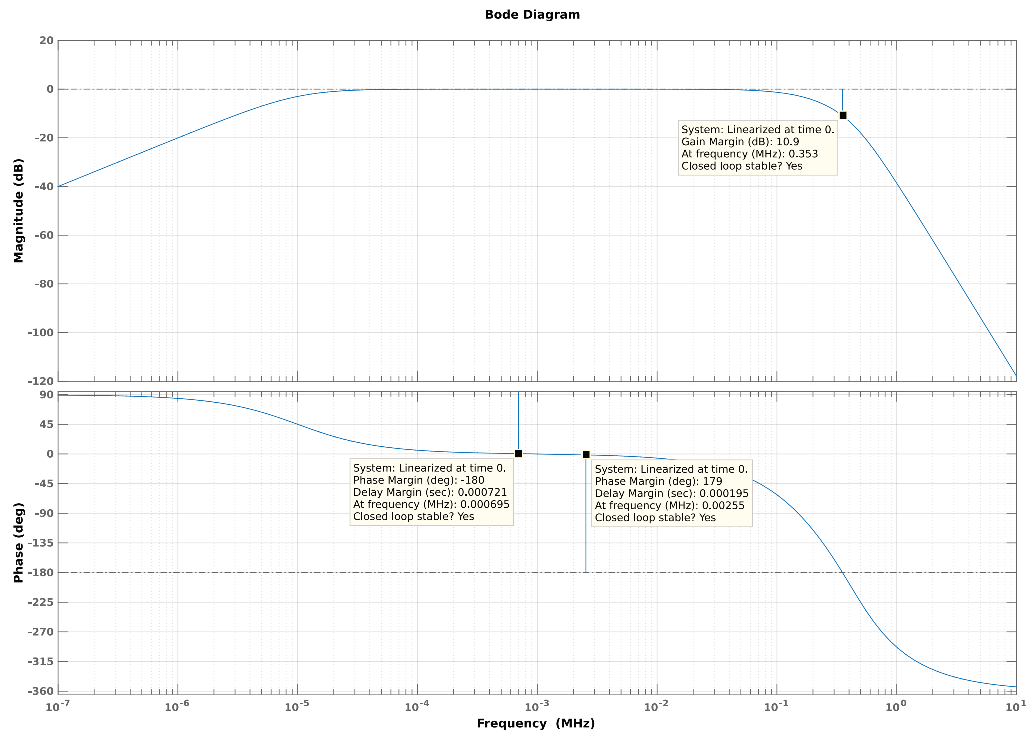


Figure 2.52: The Stability margins of the overall filter system.

Filters can be classified in several different groups, depending on what criteria are used for classification. The two major types of digital filters are finite impulse response digital filters (FIR filters) and infinite impulse response digital filters (IIR). Table 1 shows the different properties between these filters.

FIR-filters	**IIR-filters**
FIR filters have an impulse response of finite duration, i.e. with an amplitude that decays to zero in a finite number of steps (samples) or duration.	IIR filters have an impulse response that has, in theory, an infinite duration or continuous forever because of its recursive nature (feedback).
FIR filters have therefore a non-recursive structure, that is to say filters which do not have feedback.	IIR filters therefore have a recursive structure, that is to say, filters in which at least one feedback occurs.
Because there is no feedback means that the filter is always stable – it can not 'automatically' start to oscillate. They are unconditionally stable.	Because there is feedback means that the filter may start to oscillate spontaneously.
FIR filters have a linear phase characteristic. This is the case because the group delay is constant across the entire frequency range.	IIR Filters have a non-linear phase characteristic. This is the case because the group delay is variable over the entire frequency range.
System function contains only zeros.	System function contains poles and zeros.
FIR filters, however, have a low speed of execution, because complex transfer curves require long filters (i.e. require many tappings for high order) and thus long program loops.	IIR filters on the other hand generate with a minimum filter length (that is to say that the corresponding loop in the program code does not often have to be executed) a long (in theory infinitely long) convolution. So the same FIR filter can be designed with IIR filters using a small filter order.

Table 2.4: Main characteristics compared between the FIR and IIR filter.

The major distinction between IIR and FIR filters lies in the fact that not only zeros but also poles appear in the system function in IIR filters. This gives a certain direct connection in some respects to conventional continuous filters (i.e., analog filters), where the design is also often based on a description of poles and zeros. There are therefore also a number of techniques to design discrete filters, based on existing continuous filters (eg Butterworth, Bessel, Chebyshev and elliptic or Caurer-filters [24]. So, because of the similarities between the transfer functions of IIR digital filters and those of analog filters, the most popular techniques for designing IIR digital filters are, in some manner, digital versions of analog filter designs. FIR filters on the other hand cannot be designed from the theory of analog filters because analog filters based upon lumped circuit elements have infinite impulse response. The design problem is to find a suitable transformation for mapping the s-plane poles and zeros of the analog filter representation onto the z-plane. These techniques require the construction of simple mapping procedures to map analog filter designs into IIR digital filter designs. This means that the design of an IIR digital filter involves the following two steps:

1. Design an analog filter by obtaining an appropriate transfer function $H(s)$ to meet the signal processing requirements.
2. Construct a mapping procedure to transform $H(s)$ into an appropriate transfer function $H(z)$, thus resulting in an IIR digital filter design that will meet the specifications.

This two-step procedure for designing IIR digital filters is illustrated in Figure 2.53.

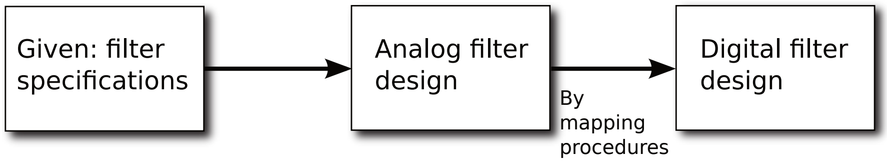


Figure 2.53: A two-step procedure for designing IIR digital filters.

In the following two sections we show a method in Matlab for converting $H(s)$ (the Laplace transform of the impulse response $h(t)$ in the time-domain) to the discrete impulse response $H(z)$ in the z -domain.

Transformation of the filter to the s-domain

If we look at Figure 2.50, we see that the filter consists of a series connection of a band-pass filter, a 1^{st}-order low-pass filter and a 2^{st}-order low-pass filter. Now the form of the mathematical models, that show the relationship between the input and the output of these filters, are well known. Figure 2.54 shows the frequency response of those filters.

$$\frac{U_o(j\omega)}{U_i(j\omega)} = \frac{\tau_1 j\omega}{(\tau_1 j\omega + 1)(\tau_2 j\omega + 1)}$$

(a)

$$\frac{U_o(j\omega)}{U_i(j\omega)} = \frac{1}{(\tau j\omega + 1)}$$

(b)

$$\frac{U_o(j\omega)}{U_i(j\omega)} = \frac{1}{\frac{1}{{\omega_n}^2}(j\omega)^2 + \frac{2\xi}{\omega_n} j\omega + 1}$$

(c)

Figure 2.54: Frequency Response: Band pass filter (a) Low pass filter (b) Second-order low pass filter (c).

In the literature we often find the following expression: $H(s) = \frac{U_o(s)}{U_i(s)}$
Where $\underline{s}$ is the complex frequency. Also U_o and U_i are of course complex variables. The following applies:

$s = \sigma + j\omega$

However, if we are interested in the actual realization of a practical (i.e. tangible) filter, then we can write:

$s = j\omega$

For $\sigma = 0$ and thus $s = j\omega$ arises from the transmission ratio $H(s)$, the so-called transmission ratio or steady state frequency response $H(j\omega)$. The modulus of $H(j\omega)$ represents the gain as a function of ω, the argument of $H(j\omega)$ represents the phase shift between input and output signal, also as a function of ω.

The transmission ratio $H(s)$ is the Laplace transfer function of the system. It is the ratio of the Laplace transform of the output signal and the Laplace transform of the input signal. The transfer function $H(s)$ is the Laplace transform of the unit impulse response $h(t)$ in the time domain. If we transform the frequency responses shown in Figure 2.54 to the Laplace domain we will get the responses as shown in Figure 2.55.

$$\frac{U_o(s)}{U_i(s)} = \frac{\tau_1 s}{(\tau_1 s + 1)(\tau_2 s + 1)}$$

(a)

$$\frac{U_o(s)}{U_i(s)} = \frac{1}{(\tau s + 1)}$$

(b)

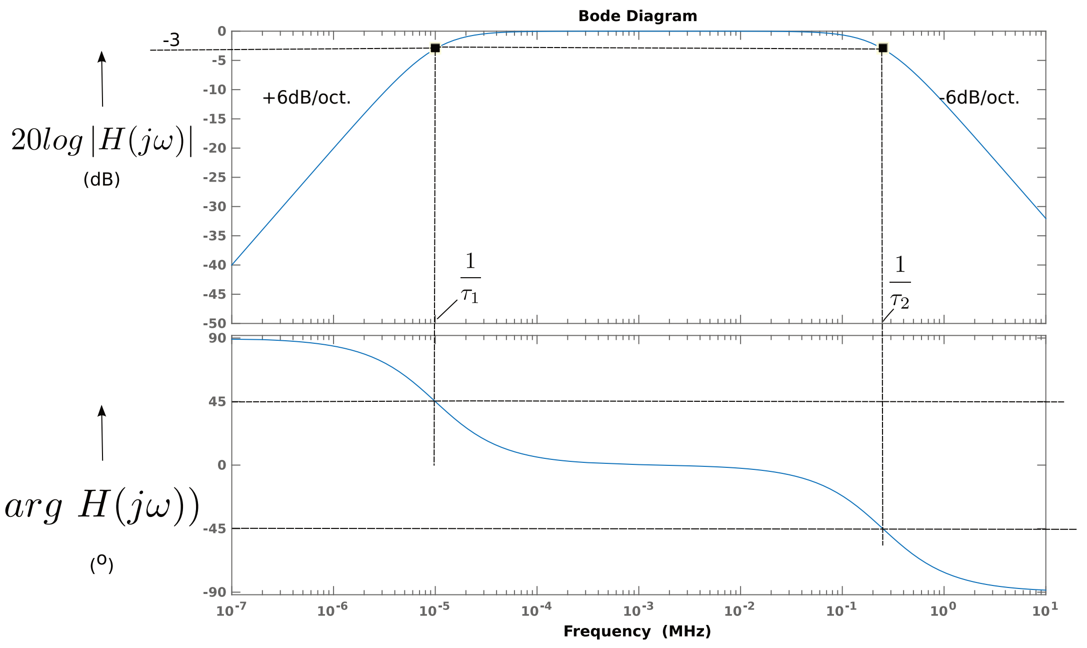
$$\frac{U_o(s)}{U_i(s)} = \frac{1}{\frac{1}{{\omega_n}^2}(s)^2 + \frac{2\xi}{\omega_n} s + 1}$$

(c)

Figure 2.55: Laplace transform: Band pass filter (a) Low pass filter (b) Second-order low pass filter (c).

The most significant advantage of the Laplace transform is that differentiation and integration become multiplication and division, respectively, by s. The transform turns integral equations and differential equations to polynomial equations, which are much easier to solve. Once solved, use of the inverse Laplace transform reverts to the time domain.

Figures 2.56, 2.57 and 2.58 shows the Bode- and pole-zero plots of these filters.



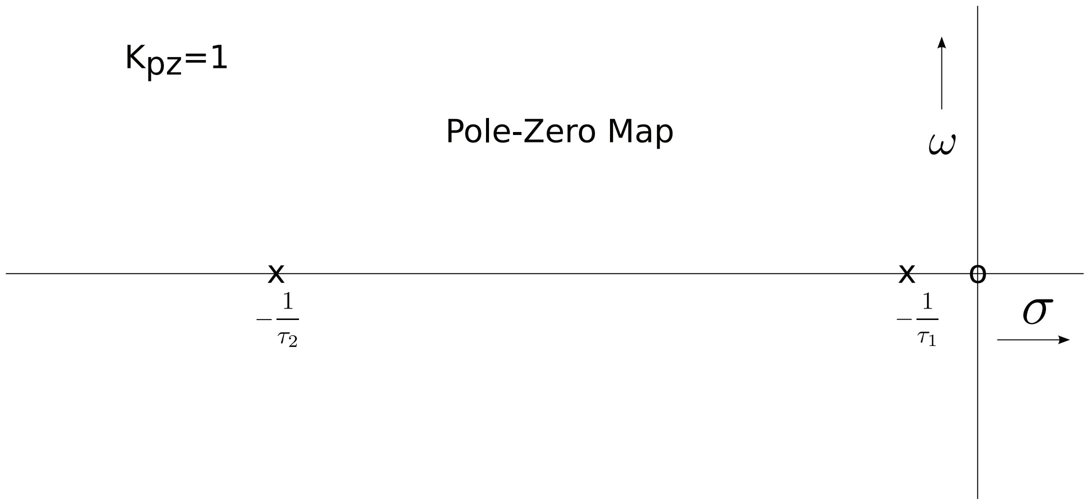
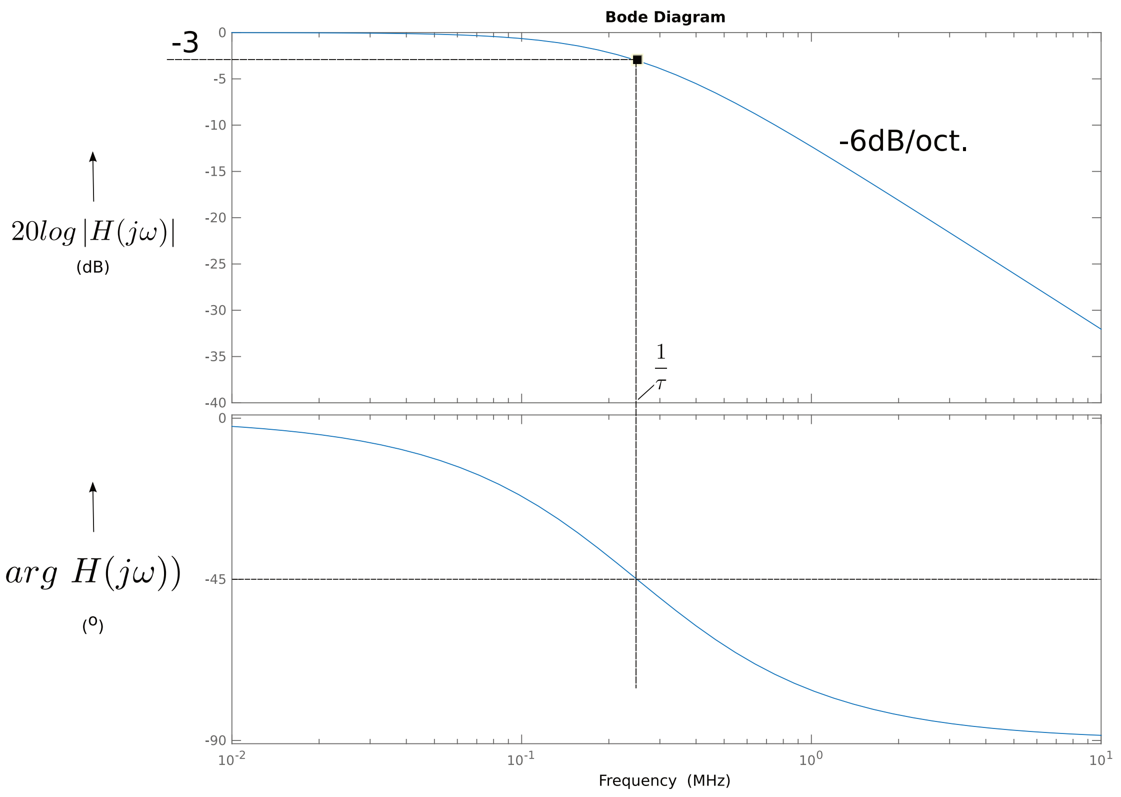


Figure 2.56: Bode and pole-zero plots of the bandpass filter.



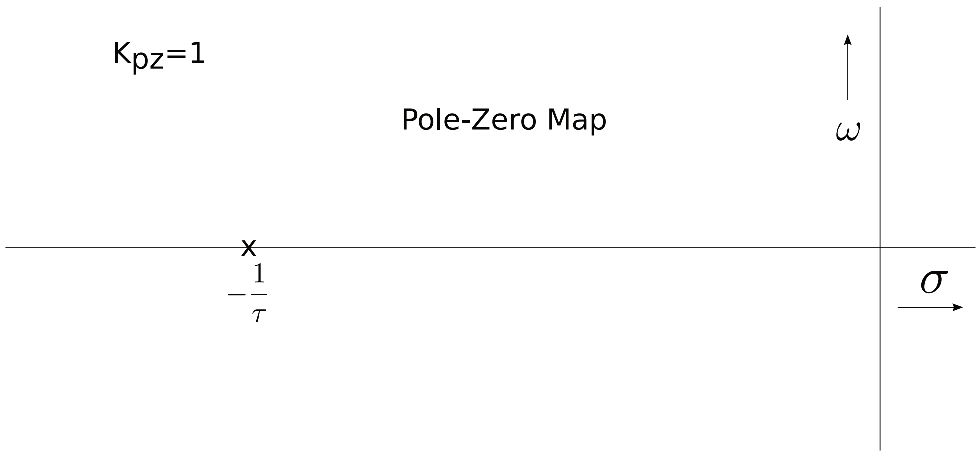
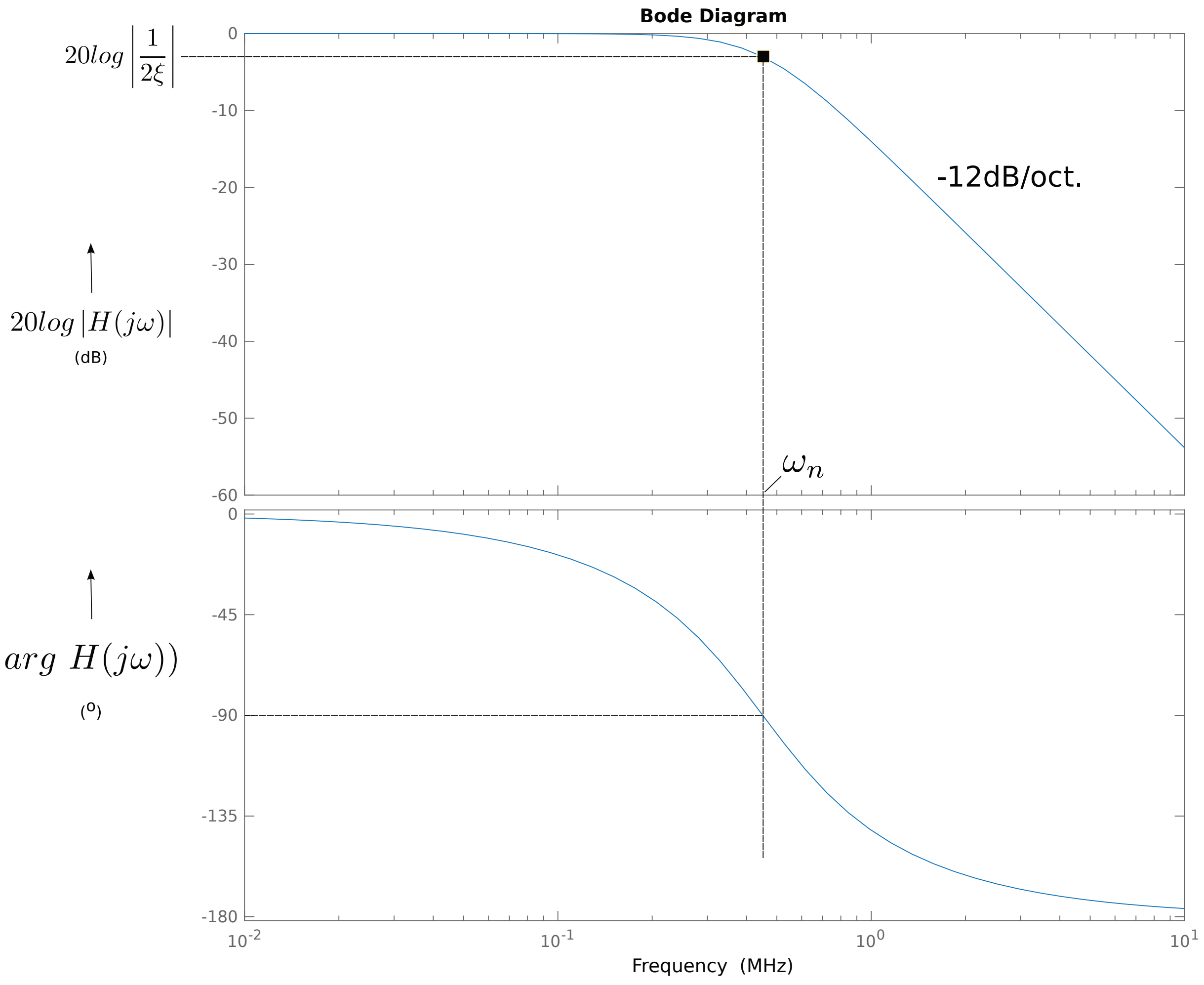
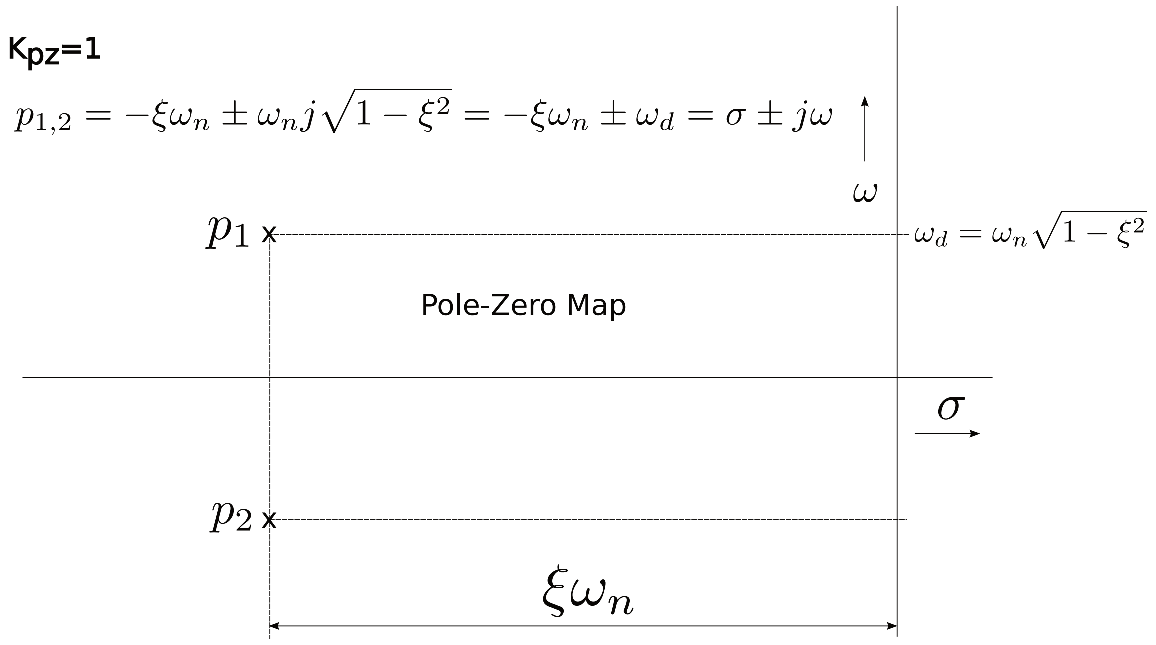


Figure 2.57: Bode and pole-zero plots of the 1st-order low pass filter.





Kpz=1
$p_{1,2} = -\xi\omega_n \pm \omega_n j\sqrt{1-\xi^2} = -\xi\omega_n \pm \omega_d = \sigma \pm j\omega$
ω
p_1
$\omega_d = \omega_n\sqrt{1-\xi^2}$
Pole-Zero Map
σ
p_2
$\xi\omega_n$

Figure 2.58: Bode and pole-zero plots of the 2^{nd}-order low pass filter.

Where the parameter ξ is the *damping ratio* and ω_n is the *undamped natural frequency* and ω_d is the *damped natural frequency.*

Now that we've looked at the frequency characteristics of the individual filters separately we need to determine the transfer function of the entire filter. The transfer functions of filters in cascade multiply together. We must be aware of the loading effects between these filter blocks. Joining blocks together can cause problems because of interaction effects. In the voltage domain, this can be solved by designing systems with low output impedance and high input impedances. It is almost impossible to achieve this consistently with passive filters. Fortunately, this is covered because the filter designs of the PhADC receiver [10] use operational amplifiers as integral parts of the filter circuits.

Now that we have the transfer functions of the individual filters we can set up the transfer function of the entire filter.

$$H(s) = \frac{\tau_1 s}{(\tau_1 s+1)(\tau_2 s+1)} \frac{1}{(\tau s+1)} \frac{1}{(\frac{1}{\omega_{n2}}(s)^2 + 2\frac{\xi}{\omega_n}s+1)} \tag{2.32}$$

The next step is to find the coefficients of the transfer function (2.32), i.e. τ_1, τ_2, τ, ω_n, ξ . We have all the information about the cut-off frequencies of the filters and so we can derive all coefficients. But let's first look at the type of filters that are applied and infer relationships between the input and output of these filters.

Band-pass low noise amplifier

The PhADC receiver [10] uses a fully-differential low noise amplifier (LNA) with band-pass filter that is shown in Figure 2.59(a).

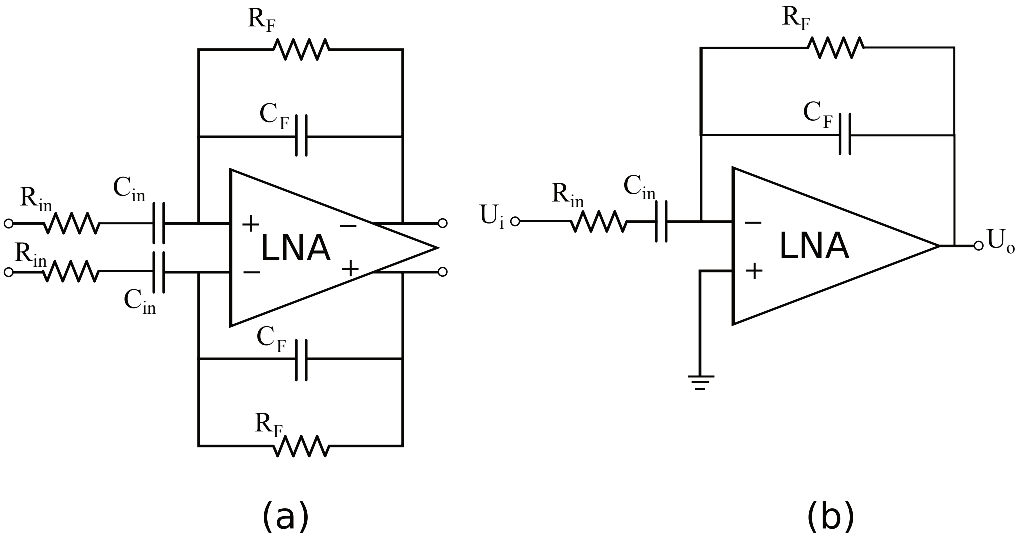


Figure 2.59: Fully-differential band-pass low noise amplifier (a) Single-ended band-pass low noise amplifier.

But let us determine the transfer function of the filter with the aid of a single-ended band-pass LNA as shown in Figure 2.59(b).

$$\frac{U_o(j\omega)}{U_i(j\omega)} = \frac{\frac{R_F}{R_F C_F j\omega+1}}{R_{in} + \frac{1}{C_{in}} j\omega} = -\frac{R_F C_{in} j\omega}{(\tau_1 j\omega+1)(\tau_2 j\omega+1)} = -\frac{R_F}{R_{in}} \frac{\tau_1 j\omega}{(\tau_1 j\omega+1)(\tau_2 j\omega+1)} \tag{2.33}$$

Where $\tau_1 = R_{in}C_{in}$, $\tau_2 = R_F C_F$. For a unity pass-band gain we have $R_F = R_{in}$.

For the fully-differential band-pass LNA we can write then:

$$\frac{U_o(j\omega)}{U_i(j\omega)} = \frac{\tau_1 j\omega}{(\tau_1 j\omega + 1)(\tau_2 j\omega + 1)} \tag{2.34}$$

The high-pass and low-pass cut-off frequencies are set at 10Hz and 250kHz respectively. Then it follows:

$$\omega_l(high-pass) = 2\pi f_l = \frac{1}{\tau_1}\ rad/s \quad \rightarrow \quad \tau_1 = \frac{1}{20\pi} \approx 15.9155.10^{-3}\ sec$$

$$\omega_u(low-pass) = 2\pi f_u = \frac{1}{\tau_2}\ rad/s \quad \rightarrow \quad \tau_2 = \frac{1}{500.10^3\pi} \approx 6.3662.10^{-7}\ sec$$

Substituting these values in (2.34) and writing the Laplace transformed transfer function we find:

$$\frac{U_o(j\omega)}{U_i(j\omega)} = \frac{15.9155.10^{-3} j\omega}{(15.9155.10^{-3} j\omega + 1)(6.3662.10^{-7} j\omega + 1)} \rightarrow \mathscr{L} \rightarrow \frac{U_o}{U_i} = \frac{15.9155.10^{-3} s}{(15.9155.10^{-3} s + 1)(6.3662.10^{-7} s + 1)} \tag{2.35}$$

Programmable gain amplifier (PGA)

The circuit of the programmable gain amplifier is shown in Figure 2.60.

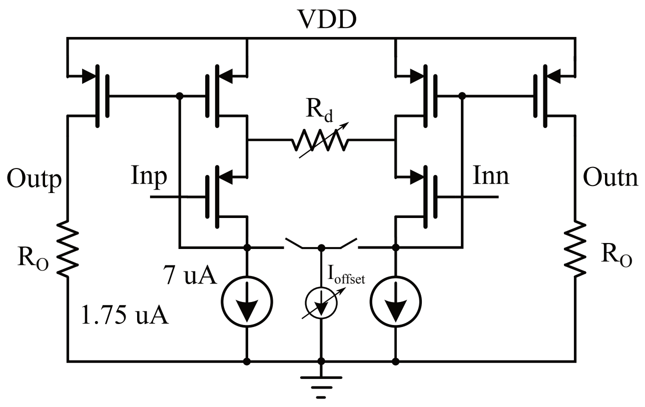


Figure 2.60: PGA circuit.

The amplifier employs a gm -boosted differential pair with source degeneration [6, 12]. In this configuration, the pass-band gain is determined by the degeneration resistor R_d, output resistor R_o, and the current ratio of the two stages (4:1 in our case), which is $\frac{R_o}{2R_d}$ for this implementation. The low pass filter dominant pole at the PGA output is determined by the load resistance R_o and output capacitance C_o and is therefore:

$$f_c = \frac{1}{2\pi R_o C_o} = \frac{1}{2\pi\tau} \tag{2.36}$$

with $\tau = R_o C_o$. The pole is kept constant due to the fact that a degeneration resistor R_d is made programmable while maintaining a constant load resistor. The low-pass cut-off frequency is kept at 250kHz. That means $\tau = \frac{1}{2\pi.500.10^3} \approx 6.3662.10^{-7} sec$. Using the formula shown in Figure 2.54(b) we can write for the low-pass filter transfer function:

$$\frac{U_o(j\omega)}{U_i(j\omega)} = \frac{1}{(\tau j\omega + 1)} = \frac{1}{6.3662.10^{-7} j\omega + 1} \quad \rightarrow \mathscr{L} \rightarrow \quad \frac{U_o(s)}{U_i(s)} = \frac{1}{\tau s + 1} = \frac{1}{6.3662.10^{-7} s + 1} \tag{2.37}$$

Sallen & **Key** 2^{nd}**-order low pass filter**

A 2^{nd}-order filter is necessary to attenuate the close adjacent and alternate channel interference [10]. The 2^{nd}-order filter employs a Sallen & Key topology, as shown in Figure 2.61(b).

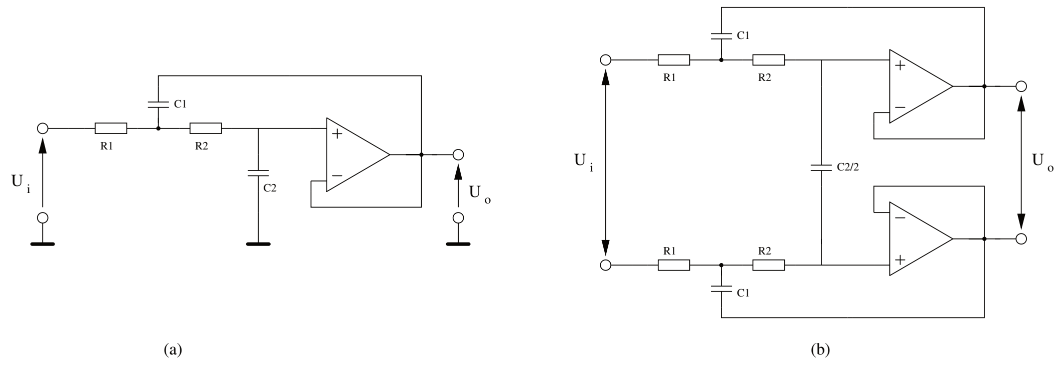


Figure 2.61: Sallen-Key low pass filter: single ended (a) and fully differential (b)

The cut-off frequency is given by $\frac{1}{2\pi}\sqrt{R_1 R_2 C_1 C_2}$ for the topology. In this implementation, C_1 and C_2 are chosen 832fF and 416fF, respectively, while R_1 and R_2 have a nominal value of $600k\Omega$, giving a nominal cut-off frequency of 450 kHz. Using Figure 2.61(a), for the frequency characteristic of the transfer function, the transfer function of the signal at the output U_o of the filter in relation to the input signal U_i, we find:

$$\frac{U_o(j\omega)}{U_i(j\omega)} = \frac{1}{1 + j\omega C_2(R_1 + R_2) + (j\omega)^2 R_1 R_2 C_1 C_2} \tag{2.38}$$

Transforming it to the Laplace domain yields:

$$\frac{U_o(s)}{U_i(s)} = \frac{1}{1 + sC_2(R_1 + R_2) + s^2 R_1 R_2 C_1 C_2} \tag{2.39}$$

If we fill in the values for C_1, C_2, R_1 and R_2, then we get:

$$\frac{U_o(s)}{U_i(s)} = \frac{1}{s^2 1.246.10^{-13} + s4.992.10^{-7} + 1} \tag{2.40}$$

Construction of the Laplace transfer function of the overall filter

Using (2.35), (2.37) and (2.40) in Matlab we can construct the Laplace transfer function of the overall filter system:

$$H(s) = \frac{s15.9155.10^{-3}}{(s15.9155.10^{-3}+1)(s6.3662.10^{-7}+1)} \cdot \frac{1}{(s6.3662.10^{-7}+1)} \cdot \frac{1}{(s^2 1.246.10^{-13}+s4.992.10^{-7}+1)}$$

after simplification it yields:

$$H(s) = \frac{15.9155.10^{-3} s}{8.037.10^{-28} s^5 + 5.745.10^{-21} s^4 + 1.855.10^{-14} s^3 + 2.821.10^{-8} s^2 + 15.9155.10^{-3} s + 1} \tag{2.41}$$

This is the Laplace transfer function of the overall filter system. It is the ratio of the Laplace transform of the output signal and the Laplace transform of the input. The transfer function $H(s)$ is the Laplace transform of the unit impulse response $h(t)$ in the time-domain. (2.41) shows a 5^{th}-order filter system with 5 poles, two poles in Butterworth, one low and two high poles on the real axis of the s-plane and one zero in the origin.

Transformation of the filter to the z-domain

Now that we have characterized our analog filter we have to convert it to a discrete filter design to implement it as a IIR biquad digital filter in our Simulink model shown in Figure 2.48. The biquad filter consists of a number of cascaded second-order sections (sos). In signal processing, a digital biquad filter is a second-order recursive linear filter, containing two poles and two zeros. *Biquad* is an abbreviation of *biquadratic*, which refers to the fact that in the $\mathcal{Z}$-domain, its transfer function is the ratio of two quadratic functions:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.42}$$

The coefficients are often normalized such that $a_0 = 1$.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2.43}$$

High-order IIR filters can be highly sensitive to quantization of their coefficients, and can easily become unstable. This is much less of a problem with first and second-order filters; therefore, higher-order filters are typically implemented as cascaded biquad sections (and a first-order filter, if necessary). The two poles of the biquad filter must be inside the unit circle for it to be stable. In general, this is true for all filters, i.e. all poles must be inside the unit circle for the filter to be stable. Matlab provides different discretization methods to convert a continuous-time dynamic system to discrete time. We will not use the impulse invariant discretization method, but the ZOH (zero-order hold) to the input of the filter since the filtered signal is passed to the input of the Phase-domain ADC which does a track and hold that is a practical version of a sampling circuit that is based on the ZOH. Actually, the combination of the front-end track and hold and the Phase AD converter operates as an ideal sampling circuit [16]. The Phase AD converter senses the output of the front-end sampler only during the hold mode, and hence the digitized value corresponds to sampled points on the input waveform. In high-speed systems, the distinction between the outputs of the zero-order hold and track-and-hold schemes begins to diminish because the sampling window width becomes comparable to the sampling period. In Matlab we can represent numeric system components using any model type. For instance the Laplace model shown in formula (2.41) can be represented in analytic model types such as tf (transfer function model) using numeric row vectors for the numerator and denominator coefficients ordered in descending powers of *s* or a *zpk* (zero-pole-gain model), *z*, *p* being vectors of real- or complex-valued zeros and poles, and *k* being the real- or complex-valued scalar gain. However, numeric LTI (Linear Time Invariant) model types are not equally well-suited for numerical computations. Generally, it is recommended to work with state-space (ss) [18] or frequency response data (*frd*) models [21], for the following reasons:

- The accuracy of computations using high-order transfer functions (*tf* or *zpk* models) is sometimes poor, particularly for MIMO (multi-input multi-output) or higher-order systems. Conversion to a transfer function representation can incur loss of accuracy.
- When you convert tf or zpk models to create state space using ss, the software automatically performs balancing and scaling operations. Balancing and scaling improves the numeric accuracy of computations involving the model. For more information about balancing and scaling state-space models, see Matlab help files.
  In addition, converting back and forth between model types can introduce additional state or orders, or introduce numeric inaccuracies. For example, conversions to state space are not uniquely defined, and are not guaranteed to produce a minimal realization for MIMO models. For a given state-space model sys,

ss(tf(sys))

can return a model with different state-space matrices, or even a different number of states in the MIMO case.

In the following text we show a method in Matlab for converting $H(s)$ (the Laplace transform of the impulse response $h(t)$ in the time-domain) from formula (2.41) to the discrete impulse response $H(z)$ in the z-domain using the zero-order hold discretization. The zero-order hold discretized IIR filter is realized by the following steps:

- First, we define two numeric row vectors for the numerator and denominator coefficients ordered in descending powers of *s*.
- Then we use the Matlab function *tf2ss* to convert the transfer function to a state-space presentation. It creates matrices *A, B, C, D* of the state-space data.
- Next, we use the Matlab function *ss* to construct the state-space model by accessing the state-space data in the *A, B, C, D* matrices.
- Next, we use the Matlab function *c2d* to convert the continuous-time dynamic system to the discrete time using the state-space model with sample time T_s that approximates the continuous-time model $H(s)$.
- Next, we use the Matlab function *ssdata* to access the state-space data of the discrete time model to create *E, F, G, H* matrices.
- Next, using the Matlab *ss2sos* function to convert state-space of this discrete model to second-order sections model by accessing the state-space data in the *E, F, G, H* matrices of the discrete model.

But before we start taking the above steps, we go into selecting the correct sampling time. This is treated in the next section.

Specify the sample time T_s for the discrete filter

First we need to determine the sampling frequency for the discrete filter. According to the sampling theorem the sampling frequency should be at least twice the highest frequency in the signal to be sampled. But the discrete filter must be able to generate enough samples for the bandwidth so that it can sample at the correct moments. In practice, in order to preserve the fidelity, the sampling is taken to be between 5 – 10 times the highest frequency of the signal to be sampled. The filter needs to take enough samples so that the downsampler can detect the smooth transitions of the pulse shaped signals. Ideally, you want the discrete filter to perfectly mimic the low and high frequencies behavior of the analog filter. This must be a discrete filter with a very high sample rate. But is difficult to implement, i.e, slow simulation speeds, unnecessary much signal processing. But an approach is also good enough. How well that approach should be is not really clear.
There is a MATLAB function called *c2d* that converts a given continuous system (either in transfer function or state-space form) to a discrete system using the zero-order hold discretization. The bit time (i.e. bit rate $1/T_b$=375kHz) in our Simulink simulation model is $T_b = 2.67\mu s$. A convenient number T for the sampling time is chosen to be $T_s = \frac{T_b}{8} \approx 0.3333\mu s$. With this selection, the sample rate of 8 times the bit rate is high enough to build a discrete filter, to a certain extent, with the same specifications as the analog filter. In any case, we more than amply meet the Nyquist sampling theorem (i.e. $T_s \leqslant T_b/2$). The Simulink Simulation Transceiver system is sampled at $2.67\mu s$ and what happens between these samples does not add any additional useful information for the downsampler. Figure 2.62 illustrates this. So the information only changes every $2.67\mu s$ and so we need to make sure that the downsampler samples at the right time.

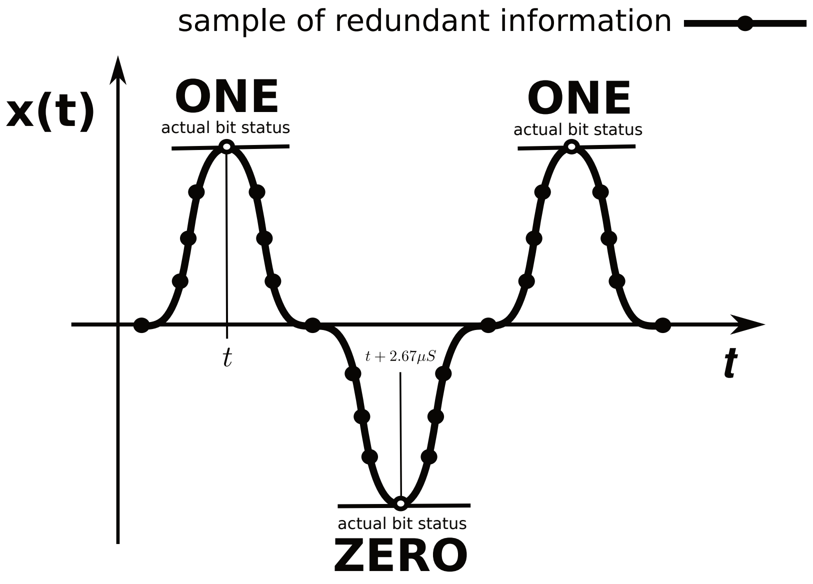


Figure 2.62: Smoothed data transitions with extra samples taken between ones and zeros.

But why do we have chosen a discrete filter sample time of $T_b/8$? In the transmitter the DQPSK modulated signal is up-sampled by a factor of 8 by the SRRC, at the receiver side the same signal, after filtering, must be down-sampled with the same factor. This has an advantage for the system simulation, the received signal, after the down-sampler, will in turn have a period equal to the bit time T_b=2.67μs of the original signal. Figure 2.63 shows this schematically.

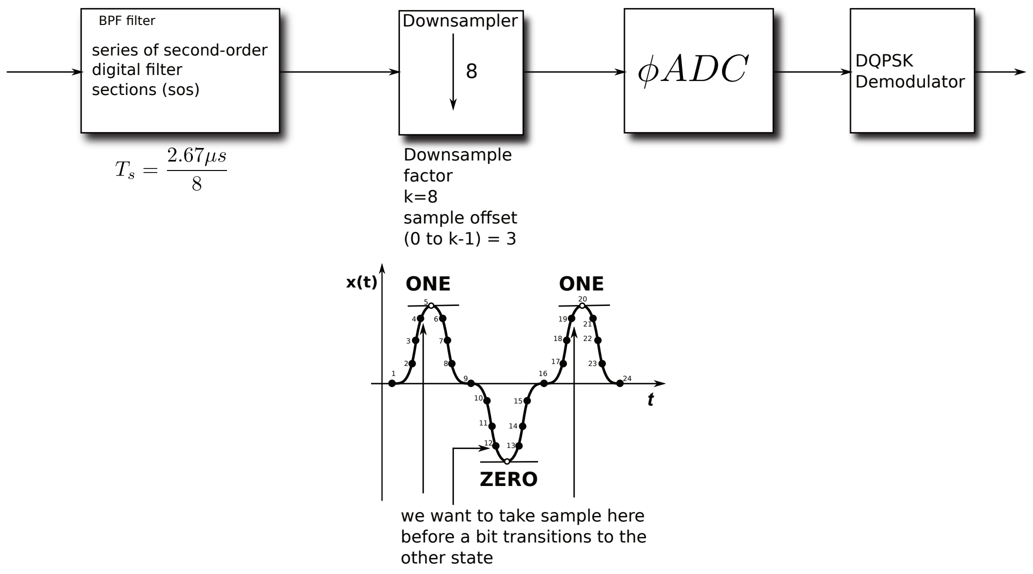

BPF filter
series of second-order digital filter sections (sos)
$T_s = \frac{2.67\mu s}{8}$
Downsampler
8
Downsample factor k=8 sample offset (0 to k-1) = 3
ϕADC
DQPSK Demodulator
x(t)
ONE
ONE
t
ZERO
we want to take sample here before a bit transitions to the other state

Figure 2.63: Receiver:BPF filter and sampler creating a ϕADC input signal at a sample rate of 2.67μs.

Moreover, it is important that the bit time of the signal in the Simulink simulation system is divided by an integer number, otherwise the simulation does not work.

If we look at Figure 2.62 we wonder why we must first up-sample and then filter. Very simple: to be able to see the smooth transitions of the pulse shaped signals more samples have to be taken between the ones and zeros. Hence, we must first up-sample. This effect of multiple samples between transitions can also be seen in the constellation diagram as gradual transitions from one constellation to another, see Figure 2.64. So if we have very abrupt transitions, it means that we need high bandwidth to do these quick transitions.

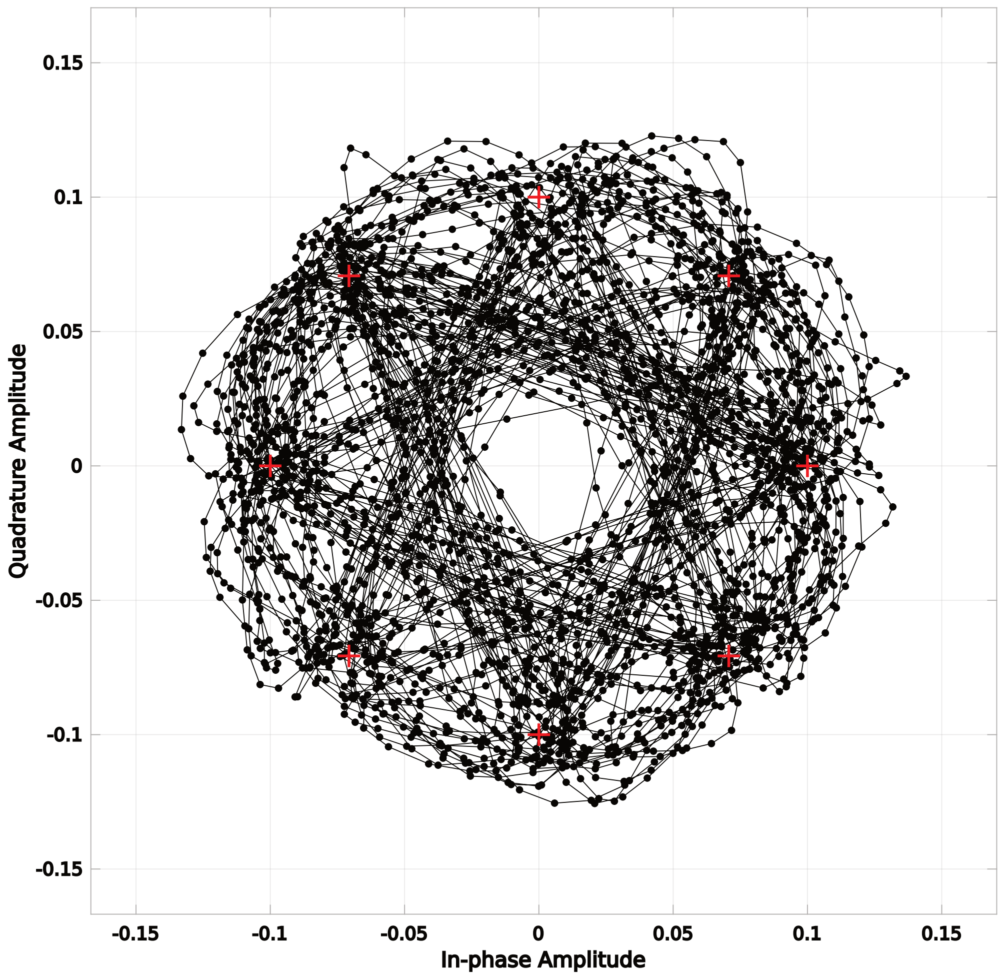


Figure 2.64: Vector transition diagram diagram showing smooth transistions.

The creation of the IIR digital filter in a form of second-order sections

Below is a complete worked out example for creating a digital filter.
The first few steps is the realization of the Laplace transfer function of the analog filter.

1. Define the system by its numerator and denominator

   $num = [0\ 0\ 0\ 0\ 15.9155.10^{-3}];$
   $den = [8.037.10^{-28}\ 5.745.10^{-21}\ 2.821.10^{-8}\ 15.9155.10^{-3}]$

The second few steps is the realization of the discrete transfer function from the Laplace transfer function $H(s)$.

2. We use the Matlab function *tf2ss* to convert the transfer function to state-space presentation. It creates matrices *A*, *B*, *C*, *D* of the state-space data.

   $[A, B, C, D] = tf2ss(num, den);$

3. The next step is to use the Matlab function *ss* to construct the state-space model by accessing the state-space data in the *A*, *B*, *C*, *D* matrices.

$H_{ss} = ss(A, B, C, D);$

4. Next, we use the Matlab function *c2d* to convert continuous-time dynamic system to discrete time using the state-space model with sample time T_s that approximates the continuous-time model $H(s)$. The method used is zero-order hold descritization.

   $H_z = c2d(H_{ss}, Ts,' zoh')$

The next few steps is to convert state-space of this discrete model to second-order sections model.

5. We use the Matlab function *ssdata* to access the state-space data of the discrete time model to create *E, F, G, H* matrices.

   $[E, F, G, H] = ssdata(H_z)$

6. Finally, we use the *Matlab ss2sos* function to convert the state-space of this discrete model to second-order sections by accessing the state-space data in the *E, F, G, H* matrices of the discrete model.

   $[sos, g] = ss2sos(E, F, G, H);$

   where *SOS* is an L by 6 matrix with the following structure:

   SOS=[b01 b11 b21 1 a11 a21
   b02 b12 b22 1 a12 a22
   ...
   b0L b1L b2L 1 a1L a2L]

   Each row of the SOS matrix describes a 2nd order transfer function:

   $Hk(z) = \frac{b0k+b1kz^{-1}+b2kz^{-2}}{1+a1kz^{-1}+a2kz^{-2}}$

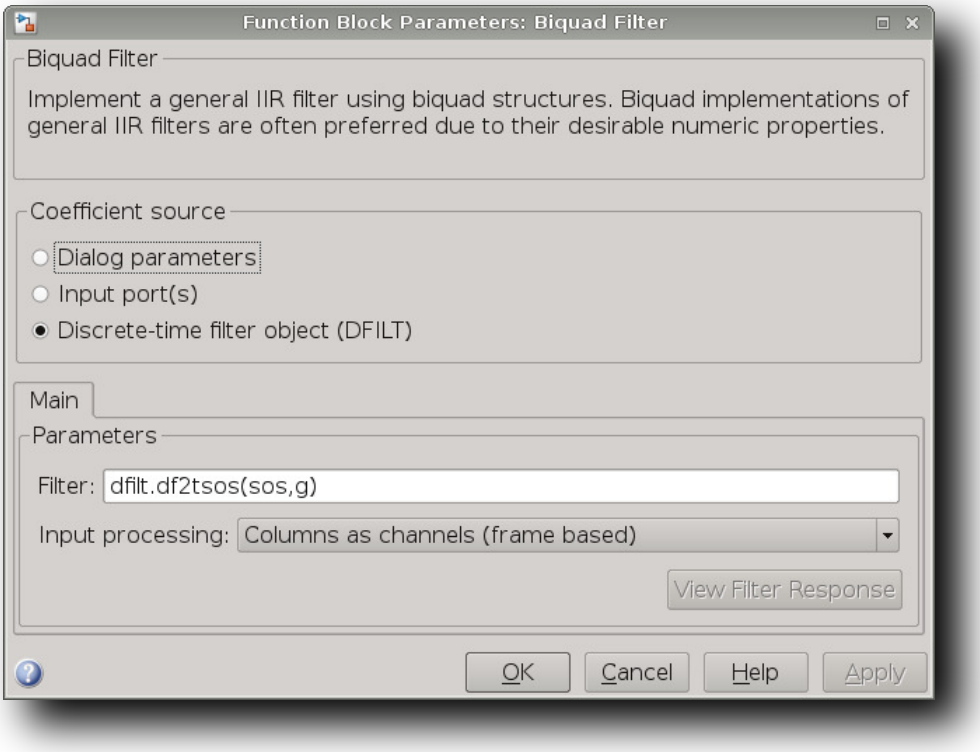
   where k is the row index.

   G is a scalar which accounts for the overall gain of the system. If G is not specified, the gain is embedded in the first section. The second order structure thus describes the system H(z) as:
   H(z) = G*H1(z)*H2(z)*...*HL(z)

In this form, the filter is now ready for use. For this purpose we use the Simulink Biquad filter block. The location of the filter is at the receiver side before the down sampler and phase-domain ADC. See Figure 2.48. If we double-click on this block a dialogue box will appear as shown in Figure 2.65.

The Biquad Filter block can operate in three different modes. We can select the mode in the Coefficient source group box. We are not going to cover all modes, for this we refer to the Matlab help files. As you can see from the dialogue box we use the discrete-time filter object mode (DFILT). Here we specify the filter using a *dfilt* object of the structure *dfilt.df2sos*. *dfilt.df2sos(sos, g)* returns a discrete-time, second-order sections, direct form II filter object, with gains for each section if *g* was a gain vector. If *g* is a scalar then it returns discrete time second order sections with a gain g for the first section and a default gain of one for all the rest of the sections.

With Matlab LTI Viewer command *ltiview* (H_s, H_z) we can make some analysis plots. See figures 2.66 to Figure 2.69. These are the plots of the impulse response (Figure 2.66), Bode diagram (Figure 2.67), Zero-Poles Map H_s (Figure 2.68) and finally Zero-Pole Map H_z (Figure 2.69).



Function Block Parameters: Biquad Filter

Biquad Filter

Implement a general IIR filter using biquad structures. Biquad implementations of general IIR filters are often preferred due to their desirable numeric properties.

Coefficient source

- ○ Dialog parameters
- ○ Input port(s)
- ● Discrete-time filter object (DFILT)

Main

Parameters

Filter: dfilt.df2tsos(sos,g)

Input processing: Columns as channels (frame based)

Figure 2.65: Function block parameters: biquad filter.

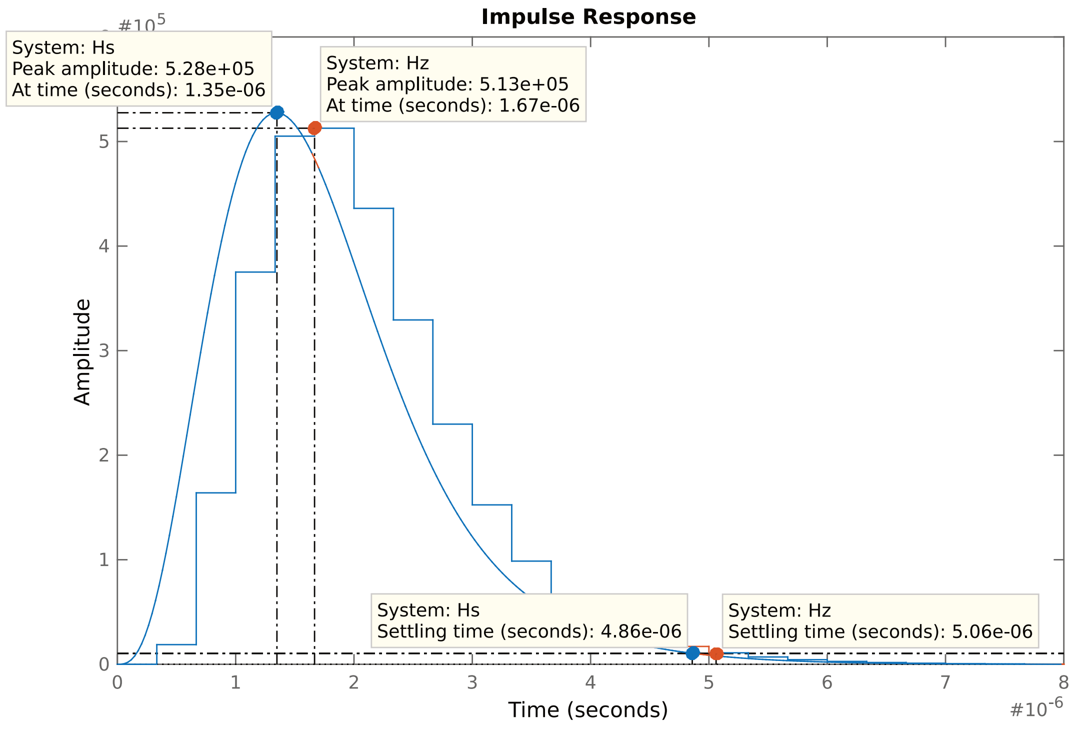


Figure 2.66: Analysis plot: impulse response biquad filter.

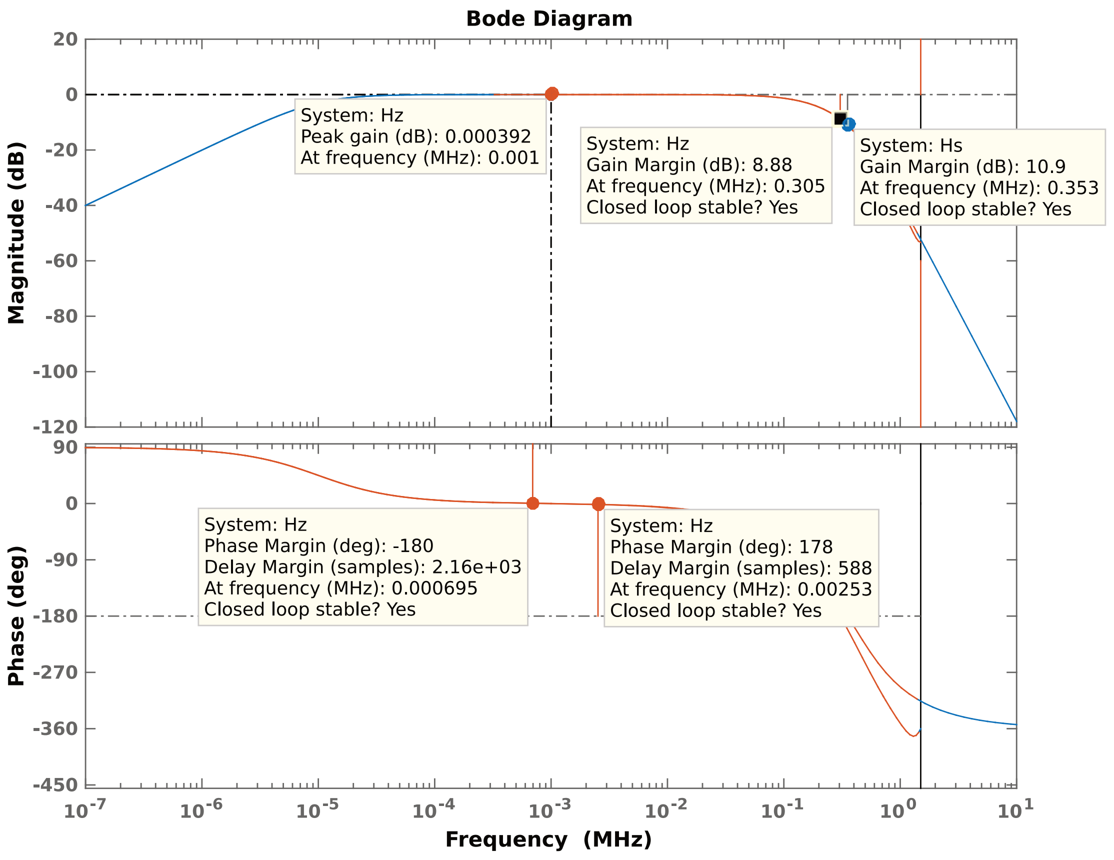


Figure 2.67: Analysis plot: Bode diagram of time-continuous system H_s, and discrete system H_z.

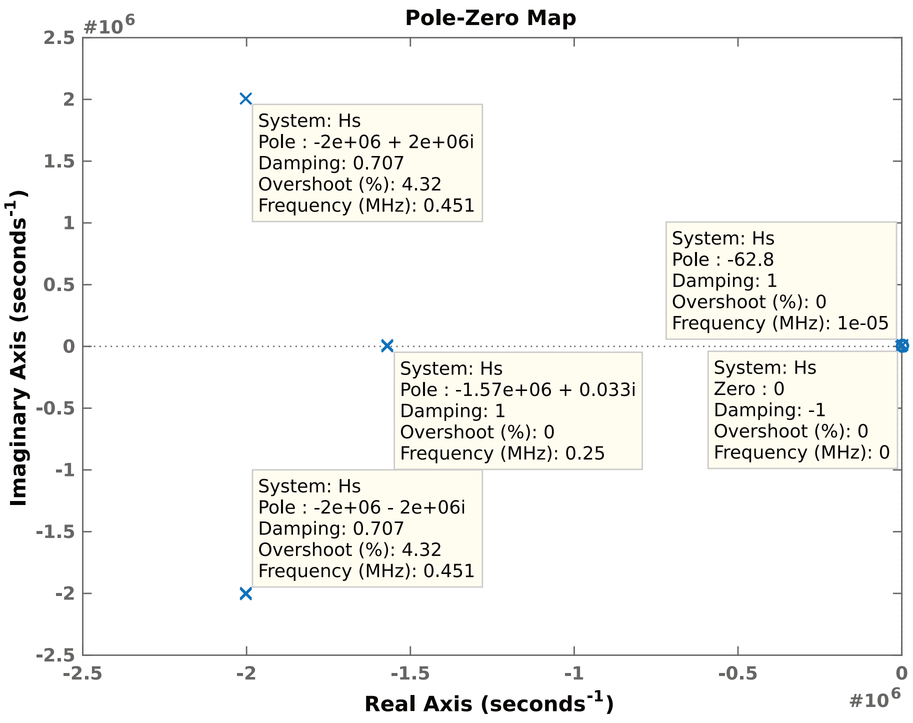


Figure 2.68: Pole-Zero Map of the time-continous system H_s.

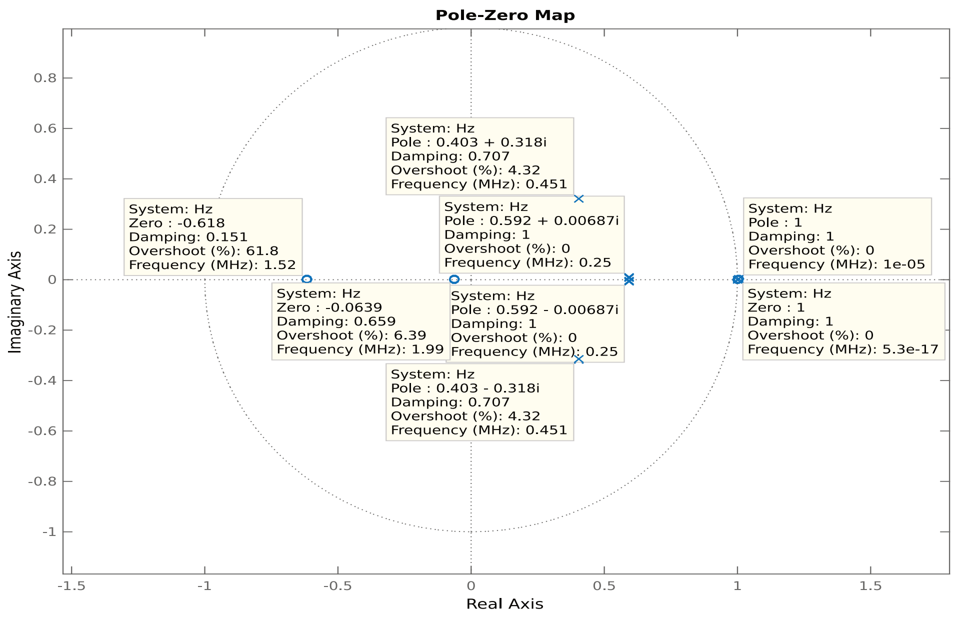


Figure 2.69: Pole-Zero Map of the disrete time system H_z.

BER as function of only IQ offset and channel noise using the Simulink tranceiver system model with matched SRRC-Biquad filtering

For the following simulations, exactly the same model has been used as shown in Figure L.1 in Appendix L Section L.1, but the only difference is that the SRRC receive filter has been replaced by the above described biquad type of receive filter. See Figure L.3 in Appendix L Section L.2.

Figures 2.70 and 2.71 show the results of a BER simulation for the transceiver system with matched SRRC-biquad filter and no phase quantization. One is shown with channel noise given in E_b/N_o and the other in SNR. Lets first compare the results from Figures 2.71 and 2.73. We see that in the latter Figure BER is starting to increase at channel noise $SNR = 16dB$ and this is, as mentioned earlier, exactly the moment where channel and phase quantization noise coincide together and the channel noise becomes dominant.

And in general, if we compare the result with that of Subsection 2.4.3 we can see that all curves, as a result of the biquad filter implementation, have raised up. Of course, the match between SRRC-Biquad is less than that of SRRC-SRRC filtering. And in addition, the bandwidth of the biquad filter implementation is larger than the one of the SRRC filter. That means that there is more noise added and hence, an increase of bit errors. The range of maximum permissible IQ offset for a BER of 10^-5 is consequently also somewhat lower in this case.

We know that the SNR of a real 5 bit PhADC [10] deteriorates compared to the modeled ideal 5 bit PhADC and thus the maximum allowable IQ offset will even go lower. Suppose $SNR_{am,ph}$, the equivalent amplitude SNR of the ideal 5 bit PhADC, is 22 dB. And if we now take the real Phase Domain ADC [10] then $SNR_{am,ph}$ will be 16.6 dB instead of 22dB. This means an increase of 5.4 dB in noise power.
If we calculate the amount of the increased noise power and add it to the noise power of the ideal 5 bit PhADC, the $SNR_{am,ph}$ decreases. This has the consequence that all the curves in Figures 2.70 and 2.73 go up. This means that we will read a lower maximum permissible IQ offset at the same curve in which SNR_{prec}, i.e. noise preceding the PhADC, is about the same level as $SNR_{am,ph}$. In this example it is 22 dB. The maximum permissible IQ offset will be in the vicinity of 10mV.

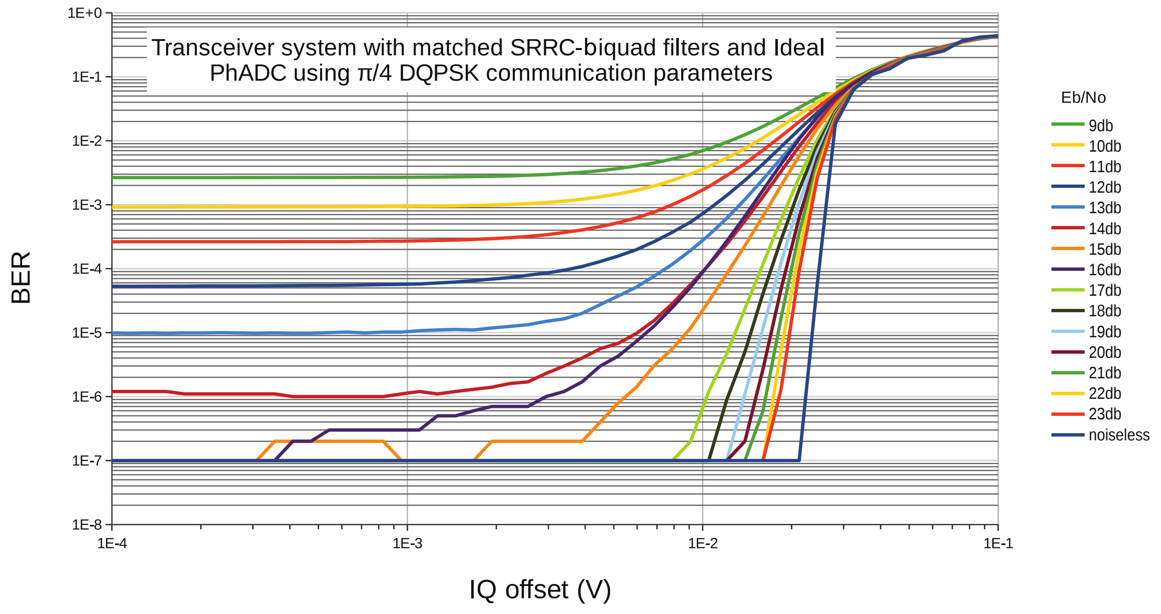


Figure 2.70: Plot of BER as a function of IQ offset and channel noise E_b/N_o of the Simulink transceiver system with matched SRRC-biquad filters and ideal PhADC using $\pi/4$ DQPSK modulation parameters.

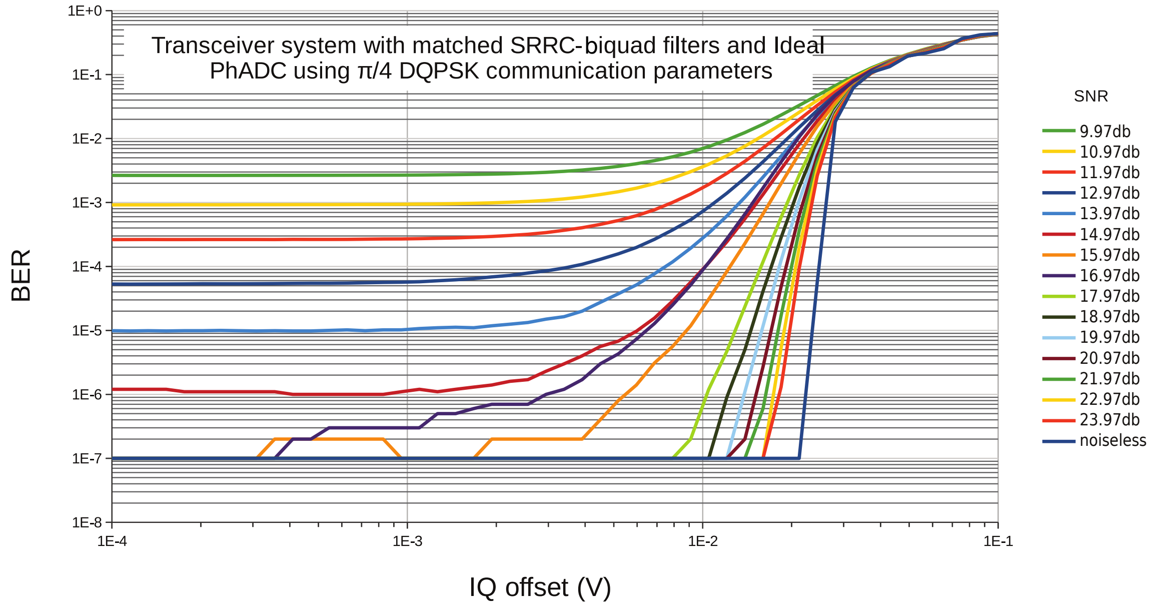


Figure 2.71: Plot of BER as a function of IQ offset and channel noise SNR of the Simulink transceiver system with matched SRRC-Receive Biquad filtering and ideal PhADC using $\pi/4$ DQPSK modulation parameters.

BER as function of IQ offset, channel- and quantization noise using the Simulink tranceiver system model with matched SRRC-biquad filtering

The Figures below have been discussed above and we see that due to phase quantization noise the curves have been shifted upwards, because more noise is added in the form of quantization noise from the PhADC and this results in more bit errors.

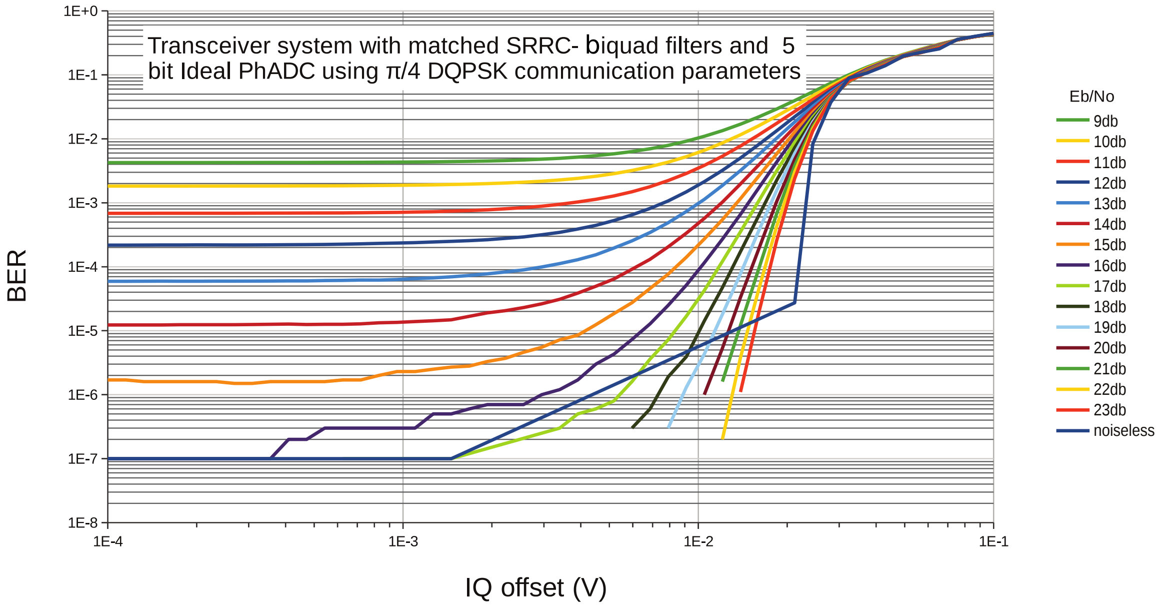


Figure 2.72: Plot of BER as a function of IQ offset and channel noise E_b/N_o including quantization noise E_b/QN_o of the Simulink transceiver system with matched SRRC-Receive Biquad filtering and 5 bit ideal PhADC using $\pi/4$ DQPSK modulation parameters.

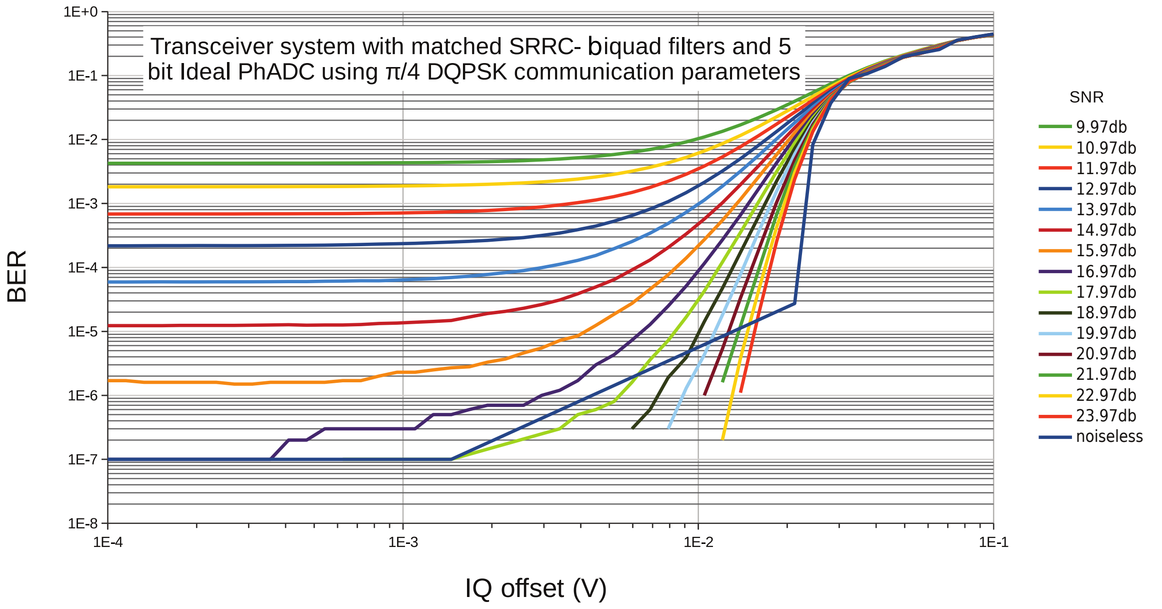


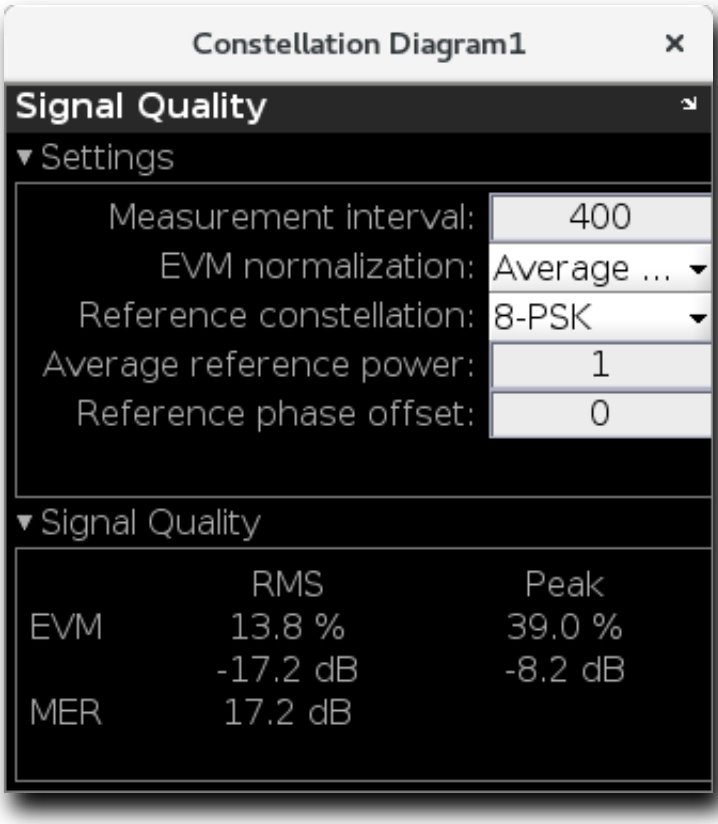
Figure 2.73: Plot of BER as a function of IQ offset and channel noise SNR including quantization noise E_b/QN_o of the Simulink transceiver system with matched SRRC-Receive Biquad filtering and 5 bit ideal PhADC using $\pi/4$ DQPSK modulation parameters.

We can conclude that IQ offset, channel and quantization noise do have effect on the BER. The IQ offset causes a shift in the constellation and an imbalance in the phase vector distribution. Due to this imbalance more phase errors occur.
In the presence of noise, a cloud forms around each reference point in the constellation, causing an error if a particular sample crosses the decision boundary. The more noise, the wider the cloud and thus the more frequent phase errors. From the plots we have seen that the more noise we add the more the curves raise and the less IQ offset is tolerated.

Signal quality of the received signal of the matched SRRC-Biquad filter model with IQ offset, channel and quantization noise

The Simulink model with the matched SRRC-Biquad filter, IQ offset, channel and quantization noise is shown in Figure F.5 in Appendix F Section F.3. This model ultimately contains all "non-idealities", i.e., channel and quantization noise, biqaud filter and 100 mV offset.
The simulation results of the signal quality is shown in Figure 2.74 after the system has been compensated and as we can see is still better than the permissible $\pi/4$ DQPSK EVM number shown in Figure 2.29. The result is obtained at;

- channel noise of $E_b/N_o = 20$ dB.
- IQ offset of 100 mV.
- quantization noise from the 5 bit PhADC.



Measurement interval: 400
EVM normalization: Average ...
Reference constellation: 8-PSK
Average reference power: 1
Reference phase offset: 0

	RMS	Peak
EVM	13.8 %	39.0 %
	-17.2 dB	-8.2 dB
MER	17.2 dB	

Figure 2.74: Signal quality dialog box.

3

IQ gain mismatch detection and compensation in a Phase-Domain ADC

The IQ gain mismatch detection principle will be described with the help of Figure 3.1 and Figure 3.2. Suppose that the amplitude of Q is larger than of I, the Q full scale range FS_Q would be larger than the I full scale range FS_I. The circular vector trajectory becomes elliptical like shown in Figure Figure 3.1. In this case there are more vectors in the Regions (1) + (3) than in the Regions (2) + (4). See red vectors in Figure 3.2(a). We say that there is an imbalance in the distribution of vectors in the Regions (1) + (3) and Regions (2) + (4). By detecting the number of vectors (i.e., the symbols containing the phase information) in these Regions we can detect the imbalance and thus the sign and the amplitude of the Q gain mismatch factor. Exactly the same principle can also be applied in case when the I full scale range FS_I is larger than the Q full scale range FS_Q. The circular vector trajectory becomes elliptical like shown in Figure 3.2(b). In this case there are more vectors in the Regions (2) + (4) than in the Regions (1) + (3). See red curves in Figure 3.2(b). Since we receive the phase digitally from the PhADC, it is very easy to realize a phase vector distribution density analysis in the digital domain.

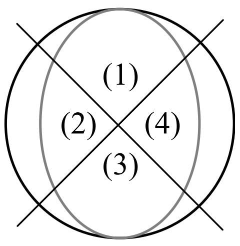


Figure 3.1: IQ gain mismatch. (Picture courtesy of Yao Liu [10].)

Similar calculations as (2.1) and (2.2) apply to the phase vector distribution imbalance IMB_{mis} and yield:

$$IMB_{mis} = 1 - \frac{4arctan(1-\alpha)}{\pi} \tag{3.1}$$

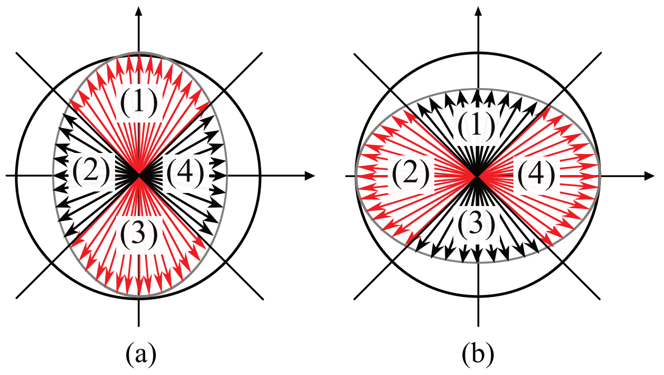


Figure 3.2: In case of a larger full scale range FS_Q the amount of red vectors in Region (1) and (3) is larger than the amount of black vectors in Region (2) and (4) (a), In case of a larger full scale range FS_I the amount of red vectors in Region (2) and (4) is larger than the amount of black vectors in Region (1) and (3) (b) .

The phase vector distribution imbalance IMB_{mis} can be found by taking the difference of the number of symbols in Regions (1) + (3) and Regions (2) + (4) and dividing this difference by the total number of symbols. See Equation (3.2). We define the joint number of symbols in Regions (1) and (3) and in Regions (2) and (4) as $Regions$ (1) + (3) and $Regions$ (2) + (4) and the total number of symbols as N.

$$IMB_{mis} = \frac{(Regions\ (1)+(3)) - (Regions\ (2)+(4))}{Regions\ (1)+(2)+(3)+(4)} = \frac{(Regions\ (1)+(3)) - (Regions\ (2)+(4))}{N} \tag{3.2}$$

Now that IMB_{mis} is known, we can easily determine the mismatch factor α with the help of (3.1).

$$\alpha = 1 - tan\left(\frac{\pi}{4}(1 - IMB_{mis})\right) \tag{3.3}$$

α is defined as the mismatch factor between the I and Q full scale range FS_I and FS_Q. See (1.4). Fortunately, for the IQ gain mismatch compensation we can suffice with just one control loop as shown in Figure 3.5. In this case the gain mismatch is controlled in Q. If the Q amplitude is greater than that of I then it is reduced to the amplitude of I. And if the amplitude of Q is smaller than that of I then it is increased to the amplitude of I. The advantage of this is that we can realize this only with one control loop thereby reducing the amount of hardware and power consumption which would have been consumed by the extra hardware.

Suppose we have an IQ amplitude compensation system like shown in Figure 3.9, where we omit for now the operation of the *Two Regions Detector*, and an IQ gain mismatch factor of 0.2 V (i.e., A_I = 0.8 V and A_Q = 1 V) has been detected and compensated. Figure 3.3 shows the feedback signal α_f in case of random IQ signals and Figure 3.4 for sinusoidal IQ signals. The simulation time has been taken long enough such that the feedback settles around the desired gain mismatch factor value of $\alpha_f = 0.2V$. α_f can be described by Equation (3.4):

$$\alpha_f = \begin{cases} 1 - \frac{A_I}{A_Q} & when\ amplitude\ mismatch\ is\ compensated \\ other & when\ amplitude\ mismatch\ is\ still\ not\ compensated \end{cases} \tag{3.4}$$

In Figure 3.3, thanks to the random behavior of phase transitions for random IQ signals, the feedback gain mismatch factor signal will average closer to the value of 0.2 V than when the phase transistions go through a circle in sequence as is the case for sinusoidal IQ signals. The feedback signal steps randomly around the desired value of 0.2 V, while for sinusoidal IQ signals the feedback signal goes periodically two steps up and two steps down as a result of detecting two samples in each Region by the two-regions detector. This is due to how the *two-regions detector* works, is discussed in Subsection 3.1.2. The information signal is sampled with a rate 8 times higher then its frequency with the results that 8 phase samples can be seen in the constellation. The detector detects two samples in each Region (1), (2), (3) and (4). And when the phase information cycles around the constellation we see a fixed pattern of two logic ones, two logic zeros, two logic ones, etc. See Figure 3.4.

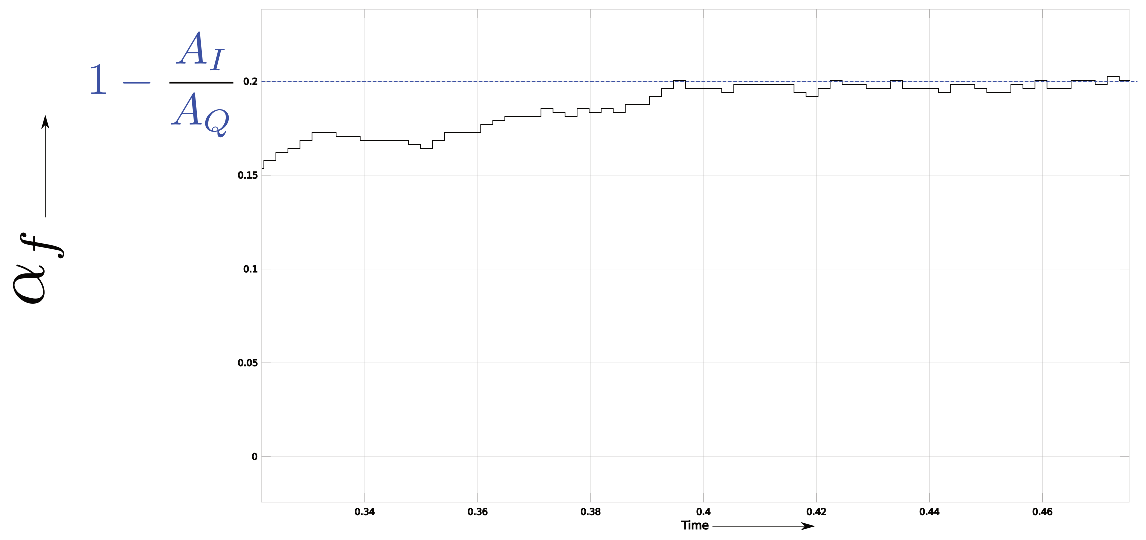


Figure 3.3: Due to the random phase transitions, the feedback mismatch signal α_f tries to settle around the desired value $1-\frac{A_I}{A_Q}$ in a random sequence of up and downs.

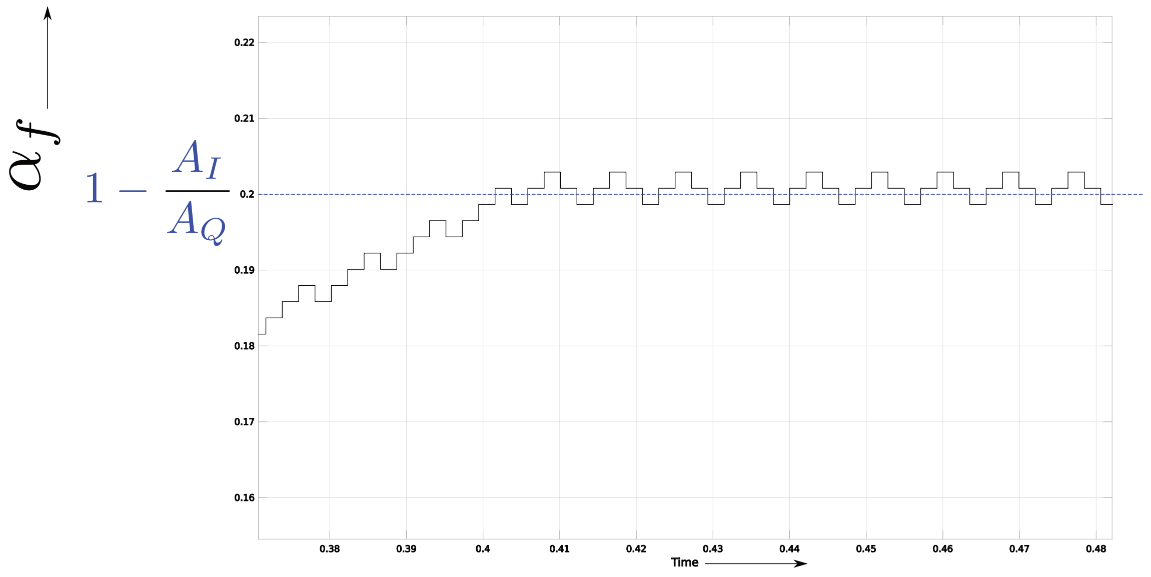


Figure 3.4: For linear input phase, the phase transitions go in a circle and the feedback signal α_f from the detector+integrator shows a fixed pattern of two pulses ups and two pulses down.

3.1. System architecture

Figure 3.5 shows the system architecture of the IQ gain mismatch compensation model based on input-output frame-based data processing. The system processes a frame of N phase samples at a rate that is $1/N$ times the phase sample rate of the original phase sample-based signal. The sample and frame-based processing is discussed in Section 2.2.
The system has only one feedback loop, namely for Q gain mismatch compensation. Basically the PhADC does an $arctan$ function as shown in Equation (1.3) and presents the 5 bit digital phase to the *IQ gain mismatch Detector*. In case of an IQ gain mismatch, the number of symbols are detected in the Regions (1) + (3) and Regions (2) + (4) of the IQ complex plane. The phase vector distribution imbalance is then detected and subsequently the detected Q gain mismatch is passed to the integrator.
The integrator is needed to eliminate the steady error between the actual and desired output phase, i.e., eliminate the IQ gain mismatch caused by the gain difference between A_I and A_Q. To enhance the speed of control

a gain block A is added in the feedback loop.

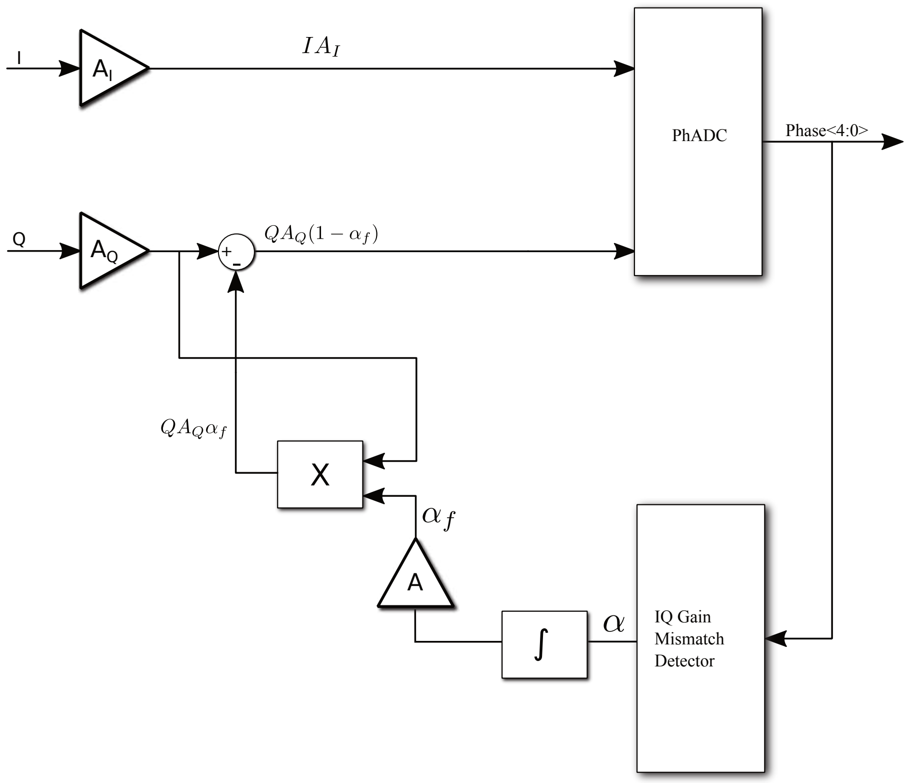


Figure 3.5: System architecture of the IQ gain mismatch detector based compensation model.

The system will be compensated when:

$IA_I = QA_Q(1-\alpha_f)$ or when $A_I = A_Q(1-\alpha_f)$

The gain mismatch factor α_f will then become equal to:

$\alpha_f = 1 - \frac{A_I}{A_Q}$

3.1.1. IQ gain mismatch detection based on IQ signal input-output data frames

Figure 3.6 shows a flowchart of the detection algorithm of the *IQ gain mismatch detector* block shown in Figure 3.5. This algorithm detects the number of symbols in Regions (1) + (3) and Regions (2) + (4) of the IQ complex plane in case of IQ gain mismatch compensation and calculates the phase vector distribution imbalance (i.e., IMB_{mis}) and feedback Q gain mismatch factor α. Frame-based operation refers to the fact that phase vector distribution imbalance IMB_{mis}, and from that the mismatch factor α, is determined on the basis of a frame of IQ data signal symbols. The algorithm first reads in one N sized frame of phase data symbols, detects its phase vector distribution in the IQ complex plane and then determines the imbalance and IQ gain mismatch. This process is iterative.

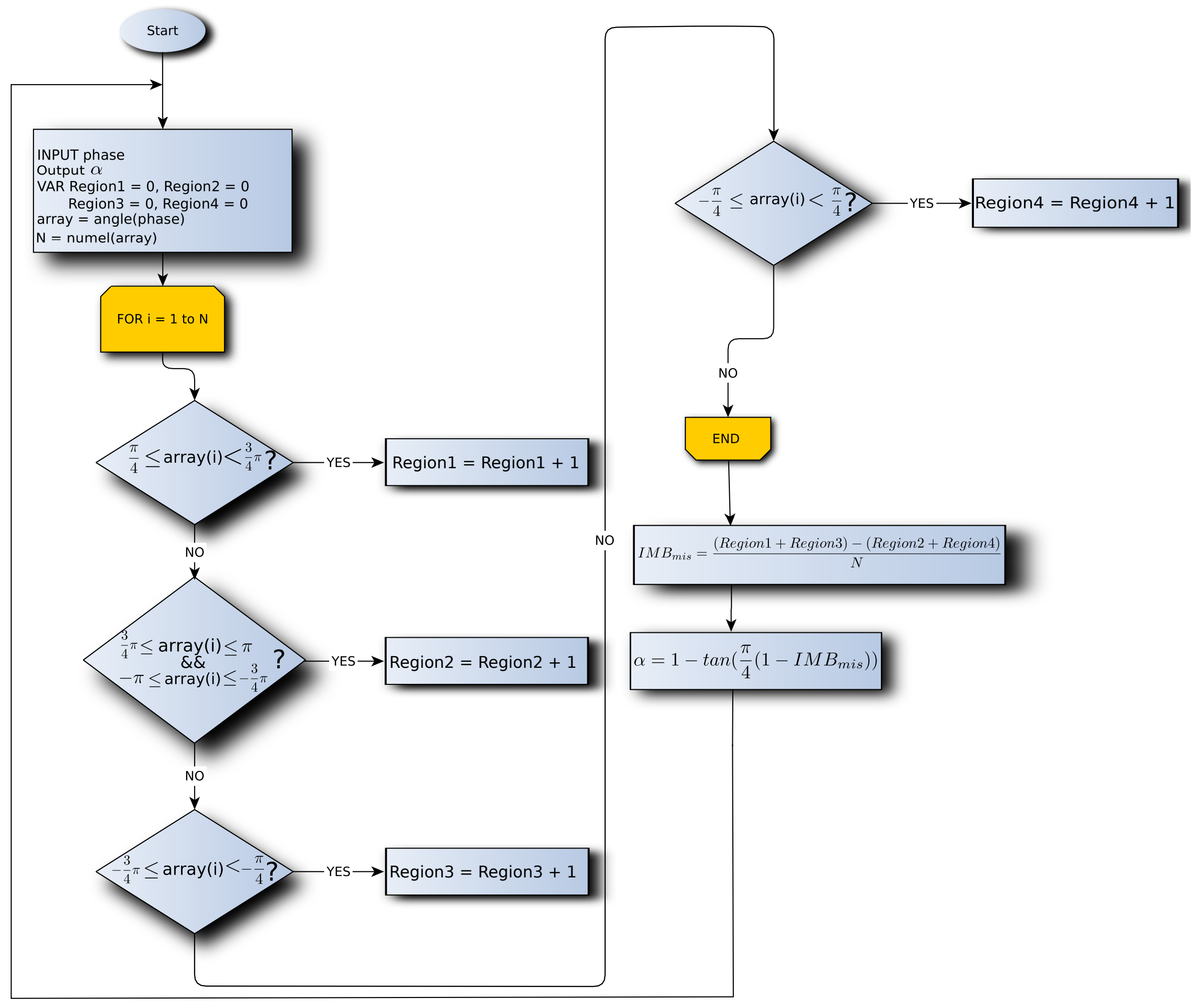

Start
INPUT phase
Output α
VAR Region1 = 0, Region2 = 0
Region3 = 0, Region4 = 0
array = angle(phase)
N = numel(array)
FOR i = 1 to N
$\frac{\pi}{4} \leq$ array(i) $< \frac{3}{4}\pi$?
YES
Region1 = Region1 + 1
NO
$\frac{3}{4}\pi \leq$ array(i) $\leq \pi$ && $-\pi \leq$ array(i) $\leq -\frac{3}{4}\pi$?
Region2 = Region2 + 1
$-\frac{3}{4}\pi \leq$ array(i) $< -\frac{\pi}{4}$?
Region3 = Region3 + 1
$-\frac{\pi}{4} \leq$ array(i) $< \frac{\pi}{4}$?
Region4 = Region4 + 1
END
$IMB_{mis} = \frac{(Region1 + Region3) - (Region2 + Region4)}{N}$
$\alpha = 1 - tan(\frac{\pi}{4}(1 - IMB_{mis}))$

Figure 3.6: Flowchart of gain mismatch detection algorithm for the frame-based *IQ gain mismatch detector* implementation.

The gain mismatch factor is determined by Equation (3.3). Using a small angle approximation, it can be written as:

$$\alpha \approx 1 - \frac{\pi}{4}(1 - IMB_{mis}) \tag{3.5}$$

and the phase-vector distribution imbalance IMB_{mis} is determined by (3.2). Averaging the difference in number of symbols in Regions (1) + (3) and in Regions (2) + (4), can be recognized as low pass filtering.
So, in frame-based operation the *IQ gain mismatch detector* averages but at the same time there is also an intergrator in the feedback path. The system thus performs a double integration compared to the single integration in sample-based operation, later more about this.
The conclusion is that the IQ gain mismatch detector is a low pass filter that behaves with integral action and has a pole not in the origin of complex s-plane.
Because the phase is offered in a digitized fashion by the PhADC, we can precisely determine the phase vector distribution imbalance, IMB_{mis}, so the gain mismatch factor α can be more or less detected accurately. The size of control steps outputted by the *IQ Gain Mismatch Detector* will then be dynamically adjusted according to the imbalance between the number of symbols in Regions (1) + (3) and Regions (2) + (4) in the IQ complex plane. The frame-based IQ gain mismatch compensation system will then , as a result, converge better to the desired final value of the feedback gain mismatch factor. See Figure 3.7.

In summary:

1. Using a small angle approximation, IQ gain mismatch is linearly related to the phase vector distribution imbalance.

2. The phase is digitally presented by the PhADC and thus the phase vector distribution imbalance can be detected accurately.

3. Because of Point 2, the IQ gain mismatch detection is also accurate compared to the sample-based operation and thus the system will do better controlling since the size of feedback control steps from *IQ gain mismatch Detector + integrator* are dynamically adjusted according to the imbalance of phase vector distribution between Regions (1) + (3) and Regions (2) + (4) in the IQ complex plane.

4. The frame-based *IQ gain mismatch detector* is a low pass filter with integral action and thus does have a pole not in the origin of complex s-plane.

If we now use the system shown in Figure 3.5 with this detection algorithm implementation in the *IQ gain mismatch detector* block then the feedback signal α_f looks like shown in Figure 3.7. The feedback signal fluctuate closer to the desired value of 0.2 V, but the size of control steps are also smaller and change dynamically. Both the frame-based and sample-based compensation system converge very well since the transient signal α_f decays to small value so that is almost in the steady state and is far within the 1% variation around the desired end value, in this case 0.2 V.

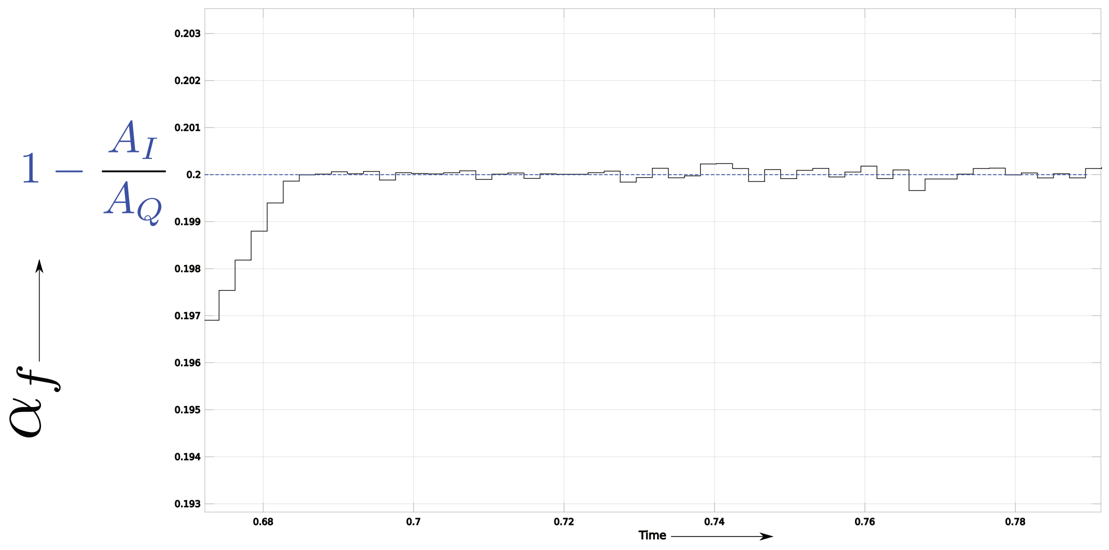


Figure 3.7: This is the feedback signal from the frame-based compensation system. The control action shows a fast and precise transition to the desired end value.

But the frame-based operation method also has its disadvantages. Firstly, arithmetics are performed and data needs to be stored, and for that we need a micro-controller (μC) and memory. Secondly, this will cost more power and circuit area on chip. Especially in the case of wireless body area networks, ultra low power consumption and size are very important. So, we have to look for an even simpler implementation of this gain mismatch compensation principle, in which we do not need a μC and memory.

3.1.2. IQ gain mismatch detection based on IQ signal input-output data samples

The detection algorithm for IQ gain mismatch compensation can even be made simpler. The method of detecting symbols in the IQ complex plane is the same as in the previous system but the difference between frame-based approach is that it works on an input-output sample-based processing principle. This means that the system tries to compensate for each passing symbol individually instead of on every frame of symbols as in frame-based processing.

In case of gain mismatch compensation, a symbol detected in Regions (1) + (3) in the IQ complex plane will output a logical '1' and a symbol detected in Regions (2) + (4) will output a logical '0', as shown in Figure 3.8.

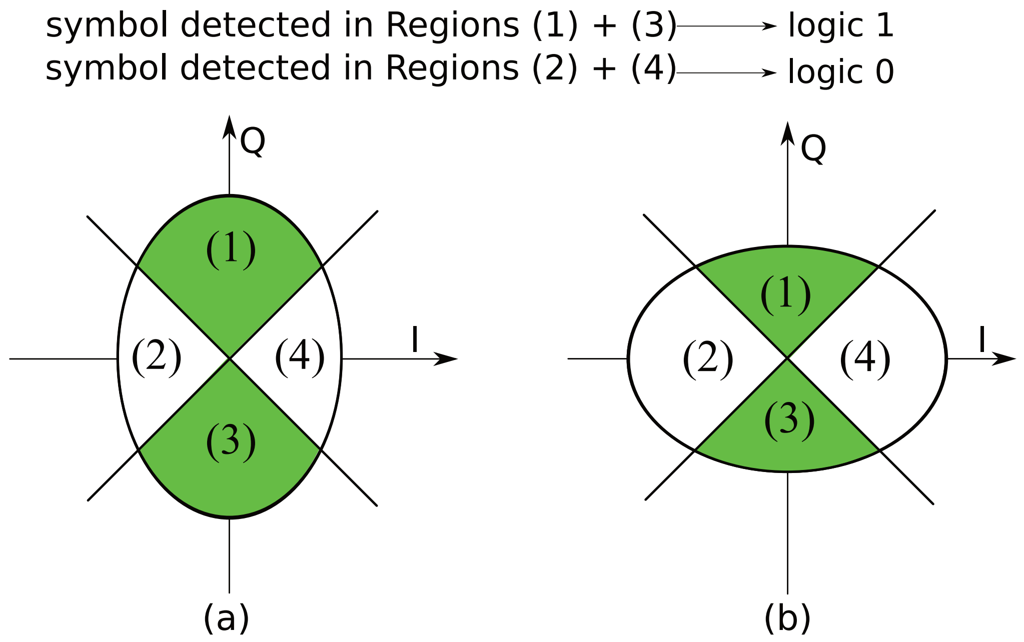

symbol detected in Regions (1) + (3) → logic 1
symbol detected in Regions (2) + (4) → logic 0
Q
I
(1)
(2)
(3)
(4)
(a)
(b)

Figure 3.8: Two-regions detection for IQ gain mismatch compensation. If a symbol is detected in Regions 1 and 3 the output is a logic "1" and if a symbol is detected in Regions 2 and 4 output is a logic "0" (b).

If the Q full scale range FS_Q is larger than the I full scale range FS_I, the circular vector trajectory becomes elliptical as shown in Figure 3.8(a). The number of symbols in Regions (1) + (3) is larger than in Regions (2) + (4). There is a phase vector distribution imbalance between Regions (1) + (3) and Regions (2) + (4). According to Equation (1.4) α will become positive and thus also the mismatch factor α_f and negative feedback will ensure that the amplitude of Q is controlled down to the amplitude of I. Exactly the opposite will occur when the number of symbols in Region (2) + (4) is larger than in Region (1) + (3). The feedback gain mismatch factor α_f will become negative and negative feedback will ensure that the amplitude of Q is controlled up to the amplitude of I.

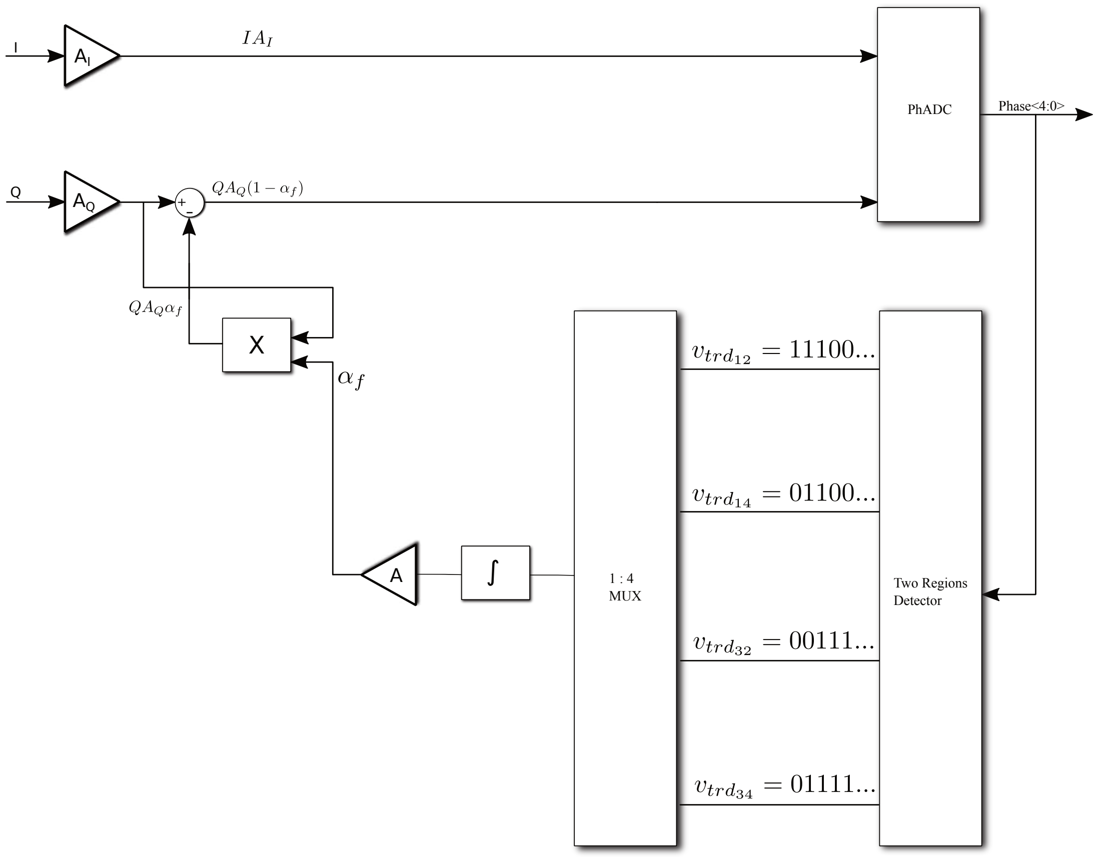


Figure 3.9: System architecture of the IQ gain mismatch compensation model based on the two-regions detector.

In order to find an implementation for the *two regions detector* detector, let's rewrite Equation (3.2 into:

$$IMB_{mis} = \frac{Regions\,(1)-(2)}{N} + \frac{Regions\,(3)-(4)}{N} \tag{3.6}$$

Equation (3.6) has a very similar structure as (2.3) and (2.4). Equation (3.6) tells us that we need to determine the phase vector distribution imbalance between Regions (1) and (2) plus the phase vector distribution imbalance between regions (3) and (4) to get the overall phase vector distribution imbalance IMB_{mis}.
But we have to pay attention to some details to make this detection principle work properly. Let's assume a random input phase and only IQ gain mismatch. The phase vector distribution in the regions around 0, $\frac{\pi}{2}$, $\pm\pi$ and $-\frac{\pi}{2}$ do not change much for IQ gain mismatch. See Figure 3.10. The imbalance occurs mainly in regions around $\pm\frac{\pi}{4}$ and $\pm 3/4\pi$, but even there, due to the elliptical shape, the phase vector distribution imbalance is very small.

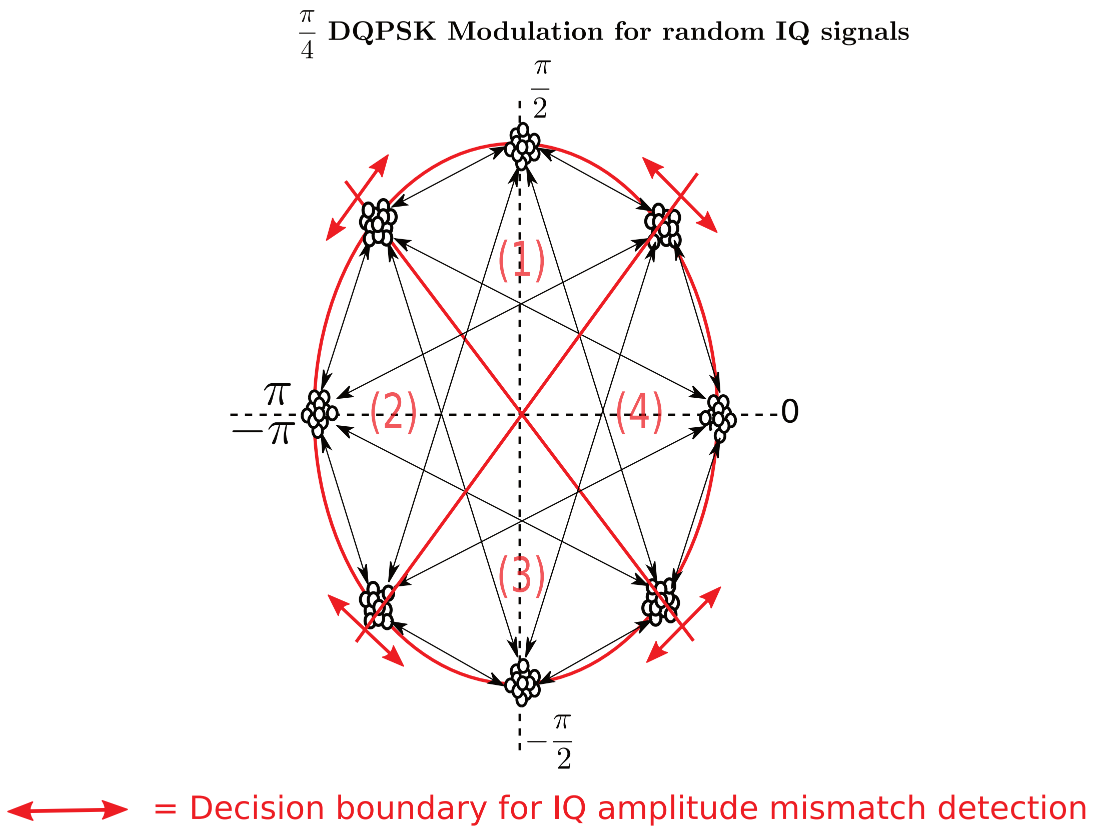

$\frac{\pi}{4}$ DQPSK Modulation for random IQ signals
$\frac{\pi}{2}$
(1)
π
$-\pi$
(2)
(4)
0
(3)
$-\frac{\pi}{2}$
= Decision boundary for IQ amplitude mismatch detection

Figure 3.10: Constellation diagram showing the decision boundaries between regions (1) and (2), (2) and (3), (3) and (4), (4) and (1) for the phase vector distribution imbalance detection used in IQ gain mismatch control.

Apart from that, due to the pulse shaping through the SRRC filters, not even yet involving channel noise, there is a cloud around the reference constellation points causing an error if a particular sample crosses the decision boundary. This random crossings from one region to another ensures that we can not properly detect the phase vector distribution imbalance at these boundaries. It will be a weak detection. The symbols at the reference locations that belong in one region may end up in the other. This random behavior is the reason that the distribution imbalance more or less averages to a low value. The strength of this detected vector distribution imbalance is just not large enough to trigger the IQ gain mismatch control, because the difference in phase vector distribution in these regions will be small in case of only IQ gain mismatch.

But we are going to find a way to increase the gain in the feedback loop using again (3.2), we can also rewrite it in a different form:

$$IMB_{mis} = \frac{Regions\ (1)-(4)}{N} + \frac{Regions\ (3)-(2)}{N} \tag{3.7}$$

it reveals that we also can control the phase vector distribution imbalance IMB_{mis} by detecting phase vector distribution imbalance between Regions (1) and (4) plus the phase vector distribution imbalance between Regions (3) and (2). Combining (3.6) and (3.7) yields:

$$\begin{aligned} IMB_{mis} = \frac{Regions\ (1)-(2)}{N} + \frac{Regions\ (3)-(4)}{N} + \frac{Regions\ (1)-(4)}{N} + \frac{Regions\ (3)-(2)}{N} = \\ 2\frac{(Regions\ (1)+(3))-(Regions\ (2)+(4))}{N} \end{aligned} \tag{3.8}$$

So all the regions interact with their neighboring regions and no two regions are left uncontrolled for phase vector distribution imbalance. Not only we detect the phase vector distribution imbalance between Regions (1) and (2), (3) and (4) , but also between Regions (2) and (3), (1) and (4).

Equation (3.8) is similar to (3.2) except a multiplication factor 2, hence we have increased the gain of the detected phase vector distribution imbalance by a factor of 2 and with another factor 4 by the multiplexer shown in Figure 3.9.

For linear input phase however, the phase vector distribution imbalance is well detectable because most of the phase samples situate in their own region and if there is a IQ gain mismatch and therefore the circular phase vector distribution becomes elliptical then a number of phase vectors are transferred to another region, as a result of which the phase vector distribution density increases there and decreases in the other.

In case of IQ offset, any shift in the constellation will cause a majority of phase samples to cross the decision boundary. See Figure 3.11(a) and 3.11(b). So at these locations the phase vector distribution imbalance can be easily detected. In this case we can conclude that the phase vector distribution imbalance detection for IQ offset compensation is more robust than that the one for IQ gain mismatch compensation.

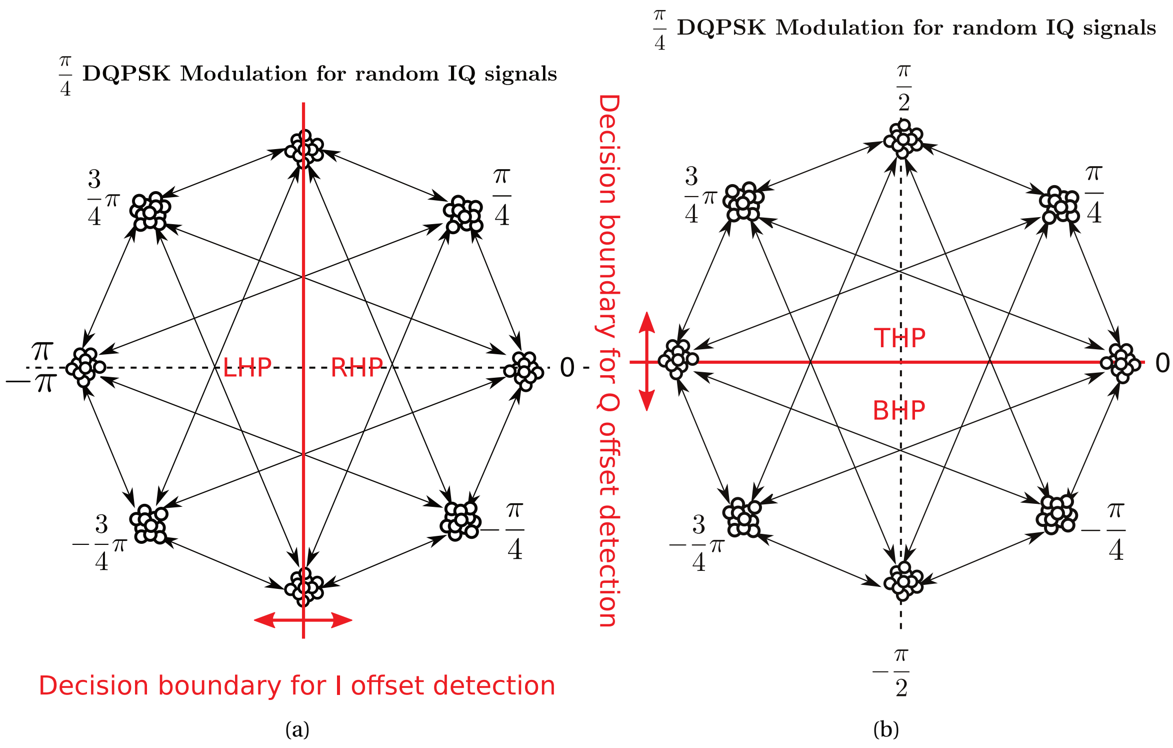


Figure 3.11: Showing decision boundary in constellation for phase vector distribution imbalance for: I offset (a), Q offset (b).

If we look at Equation 3.8 we see a summation of four terms and every term constitutes a signal. Comparing this with Equation (3.7), means we have increased the control from two signals to four signals.

So we have to make four feedback signals and use a 1:4 multiplexer (MUX) to feedback one signal.
A MUX is a circuit with multiple data inputs, one or more selection inputs and an output. The selection signals correspond to a binary code (selection address) to select one of n data inputs.
The complete IQ gain mismatch compensation system based on a so-called *Two Regions Detector* is shown in Figure 3.9. It is so named because in principle the phase vector distribution imbalance detection is always determined between two regions, i.e., between region (1) and (2), (3) and (4), (1) and (4), (3) and (2), hence the name *Two Regions Detector*.

Based on this detection principle the *Two Regions Detector* will produce binary signals of ones and zeros at the output and multiplexed to the input of the integrator as shown in Figure 3.9. The integrator produces a stept up or stept down signal which will eventually settle at a certain value when the gain mismatch is compensated. Due to the changing phase vector distribution imbalance in the 4 regions of the IQ complex plane the control signal α_f will produce to some extent dynamically changing control steps. See Figure 3.3.

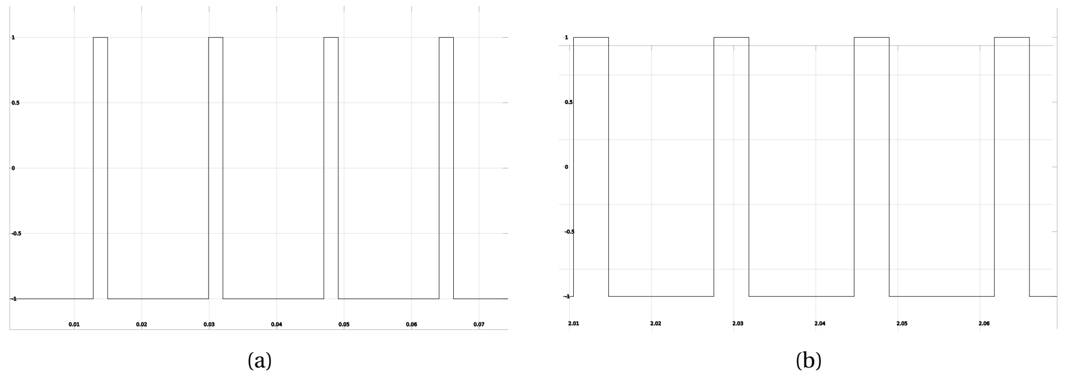


Figure 3.12: The vector distribution imnbalance translates into a non-25% duty cycle at each output $v_{trd_{xx}}$ of the *Two Regions Detector*: vector distribution not in balance, duty-cycle ≠ 25% (a), vector distribution is in balance and produces a perfect duty cycle of 25% (b).

It is interesting to see that in case of IQ sinusoidal input signals the imbalance of phase vector distribution in the IQ complex plane is translated into a duty cycle at the outputs of the *Two Regions Detector*. See Figure 3.12. A duty cycle of less than 25% shows that there is still some positive gain mismatch remaining. Figure 3.12(a) shows, for example, that there is still positive gain mismatch remaining. On the other hand, in Figure 3.12(b), the duty-cycle is exactly 25%, which means that the phase vector distribution is perfectly in balance and the gain mismatch is eliminated.

The *Two Regions Detector* will output the phase vector distribution unbalance as a group of logic "1" and logic "0" pulses and therefore we also see this in the duty cycle of each signal. The duty cycle of each signal presents the phase vector distribution imbalance between two regions in the IQ complex plane and, since we compare two quadrants, we see a duty cycle of 25% when the gain mismatch is compensated. To make it clearer, between two IQ complex halve planes the detector would produce signals with a duty cycle of 50% and between two IQ complex quarter planes it would produce signals with a 25% duty cycle ,etc.

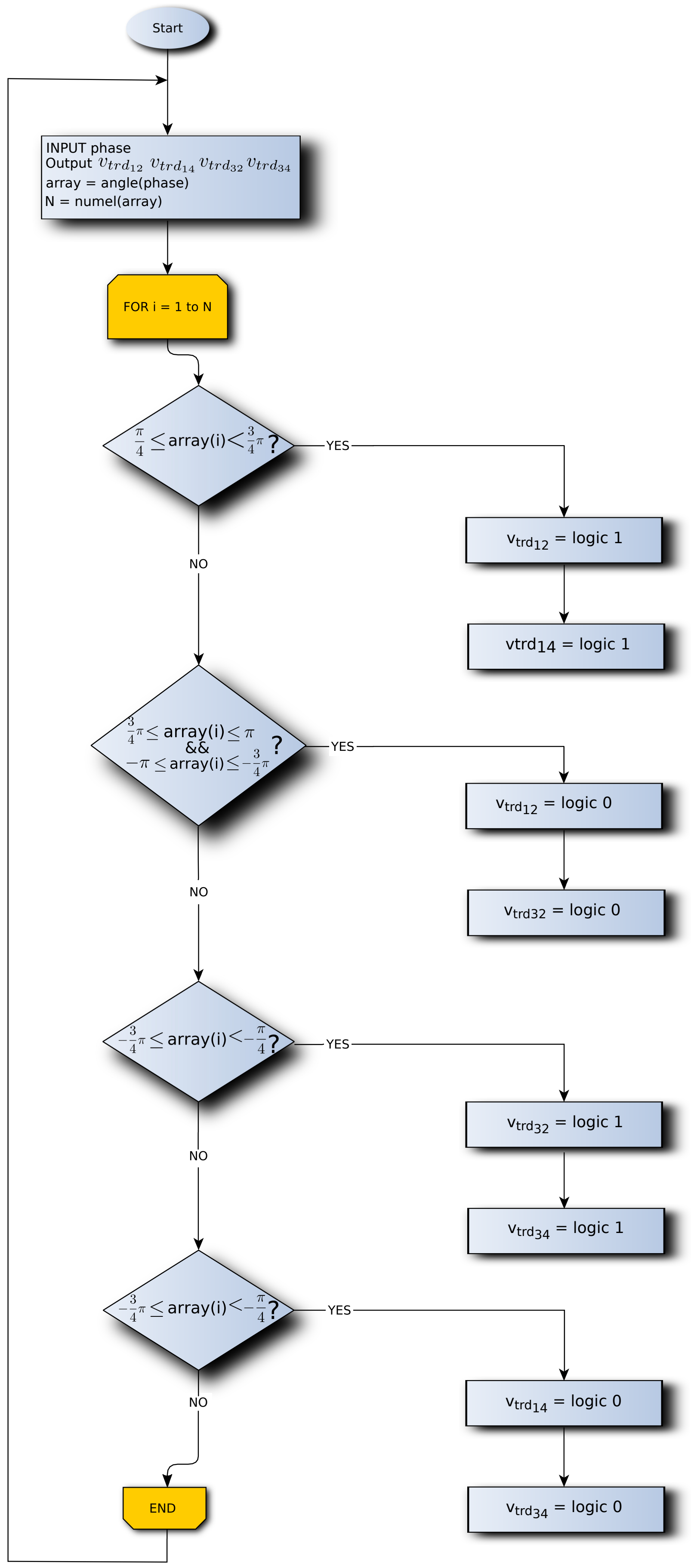

Start
INPUT phase
Output $v_{trd_{12}}$ $v_{trd_{14}}$ $v_{trd_{32}}$ $v_{trd_{34}}$
array = angle(phase)
N = numel(array)
FOR i = 1 to N
$\frac{\pi}{4} \leq$ array(i) $< \frac{3}{4}\pi$?
YES
$v_{trd_{12}}$ = logic 1
vtrd$_{14}$ = logic 1
NO
$\frac{3}{4}\pi \leq$ array(i) $\leq \pi$ && $-\pi \leq$ array(i) $\leq -\frac{3}{4}\pi$?
YES
$v_{trd_{12}}$ = logic 0
$v_{trd_{32}}$ = logic 0
NO
$-\frac{3}{4}\pi \leq$ array(i) $< -\frac{\pi}{4}$?
YES
$v_{trd_{32}}$ = logic 1
$v_{trd_{34}}$ = logic 1
NO
$-\frac{3}{4}\pi \leq$ array(i) $< -\frac{\pi}{4}$?
YES
$v_{trd_{14}}$ = logic 0
$v_{trd_{34}}$ = logic 0
NO
END

Figure 3.13: Flowchart of the two-regions detection algorithm for the *Two Regions Detector* implementation.

Figure 3.13 shows a flowchart of the two-regions detection algorithm of the *Two Regions Detector* shown in Figure 3.9.
If we compare this flowchart with the one shown in Figure 3.6 we see that the outputs $v_{trd_{12}}$, $v_{trd_{14}}$, $v_{trd_{32}}$, $v_{trd_{34}}$ are directly controlled within the *For loop*. The control algorithm shown in Figure 3.9 will output α after its *For loop* has been finished. This clearly shows the difference between frame- and sample based processing. The two-regions detection algorithm reacts on each symbol individually while the IQ gain mismatch detection algorithm first processes the whole frame of phase data symbols and then outputs α. And in contrast, the sample-based operation using the *Two Regions Detector* has no integral action but outputs a logical "0" or a logical "1" based on the position of a symbol in the IQ complex plane. It is not detecting the IQ gain mismatch but only position of a phase symbol in one of the regions (1), (2), (3) and (4). So we can assume that the *Two Regions Detector*, unlike IQ *Gain Mismatch Detector*, is not a filter and therefore has no pole but has a gain that is highly non-linear.

In summary:

1. The *Two Regions Detector* does not detect IQ gain mismatch but outputs a logical "0" or a logical "1" based on the position of a symbol in the regions (1), (2), (3) and (4) in the IQ complex plane.

2. So, the *Two Regions Detector* is not a filter that has integral action but has a gain that is highly non-linear.

The IQ gain mismatch compensation system based on the *IQ Gain Mismatch Detector* has integral behavior and the IQ gain mismatch compensation system based on the *Two Regions Detector* has no integral behavior, but both systems have an integrator in the feedback path. And that is why we can say that the *IQ Gain mismatch Detector* based compensation system is a second order system and the compensation system based on *Two Regions Detector* is a first order system. The response of both systems should therefore be different.

Concerning the speed of compensation of both systems, as discussed in Subsection 2.2.2, depending on the position of the poles and the loop gain one can be faster than the other.
To test the speed of both systems we use a Simulink simulation system setup like shown in Figures O.1 and O.3 in Appendix O Sections O.1 and O.2.

For measuring the speed of IQ gain mismatch compensation we can not use the bit error rate as a measure since no noise is added to the channel and also the PhADC is ideal and thus immune to phase magnitude variations. As a consequence, we can not observe any bit errors except 1 error which arise from one single artefact in the beginning of the test sequence. In other words, there's no error mechanism that can cause the bits to be incorrectly detected. We can only observe the moment when the IQ gain mismatch is compensated and note the three parameters; bit error rate, number or bit errors and total number of bits.
Tabel 3.1 shows the simulation results. When we look at the total simulation time of both systems, we see that the system with the *Gain Mismatch Detector* is twice as fast as the system with the *IQ Two Regions Detector*. We could ask ourselves what happens if we would not choose a first order integrator ($1/s$), but a second order ($1/s^2$) integrator in the feedback path of the sample-based system just like the frame-based which is based on second order compensation. Perhaps then the speed of both system will be the same.

IQ gain mismatch compensation system based on	**BER Measurement**	**Result**	**Total simulation time (s)**
Two regions detection	Bit Error rate: Number of error bits: Totaal number of bits:	$5.32.10^{-6}$ 1 $1.88.10^{5}$	1.4
IQ gain mismatch detection	Bit error rate: Number of error bits: Totaal number of bits:	$3.87.10^{-6}$ 1 $2.584.10^{5}$	0.69

Table 3.1: BER measurements of the IQ gain mismatch compensation system based on IQ amplitude- and Two regions detection.

Figure 3.14 shows the response of feedback gain mismatch signal α_f of both systems.

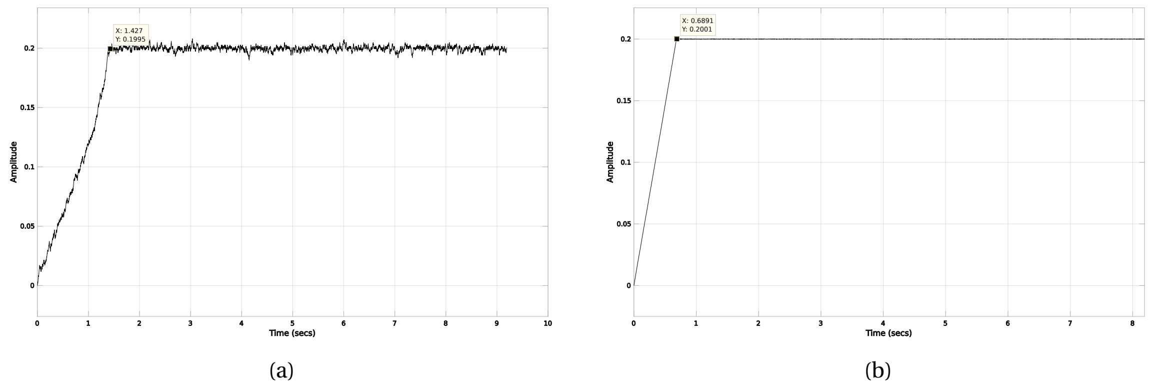


Figure 3.14: System simulation: response of feedback gain mismatch signal α_f after compensation based on *Two regions detection* (a) and *IQ Gain Mismatch Detection* (b).

The sample-based model has some issues to be considered:

- the sample-based compensation is not smooth since it does only a *Two Regions* and no gain mismatch detection.
- so, it does not converge as well as the frame-based compensation.
- The gain in the feedback path has to be chosen properly since it is sensitive to large gains and tends to make the system instable.

But has also very attractive advantages:

- the control system can be simple to implement.
- no need for memory and μC, just simple combinatorial logic is sufficient. As a consequence, less energy consumption due to simple architecture.
- it can be implemented fully digitally, no need for analog circuits.

So given the advantages, despite the disadvantages, a sample-based model is a better choice.

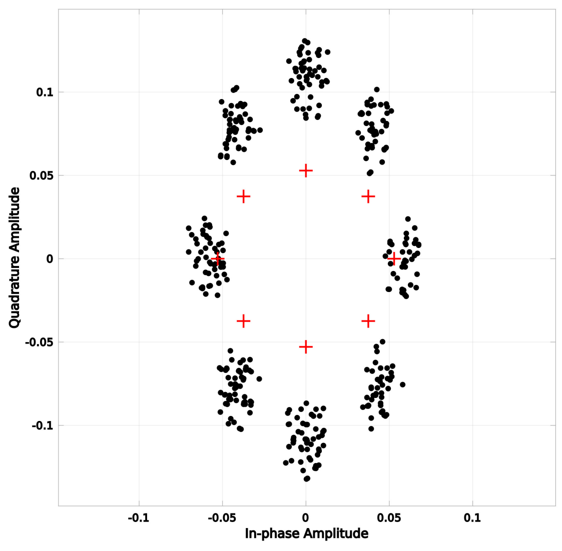
3.2. Functional performance Simulink system model

In this subsection we will test the functionality of the IQ gain mismatch compensation system. Both the sample-based and the frame-based systems have been tested and for this we use the Simulink models from

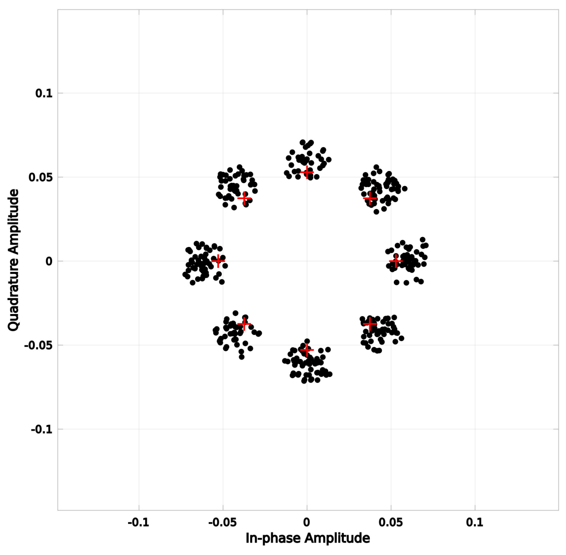
Appendix O, Sections O.1 and O.2. The settings of each block in the models are indicated in boxes in the relevant model. Detailed information about the blocks used in model can be found in Appendix G.
The information from Section 2.3 about the SRRC Transmit and Receive filter and the Simulink model of the PhADC also applies here.

3.2.1. Simulation of IQ gain mismatch compensation

Figure 3.15(a) shows the constellation of receiver input signal in black after gain mismatch is added with a gain for Q twice as large as the that of I signal, i.e., $A_I = 0.5$ and $A_Q = 1$, resulting in a gain mismatch of 6 dB. Incidentally, this is a pretty large and unrealistic value, but this has been done to clarify the concept of IQ gain mismatch. As we can see in the figure, the clouded groups of constellation points are arranged in such a way that it describes an elliptical shape. The Q fullscale range FS_Q is twice as large as the I fullscale range FS_I. Ideally, these clouds should be arranged in a circular fashion, viz., around the origin, in the reference points indicated with the red crosses. These red crosses define the exact locations of the constellation points for $\pi/4$ DQPSK modulation with no disturbances. But after gain mismatch compensation the constellation points are again at the reference locations. See Figure 3.15(b). So we can conclude that the system is working.



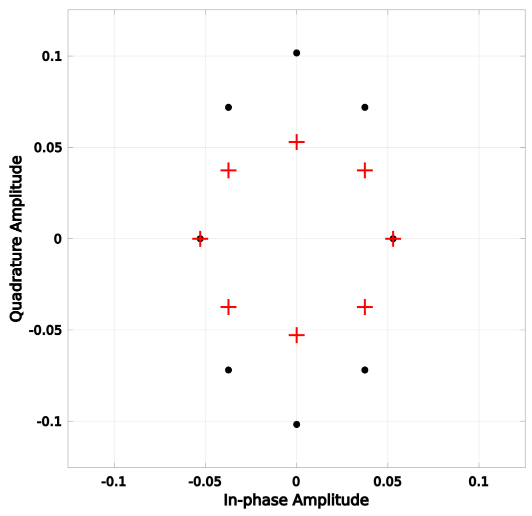
(a)



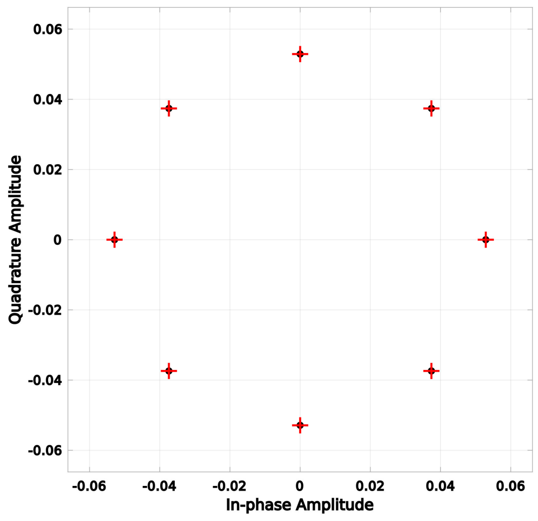
(b)

Figure 3.15: Reference constellation (in red) and receiver input signal (in black) for uncompensated IQ gain mismatch (a) and compensated IQ gain mismatch (b).

Figure 3.16(a) shows the input signal of the PhADC. The clouds around the reference points in the constellation are now much smaller due to the raised cosine filtering and the symbols will be closer to the reference locations when the system is compensated. After gain mismatch compensation it looks like Figure 3.16(b).



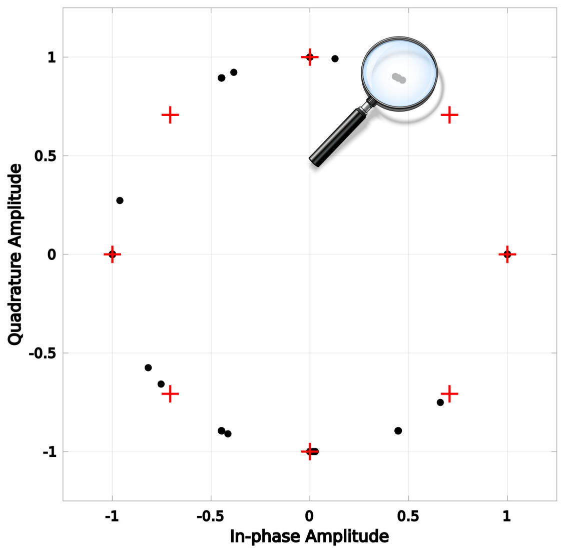
(a)



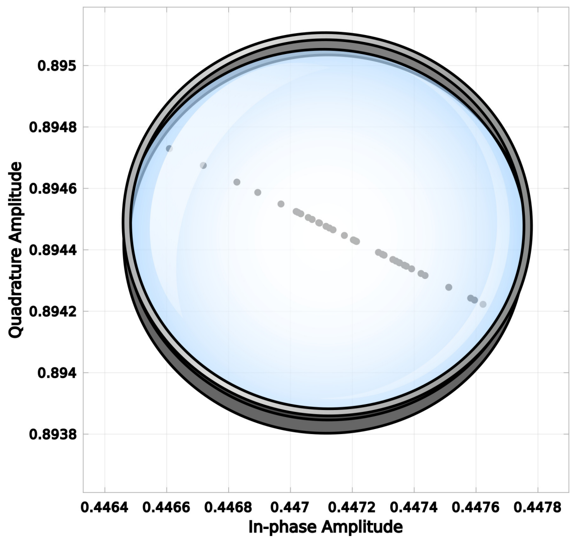
(b)

Figure 3.16: Reference constellation (in red) and Phase-Domain ADC input signal (in black) for uncompensated IQ gain mismatch (a) and compensated IQ gain mismatch (b).

It is also interesting to see what happens at the output of the PhADC, see Figure 3.17(a). At the beginning of gain mismatch compensation, much of the symbols are concentrated in Regions (1) and (2) of the constellation diagram as a result of the gain mismatch. During compensation they move gradually to their reference points. See Figures 3.17(a) and 3.18(a). The concentration of symbols in the Regions (1), (2), (3) and (4) are spread over the constellation circle in the direction of the reference locations. It is also important to see that the phase vector magnitudes, the tip of these vectors indicates the location of a symbol in the constellation, are clipped to a magnitude of one. Hence, the symbols describe a unity circle. The ideal PhADC is immune to phase vector magnitude variations and that is the reason why it is clipped to a magnitude of one. Figure 3.17(b) shows a zoomed-in small part of the symbols distribution. Finally, after compensation completion, all the symbols are at their reference locations. See Figure 3.18(b).



(a)



(b)

Figure 3.17: Reference constellation in (red) and DQPSK demodulator input signal (in black) for uncompensated IQ gain mismatch (a) and zoomed in part of a black spot in (a).

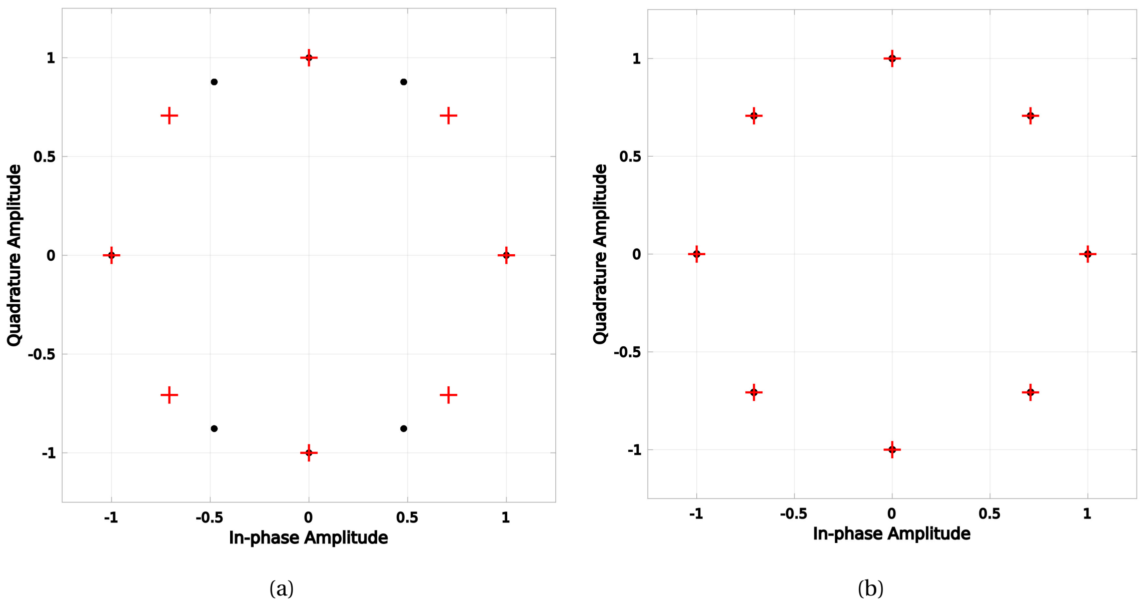


Figure 3.18: Reference constellation (in red) and partially compensated DQPSK demodulator input signal (in black) (a) and fully compensated IQ gain mismatch.

Figure 3.19 shows the phase vector trajectory of the transmitter and receiver signal. As we can see in Figure 3.19(a), due to the pulse shaping by the SRRC transmit filter the phase symbols are scattered around the ideal points and the phase transitions are gradual and not abrupt. At the end of receiver chain we see perfect $\pi/4$ DQPSK phase transitions. See Figure 3.19(b).

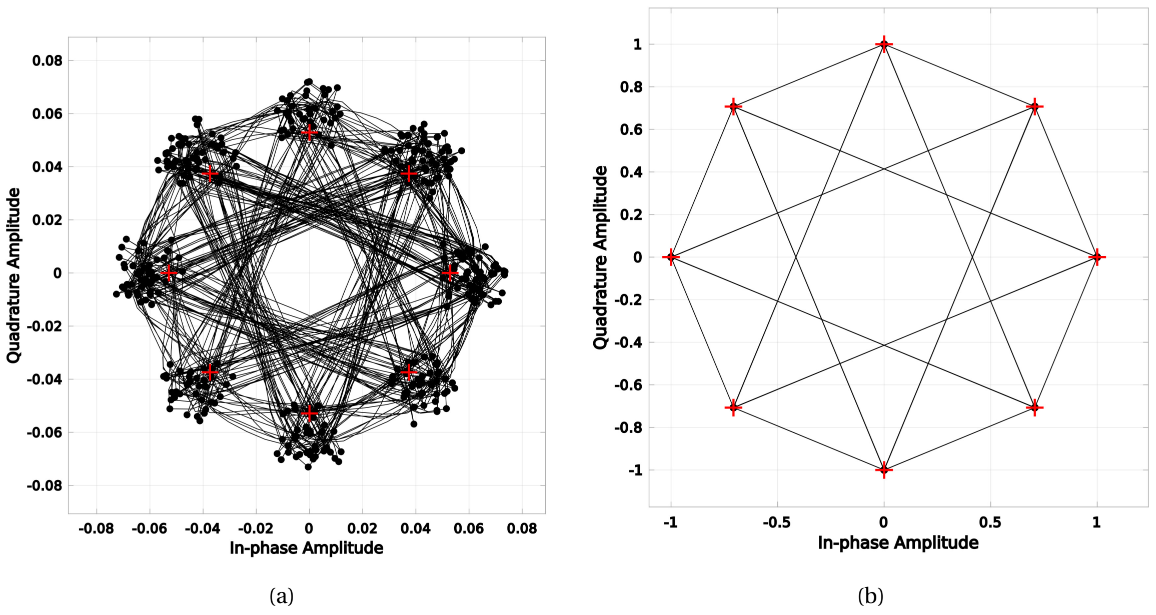
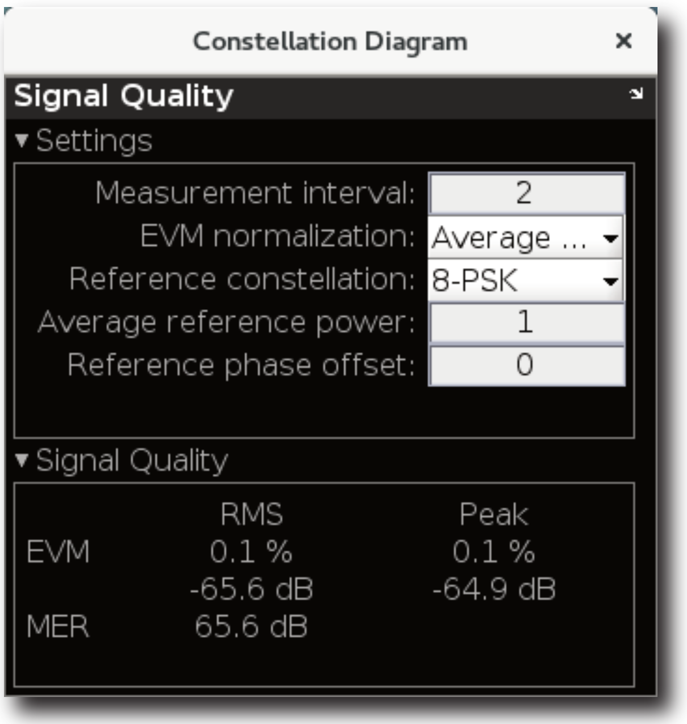


Figure 3.19: Shows signal trajectory of transmitted T signal (a) and received R signal (b).

3.2.2. Signal quality of received signal

The definitions of the performance metrics EVM and MER are given in Subsection 2.3.2. It is therefore sufficient to give only the results of the simulation. See Figure 3.20.


Constellation Diagram

Signal Quality

▾ Settings

Measurement interval:	2
EVM normalization:	Average ...
Reference constellation:	8-PSK
Average reference power:	1
Reference phase offset:	0

▾ Signal Quality

	RMS	Peak
EVM	0.1 %	0.1 %
	-65.6 dB	-64.9 dB
MER	65.6 dB	

Figure 3.20: Signal quality dialog box.

As we can see, the EVM is much better than the permissible EVM number for a $\pi/4$ DQPSK modulation shown in Table 2.29. But again, is not a fair comparison because no channel and quantization noise has been added yet and the SRRC receive filter has not yet been replaced by an analog channel select filter. This will be dealt with in Subsections 3.3.2 and 3.3.3.

3.3. System performance analysis

The information about the effect of channel and quantization noise on the bit error rate from Section 2.4 also applies here.

3.3.1. Bit Error Rate (BER) measurements as a function of IQ gain mismatch

For this simulation we only look at the effect of IQ gain mismatch on the BER. Noise is not included. For this simulation the model is applied as shown in Figure P.1 of Appendix P section P.1. In the model we see that the AWGN channel noise block has been removed and instead a gain mismatch is added.

Without noise and IQ amplitude mismatch, there should be no bit errors as the number of bits used in $\pi/4$ DQPSK is 2 and the resolution of the PhADC is infinite, which of course is infinitely more than needed, and immune to phase vector magnitude variations. In other words, there is no error mechanism that can cause the bits to be incorrectly detected. Hence, the BER will be 0.
Let's take a look at the results shown in Figure 3.21. This test was done at 10^7 bits and as we can see, up to a gain mismatch of about ±25 dB there is just one bit error. It is expected to be 0, but we do see 1 bit error (i.e., BER=10^-7) and this might arise from one single artefact at the beginning of the sequence of 10^7 bits. Of course, this result is still much better than the required BER of 10^{-5}. So, this part of the curve tells us that an IQ gain mismatch up to a maximum of ±25 dB is tolerable and still gives virtually no bit errors. If the gain mismatch is increased above this value then we see bit errors. This region shows a quite sharp transition, a threshold. Above -25 dB and below 25 dB we hardly see errors and between ±25 and ±30 dB we will have a BER in the order of 1.10^-7 to 2.10^-3, i.e., from 1.10^-5% to 0.2%. Since gain mismatch originates from device mismatches in the passive front-end (PFE), baseband LNA (BB LNA) and analog baseband (PGA and 2^{nd}-

order filter)[10], it is a constant deterministic interference. The relationship with a sharp transition makes sense suggesting that the BER doesn't increase smoothly with IQ gain mismatch as is in the case of noise as shown in Figures 2.36 and 2.37. Noise in the signal chain prior to the PhADC also blurs the constellation (the points become clouds) and this every now and then leads to incorrect detection of the phase. Of course, the more noise, the wider the cloud and thus the more frequent incorrect detection of the phase occurs. For IQ gain mismatch it is different. For an IQ gain mismatch between -25 dB and 25 dB we have no errors at all and a little below -25 and above 25 dB, we will have a BER in the order of 1.10^{-5}% to 0.2%. Continuing with Figure 3.21, below a IQ gain mismatch of -30 dB and above 30 dB, the BER increases at a smaller rate to a BER of 0.1.

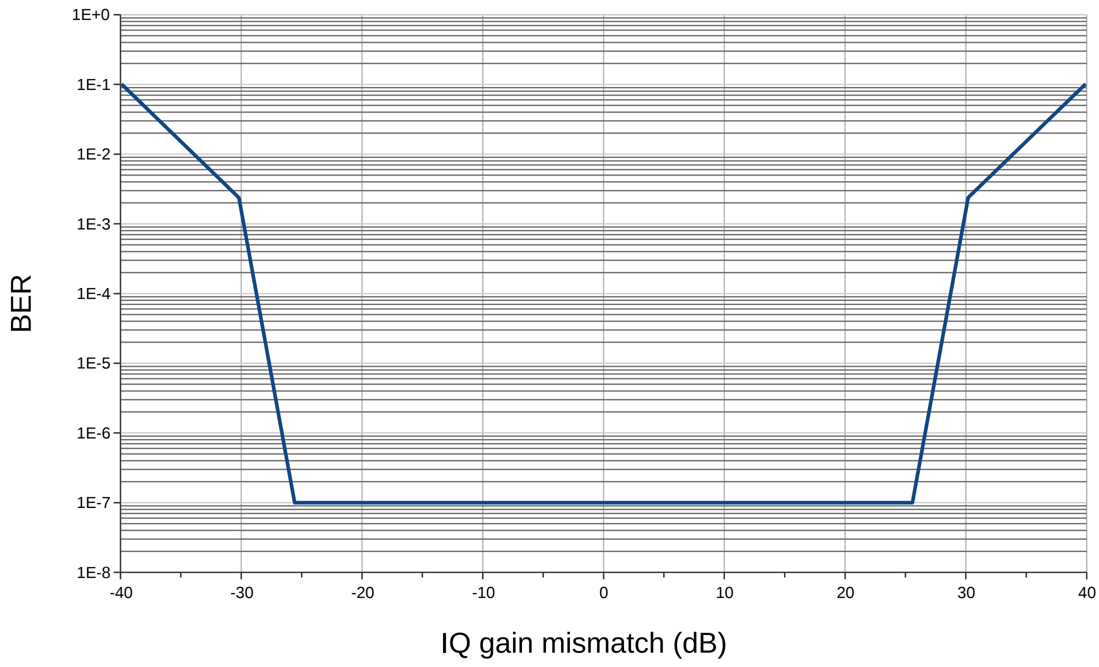


Figure 3.21: Plot of BER as a function of IQ gain mismatch of the Simulink transceiver system with matched SRRC filters and ideal PhADC using $\pi/4$ DQPSK communication parameters.

3.3.2. Bit Error Rate (BER) measurements as a function of both IQ gain mismatch and noise

Previously we have tested the effect of IQ gain mismatch on the BER, but now noise is included to the gain mismatch. At each injected noise value, the gain mismatch is swept from -40 to 40 dB and a new BER curve as a function of this gain mismatch is included in the graphs below. Noise in the signal chain prior to the PhADC will blur the constellation (i.e., the points become clouds) and this will lead to an increase of phase errors, hence, BER. The more noise the wider the cloud and thus the more frequently phase errors occur.

BER as a function of only IQ gain mismatch and channel noise

Figures 3.22 and 3.23 show the BER as a function of IQ gain mismatch and channel noise, but without quantization noise. The PhADC is ideal in the sense of having no phase quantization, i.e., has an infinite resolution. The figures show identical curves but in Figure 3.23 the injected channel noise is in terms of E_b/N_o and in Figure 3.23 in terms of the SNR.

If we continue to compare these with Figures 3.24 and 3.25, they are basically the same as the ones previuosly mentioned, but the curves in the latter are all shifted upwards due to the presence of phase quantization noise of the PhADC and hence the range of allowable IQ gain mismatch is smaller. And since we want to implement a 5 bit PhADC in the Simulink model, we go directly to the last two Figures to interpret the simulation data there. It has no added value to treat both 3.22 and 3.23 here. The transceiver model with the 5 bit PhADC implementation gives a more realistic picture of the real Phase-Domain ADC receiver [10] implementation.

Using the ideal PhADC, i.e., with no phase quantization, is just a tool to show the effect of only IQ gain mismatch and channel noise on the BER without phase quantization. Hence, we can see how much the BER has increased when we add phase quantization to the model.

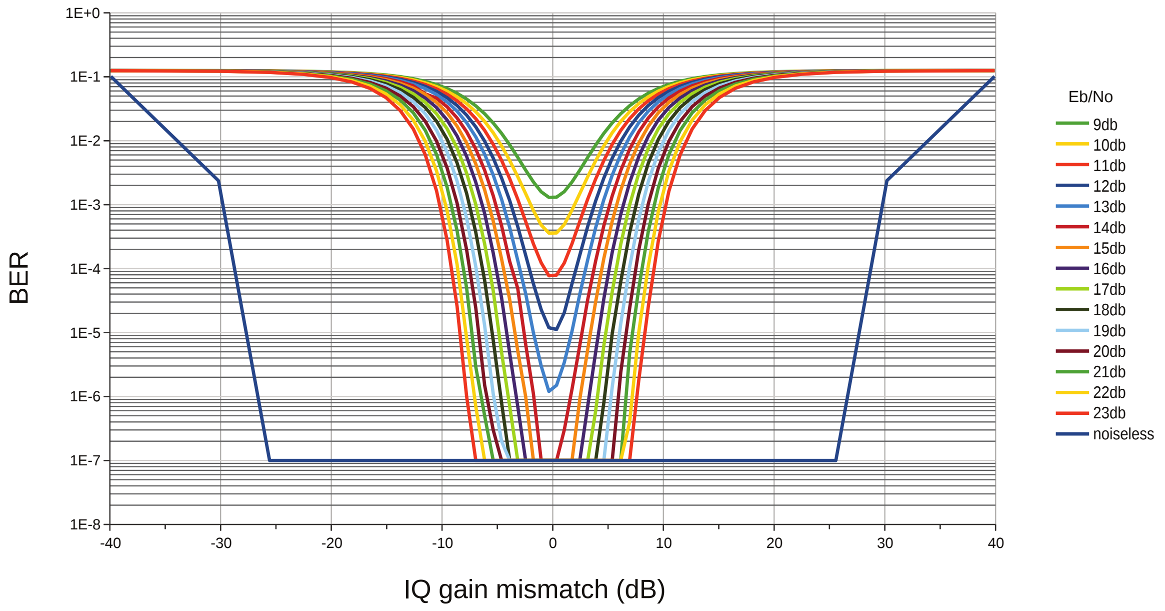


Figure 3.22: Plot of BER as a function of IQ gain mismatch and channel noise E_b/N_o of the Simulink transceiver system with matched SRRC filters and Ideal PhADC using $\pi/4$ DQPSK communication parameters.

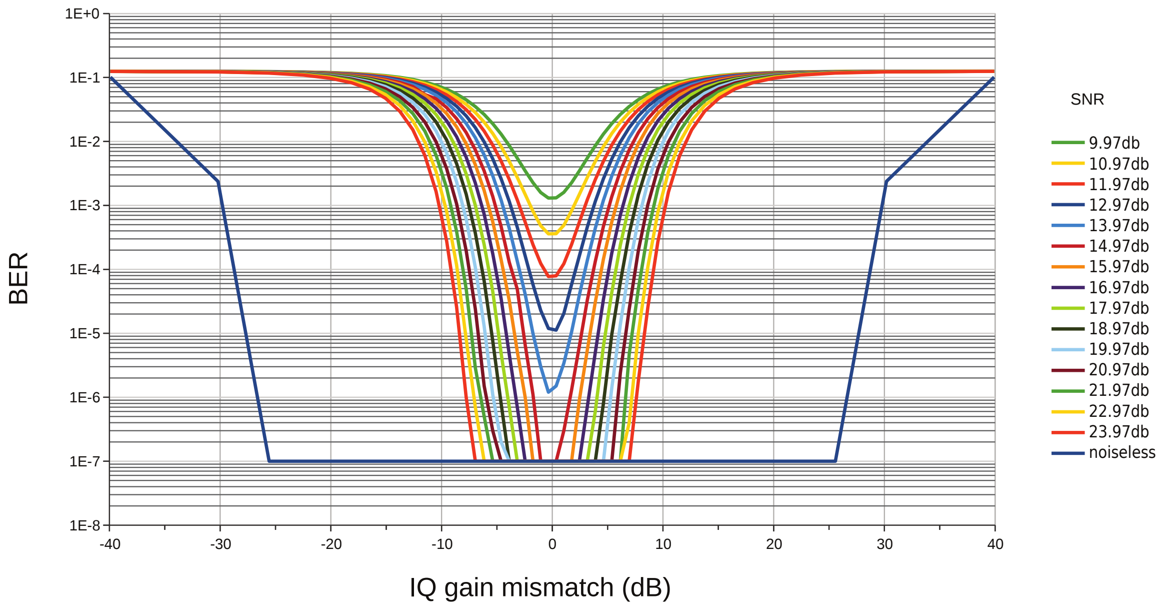


Figure 3.23: Plot of BER as a function of IQ gain mismatch and channel noise SNR of the Simulink transceiver system with matched SRRC filters and Ideal PhADC using $\pi/4$ DQPSK communication parameters.

BER as function of only IQ gain mismatch, channel noise and phase quantization noise

The analysis in Subsection 2.4.3 of the IQ offset as a function of channel and quantization noise also applies to the IQ gain mismatch and therefore we go directly to the determination of the maximum permissible IQ gain mismatch for a maximum allowable BER of 10^{-5} for $\pi/4$ DQPSK modulation, according to the IEEE 802.15.6 standard. For this we use Figure 3.25. An important note is that when we compare Figures 3.24 and 3.25 with Figures 3.22 and 3.23 then we see that the allowable range of IQ gain mismatch has become even smaller. This is due to the addition of phase quantization noise.

The preceding SNR (SNR_{prec}), i.e., the SNR at input of PhADC without considering the noise of the PhADC, is 21 dB (-6.5 dBm + 27.5 dBm). See Figure 2.35(a). This is represented by the brown curve (ie, SNR 20.97 dB) in Figure 3.25 and if we determine the x-axis value at the point on the curve for which the BER is 10^{-5}, we arrive at a maximum permissible IQ gain mismatch of about ±5 dB.

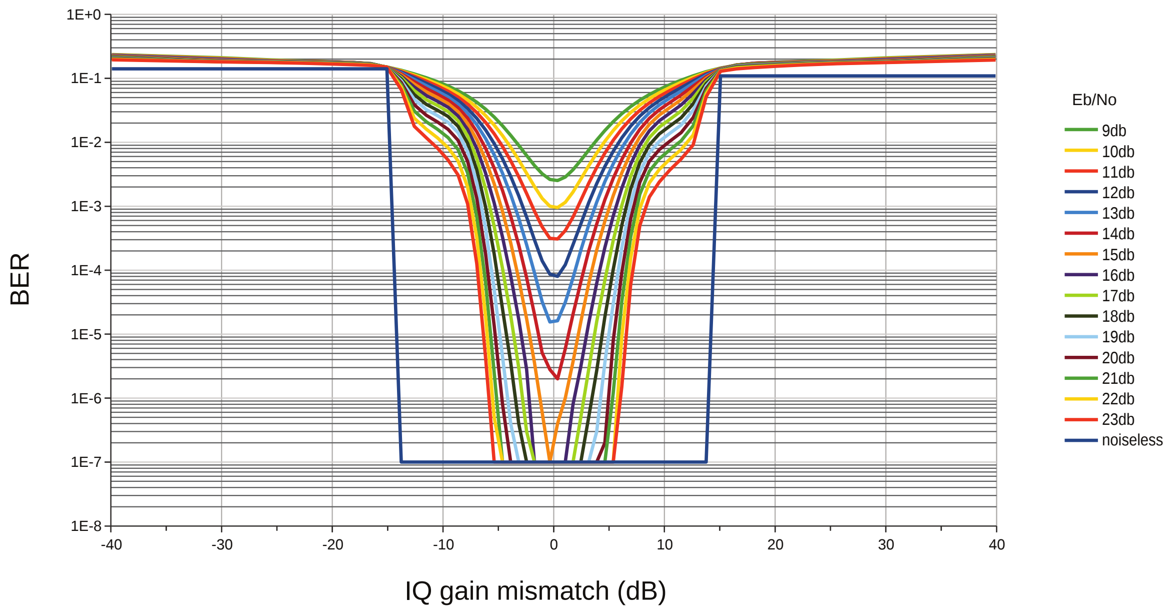


Figure 3.24: Plot of BER as a function of IQ gain mismatch and channel noise E_b/N_o including quantization noise E_b/QN_o of the Simulink transceiver system with matched SRRC filters and 5 bit ideal PhADC using $\pi/4$ DQPSK communication parameters.

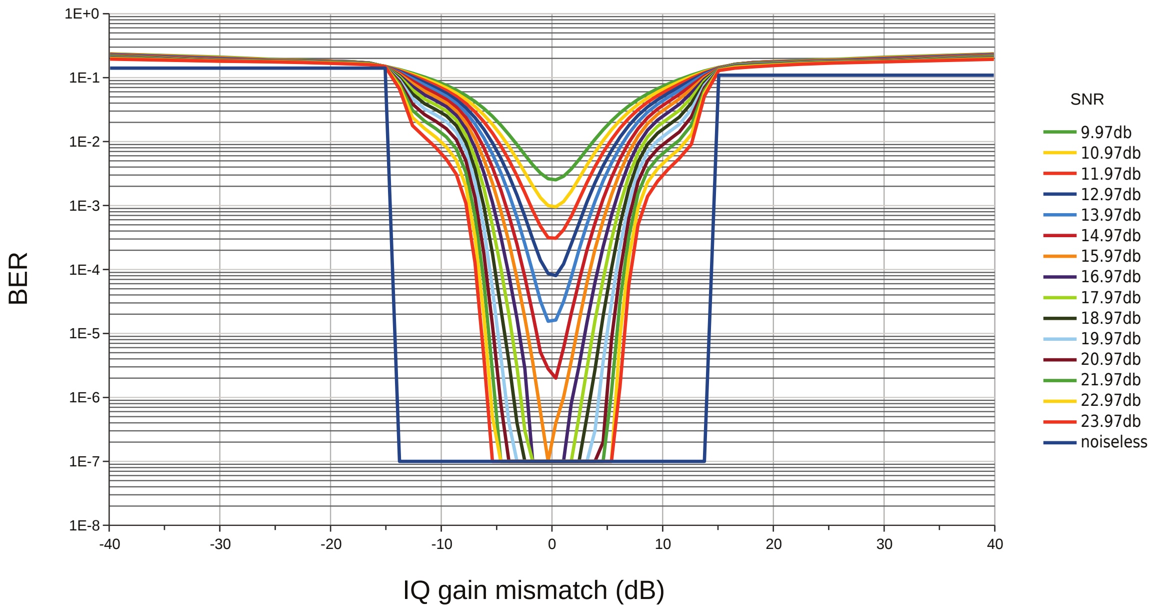


Figure 3.25: Plot of BER as a function of IQ gain mismatch and channel noise SNR including quantization noise SQNR of the Simulink transceiver system with matched SRRC filters and 5 bit ideal PhADC using $\pi/4$ DQPSK communication parameters.

3.3.3. Bit Error Rate (BER) measurements as a function of both IQ gain mismatch and noise for a $\pi/4$ DQPSK transceiver system with matched SRRC-biquad filter

These are the simulation results for a matched SRRC-Biquad filter transceiver system. In other words, on the receiver side the SRRC filter has been replaced by the Biquad filter from Subsection 2.4.4.

Figures 3.26 and 3.27 show the results for BER as a function of only IQ gain mismatch and channel noise. Phase quantization is not modeled here. If we compare this with Figures 3.22 and 3.23, in which the transceiver system has a matched SRRC-SRRC filter, then we see that by using the biquad filter in the receiver chain, the curves in both graphs have raised up somewhat and the allowable range of IQ gain mismatch has even become much smaller. This is due to the fact that the SRRC transmit filter is not the same as the biquad filter, hence, does not posses the complete zero-ISI property. The bandwidth of the biquad filter is always sligtly more than the bandwidth of the SRRC filter and this means that more noise is added, hence, resulting in a higher BER.

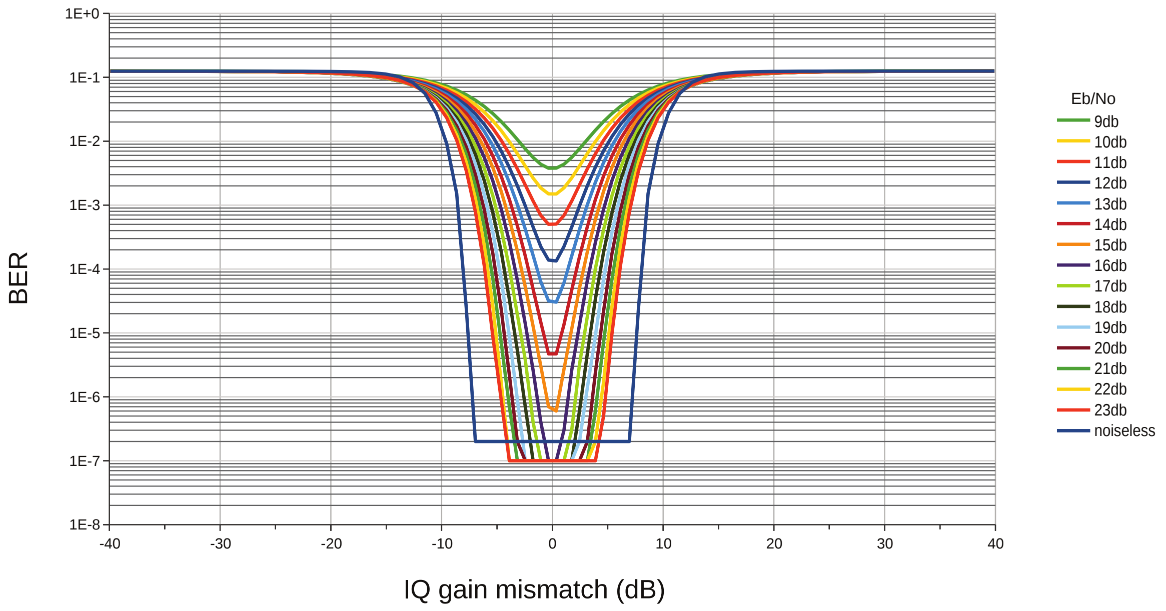


Figure 3.26: Plot of BER as a function of IQ gain mismatch and channel noise E_b/N_o of the Simulink transceiver system with matched SRRC-Receive biquad filtering and ideal PhADC using $\pi/4$ DQPSK modulation parameters.

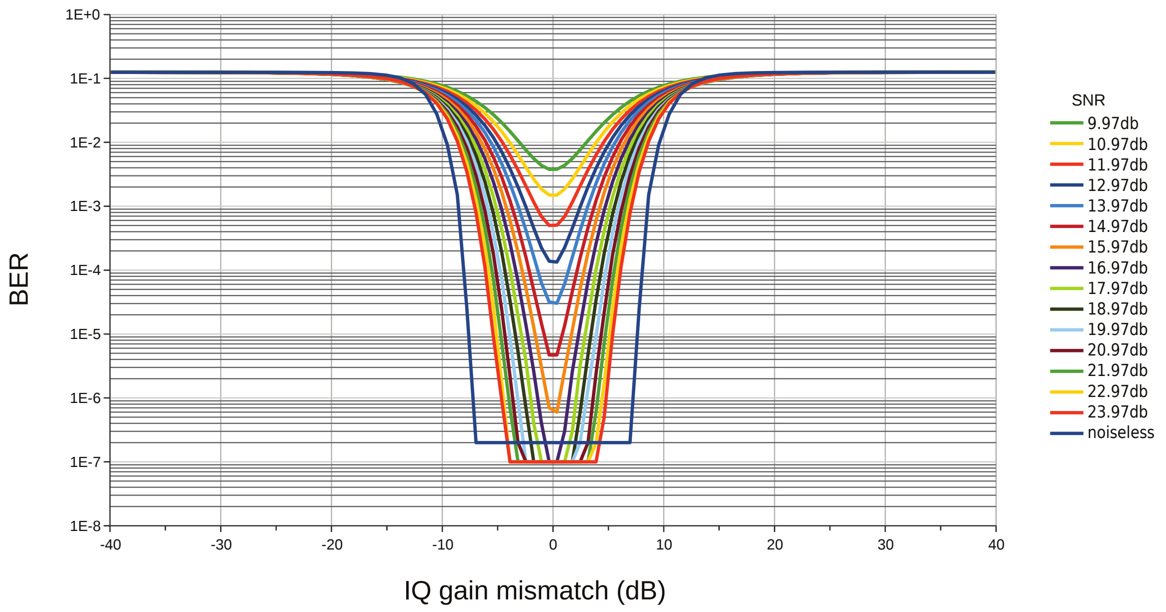


Figure 3.27: Plot of BER as a function of IQ gain mismatch and channel noise SNR of the Simulink transceiver system with matched SRRC-Receive biquad filtering and ideal PhADC using $\pi/4$ DQPSK modulation parameters.

To complete the model, the Figures 3.28 and 3.29 show the BER simulation results in which not only IQ gain mismatch and channel noise but also phase quantization noise is added. The curves have clearly raised up even more and the allowable range of IQ gain mismatch is further narrowed.
For determining the ultimate maximum allowable IQ gain mismatch for a BER of 10^{-5} using $\pi/4$ DQPSK modulation we use therefore Figure 3.29.

When we compare this figure with Figure 3.25 we see that due to the biquad receive filter the brown curve (i.e, curve at SNR 20.97 dB) runs narrower towards the origin. So we can conclude that for a required BER of 10^{-5} for $\pi/4$ DQPSK modulation, we can tolerate a maximum IQ gain mismatch that is between ±3 en ±4 dB.

For an even more reliable IQ gain mismatch compensation and convenience we choose a lowest value of ±3 dB.

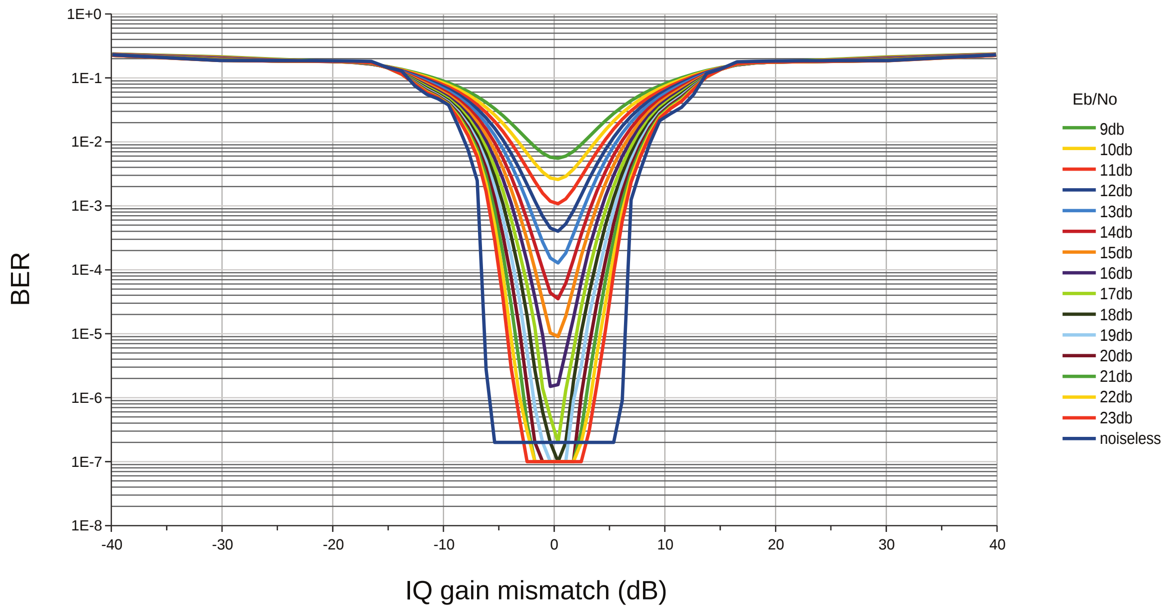


Figure 3.28: Plot of BER as a function of IQ gain mismatch and channel noise E_b/N_o including quantization noise E_b/QN_o of the Simulink transceiver system with matched SRRC-Receive biquad filtering and 5 bit ideal PhADC using $\pi/4$ DQPSK modulation parameters.

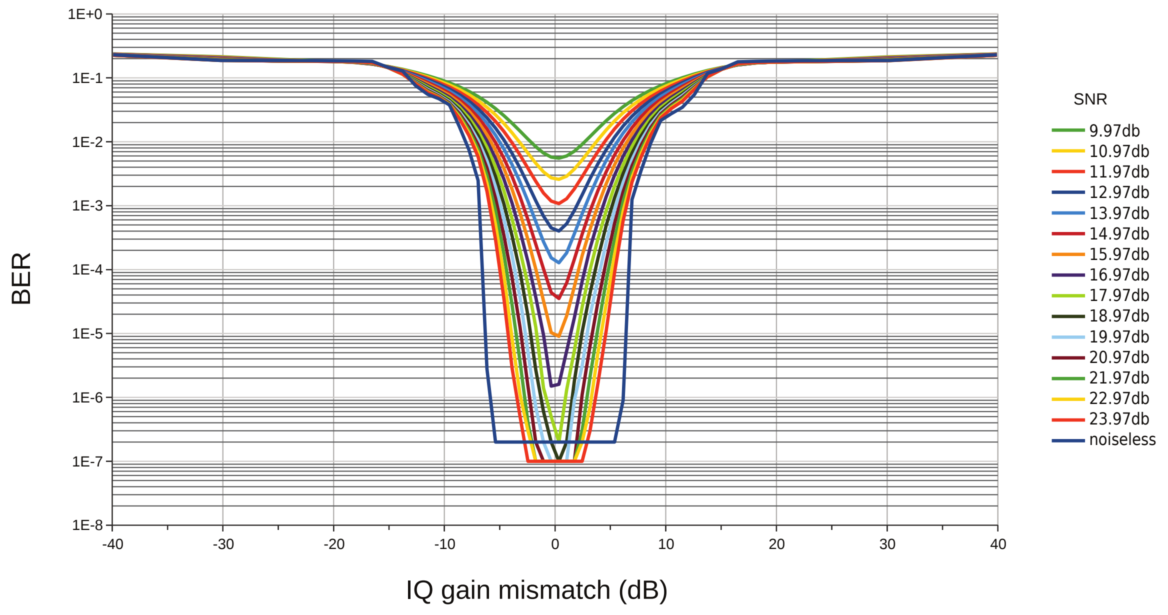
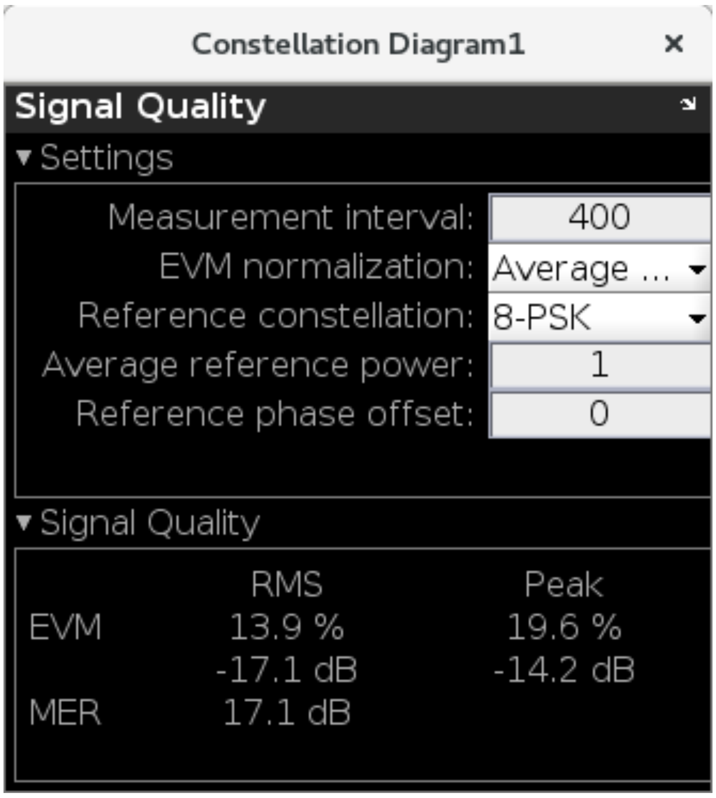


Figure 3.29: Plot of BER as a function of IQ gain mismatch and channel noise SNR including quantization noise E_b/QN_o of the Simulink transceiver system with matched SRRC-Receive biquad filtering and 5 bit ideal PhADC using $\pi/4$ DQPSK modulation parameters.

Signal quality of the received signal of the matched SRRC-Biquad filter model with IQ gain mismatch, channel and quantization noise

The Simulink model with the matched SRRC-Biquad filter, IQ gain mismatch, channel and quantization noise is shown in Figure O.5 in Appendix O Section O.3. This model ultimately contains all "non-idealities", i.e., channel and quantization noise, biqaud filter and ± 3.5 dB gain mismatch.
The simulation results of the signal quality is shown in Figure 3.30 after the system has been compensated and as we can see is still better than the permissible $\pi/4$ DQPSK EVM number shown in Figure 2.29. The result is obtained at;

- channel noise of $E_b/N_o = 20$ dB.
- IQ gain mismatch of ± 3.5 dB.
- quantization noise from the 5 bit PhADC.


Constellation Diagram1

Signal Quality

Settings

Measurement interval:	400
EVM normalization:	Average ...
Reference constellation:	8-PSK
Average reference power:	1
Reference phase offset:	0

Signal Quality

	RMS	Peak
EVM	13.9 %	19.6 %
	-17.1 dB	-14.2 dB
MER	17.1 dB	

Figure 3.30: Signal quality dialog box.

4
System design in transistor and VerilogAMS level

Verilog-A is the counterpart of Verilog-HDL that is used for the hardware behavior description of logic functions. Both are programming languages for the behavior description of hardware. Verilog-AMS has emerged from the combination of both languages. So Verilog-AMS contains both instruction sets. Cadence Design Systems was first to release an implementation of this new language, in a product named AMS Designer that combines their Verilog and Spectre simulation engines. It is a mixed signal simulator that can simulate a combination of analog and digital circuitry [4].

The intent of Verilog-AMS is to let designers of analog and mixed-signal systems and circuits create and use models that describe their designs. Once a design is described in Verilog-AMS, simulators are used to help designers to better understand and verify their designs. Verilog-AMS allows designs to be described at the same level as does SPICE , but at the same time allows designs to also be described at higher more abstract levels. This range is needed for the larger more complex mixed-signal designs that are becoming commonplace today.

So the first step towards a transistor level implementation is to describe the system, where possible, at a higher, more abstract level. It not only provides more insight into a complex system, but it is also good for simulation performance. Because at a transistor level the simulations take much longer and therefore will be expensive. Moreover, it also offers the possibility to synthesize the digital system, described in a so-called RTL logic (register-transfer level description) into a real transistor-level design. Most foundries of CMOS technologies offer Cadence designkits with this possibility and the process is optimized for such implementations. So it is possible to make a first-time-right optimal design.

4.1. Modeling and design of the IQ offset compensation system

Finding a location to control the offset

let's proceed to the system-level description of the IQ offset compensation system. The model is shown in Figure 4.1. First we will take a closer look at the system level of PhADC [10], In particular its front-end because it will give us more insight into the way we are going to implement the control of the offset. As we can see in Figure 4.2, the first stage of the PhADC is a Track-and-Hold circuitry. In addition, the I and Q signals are differentially implemented which makes the implementation of the IQ offset compensation scheme more complex.

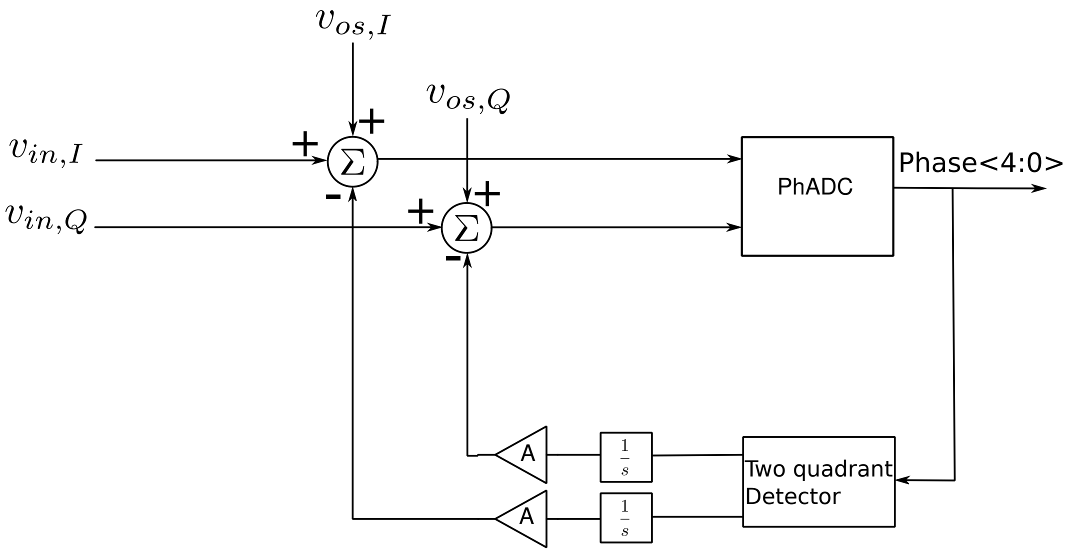


Figure 4.1: System level model IQ offset compensation.

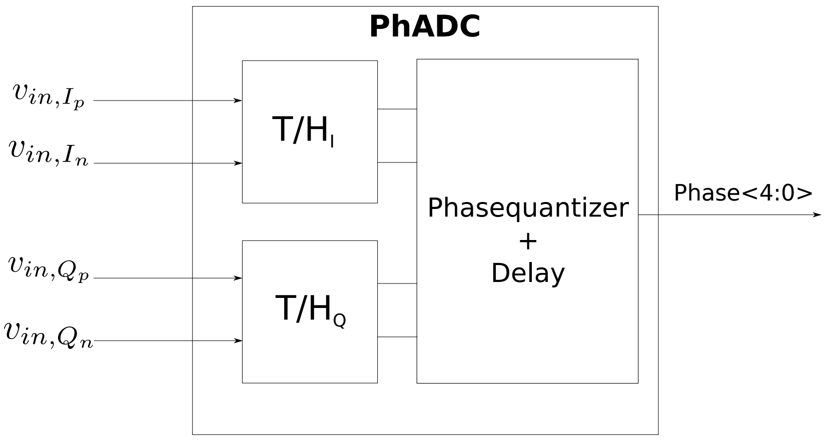


Figure 4.2: System level model of Phase domain ADC with its differential I and Q inputs.

From now on, let's just use the I signal as an example because the implementation for Q is exactly the same. The track-and-hold circuit looks in its simple form as shown in Figure 4.3. We can see the positive half of the I signal fed to the input of the track-and-hold transistor, which is an source-follower, hence a unity gain stage. This offers the possibility to adjust the offset in some way at this stage.
One possibility would be to add an extra unity gain stage at the input of $T\&H$ and adjust the offset voltage at the input of the $T\&H$ transistor as shown in Figure 4.4 by scaling the bias current I_{bias} of this stage through the controlled current source I_{comp}. It is true that the offset scheme can only go one way, namely only downwards. The disadvantage of this method is that an extra stage before the PhADC must be implemented in front of the summing point.

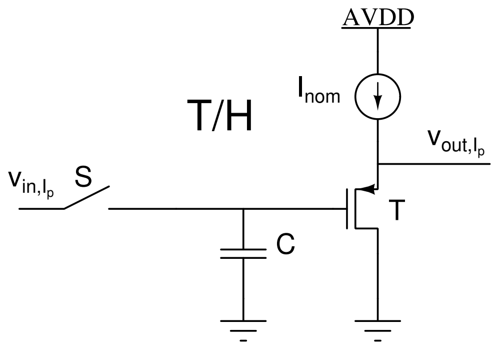


Figure 4.3: First stage of the PhADC: a simplified track-and-hold circuit.

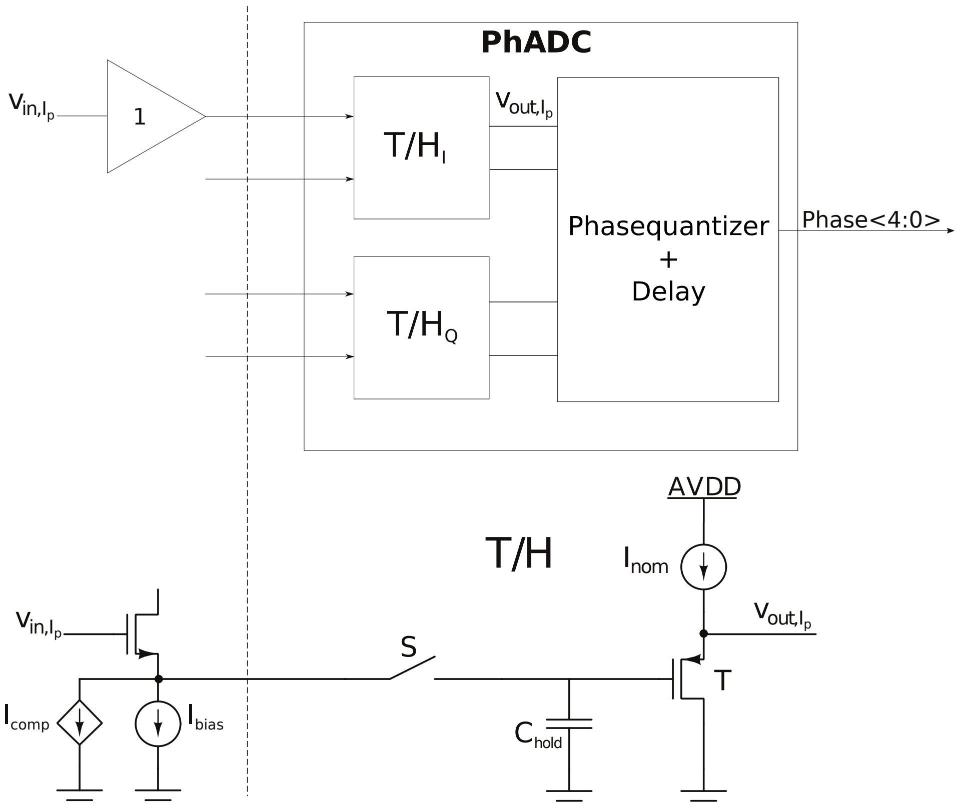


Figure 4.4: Control of offset outside the PhADC by substracting a current I_{comp} from a unity gain stage's I_{bias} current.

But a better solution would be if we control the offset in PhADC itself and therefore in the $T\&H$ transistors. An additional advantage would be that, there is no need for an extra stage and, moreover, the $T\&H$ transistors are already nominally biased and we only need to set the control range with the current-steering DAC current source I_{comp} shown in Figure 4.5. This offset adjustment also only goes one way, namely up; we can only source a current into the $T\&H$ transistor. But the other way is also possible, by sinking the current. Offset control can then only go down.

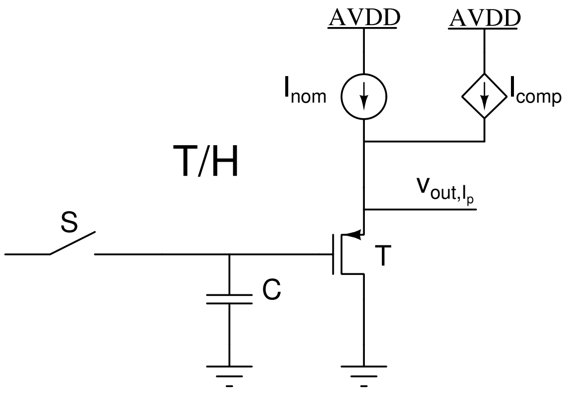


Figure 4.5: Control of offset inside the PhADC by adding a current I_{comp} to the track-and-hold transistor's I_{nom} bias current.

Implementation of the integrator

Figure 4.6 shows a mixed signal solution of a ramp up and down integrator. A digital signal is generated by the two quadrant detector that controls Switches S1 and S2. At a logic level high switch S1 closes, S2 opens, and then the analog integrator, i.e., the capacitor C_{pump}, is charged by current I_1. At logic level low, Switch S2 closes and S1 opens, a current I_2 is sunk from the integrator and thus the capacitor C_{pump} is discharged. By using the charge pump we can generate a ramp up and down signal that is essential for adjusting the offset.

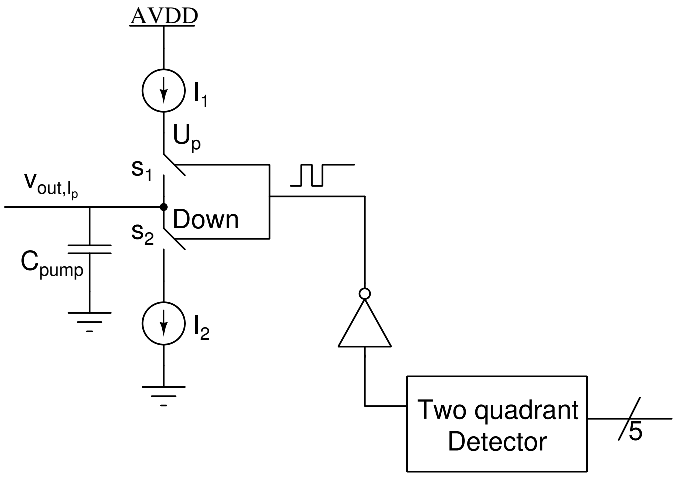


Figure 4.6: Mixed signal implementation of a ramp up and ramp down integrator. Only positive half of I signal is shown.

But the disadvantage of this integrator implementation is charge leakage from the capacitor. Since the purpose of this entire system implementation is ultimately to make it suitable for wireless communication, the question is how much charge, and thus offset change, occurs between the data packets (preamble), where there is no offset regulation. Because there is no data, hence no compensation, available during switching to another channel or between the packets, so the offset can change.
For example, for *Blue-tooth Low Energy*, the time between data packets is defined as the *Inter Framespace* (T_IFS), which is $150\mu s$. See Figure 4.7. If this IQ offset regulation system is continuous in time and also adapted to e.g. Blue-tooth LE, the settling time of this scheme must be faster than this time before the information data is sent. But perhaps this is not necessary at all to compensate IQ offset between the data frames. In communication, there's always training data sent that determines the synchronization and only then the header data is recognized. So a lot of data bits precede the actual communication. In beginning, the data is never free of errors, which is really the case in a practical communication system. And the data frames contain a lot more than besides only information.

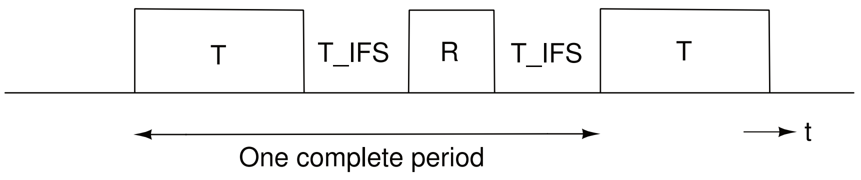


Figure 4.7: Bluetooth Low-Energy connection, one complete transmission period from one device to its peer.

After we have now dealt with all parts of the system, we can merge them and build the system shown in Figure 4.8. As mentioned earlier, the Q signal path has been deliberately omitted from the figure, but, again, it is exactly the same as the I signal path and will therefore not provide any additional information and clarity to the explanation.
It is important to see that we need a G_m stage in the feedback path of the system to convert the integrator voltage into a current, because we do not set a voltage but inject current into the T&H transistor.

An additional disadvantage of this system is that because of the analog implementation of the integrator it does not really offer flexibility compared to a digital one. We can not store the feedback offset value because we do not have the offset digitally available. So there's a good reason to seek for a digital implementation of the integrator.

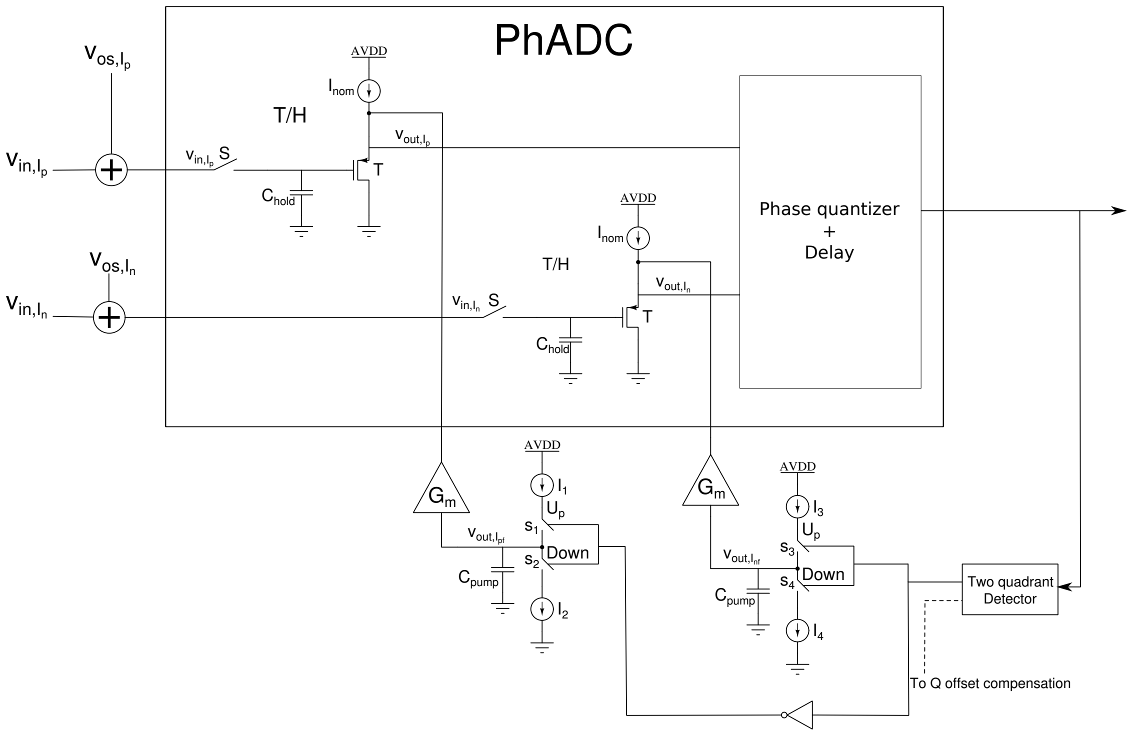


Figure 4.8: IQ offset compensation system using analog integrators and G_m stages. Only I offset compensation is shown. For Q offset compensation, exactly the same circuit applies.

Fully digital implementation of the integrator

The ramp up and ramp down integrator can be implemented in a digital circuit. The charge-pump is simply replaced by an Up and Down counter. Basically, an analog integrator is the analog counterpart of an Up and Down counter in the digital domain.
But an additional problem for this implementation is that the output of this integrator is digital and must therefore first be converted to an analogue current before it can be injected into the $T\&H$ transistors. An obvious solution is therefore a current-steering DAC between the $T\&H$ and Up/Down counter.

Figure 4.9 shows an implementation with the current-steering DAC and Up/Down counter. This system offers more flexibility than the previous one because now we can store the offset digitally. So it is also suitable for one-time calibration. We then have to compensate once and then store the offset which can be restored

again in case the communication starts.
IQ offset and gain mismatch is static, as these are caused by mismatches in the devices. So we consider there's enough time to do a one-time calibration and we consider that IQ offset and mismatch do not change over time. Temperature can also be considered constant. However, there are two conditions that cause offset change:

1. Frequency change due to channel switching.
2. Different gain setting.

The settling time of the compensation system still needs to be fast enough but fortunately in a practical communication there's always payload data before it actually starts to send information so there should be enough time to compensate and store the offset.

Another possibility that the system offers is digitally disabling the control loop. This can be done very easily by adding for instance an enable signal to the Up and Down counter. If it does not count, there is no offset compensation. This is a very useful option which is much more difficult to implement in the analog domain and that makes this system more robust.

But in this system we must pay attention to the resolution of the current-steering DAC. A certain step size (1 LSB) of the DAC is required to arrive at a good compensation system. So, how much resolution do we need for the DAC to come to a certain offset compensation? We do not know in advance exactly how large the resolution should be. We can ask ourselves what is the minimum and maximum offset that needs to be eliminated? This will be discussed later in Subsection 4.1.1.

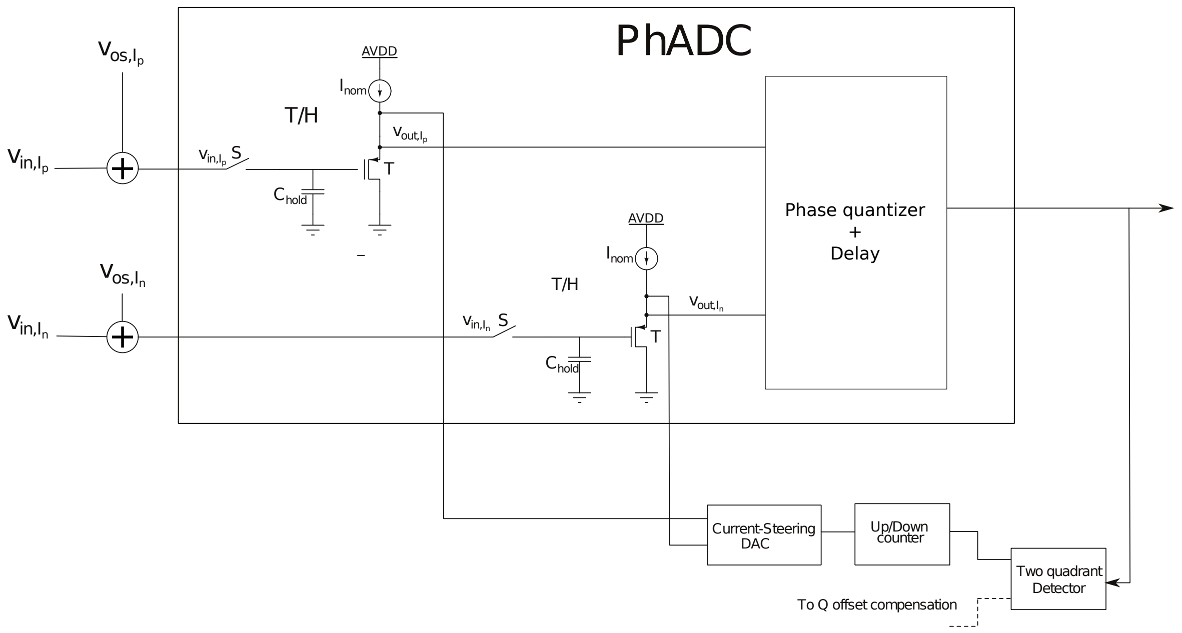


Figure 4.9: IQ offset compensation system using a digital Up and Down counter and a Current-steering DAC. Only I offset compensation is shown. For Q offset compensation, exactly the same circuit applies.

Because the I and Q signals are differential, the current-steering DAC must therefore also control two signals instead of one. And this requires a well-considered implementation of the current control. Next, we will go deeper into this.

Implementation of the Current-Steering DAC

The left side of the Table 4.1 shows an example of a 8 bit Up and Down counter output codes and the right side of the table the same code mapped to a control code for the current-steering DAC.
Offset binary coding, is a digital coding scheme where 10000000 corresponds to the midrange and 11111111 is the maximal value of the compensating current source I_{comp} on the negative side and 01111111 is the maximal value of the compensating current source I_{comp} on the positive side, see the blue and red area shown in the Table. The MSB bit is used as a selection bit to control Switches S1 and S2 shown in Figure 4.10. If MSB=1 then S2 is closed, S1 open and if MSB=0 then S1 closes, S2 opens. If the counter output is in the mid-range then no compensating current will flow on both sides and the source-followers are only biased by their nominal current source I_{nom}.

Relation between Up/Down counter output code — control code for current-steering DAC				
Counter offset binary code N = 8	Decimal value $B_6....B_0$	CS-DAC control code		
11111111	127	11111111	Maximum value	This code range is used to control for positive offset
11111110	126	11111110		
11111101	125	11111101		
....				
....				
....				
10000001	1	10000001	Minimum value	
10000000	0	10000000	Midrange	
01111111	127	01111111	Maximum value	This code range is used to control for negative offset
01111110	126	01111110		
01111101	125	01111101		
....				
....				
....				
00000010	2	00000010		
00000001	1	00000001		
00000000	0	00000000	Minimum value	

Table 4.1: Relation between 8 bit Up/Down counter output code and control code for the current-steering DAC.

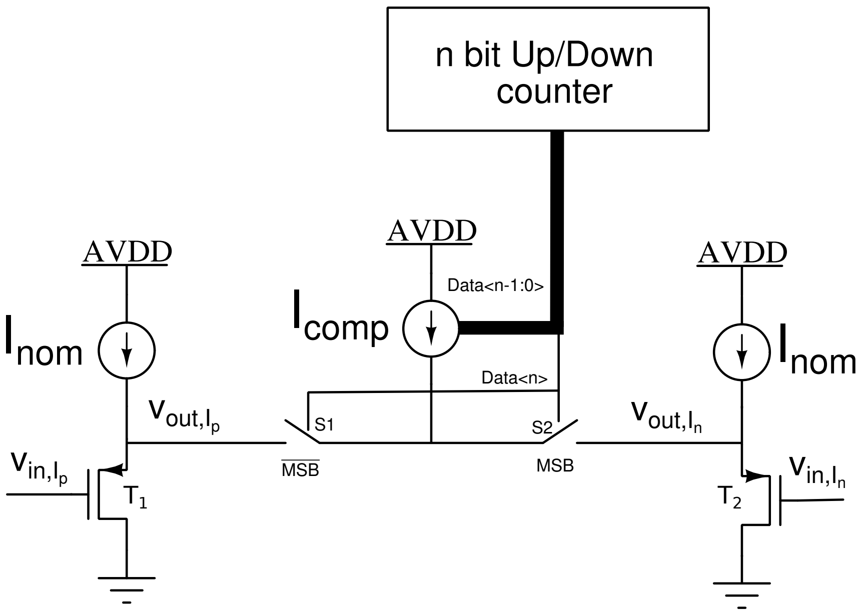


Figure 4.10: Control of the compensating current source I_{comp} and injecting it in the left or right branch using switches S_1 and S_2.

In the initial state of the circuit shown in Figure 4.10, where the Up and Down counter output is set to 10000000, no compensating current I_{comp} will flow to either side. Both source-follower transistors T_1 and T_2 are only biased by their nominal current sources I_{nom}. If both v_{in,I_p} and V_{in,I_n} are equal then no offset occurs at this counter code output . When there's positive I offset, this means that the number of symbols in the right half of IQ complex plane is higher than in the left half and since, the counter will count up for a symbol in the right half and count down for a one in left half, the counter output should show a value above the mid-range (10000000). In short, there's a positive vector distribution imbalance between the right and left half planes. The output voltage v_{out,I_p} at the source of source-follower transistor T_1 is then higher than the voltage v_{out,I_n} at the source of T_2. An obvious solution for offset compensation is to let the voltage v_{out,I_p} untouched and increase the v_{out,I_n} to the level of v_{out,I_p}. When these become equal or more precisely, when the level shifts of both source followers become equal, the system is compensated. So we use the Up and Down counter whose output is above the mid-range for controlling the I_{comp} compensating current source. Thus, in case MSB=1, switch S2 is closed and S1 is opened to let the Up and Down counter control the I_{comp} current through transistor T_2. Figure 4.11 illustrates the compensation.

v_{out,I_p}
v_{out,I_n}

Figure 4.11: Positive I offset compensation. v_{out,I_n} is raised to the level of v_{out,I_p}.

When there's a negative I offset, this means that there are more symbols in the left half of the IQ complex plane. The number of symbols in the left half is higher than in the right half and since the Up-Down counter will count up for a symbol in the right half plane and count down for a one in left half plane the counter output should show a value below the mid-range (10000000). In short, there is a negative vector distribution imbalance between the right and left half IQ complex planes. The output voltage v_{out,I_n} at the source of source-follower transistor T_2 is then higher than the voltage $v_{out,Ip}$ at the source of T_1. An obvious solution for offset compensation is to let the voltage $v_{out,In}$ untouched and increase $v_{out,Ip}$ to the level of $v_{out,In}$. When these are equal the system is compensated. So we use the Up and Down counter whose output is below the mid-range for controlling the I_{comp} compensating current source. Thus, in case MSB=0, switch S1 is closed and S2 is opened to let the Up and Down counter control the I_{comp} current through transistor T_1. Figure 4.12 illustrates the compensation.

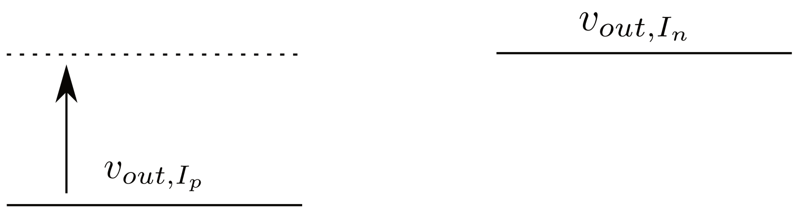


Figure 4.12: Negative I offset compensation. v_{out,I_p} is raised to the level of v_{out,I_n}

This control scheme clearly has its advantages:

1. We don't need an extra block between the Up-Down counter and the compensating current source I_{comp} to do an additional code mapping because we can use the counter output directly to control the compensating current source I_{comp}. The current control implementation is just simply the UP/DOWN counter that counts the pulses from *two quadrant detector*.

2. We don't need two separate control current sources I_{comp} to adjust each branch separately, hence, only one control data-bus is needed from the Up and Down counter. So the circuit becomes smaller and consumes less energy.

3. The only extra addition are two switches S1 and S2 for the MSB bit. And probably some inverters if we are going to use pMOSFETs for implementing the compensating current source I_{comp}.

Implementation of Up and Down counter

The counter should not count upwards when it is in position 11111111 and not count down when it is in 00000000. We would otherwise be able to get a transition from 11111111 → 00000000 and 00000000 → 11111111 which is not the intention. If the counter output has reached one of these positions means that it has reached its maximum or minimum of the control range.

4.1.1. Verilog Analog Modeling

Testbench setup feedforward path of the IQ compensation system

To simulate the circuit at system level and to avoid long simulation times, we have replaced the transistor-level circuit design of the PhADC [10] with a VerilogAMS PhADC model. They do exactly the same thing, namely extracting phase from I and Q, but the VerilogAMS model simulates much faster. The only thing that is retained from the real PhADC [10] is the track-and-hold circuit because this is our summation point for the control of IQ offset, so we took it outside the PhADC. The track-and-hold together with the VerilogAMS PhADC model then performs the same function as the real PhADC. The feed-forward path of the IQ compensation model is shown in Figure 4.13.

The transformer used in the system is called a *balun*, an acronym for *balanced-to-unbalanced* conversion because it can also perform differential to single ended conversion if its two ports are swapped. the I and Q signals come in as single-ended and are converted to differential for $T\&H$ inputs and then back to single-ended for the PhADC input.

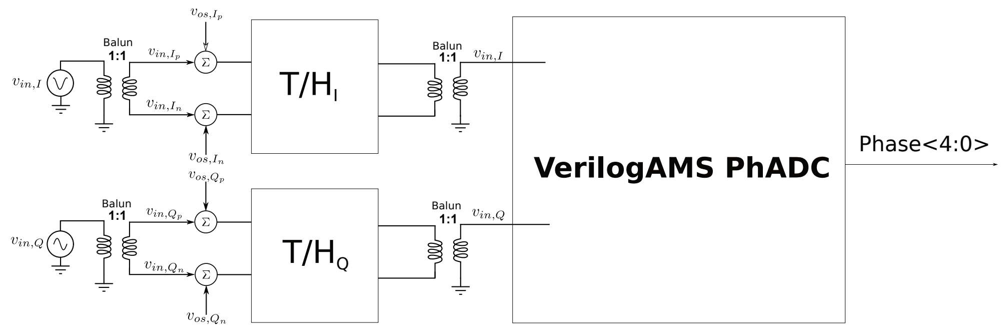


Figure 4.13: Track-and-Hold circuits have been taken out the PhADC and the PhADC itself has been replaced by a VerilogAMS PhADC model.

VerilogAMS modeling of Phase-Domain ADC

Figure N.1 in Appendix N Section N.1 shows the VerilogAMS code for the PhADC implementation. The ADC resolution is defined (i.e., *'define bits 5*) as 5 bit and operates at a supply voltage (VDD) of 1.2 V but the full scale range (fullscale) of ADC is set to 1 V. The threshold voltage (thresh) for the clock source is set to 0.6 V. And furthermore we can also set the transition time (*tt*) of the digital output signals. Here it is set to 1ns. The PhADC has no delay time (*td*). In the analog behavior description (i.e., section *analog begin ... end*) we can see that an $atan$ function is been performed to extract the phase from the IQ signals. The same coding scheme as shown in Figure 1.8 has been used to map the real phase to a digitally coded phase. This model has been programmed, tested and verified.

Design of binary-weighted current-steering DAC

The amount of bits for the Up-Down counter and the current-steering DAC is determined by the minimum offset that remains after the compensation and that is still small enough that no bit error occurs. And in addition meets the IEEE 802.15.6 standard requirement having a maximum BER of 10^{-5} at an E_b/N_o of 12 dB for $\pi/4$ DQPSK demodulation. The range of current-steering DAC is determined by the maximum offset that we want to control.

The offset at the input of the PhADC [10] is determined by the receiver portion that is connected to it, namely, the passive front-end (PFE), baseband low-noise amplifier(BB LNA) and the programmable gain amplifier (PGA). Assuming the DC offset generated by the mixer, which is part of the PFE, is strongly supressed by the bandpass BB LNA, the DC offset of the BB LNA and analog BB has a simulated standard deviation σ_{PFE+BB} of 3.4 mV for the minimum gain and 23 mV for the maximum gain [10]. If we choose a $3\sigma_{PFE+BB}$ in which almost the total spread relative to the average offset outcome in the simulation is covered, then we have a worstcase offset of at most 69mV [10].

There is also the offset of PhADC itself. It has a simulated standard deviation σ_{ph} of 1.44 mV and a $3\sigma_{ph}$ of 4.32 mV.

We should not just sum these two values but take the root square of the sum of the squared noise voltages otherwise we get an overestimation of the maximum occurring offset. Then we will have a value of:

$$\sqrt{(3\sigma_{PFE+BB})^2 + (3\sigma_p h)^2} = \sqrt{69^2 + 4.32^2} \approx 69.1mV$$

The maximum allowable IQ offset was determined at the end of Subsection 2.4.4 and is 10 mV.

When the IQ offset is compensated, all that remains is a residue, lower or equal to the maximum allowable offset at the source of T&H transistor $\Delta V_{S,TH_{max}}$, hence the name $\Delta V_{S,TH_{max}}$. We also found that this offset is 10 mV and determined by the required BER of 10^{-5} for $\pi/4$ DQPSK demodulation. One step within that 10 mV range is equal to 1 LSB. This is illustrated in Figure 4.14. An equivalent current that generates such an offset step is the LSB current of the current-steering DAC.

The relationship between the error band ϵ_{band} and the number of PhADC samples is defined as

$$\frac{\epsilon_{band}}{LSB} = number\ of\ PhADC\ samples\ per\ 1/2\ cycle \tag{4.1}$$

In the example of Figure 4.14 one cylce is considered to be 4 control steps up and 4 control steps down. In case of linear input phase, this error band must not exceed the maximum permissible IQ offset of 10 mV. For $\pi/4$ DQPSK modulation, the following rule must be met

$$\frac{f_{rot}}{f_{sym}} \geq \frac{1}{8} = 1/(4\ LSB\ per\ \frac{1}{2}\ cycle) \tag{4.2}$$

where,
f_{rot} = rotation frequency, is equal to the reciprocal of the time the phase information completes one cylce. Equal to the information frequency.
f_{sym} = the symbol rate, is equal to the reciprocal of the time between each two consecutive symbols.

This means that the sampling rate of the IQ signal must not exceed 8x the signal frequency so that the offset voltage variation at the source of the track&hold transistor remains within that error band.

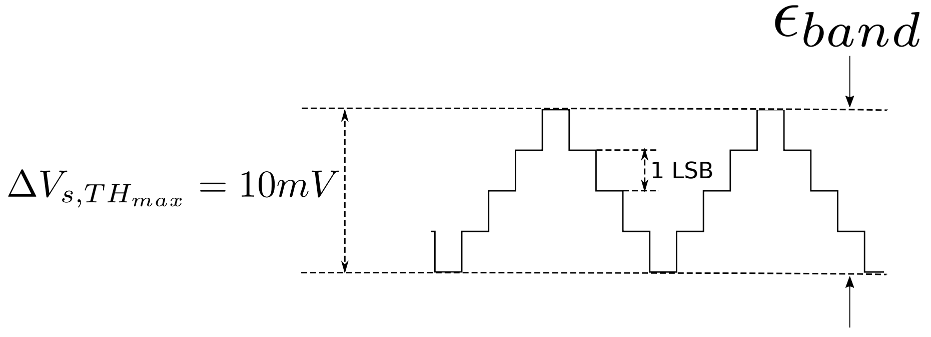


Figure 4.14: 1 LSB is determined by the number of steps that fit in the range of the maximum allowable IQ offset $\Delta V_{S,TH_{max}}$.

Using the voltage-current relation of the PMOS fet T&H transistor, the relation between I_{LSB} and $\Delta V_{S,TH_{max}}$ is defined as

$$4.I_{LSB}.\frac{1}{g_{m,TH}} \leq \Delta V_{S,TH_{max}} \tag{4.3}$$

where $g_{m,TH}$ is the small signal transconductance of the track-and-hold pMOSFET transistor. The 1 LSB current is relatively small compared to I_{nom} and so $g_{m,TH}$ has to be determined around I_{nom}. Figure 4.15 shows a parametric sweep analysis of the track-and-hold transistor source voltage $V_{S,TH}$ and its transconductance g_m as a function of the compensation current I_{comp}. The voltage at the source is nominally, $V_{S,TH_{nom}} = 659\ mV$, and here we see that $g_m = 8.45\mu S$. So, for a maximum allowable offset of $\Delta V_{S,TH_{max}}$=10mV and using (4.3), I_{LSB} should not be greater then 21 nA.

Hence, $I_{LSB} \leq 21nA$

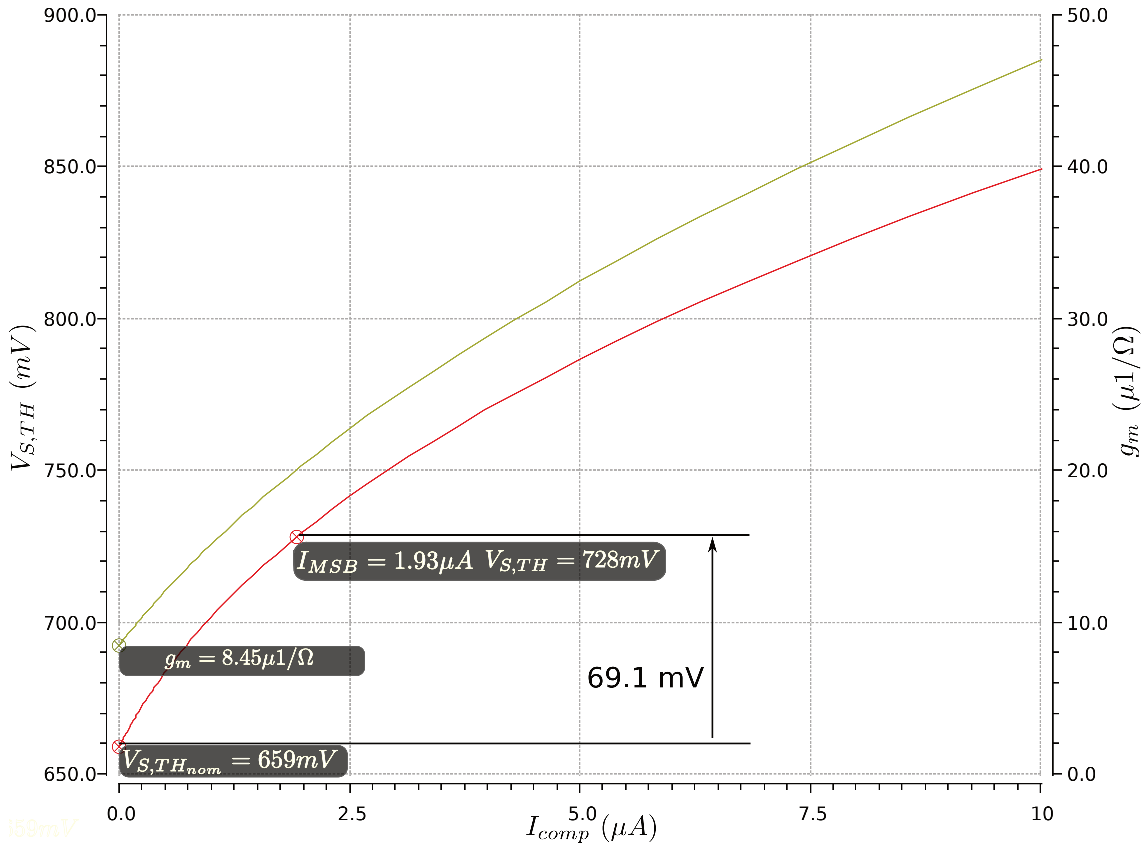


Figure 4.15: Track-and-hold pMOSFET source voltage $V_{S,TH}$ and its transconductance g_m versus compensation current I_{comp}.

The maximum occurring offset voltage is approximately 69.1 mV and if we add this to the nominal voltage $V_{S,TH} = 659mV$ we get 728 mV. The compensation current at this voltage equals I_{MSB} which is $1.93\mu A$.

The resolution of the current-steering DAC can be found from following requirement

$$2^{N-1} \geq \frac{I_{MSB}}{I_{LSB}} \tag{4.4}$$

That means we need a N-1=7 bits current-steering DAC, but we also need an additional MSB bit to control the MSB switches. Thus a total of N=8 bits. Figure 4.16 shows the implementation of a binary-weighted current-steering DAC. The Up-Down counter is a 8 bit, of which 7 bits are used for the current-steering DAC and the MSB bit for the MSB switches.

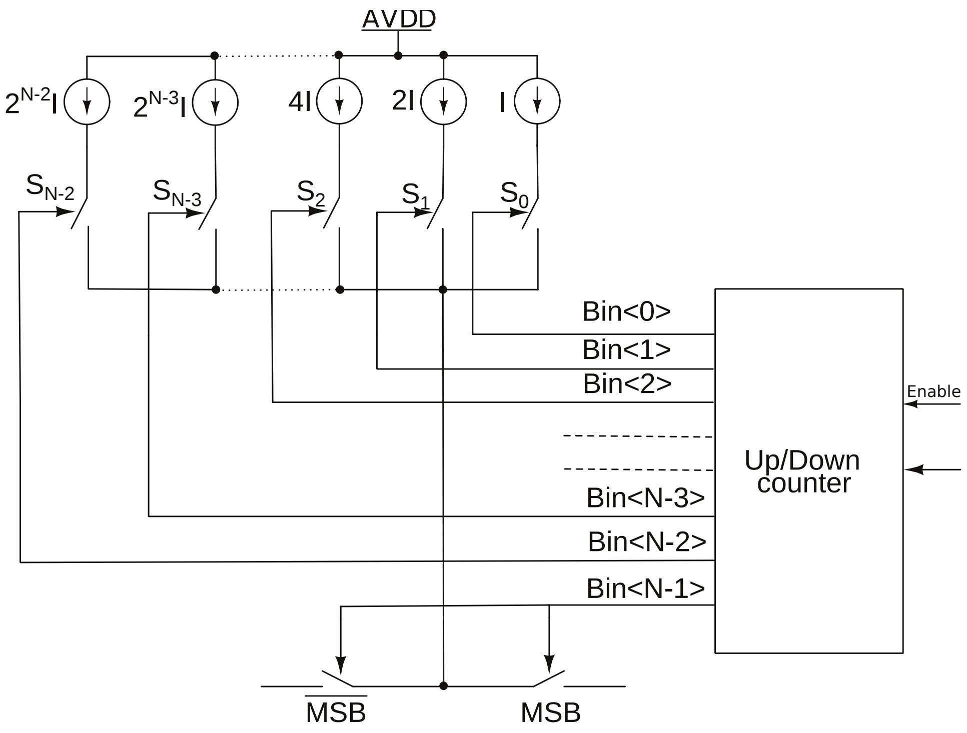


Figure 4.16: Binary coded current-steering DAC controlled by the Up/Down counter.

The problem with a binary weighted design is that a large glitch can occur at the output when several current sources are being switched at once. The Glitches are mainly the result of different delays occurring when switching several signals. For example, when the digital input code changes from 0111...1 to 100...0, all of the N-1 LSBs turn off and the MSB turns on. However, it is possible that the currents due to the LSB switches will turn off slightly before the MSB current, causing the current to temporarily fall to zero. Alternatively, if the LSB switches turn off slightly after the MSB current, the current will temporarily go to its maximum value. In either case , a glitch occurs at the output unless these delays are exactly matched, which is highly unlikely since branches have different currents. This glitch phenomenon is illustrated in Figure 4.17.
So, monotonicity is not guaranteed. The binary weighted current steering DAC suffers from large differential nonlinearity, and from the presence of glitches that degrade its dynamic performance [23]. Monotonicity is the ability of the output signal to change in the same direction, or to preserve state for every increase/decrease in the input signal. A monotonic DAC has an analog output that never decreases with an increasing decimal value corresponding to the digital input code, and conversely.

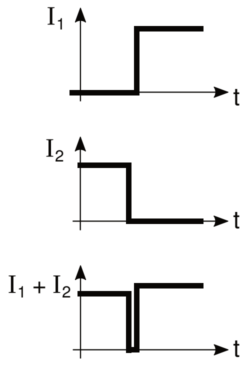


Figure 4.17: Glitches. I_1 represents the MSB current and I_2 the sum of N-1 LSB currents. Here, the LSB currents turns off slightly early, causing a glitch of zero current.

Thermometer coding is preferred over binary because only one bit switches at a time, eliminating the MSB current source glitch. The difference between binary- and thermometer code can be seen in Table 4.2.

The advantages of thermometer-coded current-steering DACs are guaranteed monotonicity, good differential nonlinearity, and very low glitches, i.e., small dynamic switching errors. However, they suffer from complexity, large area, and power consumption, due to the thermometer decoder circuit. But the last mentioned three disadvantages only apply to high-resolution Thermometer-coded DACs and here is therefore not the case.

Decimal	Binary	Thermometer
0	000	0000000
1	001	0000001
2	010	0000011
3	011	0000111
4	100	0001111
5	101	0011111
6	110	0111111
7	111	1111111

Table 4.2: Thermometer-code representation for 3 bit binary values.

Figure 4.18 shows an implementation of the thermometer-coded current -steering DAC. The difference between binary-weighted current-steering DAC from Figure 4.16 is that a binary-to-thermometer encoder is needed between the Up-Down counter and the current sources. And besides, the current sources are not binary scaled and so are all the same.

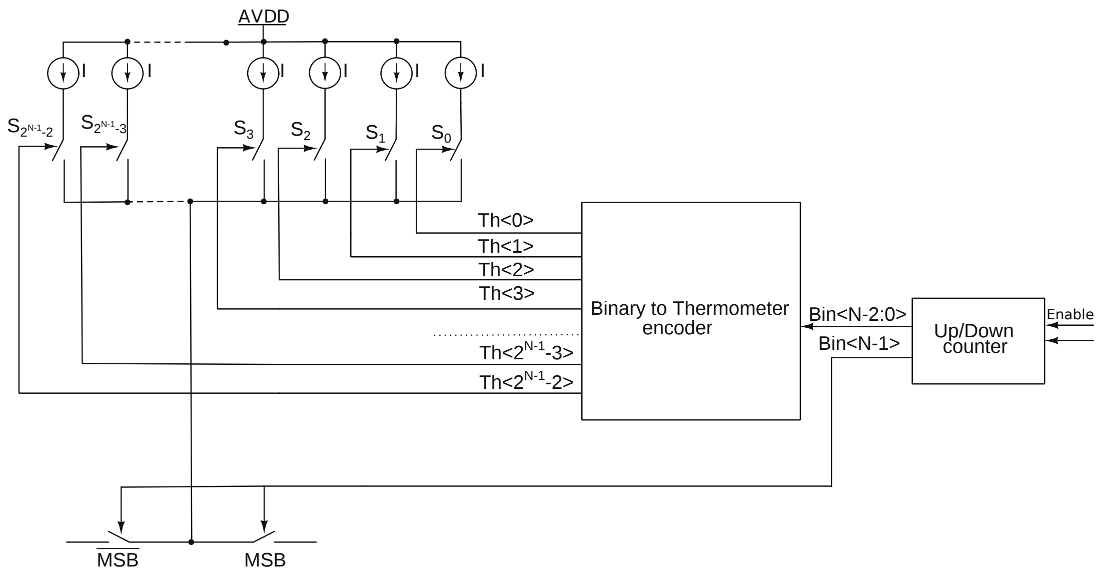


Figure 4.18: Thermometer coded current-steering DAC controlled by Up/Down counter.

Figures N.4 and N.5 in Appendix N Section N.4 shows a VerilogAMS listing of the current sources and the switches. For current source model we can change the DC value, the g_m and possibly we can add an ac component to the source. So we can make this source non-ideal by choosing a finite impedance. This offers an extra freedom factor when examining the compensation system. This also applies to the switch with its R_{on} and R_{off} resistance settings. Also notice that it reassembles a pMOS switch because a logical high on the controlling node s will open the switch and a logical low close it. That means is a low active switch, but we defined a level for current turning on as a logical high. Hence, the $\overline{MSB}$ switch is preceded by an inverter to make it high active. The *Cadence ahdlLib* includes a veriloga coded inverter called not_gate that we can use for the system model implementation.

The listing of binary-to-thermometer encoder is shown in Figure N.6 in Appendix N. It has been tested and verified. To understand the verilogAMS code, and this applies to all presented listings here, I refer to the Verilog-AMS website [9].

The code for the two-quadrant detector is shown in Figure N.2. Again, to understand the functionality of VerilogAMS models you will first have to dig a little bit into how VerilogAMS works and to explain this here in

detail is beyond the scope of this research.
The last model is shown in Figure N.3 of the Up and Down counter. The binary-to-thermometer encoder, two-quadrant-detector and Up/Down counter are all digital blocks and so it is possible to translate the models into a RTL code description and synthesize the code e.g. with Cadence Encounter to a transistor-level implementation. So for analog design part we only have to focus on the current-steering DAC and this considerably shortens the design trajectory.

4.1.2. Simulation

Performance comparison between the Verilog-AMS binary-weighted and thermometer coded current-steering DAC

LSB current I_{LSB} of both 7 bit DACs is set at $21nA$ and the output full-scale would therefore be $(2^7 - 1)I_{LSB} = 2.667\mu A$.
In ideal 7 bit DAC, each adjacent output increment should be exactly one-one hundred and twenty seventh if the output has been normalized to the reference current of $2^7.I_{LSB} = 2.688\mu A$. However, non-ideal components cause the analog increments to differ from their ideal values. The difference between ideal and non-ideal values is known as *differential nonlinearity*, or DNL and is defined as

$$DNL_n = Actual\ increment\ height\ of\ transition\ n\ -\ Ideal\ increment\ height \tag{4.5}$$

where n is the number corresponding to the digital input transition. The DNL specification measures how well a DAC can generate uniform analog LSB multiples at its output.

Figure 4.19 shows the glitch-free input signal to an ideal ADC to generate input codes for both binary and thermo coded DACs. The signal has a full scale range of V_{REF}=1.2 V. One LSB can be defined as

$$1LSB = \frac{V_{REF}}{2^N} = \frac{1.2V}{128} = 9.375\ mV \tag{4.6}$$

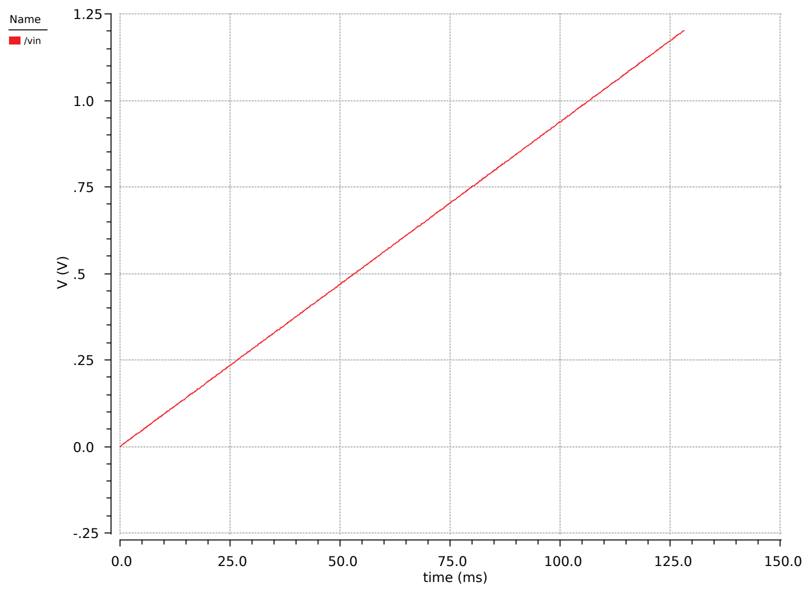


Figure 4.19: The glitch-free input signal to an ideal 7 bit ADC to generate input codes for both binary and thermo coded DACs.

Since the glitches are a time-domain behavior the output currents of both the binary and thermo coded current steering DACs are been presented as a function of time in Figure 4.20 and in 4.20(a) shows the transfer characteristics of the 7 bit binary-weighted current steering DAC and we can clearly see the glitches while in

Figure 4.20(b) for the thermo coded DAC is glitch free.
For the determination of the DNL and INL a static characterization as shown in Figure 4.21 is a more suitable one. Actually the x-axis should show the input digital codes, but since we are verifying in an analog domain, it is difficult to create such a plot. But the interesting thing is that we also see the glitches in Figure 4.21(a) which shows a static behavior.

Maximum Differential Nonlinearity (DNL) measured is 0.072159 at code 63 as shown in Figure 4.22(a). This is of course exactly the moment at which the DAC goes from code 63 to 64, i.e., 011 1111 → 100 0000 where all the LSB bits flip from one to zero. Figure 4.22(a) shows plot of the DNL values (in LSBs) versus the input digital code given in decimal.The DNL for entire converter is +0.013488/-0.072159 LSB since the overall error of the DAC is defined by its worst-case DNL.

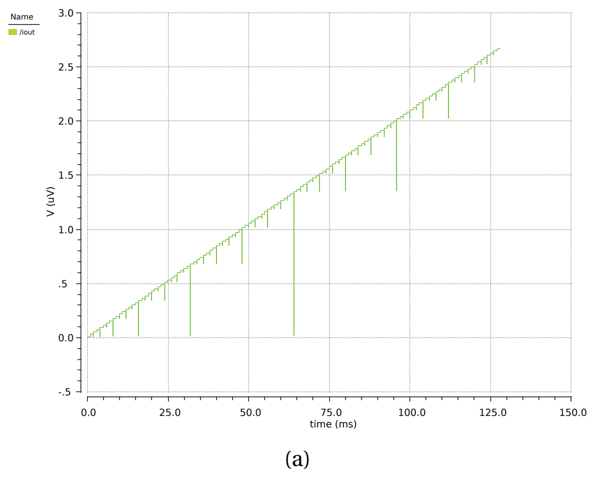


Figure 4.20: Output in the time domain of a 7 bit current steering DAC with 1 LSB = 21nA: Binary-coded (a), Thermometer-coded (b).

Figure 4.22(b) shows the transfer curve of 7 bit thermometer-coded current-steering DAC. As we can see there are no glitches and the DNL for entire converter is simply zero as shown in Figure 4.22(b) and so perfectly linear in terms of differential non-linearity.

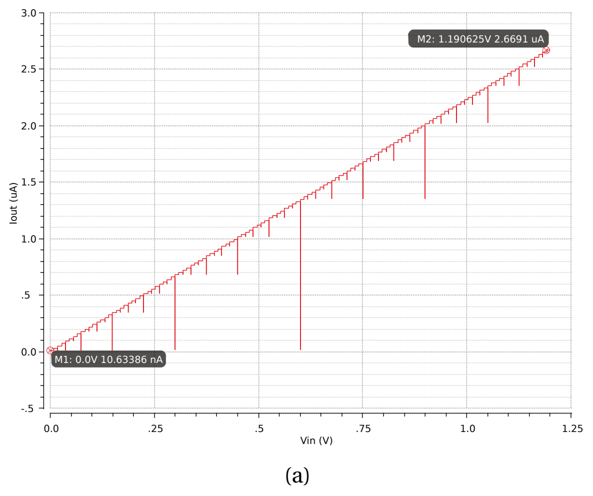


Figure 4.21: Output of a 7 bit current steering DAC with 1 LSB = 21nA: Binary-coded (a), Thermometer-coded (b).

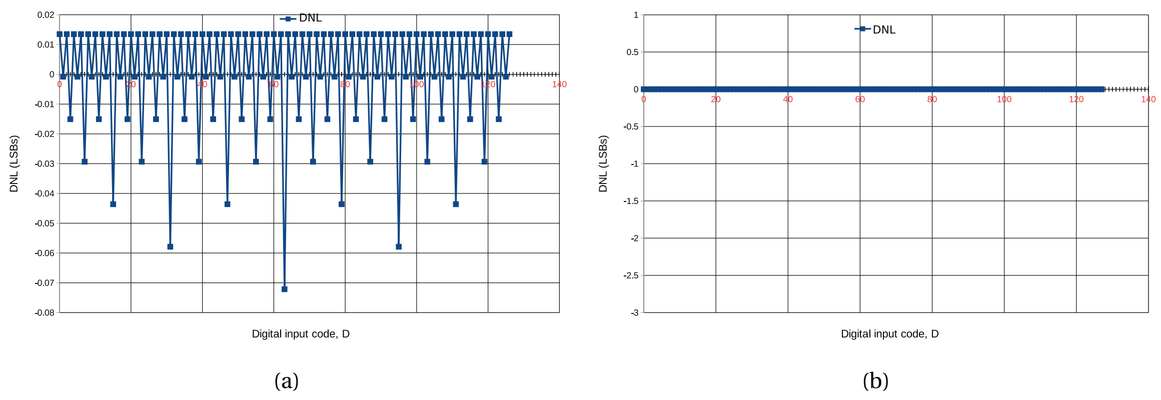


Figure 4.22: DNL curve for the 7 bit current steering DAC: Binary-coded (a), Thermometer-coded (b).

Another important static characteristic of DACs is called *integral nonlinearity (INL)*. Defined as the difference between data converter output values and a reference straight line drawn through the first and last output values, INL defines the linearity of the overall transfer curve and can be described as

$$INL_n = Output\ value\ for\ input\ code\ n - Output\ value\ of\ the\ reference\ line\ at\ that\ point \quad (4.7)$$

and the INL curves for both DACs are shown in Figure 4.23.

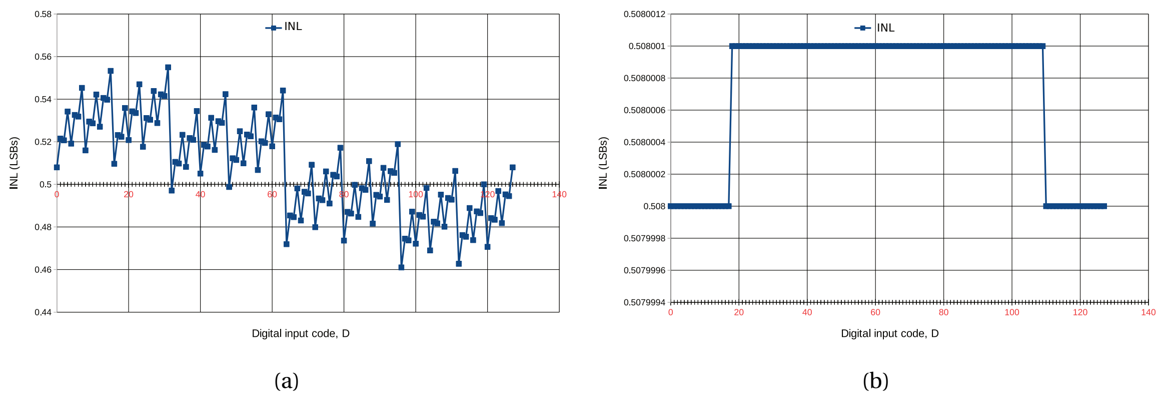


Figure 4.23: INL curve for the 7 bit current steering DAC: Binary-coded (a), Thermometer-coded (b).

The maximum INL measured is 0.554982 at code 31 for the binary-weighted current-steering DAC and for the entire converter is +0.554982/+0.461018 LSB as shown in Figure 4.23(a). For the thermometer-coded current-steering DAC the maximum INL measured is 0.508001 at code 64 and for the entire converter is +0.508001/+0.508, see Figure 4.23(b).
It is common practice to assume that a converter with N-bit resolution will have less than $\pm LSB$ of DNL and INL.

The analog output of both DACs should be 0 A for D = 0. However, an offset exists if the analog output current is not equal to zero. For the binary-weighted current-steering DAC is 10.6336 nA as shown in Figure 4.21(a). And for the thermometer-coded current-steering DAC it's 10.77729 nA, see Figure 4.21(b).

Another important performance parameter is the *Dynamic Range (DR)*, in other words the *signal-to-noise ratio (SNR)* and if we consider only quantization noise it is *signal-to-quantization noise ratio (SQNR)*. For both DACs, the dynamic range is related to the resolution of converter. For example, an N-bit DAC can produce a maximum output of 2^{N-1} multiples of LSBs and a minimum value of 1 LSB. Therefore, the dynamic range in decibels is simply

$$DR = SQNR = 20log\left(\frac{2^N - 1}{1}\right) \approx 6.02.N\ dB \tag{4.8}$$

So in this case we have 7 bits and thus a $DR = SQNR = 20log(127) \approx 12.665\ dB$

Functional performance of the mixed-signal IQ offset compensation system level design

The complete IQ offset compensation system is built in *Cadence design systems ic design software* [4] using the *Analog/Mixed Signal High Performance Interface Tool Kit (AMS HIT-KIT)* 0.18*µm high-voltage CMOS (HVCMOS) technology* [1]. Table 4.3 shows a view of the Cadence cells being used. As we can see, for now, the only transistor-level design is the track-and-hold and bias (TH & Bias) circuitry and the rest is build with Verilog-AMS.

Cadence Cell	**Model used**
TH & Bias	Transistor-Level
two-quadrant detector	Verilog-AMS
Up/Down counter	Verilog-AMS
binary-to-thermometer converter	Verilog-AMS
current-steering DAC	Verilog-AMS
Summator for IQ offset injection	Verilog-AMS

Table 4.3: View of the cells used in Cadence for the IQ offset compensation system level design.

As discussed in subsection 2.2.2, the duty cycle of the two-quadrant detector output signal is an excellent tool to monitor IQ offset compensation and its quality. For IQ sinusoidal signals with a certain frequency, perfect offset compensation should be a detector output signal with 50% duty-cycle and having same frequency. The up-down counter should then count an equal number of high- and low pulses. Figure 4.24 shows an output from two-quadrant detector for an IQ offset compensated signal of 10 kHz. The number of high- and low pulses are counted by an 8 bit Up-Down counter, MSB bit not included, and will therefore count 127 high- and 127 low pulses. The offset will always be around the ideal position, i.e. where the high pulses end, and this can be seen as 4 pulses above- and 4 below this reference position for sinusoidal IQ signals in a $\pi/4$ DQPSK communication system, i.e., a modulation scheme where signal sample rate is 8x the information frequency producing 8 constellation points. This was discussed earlier at the beginning of this chapter in subsection 4.1.1.

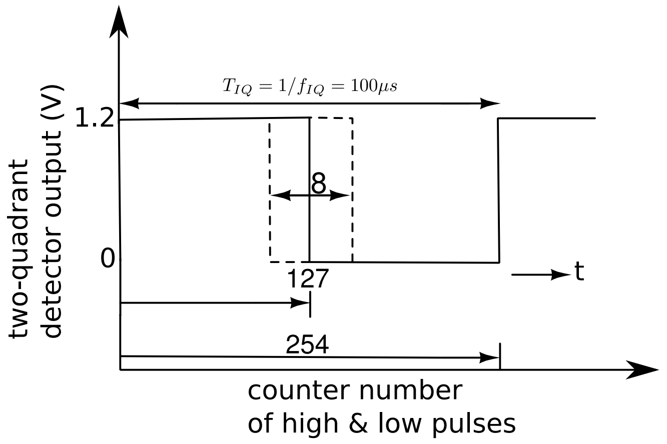

$T_{IQ} = 1/f_{IQ} = 100\mu s$
1.2
0
8
127
254
t
two-quadrant detector output (V)
counter number of high & low pulses

Figure 4.24: Two-quadrant detector output for an I or Q offset compensated signal of 10kHz. Ideally, would result in perfect pulse shaped signal with 50% duty-cycle and using an 8 bit Up/Down counter it will count 127 high- and 127 low pulses, the MSB bit not included. The offset is around the reference of 127 pulse counts and swings around 4 pulses up and 4 pules down relative to this position.

With this data we can determine the moment at which the system is compensated. The ideal duty-cycle of an offset compensation is in fact, expressed in terms of counted number high- and low pulses, equal to

(127/254) $*$ 100 = 50%. However, the duty cycle fluctuates 8 pulses around this value and hence, the duty-cycle for a compensated offset will be in the range

$\frac{127-4}{254} * 100 = 48.43\%$ and $\frac{127+4}{254} * 100 = 51.57\%$

so, if the duty cycle falls within this range, we can assume that the offset is compensated. Now we have found one of the criteria to test the speed of compensation system. We can simply monitor the two-quadrant output signal wave and determine the time at which the offset is compensated.

Another criteria is the way current sources of the current-steering are controlled. Two methods can be used

1. Simply turning on and off.
2. Switching it to ground or T/H transistor.

The first method is controlling the current source by a voltage as shown in Figure 4.25(a). When the switch S_y is open there's no voltage over the current source xI, i.e., $AVDD - AVDD$, and it turns off. And when the switch S_y is closed there will be a voltage over the current source xI of $VDD - V_{s,TH}$ and it turns on.
The second method is switching the current source xI to ground or to the T/H transistor, see Figure 4.25(b). Using this method, the system should perform faster because there's always a current standby. The only limitation is the switching speed of the switch. Disadvantage compared to the first method is that there will always be a current flowing to ground if the source is not used and the system will therefore consume more power.

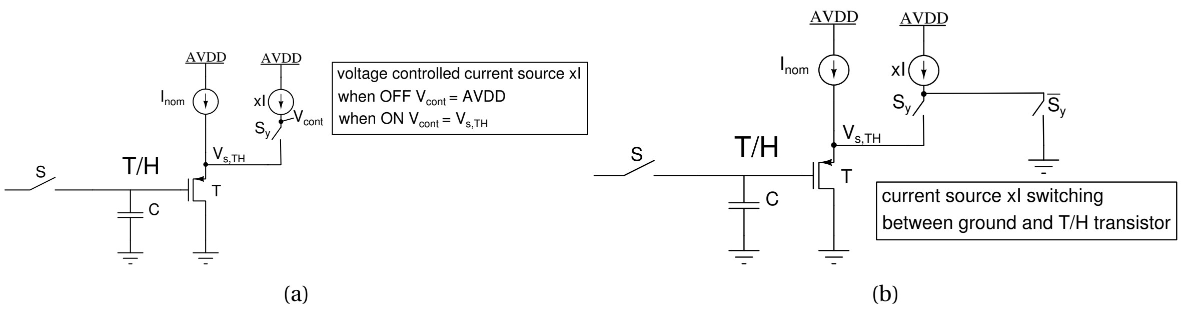


Figure 4.25: Concept of the way a current source xI from current-steering DAC is controlled: turning on/off by voltage V_{cont} (a), switching it between ground and T/H transistor (b).

We will now apply these two criteria to the simulation results below.

Figure 4.26 shows a simulation of compensation process for a negative differential I offset of 69mV. The system used is based on two-quadrant detector and using voltage controlled binary-weighted current sources of the current-steering DAC. A good look to the detector output wave reveals that the system works. At the beginning of simulation and if we look at the markers, duty-cycle is being defined as

$\frac{M19-M18}{M21-M20} * 100 = \frac{101.0303\mu s - 45.0\mu s}{145.2515\mu s - 45.0\mu s} * 100 = 55.86\%$

and looking at the end of simulation, the moment that offset is compensated, it is

$\frac{M11-M10}{M13-M12} * 100 = \frac{747.751\mu s - 699.0864\mu s}{798.878\mu s - 699.0864\mu s} * 100 = 48.76\%$

Marker M12 indicates that the system will do about 700μs until the offset is compensated. Furthermore, we see the input- and output phase waves. As we can see, the phase slope at output that's between $+\pi$ and $-\pi$ is clearly distorted at the beginning and gradually over time becomes more linear.

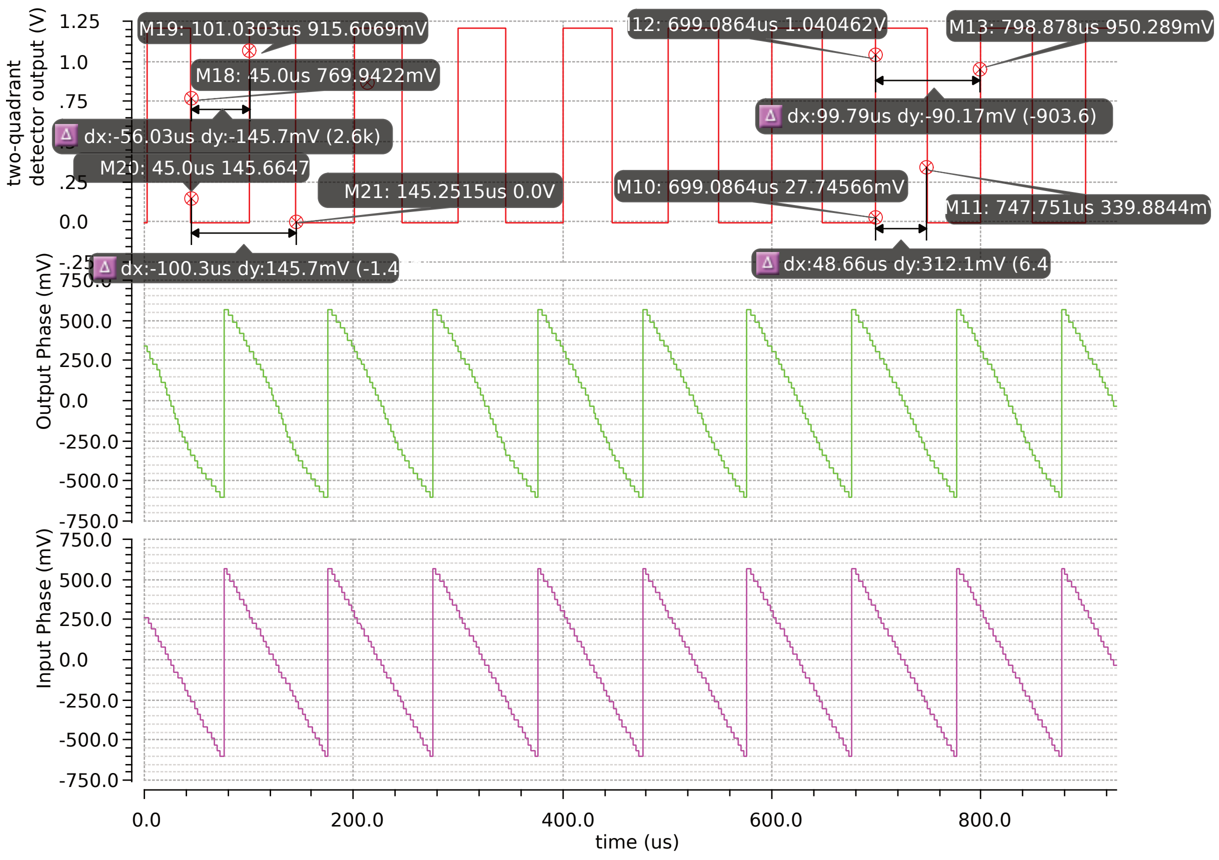


Figure 4.26: Simulation results from the IQ offset compensation system based on two-quadrant detector using voltage controlled binary-weighted current sources of the current-steering DAC for a negative differential I offset of 69 mV.

Figure 4.27 shows simulation results of IQ offset compensation system based on two-quadrant detector and using switched binary-weighted current sources of the current-steering DAC. Duty-cycle at compensation is defined as

$$\frac{M38-M37}{M40-M39} * 100 = \frac{648.25\mu s - 599.137\mu s}{699.1406\mu s - 599.137\mu s} * 100 = 49.11\%$$

This system is clearly faster than the first and confirms the fact that use of standby currents makes the system faster. To be precise, in this case $100\mu s$ quicker.

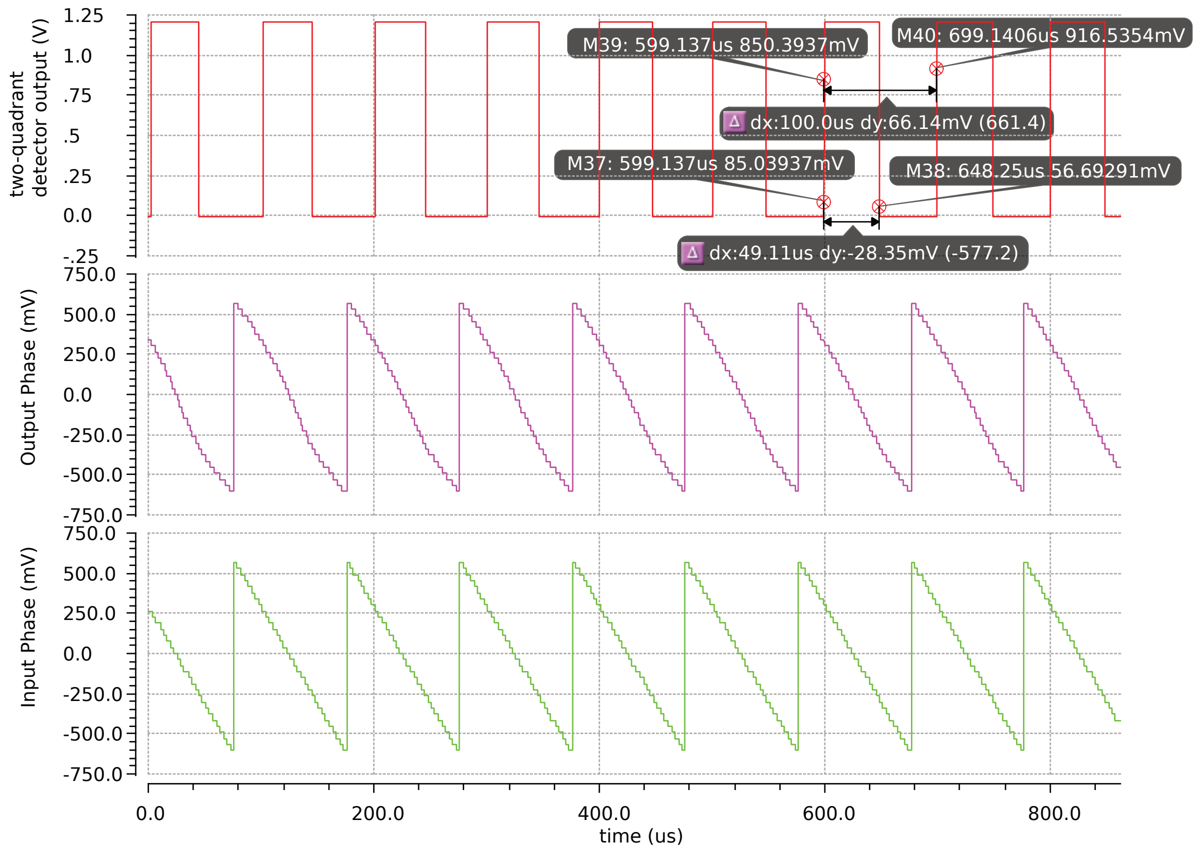


Figure 4.27: Simulation results from the IQ offset compensation system based on two-quadrant detector using switched binary-weighted current sources of the current-steering DAC for a negative differential I offset of 69 mV.

Both simulations have been conducted also under the same conditions for the same IQ offset compensation system except that it uses a thermometer-coded current-steering DAC instead of a binary-weighted. If we compare Figure 4.28 with 4.26 then we see clearly that this system is faster. It has the same speed as that of IQ offset compensation system using switched binary-weighted current sources, see Figure 4.27, but consumes less power because the currents are not standby.

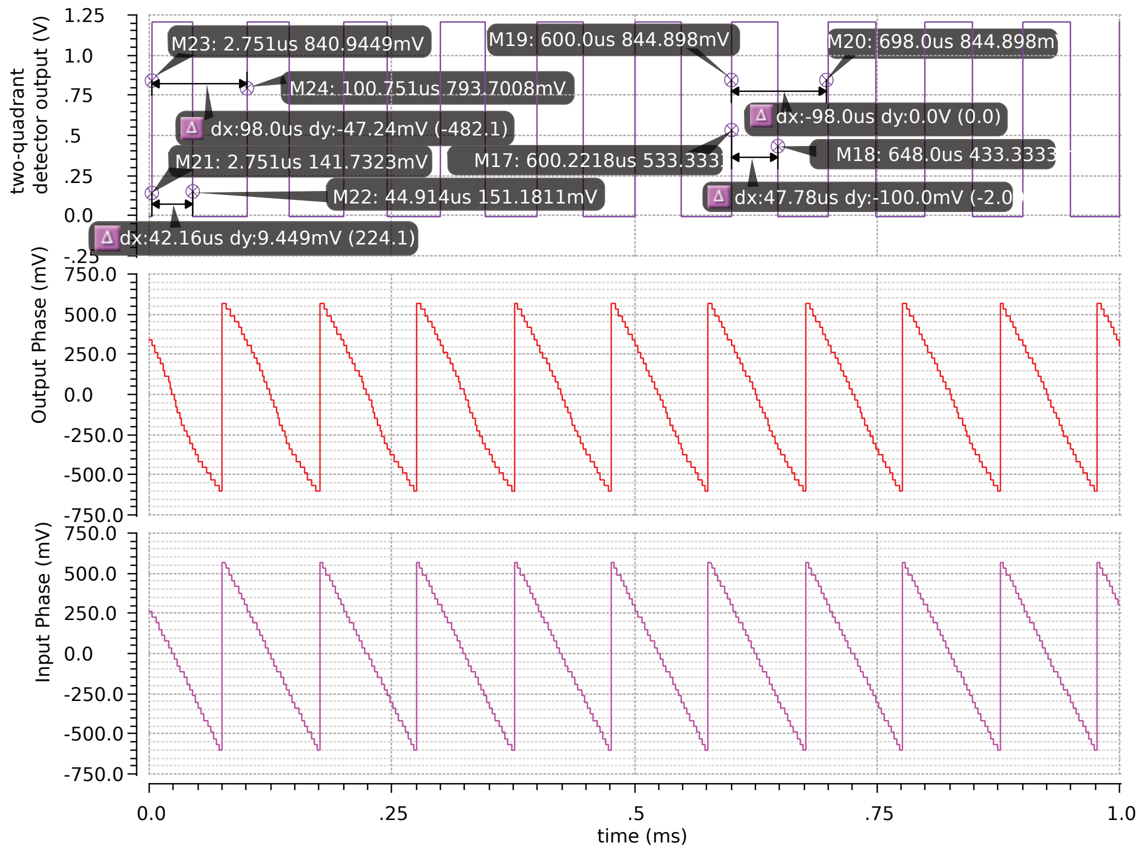


Figure 4.28: Simulation results from the IQ offset compensation system based on two-quadrant detector using voltage controlled current sources of thermometer-coded current-steering DAC for a negative differential I offset of 69 mV.

The last simulation shows the result of IQ offset compensation system using switched current sources of thermometer-coded DAC. And this reveals that we have gained no speed using this method, both systems compensate at about 600μs.

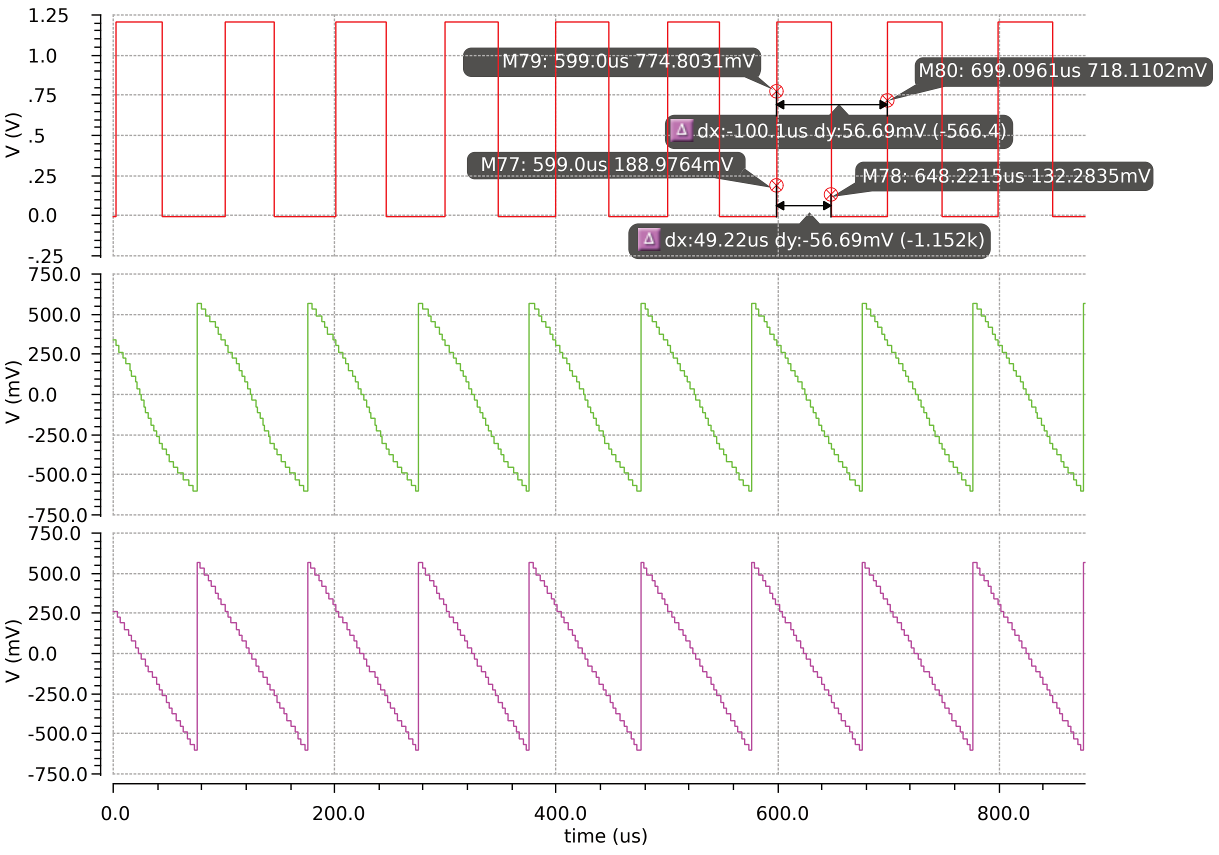


Figure 4.29: Simulation results from the IQ offset compensation system based on two-quadrant detector using switched current sources of thermometer-coded current-steering DAC for a negative differential I offset of 69 mV.

Thanks to Verilog-AMS modeling we were able to compare the linearity of the DACs and to test the IQ offset compensation system for speed. Other important performance parameters are of course also noise and cost parameters such as power consumption. These require more depth into the system and might be too cumbersome to model it here on system level. It is therefore better to test these at circuit level. The previous tests must of course be performed on the system at circuit level as well.

4.2. Modeling and design of IQ gain mismatch compensation system

What was said at the beginning of Section 4 about Verilog-AMS also applies here. Here too we try to describe the IQ gain mismatch compensation system as much as possible with Verilog-AMS. It facilitates building such a complex system and modeling it.

Finding a location in the Phase-Domain ADC receiver to control IQ gain mismatch

Let's proceed to the system-level description of the IQ gain mismatch compensation system. The model is shown in Figure 4.30. At first we need to find a way to implement the sommator (Σ) + multiplier (x). For the other blocks in the feedback loop, we can use the straightforward method that we also applied to IQ offset mismatch. In other words, for the integrator ($\frac{1}{s}$) and gain block (A) we can again replace it by a digital up/down counter. The 2 bit *Counter*, *MUX* and the *Two Regions Detector* are also fully digital implementations. As we can see in the Figure, the feedback loop consists of a *Two Regions Dectector* where the four signals are generated and the *MUX* passes the signals one at a time to the integrator with the speed of the 2 bit *Counter*. The *MUX* is therefore just an electronic switch that shares the time between the four signals, that is, multiplexing. The integrator is needed for error tracking to eliminate the steady state error between the actual- and desired output phase, i.e., eliminate the IQ gain mismatch. The gain block A is added to enhance the speed of control. If the system were compensated for IQ gain mismatch, then the output of the gain block A would be equal to the mismatch factor α_f defined by Equation (1.4).

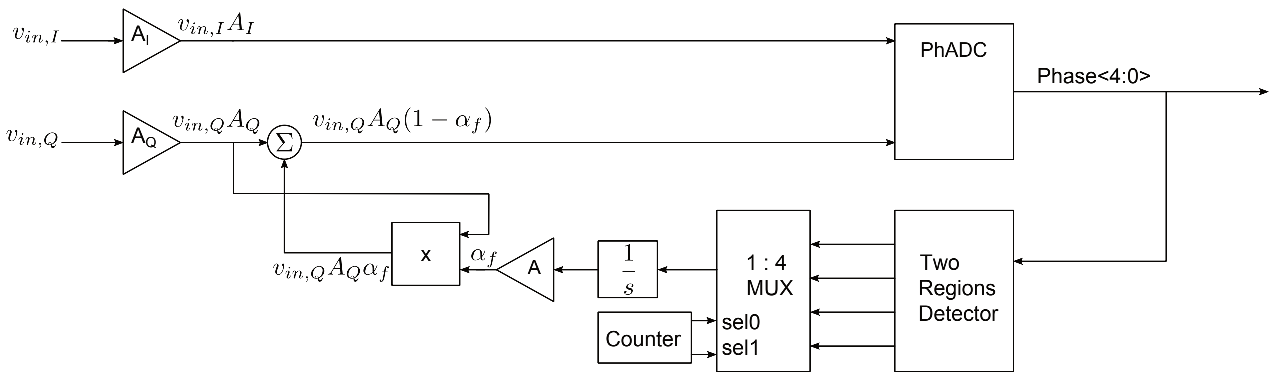


Figure 4.30: System level model for IQ gain mismatch compensation.

A good possibility for the adjustment of the gain in the Q path is offered by the PGA stage in the analog baseband of the Phase-Domain ADC receiver [10], that has been dealt with in Subsection 2.4.4. Figure 4.31 shows a system level concept of the PGA gain control.

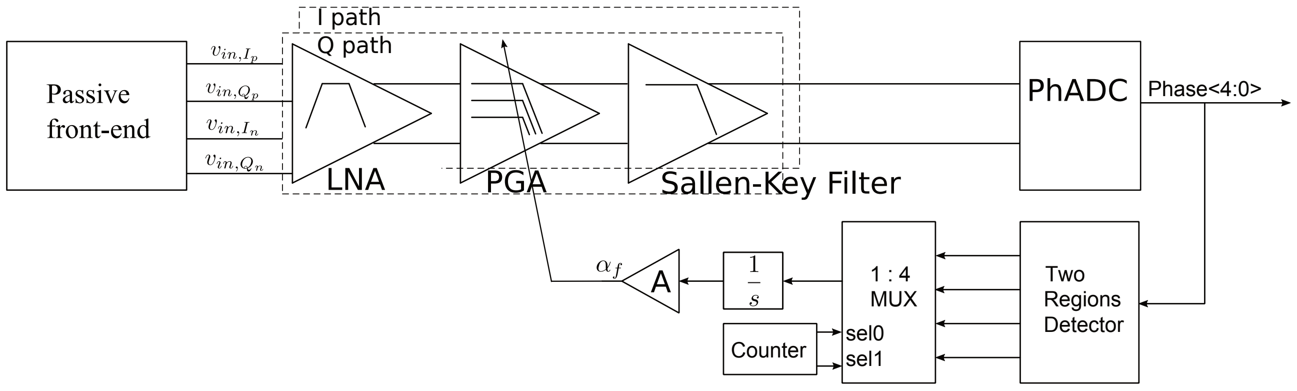


Figure 4.31: System level model for IQ gain mismatch compensation using the Programmable Gain Amplifier (PGA) to control the IQ gain.

Figure 2.60 in Subsection 2.4.4 shows the circuit of the PGA amplifier. The gain from input to output is defined as;

$$\frac{R_o}{2R_d} \tag{4.9}$$

Where,
R_o = output resistor
R_d = degeneration resistor

The gain can be progammed by adjusting the source degeneration R_d as this resistor is connected between internal nodes of the gain stage, the poles of the transfer function are kept essentially constant [12]. This is a 10 bit programmable resistance with PGA voltage gain steps of 6 dB. Unfortunately, it is not binary coded for the full range and there's no linear relation between the code and the size of resistance but has a limited number of specific settings. See Tabel 4.4 for the programming modes and the corresponding resistor values. The gain values are given for a $R_o = 400k\Omega$.

A_{PGA}	$S_{9:0}$	R_d
6 dB	1000 11 0001	$100k\Omega$
12 dB	1000 10 0010	$50k\Omega$
18 dB	1010 10 0011	$250k\Omega$
24 dB	1010 00 0100	$12.5k\Omega$
30 dB	1110 00 0111	$6.25k\Omega$
36 dB	1110 00 1000	$3.125k\Omega$
42 dB	1111 11 1000	$1.5625k\Omega$

Table 4.4: Program codes and associated resistance values for degeneration resistor R_d.

But we prefer to control the R_o since R_d is not a small resistor and the gain is very sensitive to the value of R_d. R_o is a better place than R_d to implement the finer step than 6 dB, since R_o is also much bigger than R_d. The R_o resistance is split into a fixed resistance $R_{o_{fixed}}$ and a programmable resistor $R_{o_{ctrl}}$. See Figure 4.32.

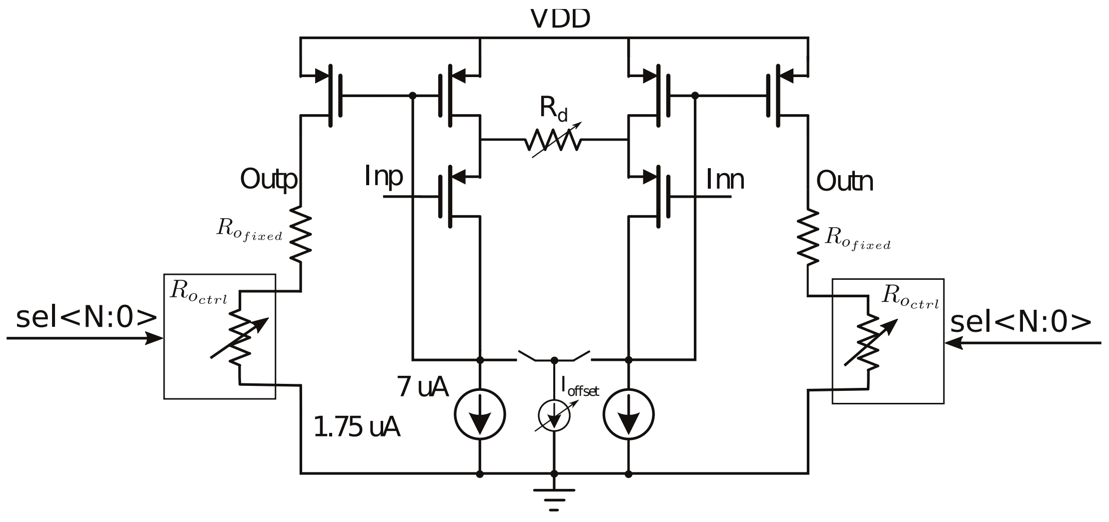


Figure 4.32: PGA circuit: R_o resistor replaced by fixed resistor $R_{o_{fixed}}$ in series with an electronic programmable resistor $R_{o_{ctrl}}$.

The output resistance R_o is defined by Equation (4.10).

$$R_o = R_{o_{fixed}} + R_{o_{ctrl}} = R_{o_{fixed}} + W.R_{step} \ \ \Omega \tag{4.10}$$

Where,
$R_{o_{fixed}}$ = fixed part of output resistor R_o.
$R_{o_{ctrl}}$ = programmable part of output resistor R_o.
W = binary to decimal converted value.
R_{step} = resistance step value.

The Equation for the conversion of binary to decimal value is shown in (4.11). It is a sum of products of a bit value $sel < N-i >$ with the base 2 to the power $N-i$, where i is from 1 to N.

$$W = sel < N-1 > \ \times 2^{N-1} \ + \ sel < N-2 > \ \times 2^{N-2} \ + \ ..... \ + sel < N-i > \ \times 2^{N-i} = \sum_{i=1}^{N} sel < N-i > \cdot 2^{N-i} \tag{4.11}$$

where,
N = the number of bits.
2 = the base for binary values.
i = value from 1 to N.
$sel < N-i >$ = bit value 0 or 1.

Figure 4.33 shows the IQ gain mismatch control system with an *Up/Down counter* for controlling the programmable part of the PGA output resistor $R_{o_{ctrl}}$.

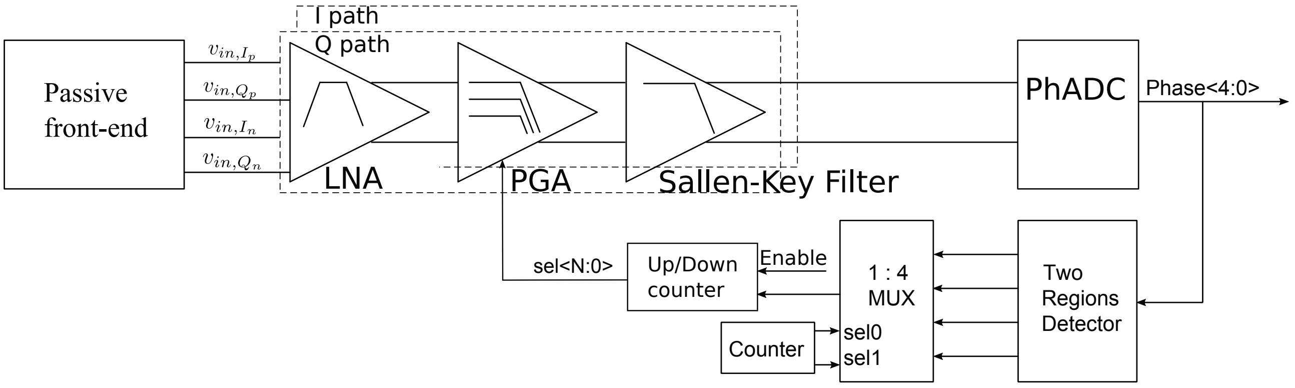


Figure 4.33: System level model for IQ gain mismatch compensation using an Up/Down counter to program the PGA.

Setting the initial gain and gain control range for the PGA amplifier

The resolution of the Up/Down counter, i.e., the number of bits N, is determined by the maximum occurring IQ gain mismatch at the input of the PhADC and a maximum permissible IQ gain mismatch that still gives a required BER of $\leq 1.10^{-5}$ for $\pi/4$ DQPSK demodulation according to the IEEE802.15.6 standard. For our PhADC receiver [10] we already know this value from the BER simulations and that is about ±3 dB. See Figure 3.29. The maximum occurring IQ gain mismatch can be determined on the basis of Monte Carlo simulations of the PhADC receiver front-end consisting of LNA, PGA and Sallen&Key filter. We do a measurement of the gain difference of the I and Q paths from the input of the LNAs to the output of the Sallen&Key filters and take the 3σ.
Unfortunately, the Monte Carlo simulations of the *pfetx* pMOS devices from the AMS hit-kit CMOS 180nm technologie were not successful. The problems are related to the AMS design kit itself, Cadence or the operating system. AMS will terminate the 180nm mpw service in 2019 and no further support is provided by ams to solve the problem. Hence, we can not solve this problem before the deadline for delivery of this thesis report.
And that is why we made a rough estimate of a maximum occurring IQ gain mismatch of ±10 dB and have made further calculations based on this. Suppose now that we want the system to be able to compensate up to a IQ gain mismatch of ±2dB to be within that required ±3dB, then we can assume that for controlling from ±10dB to ±2dB five control steps is sufficient. See Figure 4.34. And so a counter of 3 bit is needed.

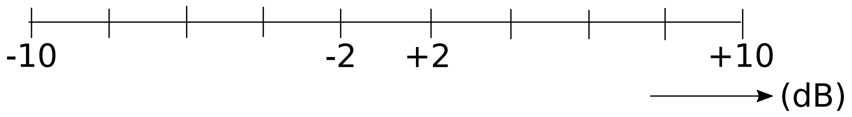


Figure 4.34: Maximum and minimum occurring IQ gain mismatch of ±10 and ±2dB, respectively.

If the IQ PGA gains are set to a nominal value of 6dB than the mismatch can be controlled from -4 to +16dB. See Figure 4.35. At a minimum gain of -4dB the PGA output resistor $R_o = R_{o_{fixed}}$. Degeneration resistor R_d is programmed for a value of $100k\Omega$ and using (4.9)

$$20log\frac{R_{o_{fixed}}}{2R_d} = -4 \text{ dB} \rightarrow R_{o_{fixed}} \approx 126k\Omega$$

For a maximum gain of 16dB the PGA output resistance R_o becomes

$$20log\frac{R_o}{2R_d} = 16 \text{ dB} \rightarrow R_o \approx 1262k\Omega$$

Using Equation (4.10) and W=8 (3 bits) R_{step} becomes

$1262k = 126k + 8 \cdot R_{step} \rightarrow R_{step} = 142k\Omega$

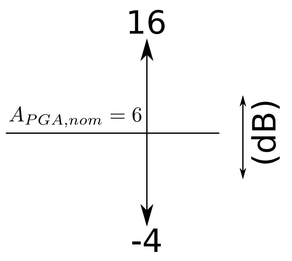


Figure 4.35: IQ gain path is set at 6dB nominal and can be controlled from -4 to +16dB.

4.2.1. Verilog Analog Modeling

Testbench setup feedforward path of IQ gain compensation system

The PhADC is implemented in the same way as described in Subsection 4.1.1, but now the front-end has been installed between the baluns and the T/H circuit, consisting of a baseband low-noise amplifier (BB LNA) and the analog baseband which consists of the programmable gain amplifier (PGA) and the Sallen & Key filter. These blocks are transistor-level designs. With the PGA, we can now implement the feedback loop for IQ gain mismatch compensation. See Figure 4.37.

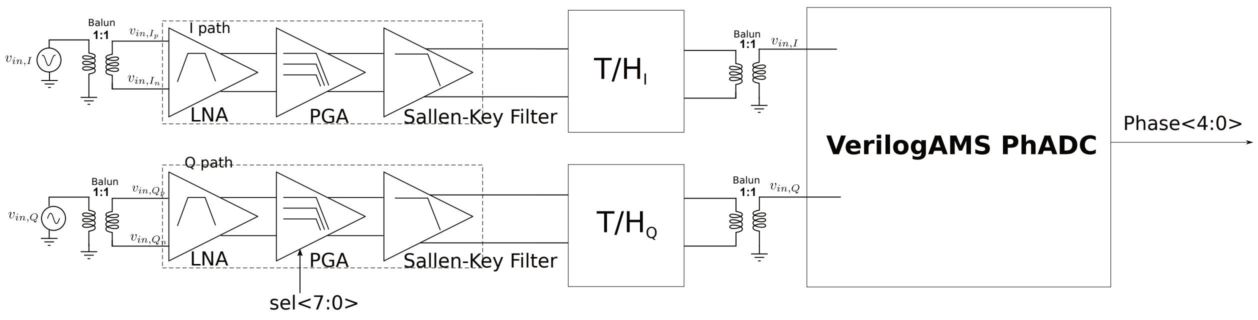


Figure 4.36: Between the input sources and Track-and-Hold circuitry the LNA, PGA and Sallen-Key filter has been added and in the Q path control lines sel<N:0> are shown to control PGA gain in the Q path.

For the correct implementation of the IQ gain mismatch compensation shown in Figure 4.37 there are a few important things that we have to take into account; If we don't want to lose information from the *Two Regions Detector*, then we have to multiplex the 4 outputs with a clock 4x the detector's clock frequency. But the disadvantage of this approach is that we then need a four times higher clock frequency. The *Up/Down counter* will oversample the output of the MUX and count more than it should do. The control step size of the programmable part of the PGA output resistance $R_{o_{ctrl}}$ thus also becomes four times as large and as a result the fineness of the control is reduced.

Another approach would be to replace the multiplexer with a four channel counter. At each transition of a signal from the *Two Regions Detector* this counter will sum the 4 output signals, hence , we don't need a clock source. Its output would then be two-channel, because we do not add the signals as the sum of the powers of 2, but we count the number of high and low pulses and so the minimum counter output is 0 and the maximum counter output is 4, so 2^N=4 and N=2 which equals the number of outputs. The *Up/Down counter* must then sum two signals, namely the two output signals from the four-channel counter. The number of outputs of the *Up/Down counter* is therefore increased by two. The control bits of the programmable part of the PGA output resistance $R_{o_{ctrl}}$ should therefore be expanded with 2 bits.

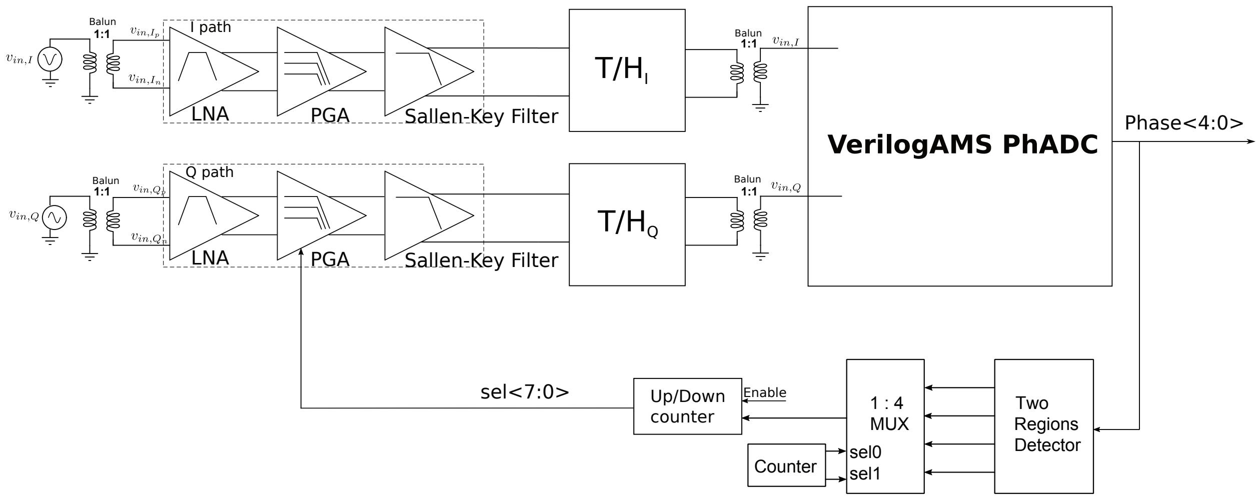


Figure 4.37: Between the baluns and Track-and-Hold circuitry the LNA, PGA and Sallen-Key filter has been added together with the IQ gain mismatch compensation loop.

This makes the design of the control hardware somewhat complex but instead should be kept simple; All the control hardware in the IQ gain mismatch feedback loop and the PhADC runs on the same clock. But with this approach we lose information from the *Two Regions Detector*, because we down-sample with a factor of four. We take 1 sample and then after another 3 samples, etc. But simulations show that, despite this limited information, the IQ gain mismatch loop still works, although the compensation speed is of course lower than when all the information from the detector is applied as with the four-channel counter approach. Down-sampling with a factor of 4 means also decreasing the loop-gain of the IQ gain mismatch compensation system with a factor of 4.

Another important aspect is the clock timing of the different hardware components of the control loop. The clocking in of data at the input of the *MUX*, *Two Regions Detector* and the 2 bit *Counter* may all occur at the same moment, but not for the *Up/Down counter*. The data at the input of this counter must first stabilize after the clock transition and only then can it be clocked in. A solution is to have the *MUX*, *Two Regions Detector* and the 2 bit *Counter* clocked on the positive edge and the *Up/Down counter* on the negative clock edge and with this we create a delay of half a clock cycle.

4.2.2. Simulation

In this Subsection, the Cadence System design in transistor and VerilogAMS level is simulated for its functionality. But before we start treating the simulation results, we first look at an important aspect in relation with the circuit level simulation of the system.

The high-pass 3 dB point f_{HP} of the analog filter from Subsection 2.4.4 is set by a feedback resistor R_f and capacitor C_F (i.e., $1/2\pi R_F C_F$) of the low noise amplifier (LNA). See Figure 2.59. This is needed to suppress the DC offset generated by the preceding stage, the passive mixer [10]. In order to make the inter-symbol-interference (ISI) negligible, the f_{HP} must be less than one-thousandth of the symbol rate [8], which is 187.5 kHz in the phase-domain ADC receiver design [10]. For a small channel integrated circuit design process like the AMS 180nm H18 technology the unit size of a capacitor is nominally 2.05 $fF/\mu m^2$ and due to the large size on chip, values usually are limited in the range of pF. The unit size of a poly resistor in the AMS 180nm H18 technlogy is nominally 260 Ω/sq and due to its large size on chip, values usually are limited in the range of $k\Omega$. So, to realize such a low f_{HP} a very large resistor is needed. To limit the size on chip, a pseudo-resistor [10] is employed to realize the big resistance required by f_{HP}. But these suffer from poor linearity. In practice, the nominal value of R_F is much greater than what would be required for f_{HP}. This moves the frequency range being distorted by the nonlinear R_F below 187.5 kHz, so as to give negligible impact on signal integrity. Considering the trade-off between capacitor size and the complexity of the pseudo-resistors, RF and C_F are chosen equal to $8G\Omega$ and $2pF$, respectively, yielding a f_{HP} of 10 Hz.

For our system simulations, unavoidably, the extreme small f_{HP} gives rise to a huge time constant. The system needs a certain amount of time to settle around the right DC operating point. One approach for avoiding such a long simulation time is to move f_{HP} from 10 Hz to lets say 500 Hz. This will significantly reduce the required feedback resistance, i.e., R_F = 159 $M\Omega$. For testing the functionality of the IQ gain mismatch compensation and the IQ gain mismatch this change does not matter at all for the performance.

Figure 4.38 shows a transient analysis in which the IQ gain mismatch compensation has been tested. The red wave represents the I signal at the output of the Sallen&Key filter. The signal level at the input is chosen to be 10 μV. For a total gain of 31 dB the output signal would then have to be approximately 354 μV and seen the result this is indeed the case. The green wave represents the Q signal at the output of the Sallen&Key filter. In order to realize an IQ gain mismatch, the amplitude of this signal has been chosen differently with respect to I and therefore is set to a value of 12.5 μV. As we can see from the plot, the I channel needs a time of 3 ms to settle around the right DC operating point, which is close to 0 V.

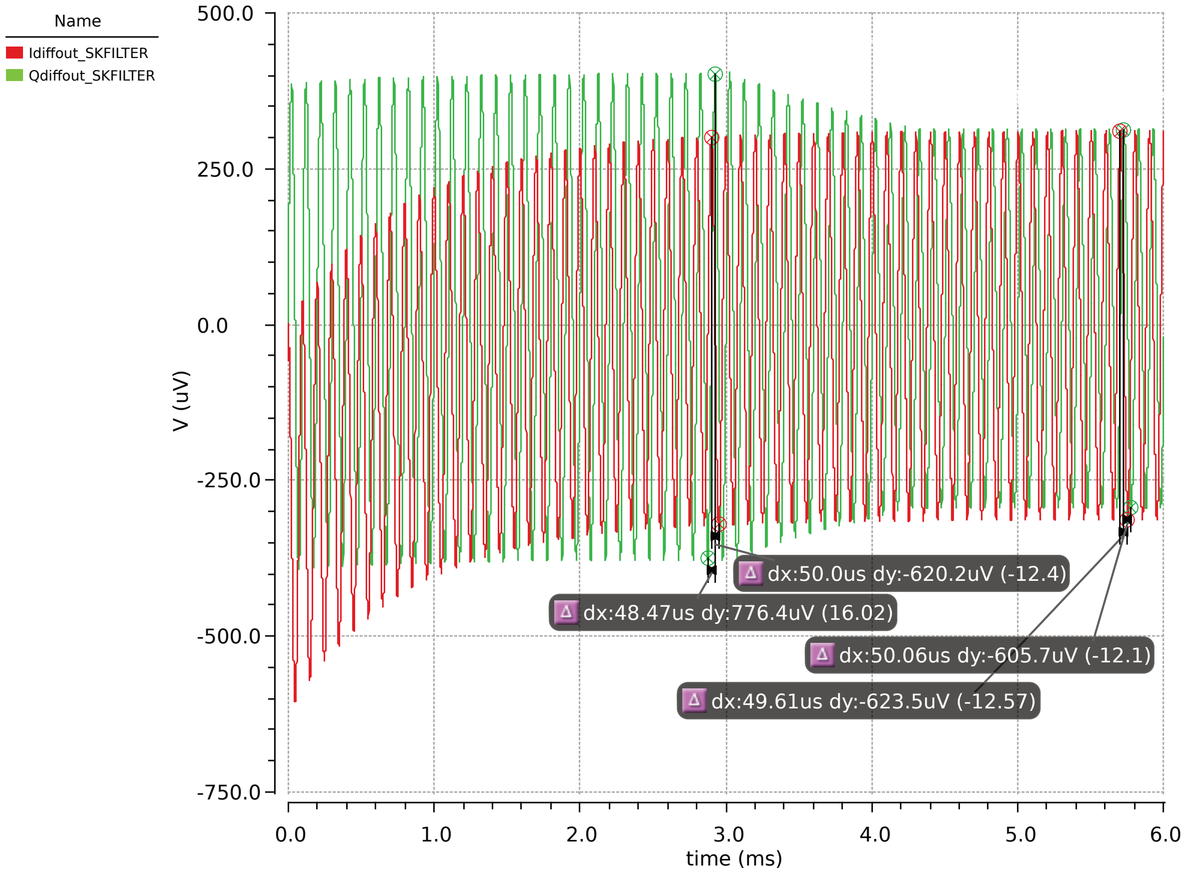


Figure 4.38: Simulation results from the IQ gain mismatch compensation system based on the *Two Regions Detector* using an up/down counter to control PGA gain.

To close the IQ gain mismatch compensation loop and start the IQ gain mismatch compensation an *enable* signal is used to trigger the *Up/Down counter*. The threshold for the triggering is set at $DVDD/2$ = 0.6 V, where $DVDD$ is the digital circuitry supply source of 1.2 V. In Cadence the enable signal is configured as a piece-wise linear source ($VPWL$). Given the results in Figure 4.39, the IQ gain mismatch compensation is started after a transient time of exactly 3 ms, so exactly at the moment the system has been settled around the right DC operating point.

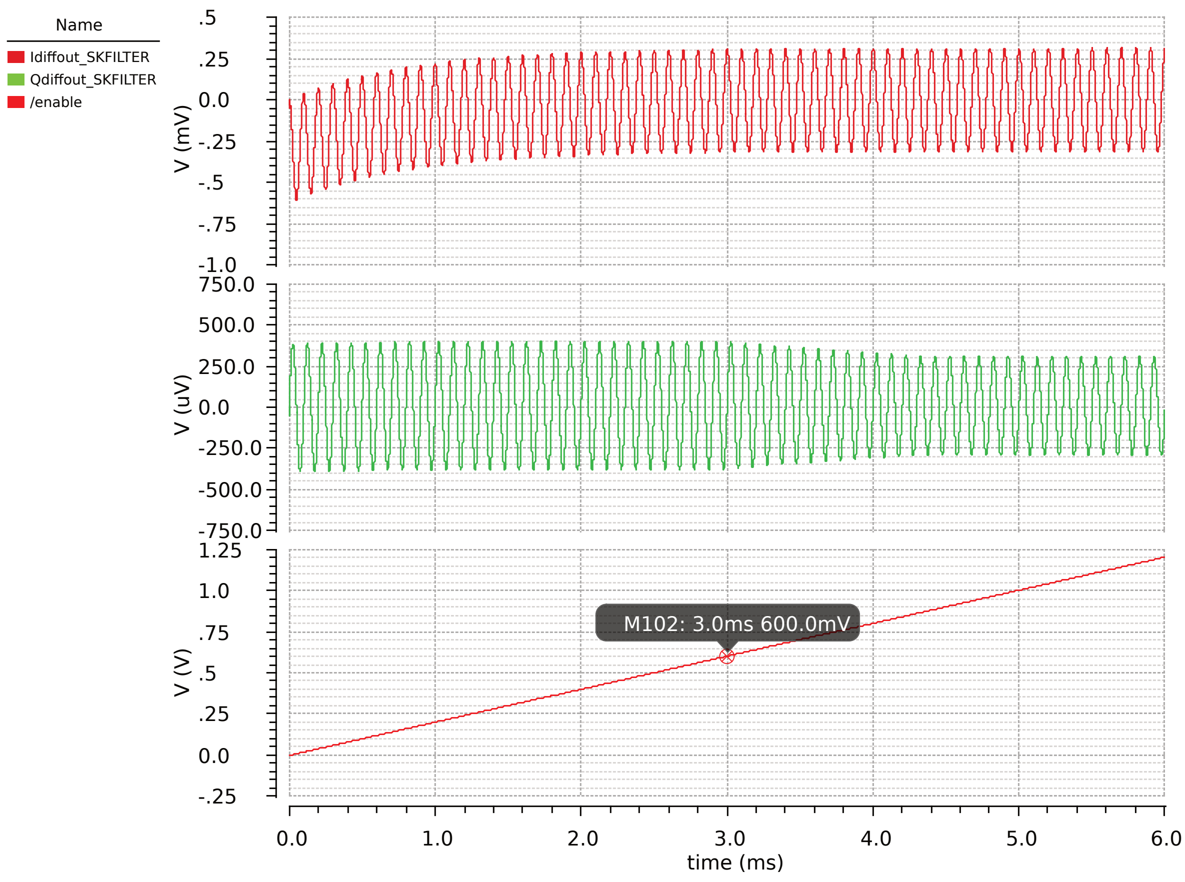


Figure 4.39: Simulation results from the IQ gain mismatch compensation system based on the *Two Regions Detector* using an enable signal for to trigger the up/down counter to control PGA gain.

As we can see clearly from Figures 4.38 and 4.40, the Q signal is controlled to the amplitude of the I signal and thus the IQ gain mismatch is compensated.

The gain mismatch before compensation is $\pm 20log(12.5\mu V/10\mu V)$ = 1.94 dB when referred to input or $\pm 20log(776.4\mu V/620.2\mu V)$ = 1.95 dB when referred to output. See Figure 4.40.
After compensation, we see from Figure 4.40 that the gain mismatch is only $\pm 20log(623.5\mu V/605.7\mu V)$ = 0.25 dB. We expect 0 dB IQ gain mismatch here but there is a residual value that can no longer be eliminated. However, this does not mean that the system isn't fully compensated; there is an integrator in the feedback loop, hence, the steady-state error is zero at the output of the PhADC and thus there are no phase errors anymore. The detector can then no longer detect the phase vector distribution unbalance and thus the feedback loop is more or less broken. And if again phase errors occur, the system will again detect and compensate.

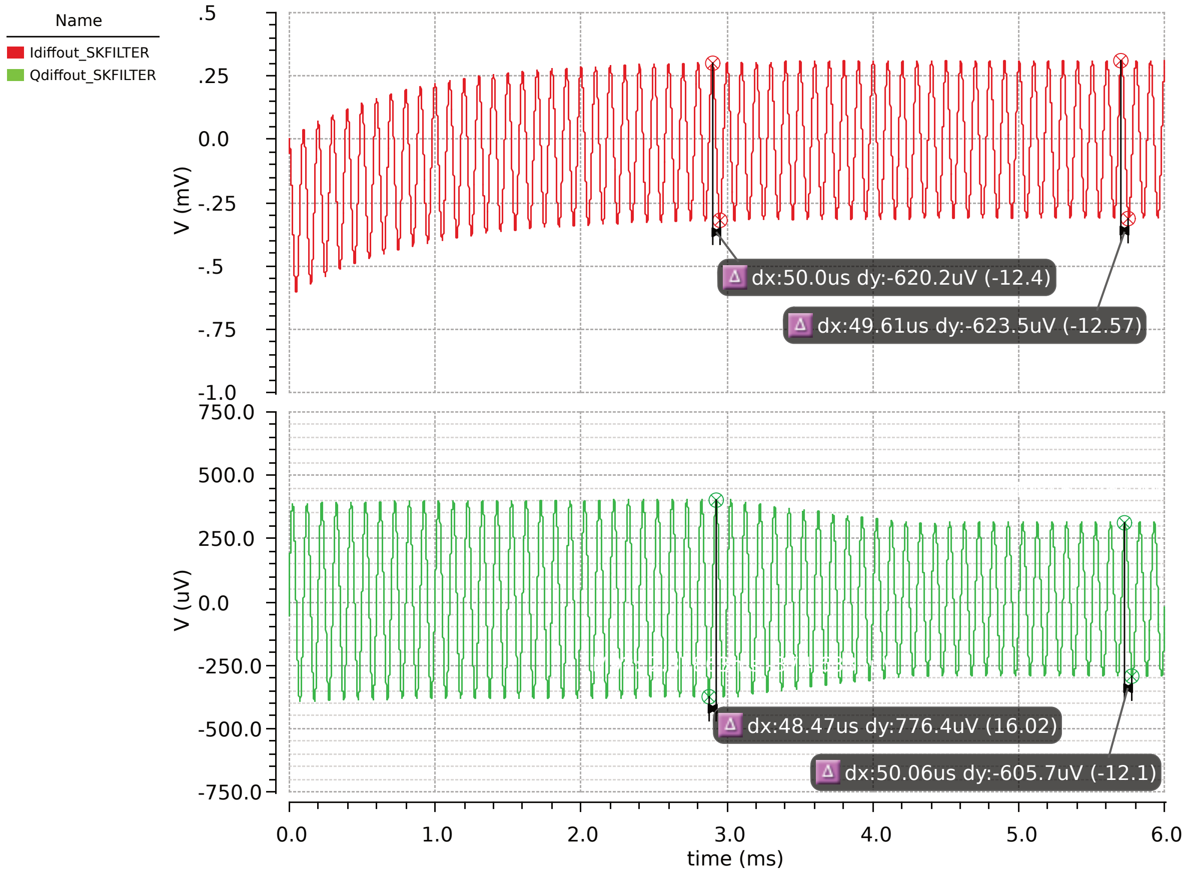


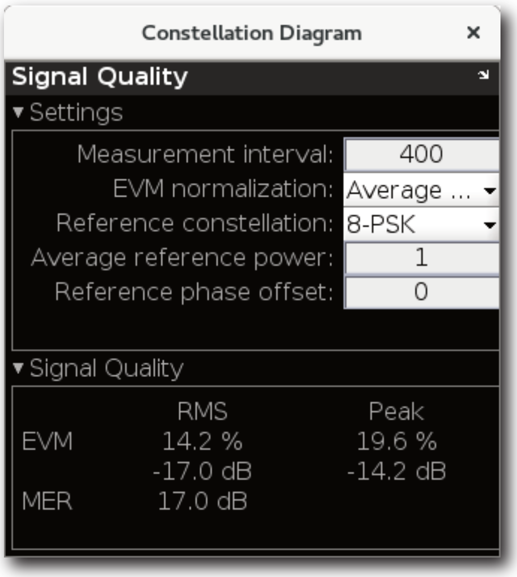
Figure 4.40: Simulation results from the IQ gain mismatch compensation system based on the *Two Regions Detector* using an up/down counter to control PGA gain with the two curves split from each other.

4.3. Overall System level Implementation

The Simulink system with both compensation loops can be found in Figure Q.1 in Appendix Q Section Q.1. In this model everything is implemented, namely; channel noise AWGN model, IQ gain mismatch, IQ offset, the biquad analog channel filter, the 5 bit PhADC, the *Two Quadrant Detector* and the *Two Regions Detector*. The system is simulated with the following conditions:

- Channel noise: $E_b/N_o = 20$ dB, equal to $SNR \approx 21$dB.
- Input receiver signal level: 0.3 V_{pp}.
- IQ offset: 100 mV
- IQ gain mismatch ± 3.5 dB.
- Modulation: $\pi/4$ DQPSK.
- Channel bandwidth: 100kHz.
- Bit sample rate: 375kHz.
- Symbol rate: 187.5 kHz.
- SRRC transmit filter with roll-off factor 0.5, filterspan: 12, upsample factor: 8. Filter gain 0 dB.
- Biquad fifth order channel select receive filter: f_{HP}=10 Hz, f_{LP} = 160 kHz. Filter gain: 0 dB. Downsample factor: 8.

As we can see from Figure 4.41, the signal quality is still acceptable. The EVM is -17 dB and is better than the permissible EVM number for $\pi/4$ DQPSK shown in Figure 2.29.



Constellation Diagram

Signal Quality

▾ Settings

Measurement interval: 400
EVM normalization: Average ...
Reference constellation: 8-PSK
Average reference power: 1
Reference phase offset: 0

▾ Signal Quality

```
        RMS        Peak
EVM     14.2 %     19.6 %
        -17.0 dB   -14.2 dB
MER     17.0 dB
```

Figure 4.41: Signal quality dialog box.

In Figure 4.42 the system is brought together, at a still high level, in a mix of analog and digital electronics. All blocks marked in blue are designed completely on transistor-level. The blocks marked in gray are modeled in VerilogAMS. Almost all the properties that a real circuit has, is modeled in it. The code can simply be converted into a gate-level netlist and synthesized in a register-level transfer (RTL) compiler such as Cadence Encounter [5]. And then in place&route tool like Cadence Innovus 4 converted to a layout.
Looking at the system, we also see that the *Up/Down counter* has an *enable* signal in each loop, this makes it possible to interrupt the compensation loop in case of, for example, calibration from one time. Since the feedback value is digital, we can save it. When the receiver is turned on, a training is used and the entire feedback loops, converge such that output phase is free of errors and the IQ offset and gain mismatch is removed. The resulting values of offset and gain mismatch are then stored by disabling the counters and remain frozen during the actual operation of the receiver.
The digital storage of offset and gain mismatch affords other capabilities, as well. For example, since the offset and gain mismatch may vary with the LO frequency or gain settings before or after the mixer, at power-up the receiver is cycled through all possible combinations of LO and gain settings, and the required values of offset and gain mismatch are stored in a small memory and when needed input to the *PGA* and or *Binary Thermometer coded encoder*.

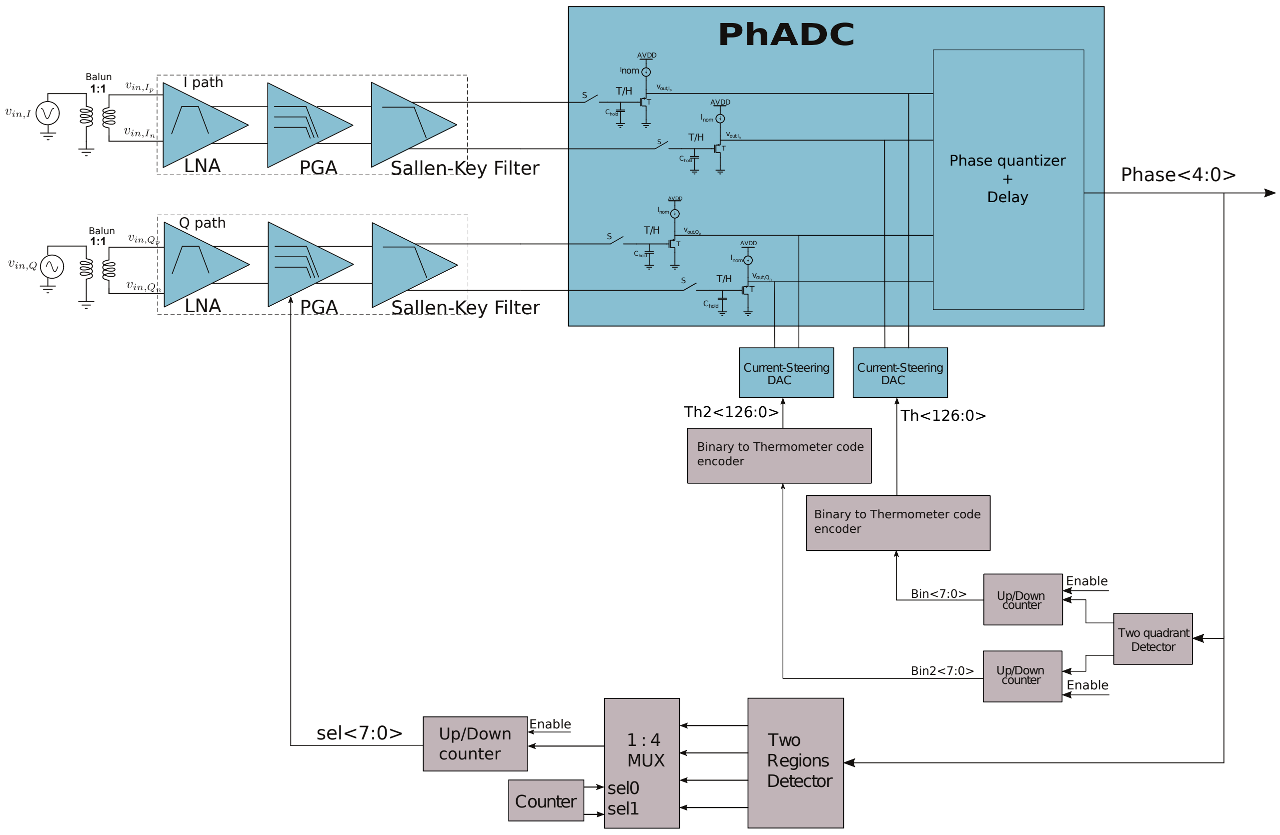


Figure 4.42: Overall system design with both IQ offset and gain mismatch compensation. Blue blocks are transistor-level and gray blocks are VerilogAMS models.

4.4. Transistor level design

One of the most important criteria for the current-steering dac is that it must be monotonous. Especially with very low LSB currents the chances are that the binary coded DAC is not monotonous due to mismatch between the transistors. Also glitches degrade its dynamic performance. So the current source has to change in the same direction or to preserve state for every increase/decrease in the input signal. And for these reasons we have also opted for a thermometer-coded dac. See Subsection 4.1.1.
Another important property is high output settling speed. It needs to be faster than the T/H speed of the PhADC, not to slow down the system. So, the output capacitance from the current steering must be as small as possible and not load the T&H transistor. The parasitic capacitance at the X node must be at least smaller than the source-drain parasitic capacitance of the T&H transistor. See Figure 4.43.
A precise matching of the current-steering dac current mirror transistors is not really required, because the current sources of the dac are still in the feedback loop of the IQ offset compensation scheme. The mismatch effects of these will hardly be noticeable, only that the system will converge into a different operating-point.

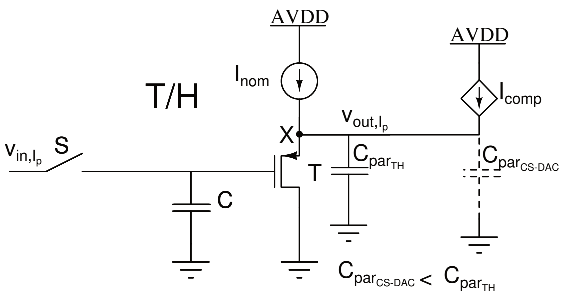


Figure 4.43: T&H transistor showing its parasitic capacitance $C_{par_{TH}}$ and the parasitic capacitance $C_{par_{CS-DAC}}$ from the current-steering dac I_{comp}.

We do not need a more enhanced current mirror source for the current-steering dac for a high output impedance to reduce channel-length modulation. The impedance seen from the source of the T&H transistor is $1/g_m$ and is therefore much lower than the drain-source impedance r_{ds} of the current mirror source. And so the effect of a high output impedance current mirror source on the channel-length modulation will be very little.

Transistor level design of the Current-Steering DAC

The reference current mirror source of the PhADC is the most obvious choice to use for the design of the current-steering dac current mirror sources $T_0 \ldots T_{2^{N-1}-2}$. In this way we create better matching with the nominal current sources I_{nom} of the T&H transistors than when a separate reference current mirror source is created. The only thing we need to do is proper scaling of the W/L dimensions. See Figure 4.44.

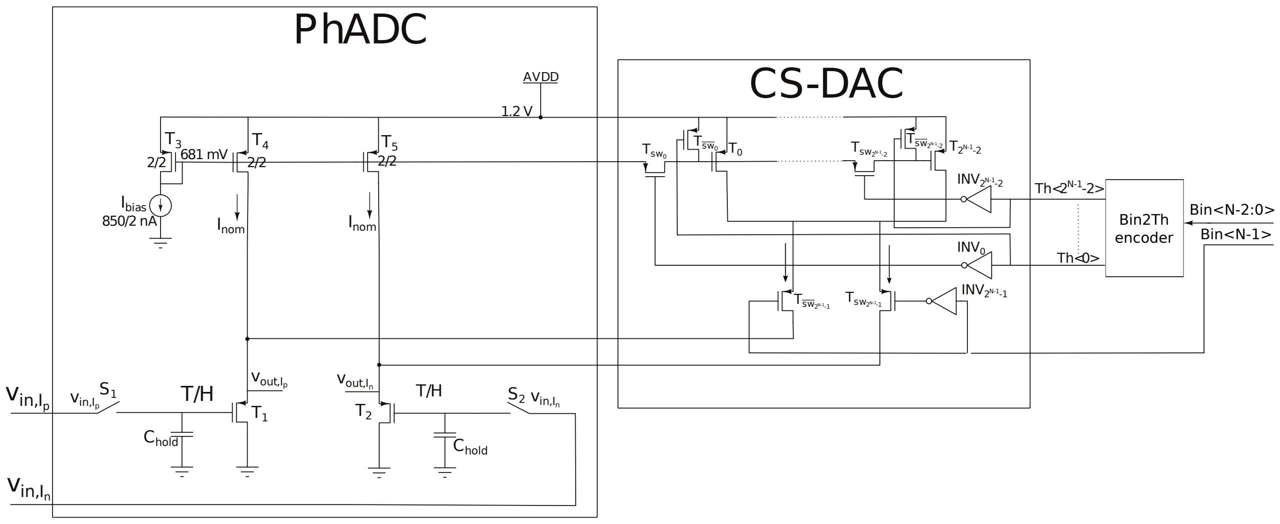


Figure 4.44: Transistor level design of the current-steering DAC (CS-DAC). The scale factor for the W/L is $1\mu m$ ($scale = 1e^{-6}$). Only the I path is shown, the Q path has exact same circuit. Only T&H and bias is shown of PhADC.

Design of the current source transistors T_0 to $T_{2^{N-1}-2}$

The reference current mirror of the PhADC I_{bias} is set at a current of 425 nA and the W/L dimensions of the reference PMOS transistor T_3 is 2/2 and the scale factor is $1\mu m$ (scale = 1.10^{-6}). The nominal current sources I_{nom} are just a copy of this current and thus the current mirror transistors T_4 and T_5 have the same dimensions, i.e., 2/2.
We need a current of 21 nA for the current-steering dac current mirror sources T_0 ... $T_{2^{N-1}-2}$ so that the I_{bias} has to be scaled with a factor of $425/21 \approx 20$. That means, keeping L=2, that we need a width of $W = 2/20 \times scale = 100\ nm$ which is not possible in the AMS CMOS H18 180nm technology. The minimum allowed width in this technology is 220 nm. This means that we can make a width of $W = 20 \times 220nm\ = 4.4\mu m$. Let's say rounded to $4\mu m$. The W/L of the reference and nominal current sources is then scaled up to 4/4. This does not make much difference for the PhADC current mirror source, the source-gate voltage and the current remain the same, but at the same time we did a correct scaling for the curren-steering dac current mirror source transistors T_0 ... $T_{2^{N-1}-2}$ which is 0.22/4.
We keep the length long which is favorable for the channel-length modulation, because the channel-length modulation factor is inversely proportional to the length of the transistor [3]. A longer length has no effect on the parasitic capacitance on the X node since the length change does affect the source-gate capacitance but not the source-drain capacitance. The width does affect the parasitic capacitance on the source-drain which is in parallel with the $T\&H$ transistor, but fortunately the width is very small, i.e., 220 nm.

Figure 4.45 shows the source-gate voltage as a function of drain current plot for the PMOS transistor with dimensions 0.22/4. According to the AMS technology data, the PFETX has a nominal threshold voltage of 405 mV which is quite similar to the value shown in the simulated graph. However, according to Figure 4.44 this transistor is biased at a source-gate voltage $V_{SG}\ = 1.2 - 0.681 = 519\ mV$ which shows a drain current of about 36 nA and is thus higher than the 21 nA we need. But it is not so bad, the current control steps of 36 nA is still small enough to be able to compensate for that 10 mV IQ offset, because a current of $4 \times I_{LSB}\ =\ 4 \times 36\ nA$ causes an increase of the T&H source voltage of 7 mV which is still smaller than the 10 mV. There is also a margin because the absolute tolerable IQ offset is approximately 16 mV instead of 10 mV and therefore should be fine. An I_{LSB} of 36 nA instead of 21 nA means that we have added a gain of 36/21 = 1.7 in the loop, hence, the system should compensate faster.

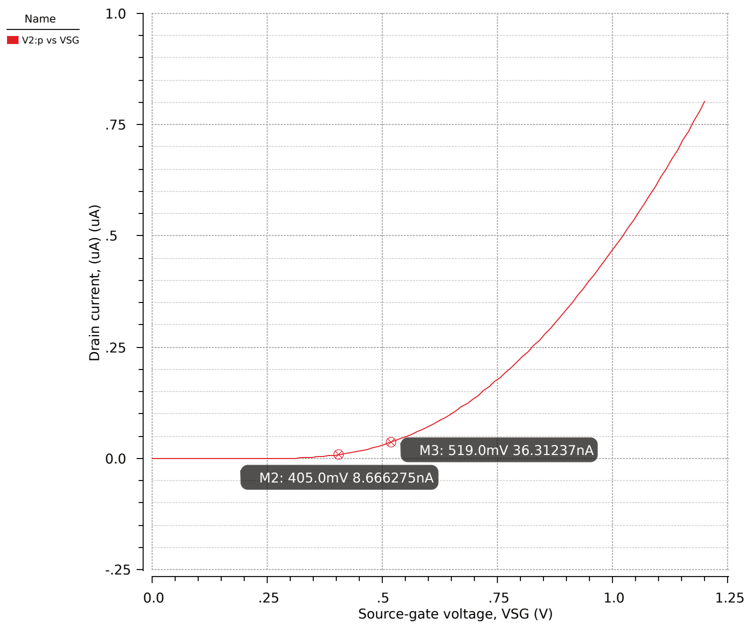


Figure 4.45: Drain current plotted against source-gate voltage with 1.2V V_{SD} for the PMOS transistor with W/L dimensions 0.22/4.

Figure 4.46 shows the source-drain voltage as a function of drain current graph at a gate-source voltage V_{SG} of 519 mV. The current shows some variation as V_{SD} changes. Knowing I_D does change with V_{SD}, we'll still simply say that the drain current is 36 nA (as an approximation).

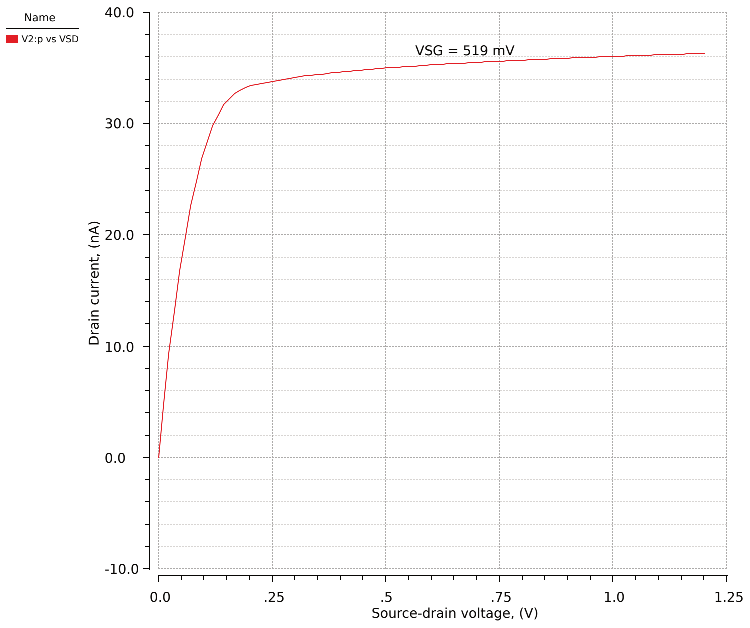


Figure 4.46: Drain current plotted against source-drain voltage with 519 mV V_{SG}.

Design of the current switch transistors T_{sw_0} to $T_{sw_{2^{N-1}-2}}$ and $T_{\overline{sw}_0}$ to $T_{\overline{sw}_{2^{N-1}-2}}$

The delays and transition times of the switching transistors, must be such that they are much faster than the system clock. The switching of the current sources, so switching on/off a compensation current, must be faster than the sample&hold time of the PhADC. This time is a quarter of the 4 MHz clock period of the PhADC, so it must be faster than 62.5 ns. There are some extremely important definitions in characterizing the time-domain characteristics of digital circuits. For instance the input rise and fall times are labeled t_r and t_f respectively.The output rise and fall times are labeled t_{LH} and t_{HL}, respectively. The delay time between the 50% points of the input and output is the propagation delay time and are labeled t_{PLH} and t_{PHL}. t_{PLH} is the time when the output changes from low to high and t_{PHL} from high to low. The definitions are shown in Figure 4.47. V_{OH} and V_{OL} are the output high and low threshold level, respectively.

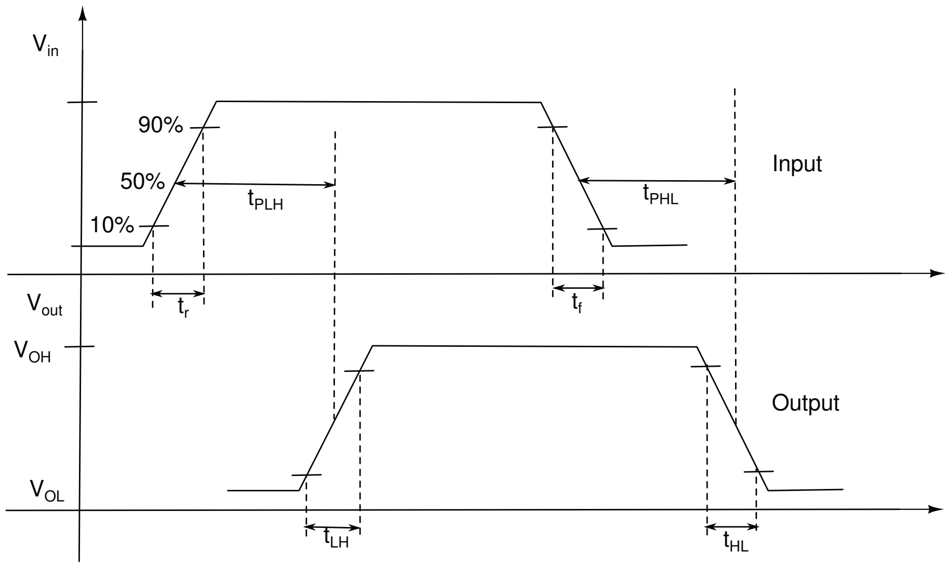


Figure 4.47: Definition of delays and transition times.

A simple digital MOSFET switch model is shown in Figure 4.48. The gate-drain capacitance which is equal to a oxide capacitance of $\frac{1}{2}C_{ox}$ has been splitted into a component from the gate to the source and from the drain to the source of value C_{ox}. The gate-source capacitance is $\frac{1}{2}C_{ox}$, hence, C_{inn} becomes $\frac{3}{2}C_{ox}$ and C_{outp} equal to C_{ox}. R_p is the effective switching resistance of the short-channel PMOS device and is defined by Equation (4.12). For the NMOS device is defined by Equation (4.13). We will soon need both for the inverter design, but first we will determine the switching times for the PMOS switch transistors T_{sw_0} to $T_{sw_{2N-1}-2}$ and $T_{\overline{sw}_0}$ to $T_{\overline{sw}_{2N-1}-2}$. For the AMS hitkit H18 180nm PFET and NFET devices the nominal on or drive current, given as a current per width, is $I_{on,p} = 260\mu A/\mu m$ and $I_{on,n} = 600\mu A/\mu m$. The nominal oxide capacitance per area C'_{ox} for this PFET and NFET is $7.5 fF/\mu m^2$ and $7.75 fF/\mu m^2$, respectively. Table shows de digital parameters to be used for hand calculations in short-channel AMS CMOS hitkit H18 180nm process.

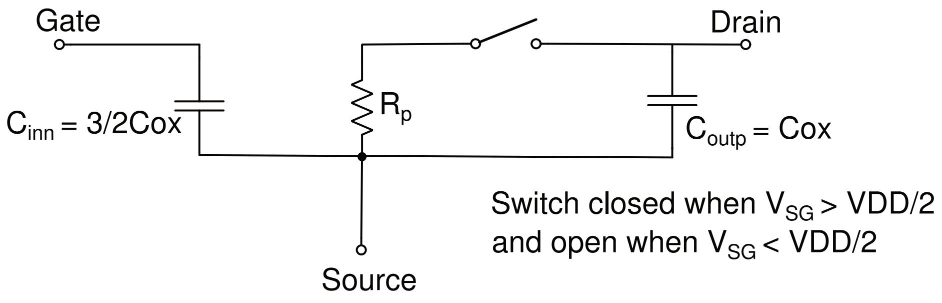


Figure 4.48: Simple digital MOSFET model.

$$R_p = \frac{VDD}{I_{D,p}} = \frac{VDD}{I_{on,p}.W.scale} = \frac{1.2V}{260\frac{\mu A}{\mu m}\cdot W\cdot scale} \rightarrow R_p \approx \frac{4.6k\cdot\mu m}{W\cdot scale} \tag{4.12}$$

$$R_n = \frac{VDD}{I_{D,n}} = \frac{VDD}{I_{on,n}.W.scale} = \frac{1.2V}{600\frac{\mu A}{\mu m}\cdot W\cdot scale} \rightarrow R_n \approx \frac{2k\cdot\mu m}{W\cdot scale} \tag{4.13}$$

Technology	R_p	R_n	**Scale factor**	$C_{ox} = C_{ox}{}' WL\cdot(scale)^2$
180 nm (short-channel)	$R_p \approx \frac{4.6k\cdot\mu m}{W\cdot scale}$	$R_n \approx \frac{2k\cdot\mu m}{W\cdot scale}$	1 μm	$C_{ox,p} = (7.5\, fF)\cdot WL$ $C_{ox,n} = (7.75\, fF)\cdot WL$

Table 4.5: Digital parameters used for hand calculations in short-channel AMS CMOS H18 180nm process.

The model circuits used to simulate the rise, fall, and delay times is shown in Figure 4.49.

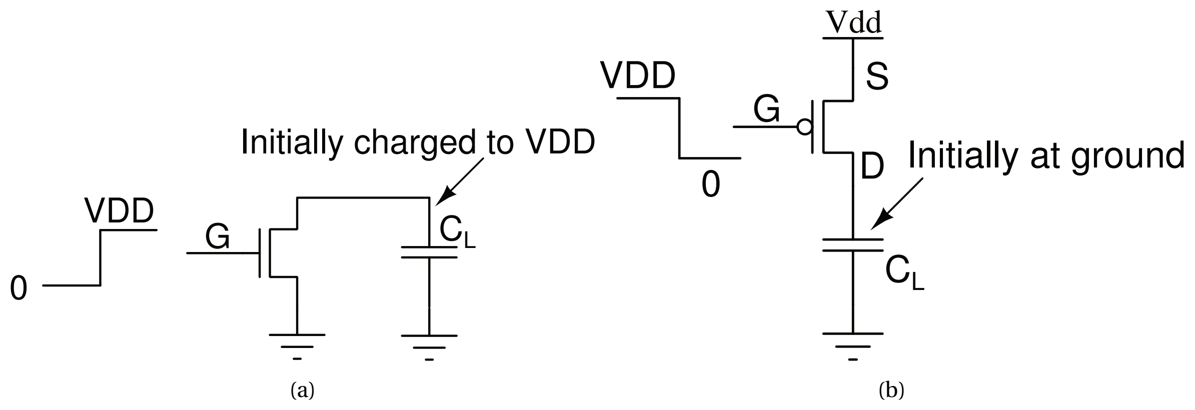


Figure 4.49: Circuits used to simulate propagation delay time t_{PLH} (a) and t_{PHL} (b).

The intrinsic switching speed of the PMOS and NMOS devices is defined as

$$\tau_p = \frac{VDD}{I_{on,p}.W.scale} \cdot C'_{ox} WL \cdot (scale)^2 = \frac{VDD \cdot C'_{ox} \cdot L \cdot scale}{I_{on,p}} \tag{4.14}$$

$$\tau_n = \frac{VDD}{I_{on,n}.W.scale} \cdot C'_{ox} WL \cdot (scale)^2 = \frac{VDD \cdot C'_{ox} \cdot L \cdot scale}{I_{on,n}} \tag{4.15}$$

In [3], the delay time of a simple RC circuit is given by

$$t_{delay} = 0.7RC \tag{4.16}$$

and the rise or fall time is given by

$$t_{rise} = 2.2RC \tag{4.17}$$

So, for our simple digital model, we will assume that the propagation delay time $tPHL$ and t_{PLH} is given by

$$t_{PLH} \approx 0.7 \cdot R_p \cdot C_{tot} \text{ and } t_{PHL} \approx 0.7 \cdot R_n \cdot C_{tot} \tag{4.18}$$

and the output rise and fall times are given by

$$t_{LH} \approx 2.2 \cdot R_p \cdot C_{tot} \text{ and } t_{HL} \approx 2.2 \cdot R_n \cdot C_{tot} \tag{4.19}$$

where C_{tot} is the total capacitance, including the load capacitance C_L, from the drain of the MOSFET to ground.

Figure 4.50 shows the MOS models used for determining the switching times. The load capacitance is much larger than the output capacitances of the switches due to the heavy load by the current-steering dac current sources T_0 ... $T_{2^{N-1}-2}$. We have determined from a good estimate that the load capacitance is about 1 pF worst case.

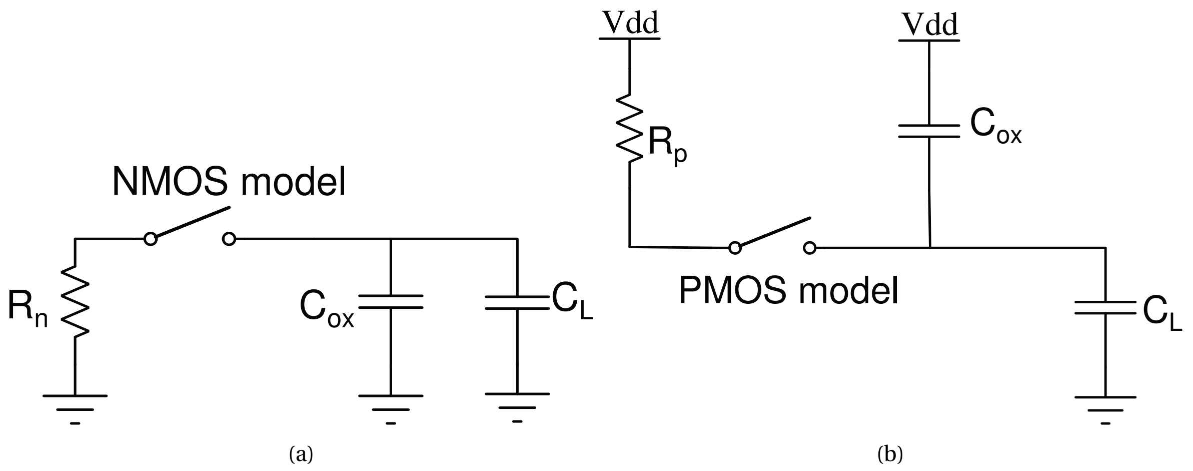


Figure 4.50: Model used to determine switching times for NMOS (a) and for PMOS (b).

Tabel 4.6 shows the timing results for NMOS and PMOS devices with W/L dimensions 1/0.18.

Device	**Effective switching resistance**	**intrinsic switching speed**	**propagation delay**	**output rise and fall time**
PMOS 1/0.18	$R_p = 4.6k$	$\tau_p = 6.2\ ps$	$t_{PLH} = 3.22\ ns$	$t_{LH} = 10\ ns$
NMOS 1/0.18	$R_n = 2k$	$\tau_n = 2.8\ ps$	$t_{PHL} = 1.4\ ns$	$t_{HL} = 4.4\ ns$

Table 4.6: Hand calculated digital parameters for short-channel AMS CMOS H18 180nm process assuming a load capacitance C_L=1pF.

Figure 4.51 shows a simulation of the circuits shown in 4.49.

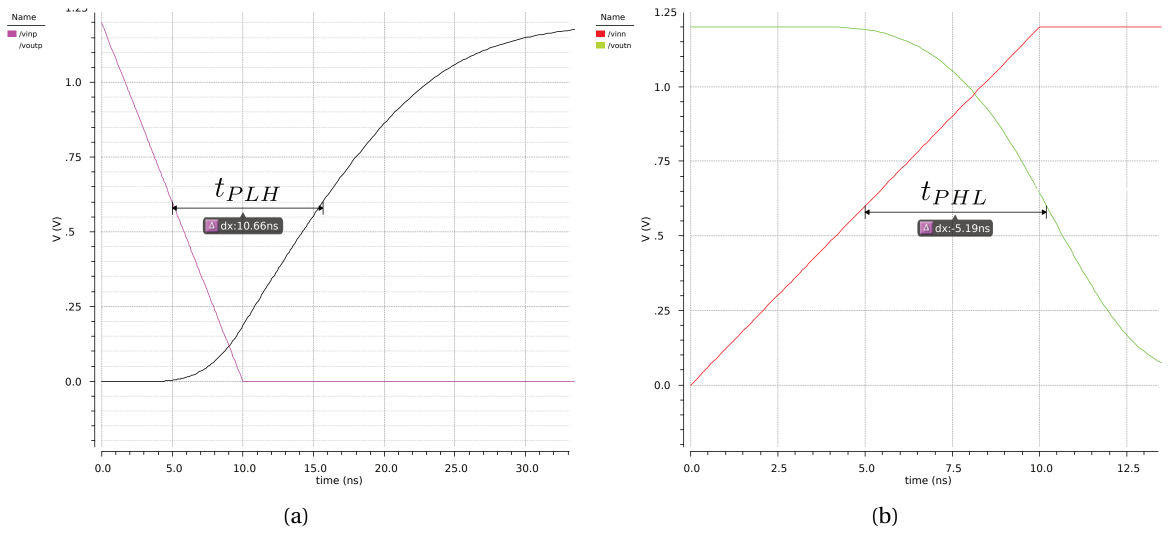


Figure 4.51: fig:Simulation of the circuits shown in Figure 4.49.

Notice the difference between calculated and simulated values. These models for hand calculations **do not give exact results**. The models are useful for determining approximate delay and transition times, usually within a factor of two to three. They can reveal the location of a speed limitation in a circuit.

Design of the inverters INV_0 to $INV_{2^{N-1}-1}$

Again, the switching delay times of the inverter must be faster than the system clock speed and the sample and hold time of the PhADC. The ideal switching point voltage, V_{SP}, of the inverter shown in Figure 4.53(a), is VDD/2. See Figure 4.52. However the switching voltage of a practical inverter is different from the ideal value of VDD/2. See dashed line in Figure. We can try to make the effective switching resistance of both the PMOS and NMOS equal, the switching point voltage would then be close to ideal. By changing the widths of the devices, we can adjust the switching point voltage. In other words, for a short-channel CMOS process we can use

$$V_{SP} = VDD \cdot \frac{R_n}{R_n + R_p} \tag{4.20}$$

Using Equation (4.12) and (4.13) in Equation (4.20) this would results in a $\frac{W_n}{W_p}$ ratio of $\frac{10}{23}$. So, if we choose a minimum length for W_n of 0.22, W_p becomes 0.506. But the AMS hitkit forces W_p to be 0.51 instead, de ratio for the inverter becomes 0.51/0.22. We shall use these values for further calculations.

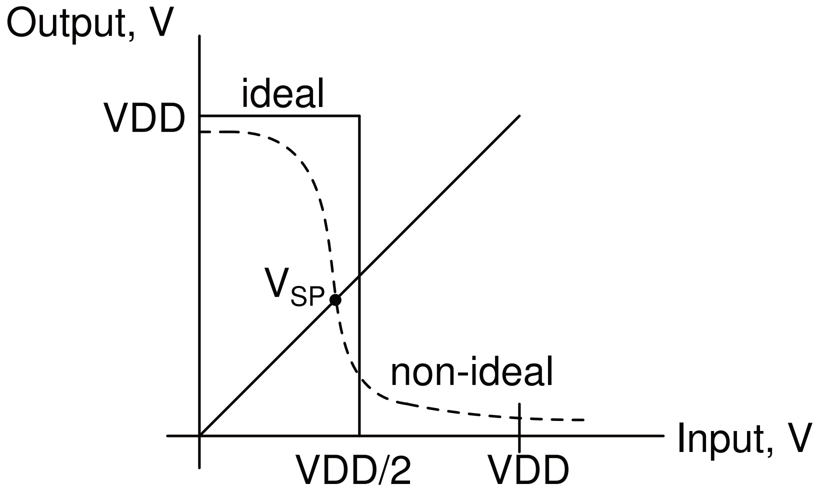


Figure 4.52: Ideal and non-ideal inverter voltage transfer function.

The switching behavior of the inverter can be examined by the capacitance and resistance associated with the inverter. Consider the inverter shown in Figure 4.53(a) with its equivalent digital model shown in Figure 4.53(b). Although the model is shown with both switches open, in practice one of the switches is closed, keeping the output connected to VDD or ground. The effective input capacitance of the inverter is

$$C_{in} = \frac{3}{2}(C_{ox1} + C_{ox2}) = C_{inn} + C_{inp} \tag{4.21}$$

The effective output capacitance of the inverter is simply

$$C_{out} = C_{ox1} + C_{ox2} = C_{outn} + C_{outp} \tag{4.22}$$

The intrinsic propagation delays of the inverter are

$$t_{PLH} = 0.7 \cdot R_{p2} \cdot C_{out} \text{ and } t_{PHL} = 0.7 \cdot R_{n1} \cdot C_{out} \tag{4.23}$$

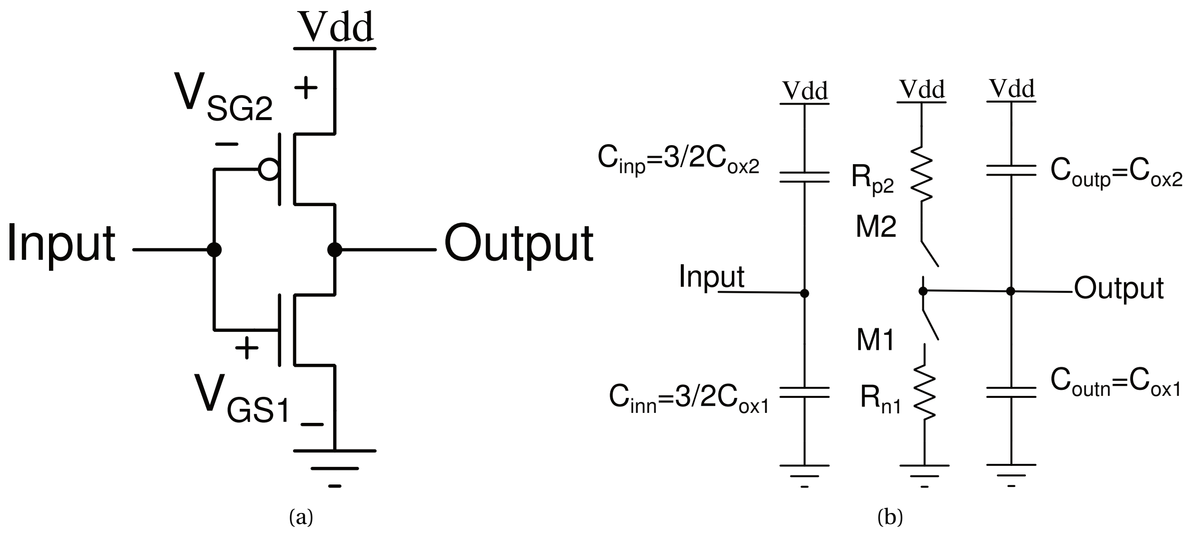


Figure 4.53: The CMOS inverter switching characteristics using the digital model.

The propagation delays for an inverter driving a capacitive load are

$$t_{PLH} = 0.7 \cdot R_{p2} \cdot C_{tot} = 0.7 \cdot R_{p2} \cdot (C_{out} + C_{load}) \tag{4.24}$$

$$t_{PHL} = 0.7 \cdot R_{n1} \cdot C_{tot} = 0.7 \cdot R_{n1} \cdot (C_{out} + C_{load}) \tag{4.25}$$

Table 4.7 shows the hand calculated values of the inverter digital parameters.

Device	**effective input capacitance** C_{in}	**effective output capacitance** C_{out}	**intrinsic delay** t_{PLH}	**intrinsic delay** t_{PHL}	**delay driving a load** t_{PLH}	**delay driving a load** t_{PHL}
inverter 0.51/0.22	1.49 fF	1 fF	6.31 ps	6.36 ps	6.31 ns	6.36 ns

Table 4.7: Hand calculated digital parameters for short channel inverter assuming a capacitive load C_L of 1 pF.

Figure 4.54 shows a simulation of the circuit shown in Figure 4.53(a) for PMOS dimensions 0.51/0.18 and NMOS dimensions 0.22/0.18.

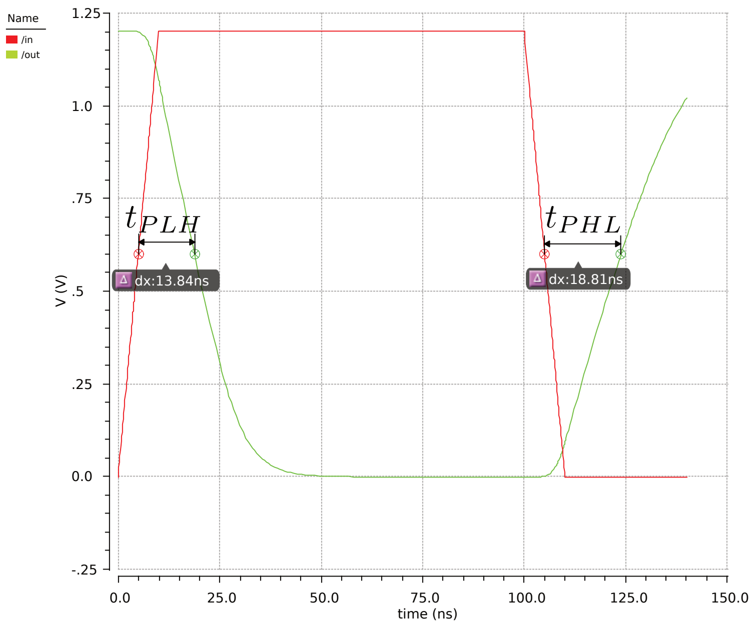


Figure 4.54: Simulation of the circuit shown in Figure 4.53(a).

Figure 4.55 shows the current-steering dac with the W/L dimensions set for each transistor and W_p/W_n dimensions for inverter using a scale factor of 1μm.

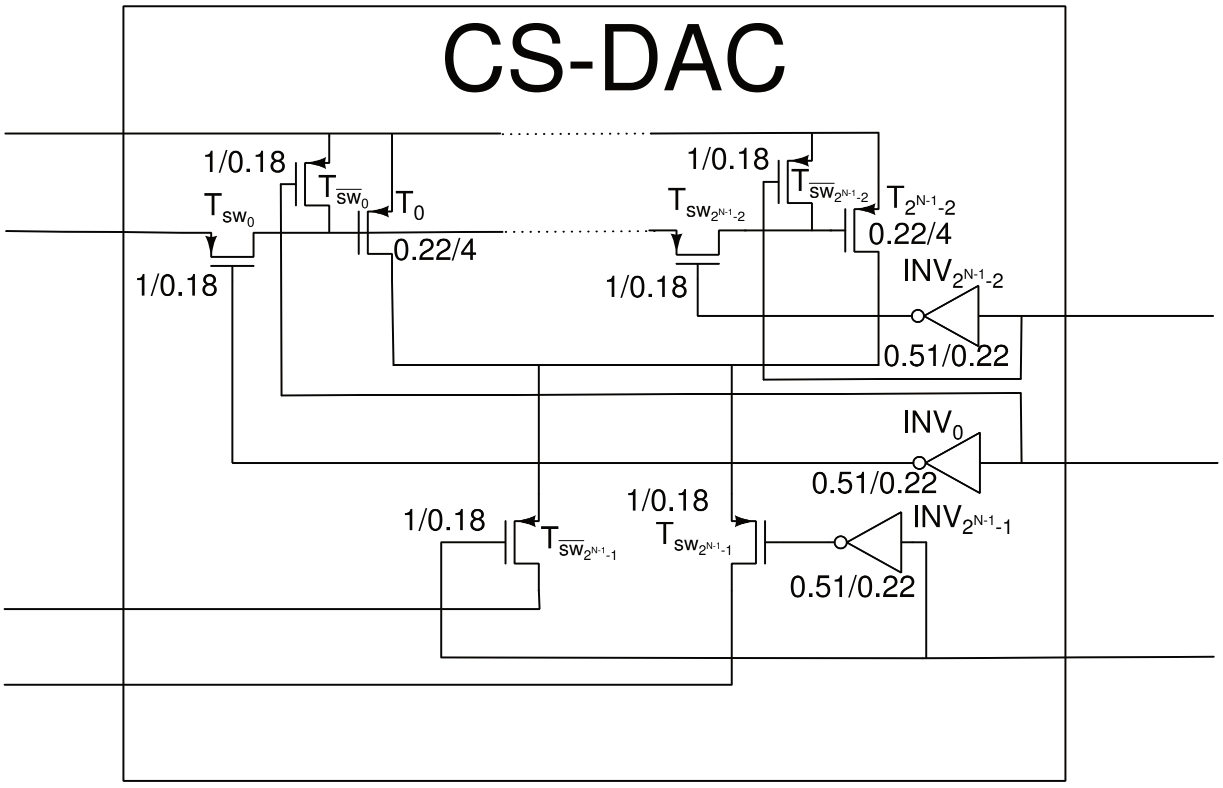


Figure 4.55: The current-steering dac with the W/L dimensions set using a scale factor of 1μm.

5
Conclusions and Recommendations

5.1. Conclusions

On the basis of the results obtained in the analysis of the basic system and design solutions for IQ offset and gain mismatch compensation for Phase-Domain ADCs (PhADC), the following conclusions are drawn:

1. The conversion of IQ magnitude to phase is not linear (i.e., arctan function) and so an IQ offset and gain mismatch will lead to non-linear phase distortion.
2. An IQ offset will shift the PhADC output phase circular vector trajectory to the left, right, up or down with respect to the origin and lead to phase errors and, hence, an increase in bit-error-rate (BER). There is a phase vector distribution imbalance between the left and right or top and bottom IQ complex half plane.
3. An IQ gain mismatch will cause the PhADC output phase circular vector trajectory to become elliptical and also leads to phase errors and an increase in BER. There is a phase vector distribution imbalance between region (1) + (3) and region (2) + (4)[*] of the IQ complex plane.
4. The PhADC offers the phase digitally and so it is very easy to realize a precise phase vector distribution density analysis process in the digital domain. The phase vector distribution imbalance can be determined by detecting the position of the phase samples in the IQ complex plane and the balance between the phase sample distribution in the regions to be compared.
5. A phase vector distribution imbalance detection algorithm has been developed and a system that detects the size and sign of the IQ offset and gain mismatch using this phase vector distribution imbalance and compensates for that.
6. Due to the simplicity of the control algorithm, it can be mapped onto electronic circuitry that is small and consists of some analog electronics and digital combinatorial logic without needing a μC or memory. As a consequence, it takes up little chip area and has a low power consumption.
7. For an ideal communication system using $\pi/4$ DQPSK modulation that requires a bit-error rate up to 10^{-5} a maximum IQ offset and gain mismatch of 28 mV and ± 25dB can be tolerated, respectively.
8. Beyond this maximum the BER shows a sharp transition from zero towards a BER of one, it is a threshold.

[*]The regions (1), (2), (3) and (4) are defined in Chapter 3

9. For an ideal communication system using $\pi/4$ DQPSK modulation that requires a bit-error rate up to 10^{-5} and a maximum added Gaussian white noise (AWGN) in the channel of E_b/N_o[¶]=12 dB a maximum IQ offset and gain mismatch of 1 mV and ± 0.35 dB can be tolerated, respectively.

10. For an ideal communication system using $\pi/4$ DQPSK modulation that requires a bit-error rate up to 10^{-5} and a maximum added Gaussian white noise (AWGN) in the channel of E_b/N_o=21 dB a maximum IQ offset and gain mismatch of 22 mV and ± 6 dB can be tolerated, respectively.

11. For a practical PhADC receiver [10] with a channel noise of $SNR = 21$ dB a maximum IQ offset and gain mismatch of 14 mV and ± 3 dB can be tolerated, respectively.

12. Signal-to-noise power ratio (SNR) is not the same as E_b/N_o and also a less understood ratio which is often confused with SNR. The relationship between SNR and E_B/N_o also depends on the modulation parameters. For a $\pi/4$ DQPSK modulation the relationship $E_B/N_o = SNR + 0.97$ dB applies.

5.2. Contributions

Concerning the analysis of the basic system and design solutions for IQ offset and gain mismatch compensation for Phase-Domain ADCs (PhADC), the following contributions have been made:

1. An IQ offset and gain mismatch detection technique for PhADCs has been developed and verified. Using the proposed technique, the IQ offset and gain mismatch of a receiver using a PhADC can be estimated by detecting the imbalance among the four phase quadrants after the analog-to-phase conversion. A feedback path is therefore constructed between the PhADC output and an offset/mismatch compensation interface to the receiver. The closed loop helps the compensation interface to settle to the proper compensation values of the offset and mismatch. Incorporating this cancellation loop in a receiver system model improves its sensitivity.

2. It is well known in practice that the mismatch and the offset can be detected and calibrated if necessary before amplitude is converted to phase for the IQ ADC receiver, but for the PhADC receiver this direct mismatch and offset detection cannot be employed due to the absence of amplitude information. So this approach is unique in the sense that we use phase information rather than amplitude information for detection and compensation. Hence, we have created a direct mismatch and offset detection technique for receivers using a PhADC for FSK and PSK modulated signals.

3. We have developed a simple control algorithm principle for IQ offset and gain mismatch detection that can also be used for other FSK and PSK modulation techniques than only $\pi/4$ DQPSK and can be implemented by means of simple combinatorial logic. The resulting system system is small, simple and uses little energy in contrast to an implementation with a μC and memory.

4. A detailed analysis has been given of the relationship between signal-to-noise-ration (SNR) with bit-energy-to-noise density (Eb/N_o) and symbol-energy-to-noise density (E_s/N_o) from the digital domain and provided evidence that these are not the same quantities and, moreover, the relationship is also dependent on modulation parameters. We used this information to show in simulations the effect of IQ offset and gain mismatch, with and without channel and quantization noise, on the PhADC receiver and provided performance metrics than can be used for design purposes.

5. We have provided a method to convert a true analog filter to a digital one for discrete modeling purposes to mimic the behavior of the PhADC receiver front-end described in[10].

6. We have given a real circuit implementation of the IQ offset and gain mismatch compensation system for the PhADC receiver described in [10] and shown that it really works.

[¶] E_b/N_o = bit-energy-to-noise density. See Chapter 2

5.3. Recommendations

Based on the assumptions stated initially and observations made during the investigation, the following recommendations are proposed for further study:

1. To describe a linear model of the IQ offset and gain mismatch compensation loop so that the system can be tested and optimized in terms of response, speed, noise, linearity, stability, etc. The system tries to compensate for IQ offset and gain mismatch but essentially what is being controlled is phase. So the input/output of this model is neither offset nor gain mismatch, but phase.
   PhADC can be modeled as a delay. Indeed, the PhADC must first sample IQ and then generate the phase. The phase quantization noise models the phase quantization error from the PhADC and is dependent on the resolution of the PhADC. The *Two Quadrant Detector* and the *Two Regions Detector* of the sample-based system can be modeled as a gain block $\partial V/\partial\varphi$ with a bigger phase quantization noise than the one of the PhADC. The phase has to be converted to an output voltage for the integrator. The sample-based linear model is the same as the frame-based model except that in the case of frame-based linear model the *Two Regions Detector* and the *Two Quadrant Detector* have to be replaced by the *IQ Gain Mismatch Detector* and the *IQ Offset Detector* Detector respectively. These blocks can be modeled as a low pass filter. The integrator can be modeled as $1/s$. Then there's a constant gain block and for the IQ offset compensation system there's a gain block $\partial\varphi/\partial V$. This block has to convert voltage back into phase to correct for phase errors and this block is highly non-linear, due to the non-linear amplitude-to-phase conversion. This gain has to do with the g_m of the track-and-hold transistor of the PhADC, because the current from the current-steering DAC is converted into a voltage. On the other hand, the current-steering DAC itself can also be modeled with a gain and some quantization noise.

2. To adapt the IQ offset and gain mismatch compensation system to accommodate a communication protocol such as *Bluetooth Low Energy*. One could choose a certain control strategy such as a one time calibration or a calibration from time to time in which training data is used to compensate the system before the actual communication is started. The compensation values can be stored digitally and the control loop can be interrupted. This is all possible thanks to the digitization of the controlling part of the feedback loop.
   But on the other hand, one can also opt for continuous control. An example is the use of a so-called *preamble*, this is the piece of training data preceding the information data. According to the *IEEE802.15.6* standard this is 50 μs and for a PhADC with a sample rate of 1 Ms/s it means 50 symbols of training data. Whether this is adequate enough for a good compensation must be investigated. But we can also see that the settling time of the compensation system must in any case be faster than 50 μs otherwise it can never calibrate the system before the communication starts. Another possibility would be to control between the data packets. In the so-called *INTER FRAMESPACE* (T_IFS). For *Bluetooth Low Energy* this is 150 μs. The reason is that during these time slots the offset can change due to switching to another channel.
   But perhaps it is not necessary to control between the frames at all. In communication, a training data is always send that determines the synchronization and only then the header data is recognized. So a lot of data bits precede the current communication.

3. To investigate the effects of interference. This makes the system for IQ offset and gain mismatch compensation much more robust. Making the system analysis more realistic improves the performance of the actually realized system. One could investigate three situations;

   - Adjacent channel effects.
   - Alternate channel effects.
   - Co-channel (i.e., in channel) effects.

   The question we have to answer is how much interference we can tolerate such that the BER is still acceptable (i.e., $\leq 10^{-5}$ for $\pi/4$ DQPSK). The three types of interferences can occur simultaneously but the cumulating effect has no significant influence on the BER, because only the tail of the interfering channel comes into the band of information. The greatest interference comes from neighboring signals coming into the information band. So we can add the above interfering channels one at a time and boost their power and see how much of this signal is tolerated so that the BER is still within acceptable limit. It requires actually five simulations to make a so called *blocking mask*. We add only one channel and see how much interference power can be tolerated for the;

- lower alternate channel.
- lower adjacent channel.
- in band channel.
- upper adjacent channel.
- upper alternate channel.

The IEEE802.15.6 communication standard states how much alternate, adjacent and co-channel interference a compliant transmitter device can tolerate. These values can be used as a reference.

4. It is highly recommended to implement the IQ offset and gain mismatch compensation system fully on chip together with the Phase-Domain ADC receiver. Since the compensation is not adapted to any communication protocol, we need to obtain our own training data to test and measure the system. We can read the output phase directly in digital form and then carry out operations on it to analyze the obtained performance for example in Matlab. The BER can be measured and data can also be mapped on a constellation diagram. In case picture of the constellation is not perfectly square or circular, it will inform us about the type of distortions and the amount experienced by the signal.

A

Line Coding and Decoding

Line coding is a part of source coding. Before the data is send to the modulator, we use certain signal mode in certain application. The consederations of selecting the digital signal modes to carry the binary data are:

1. Types of modulation
2. Types of demodulation
3. The limitation of bandwidth
4. Types of receiver

Line coding can be divided into two types, which are return-to-zero (RZ) and nonreturn-to-zero (NRZ). RZ line coding denotes for a single bit time (normally is half of a single bit time), the waveform will return to 0 V between pulses. NRZ line coding denotes for a single bit time, the waveform will not return to 0 V. As a result of the characteristics of signal, line coding also can be divided into two types, which are unipolar signal and bipolar signal. Unipolar signal denotes that the signal amplitude varies between a positive voltage level which are +V and 0 V. The only difference between bipolar signal and unipolar signal is the signal amplitude varies between a positive and a negative voltage level which are +V and -V. In the next section an example will be given of line encoder and decoder that is used for FSK and QPSK digital modulation techniques.

A.1. Bipolar nonreturn-to-zero (Bip-NRZ) Line Code Encoder

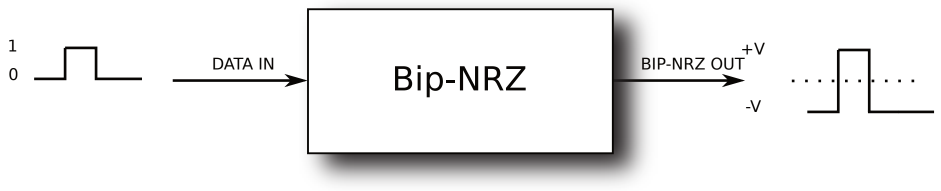


Figure A.1: Symbol of a bipolar nonreturn-to-zero (Bip-NRZ) Line Code Encoder.

Figure A.1 shows an example of a so called a bipolar nonreturn-to-zero data encoder device. When the data bit of Bip-NRZ is ’1’ or ’0’, the signal amplitude will be a positive or a negative voltage level. As for bit time, no matter the data bit is ’1’ or ’0’, the voltage level remain same.

Example:

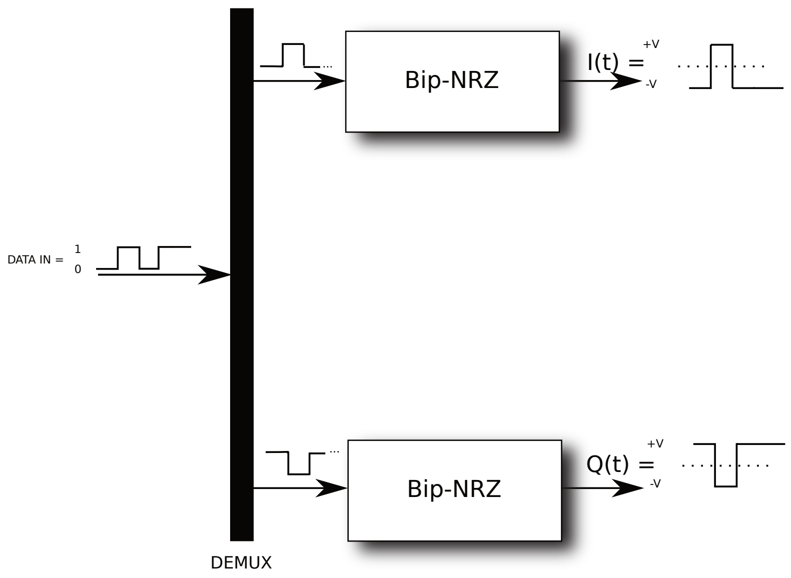


Figure A.2: Example of a bipolar nonreturn-to-zero line code encoding for IQ signals.

A.2. Bipolar return-to-zero (Bip-RZ) Line Code Decoder

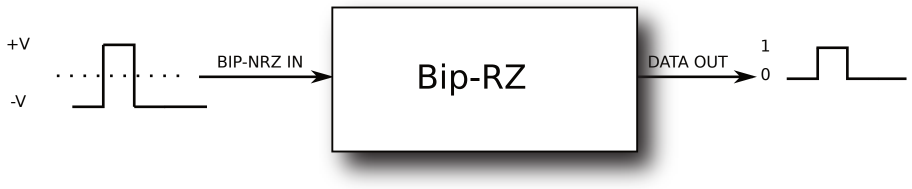


Figure A.3: Symbol of a bipolar return-to-zero (Bip-RZ) Line Code Decoder.

Line decoding is the reverse of line coding. Figure A.3 shows a bipolar return-to-zero (Bip-RZ) device. The signal amplitude of Bip-NRZ is either positive voltage or negative voltage level. Therefore, the Bip-RZ will decode the positive voltage to a data bit '1' and a negative voltage to a data bit '0'.

Example:

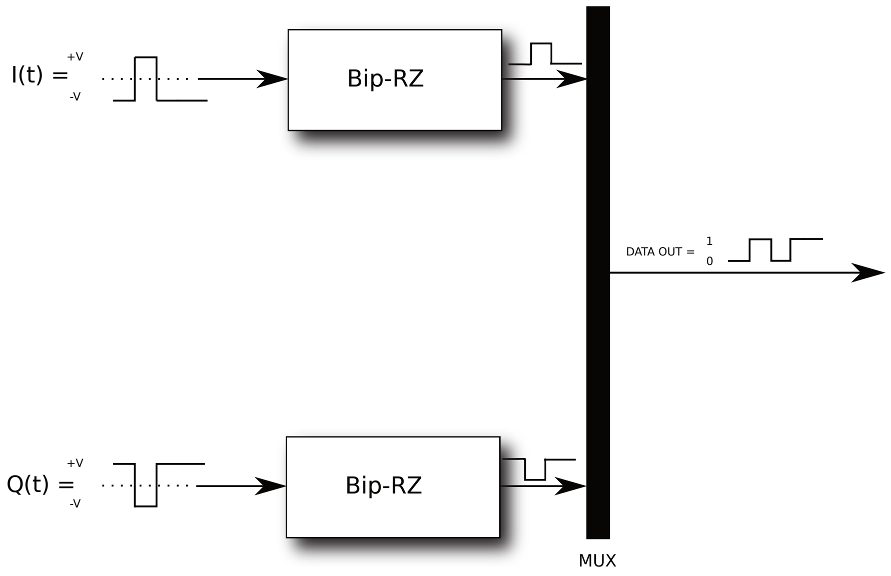


Figure A.4: Example of a bipolar return-to-zero line code decoding for IQ signals.

B
ASK modulation

An ASK signal can be expressed as

$$ASK(t) = a_n cos\omega t \tag{B.1}$$

for rectangular baseband pulses. To obtain ASK constellation, we assume bit a_n is a pulse with height of one or zero. Here we don't need to encode the source data because the binary baseband data can directly be used to modulate the carrier. We say this signal has "one basis function", and is simply defined by the possible values of the coefficient a_n. The dimensionality of a modulation is defined by the number of basis function. That makes ASK a one dimensional signal as shown in Figure B.1.

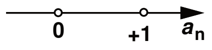


Figure B.1: ASK signal constellation.

In "amplitude shift keying" (ASK), we change the amplitude of the signal in response to information and all else is kept fixed. Bit '1' is transmitted by a signal of one particular amplitude. To transmit '0', we keep the frequency constant and change the amplitude. So a binary ASK signal is toggling between full and zero amplitudes and is also known as "on-off keying" (OOK). Equation B.2. shows a definition of the ASK signal.

$$ASK(t) = \begin{cases} sin\omega t & for \quad bit \quad 1 \quad (a_n = 1) \\ 0 & for \quad bit \quad 0 \quad (a_n = 0) \end{cases} \tag{B.2}$$

Figure B.2 shows a waveform of an ASK signal.

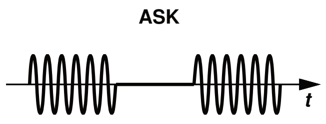


Figure B.2: ASK waveform.

Figure B.3 shows an ASK modulator and demodulator in its simplest form.

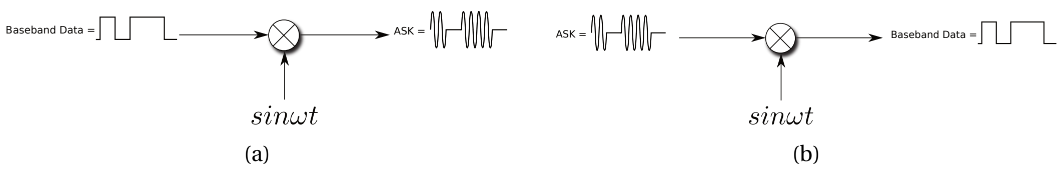


Figure B.3: ASK Modulator (a), ASK Demodulator (b).

C

FSK Modulation and Demodulation

An FSK signal can be expressed as

$$FSK(t) = a_1 cos\omega_1 t + a_2 cos\omega_2 t \tag{C.1}$$

for rectangular baseband pulses. To obtain FSK constellation, we assume bits a_1 and a_2 are pulses with a height of one or zero. Since we have only one signal source available to make a selection between carrier with frequency ω_1 and carrier with frequency ω_2 we need to encode the data to make a distinguish between the selections. Hence, the source data can also be encoded by a line encoder to obtain these signal levels as explained in appendix A. We say this signal has "two basis functions," and is simply defined by the possible values of the coefficients a_1 and a_2. The dimensionality of a modulation is defined by the number of basis function. That makes FPSK a two dimensional signal as shown in Figure C.1.

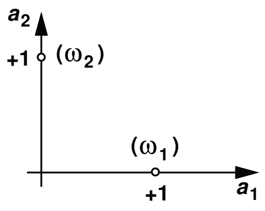


Figure C.1: FSK signal constellation.

In "frequency shift keying" (FSK), we change the frequency in response to information, one particular frequency for a '1' and other for a '0' as shown below in Equation (C.2). How it is done, can be seen in Figure C.3 for the FSK modulator example.

In the example below, frequency f_1 for bit 1 is higher than f_2 used for the 0 bit. See Figure C.2 for the FSK modulated carrier signal.

$$FSK(t) = \begin{cases} sin(2\pi f_1 t) & for\ bit\ 1\ (a_1 = 1\ a_2 = 0) \\ sin(2\pi f_2 t) & for\ bit\ 0\ (a_1 = 0\ a_2 = 1) \end{cases} \tag{C.2}$$

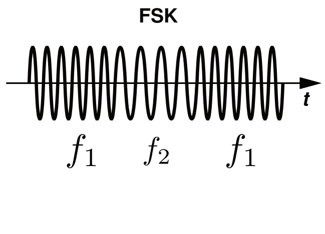
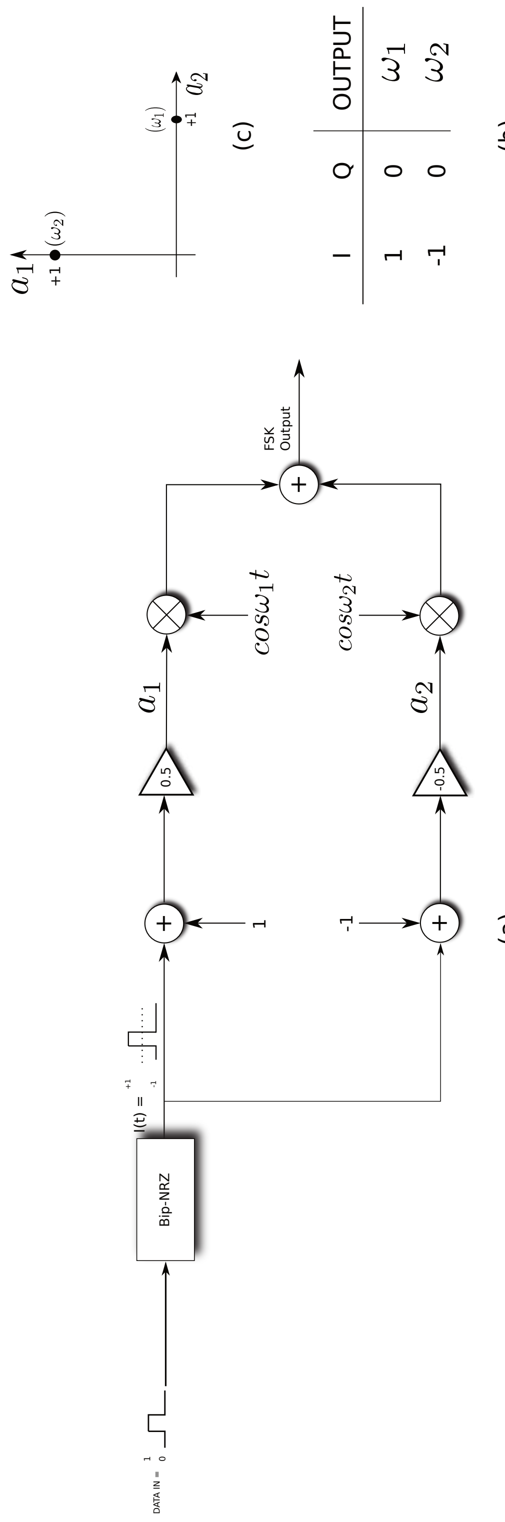


Figure C.2: FSK waveform.

C.1. FSK modulator

Figure C.3 gives an example of an FSK modulator with the IQ input to output relation and constellation of an FSK signal.



I	Q	OUTPUT
1	0	ω_1
-1	0	ω_2

Figure C.3: FSK modulator (a) IQ input to output relation (b) constellation of FSK signal (c).

C.2. FSK demodulator

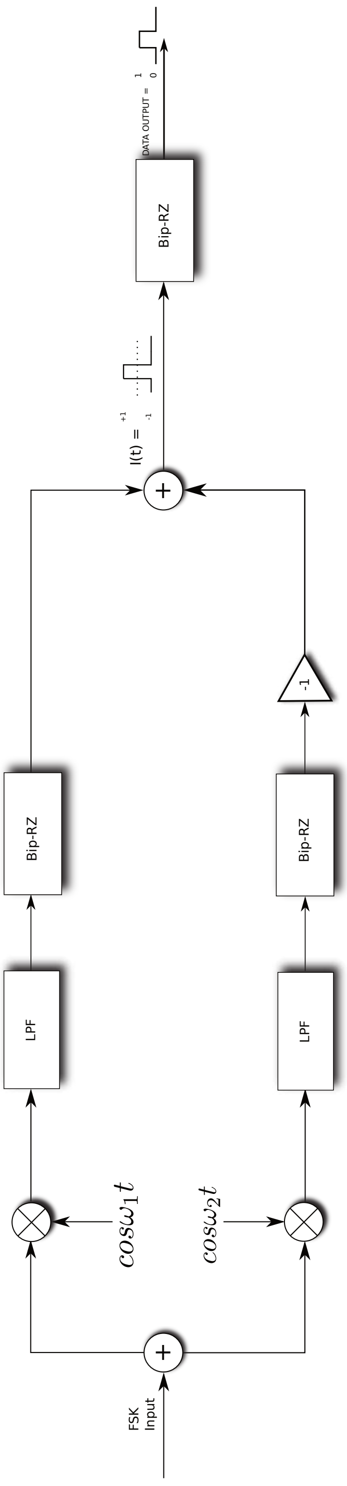


Figure C.4: FSK demodulator.

D

QPSK Modulation and Demodulation

An PSK signal can be expressed as

$$PSK(t) = a_n sin\omega t \;\; a_n = \pm 1 \tag{D.1}$$

for rectangular baseband pulses. To obtain PSK constellation, we assume bit a_n is a pulse with a height of ± 1. And as stated before in appendix B, the source data has to be encoded to obtain these signal levels using a line encoder as explained in appendix A. We say this signal has "one basis function", and is simply defined by the possible values of the coefficient a_n. The dimensionality of a modulation is defined by the number of basis function. That makes a one dimensional signal as shown in Figure D.1.

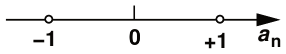


Figure D.1: PSK signal constellation.

In "phase shift keying (PSK)", we change the phase of the sinusoidal carrier to indicate information, see Equation D.2. Phase in this context is the starting angle at which the sinusoidal starts. To transmit 0, we shift the phase of the sinusoidal by 180°. Phase shift represents the change in the state of the information in this case. See Figure D.2 for the PSK modulated carrier signal.

$$PSK(t) = \begin{cases} sin(2\pi f t) & for \quad bit \quad 1 \quad (a_n = 1) \\ sin(2\pi f t + \pi) & for \quad bit \quad 0 \quad (a_n = -1) \end{cases} \tag{D.2}$$

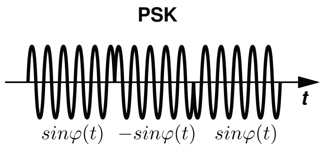
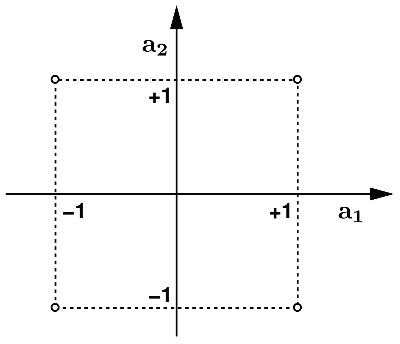


Figure D.2: PSK waveform.

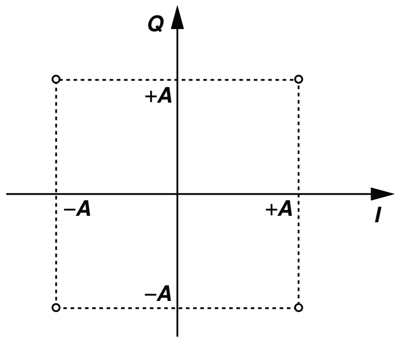
In order to further reduce the bandwidth, i.e., pack more bits in a symbol, "quadrature modulation," more specifically, "quadrature PSK" (QPSK) modulation can be performed. An QPSK signal can be expressed as

$$QPSK(t) = a_1 Acos\omega t + a_2 Asin\omega t \tag{D.3}$$

for rectangular baseband pulses. To obtain QPSK constellation, we assume bits a_1 and a_2 are pulses with height of ± 1. And here also the source data has to be encoded to obtain these signal levels using a line encoder as explained in appendix A. This signal has "two basis functions", and simply defined by the possible values of the coefficients a_1 and a_2. The dimensionality of a modulation is defined by the number of basis functions. Not because it sends two bits per symbol, but because it uses **two** independent signals (a sine and a cosine) to create the symbols. That makes QPSK a two dimensional signal as shown in Figure D.3.



(a)



(b)

Figure D.3: QPSK signal constellation in terms of a_1 and a_2 (a), and quadrature phases of carrier (b).

By varying the phase of each of these carriers we can send two bits per each signal, see Equation D.4.

$$QPSK(t) = \begin{cases} cos\omega t + sin\omega t & for \quad bits \quad 11 \quad (a_1 = 1 \quad a_2 = 1) \\ -cos\omega t + sin\omega t & for \quad bits \quad 01 \quad (a_1 = -1 \quad a_2 = 1) \\ -cos\omega t - sin\omega t & for \quad bits \quad 00 \quad (a_1 = -1 \quad a_2 = -1) \\ cos\omega t - sin\omega t & for \quad bits \quad 10 \quad (a_1 = 1 \quad a_2 = -1) \end{cases} \tag{D.4}$$

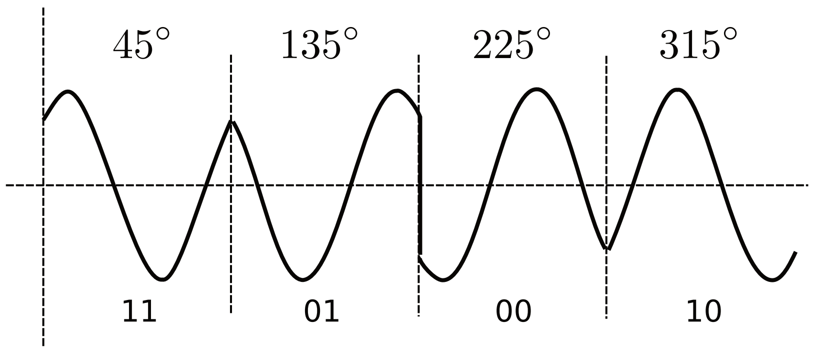
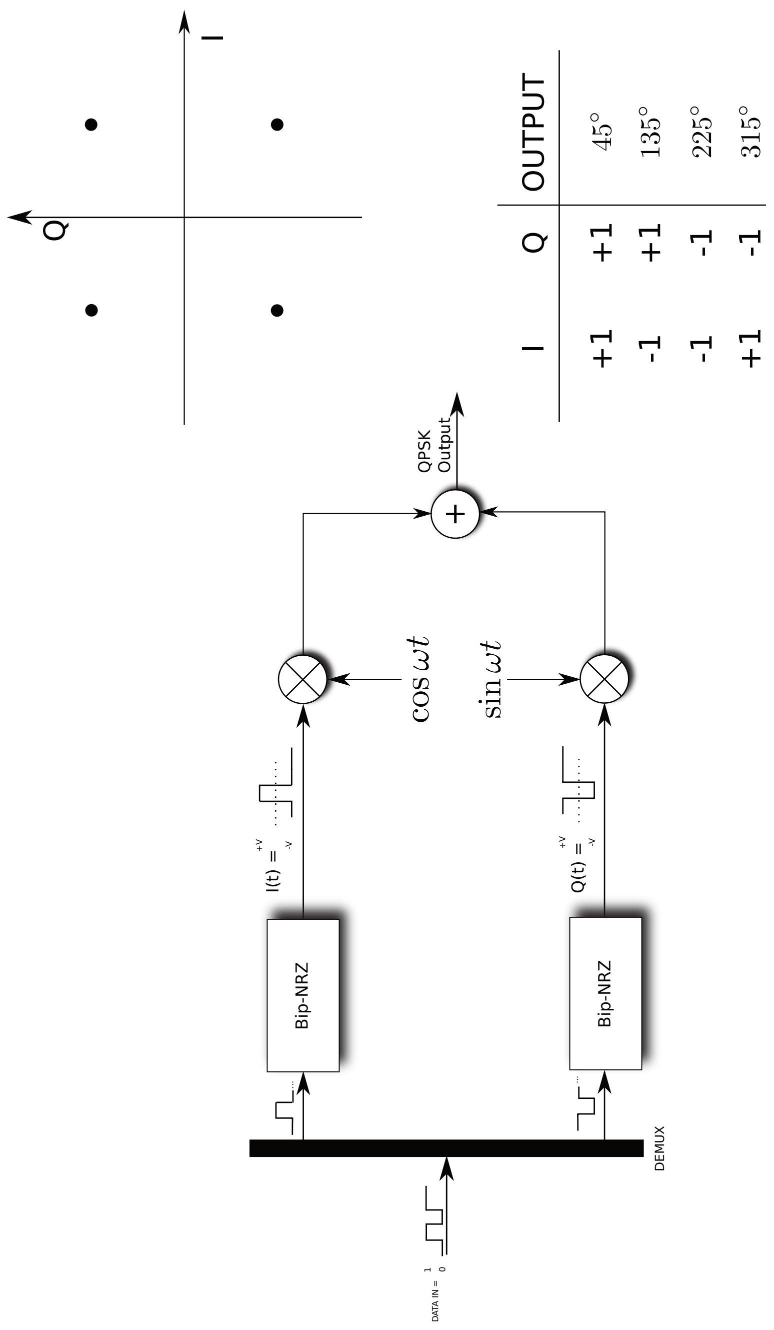


Figure D.4: QPSK waveform.

D.1. QPSK modulator



I	Q	OUTPUT
+1	+1	45°
-1	+1	135°
-1	-1	225°
+1	-1	315°

Figure D.5: QPSK modulator (a) IQ input to output relation (b) constellation of QPSK signal (c).

D.2. QPSK Demodulator

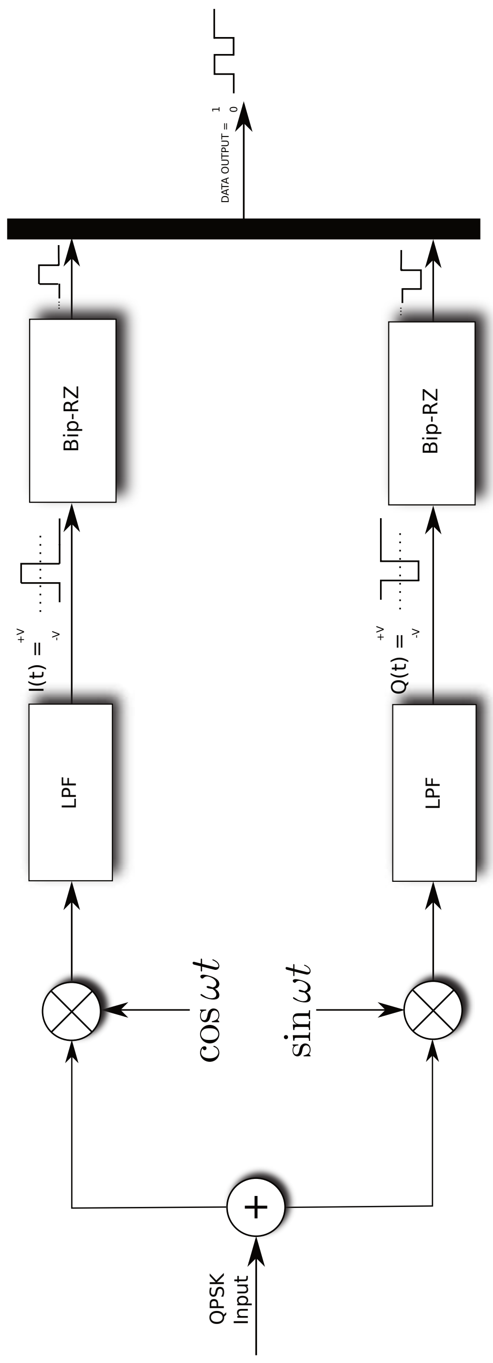
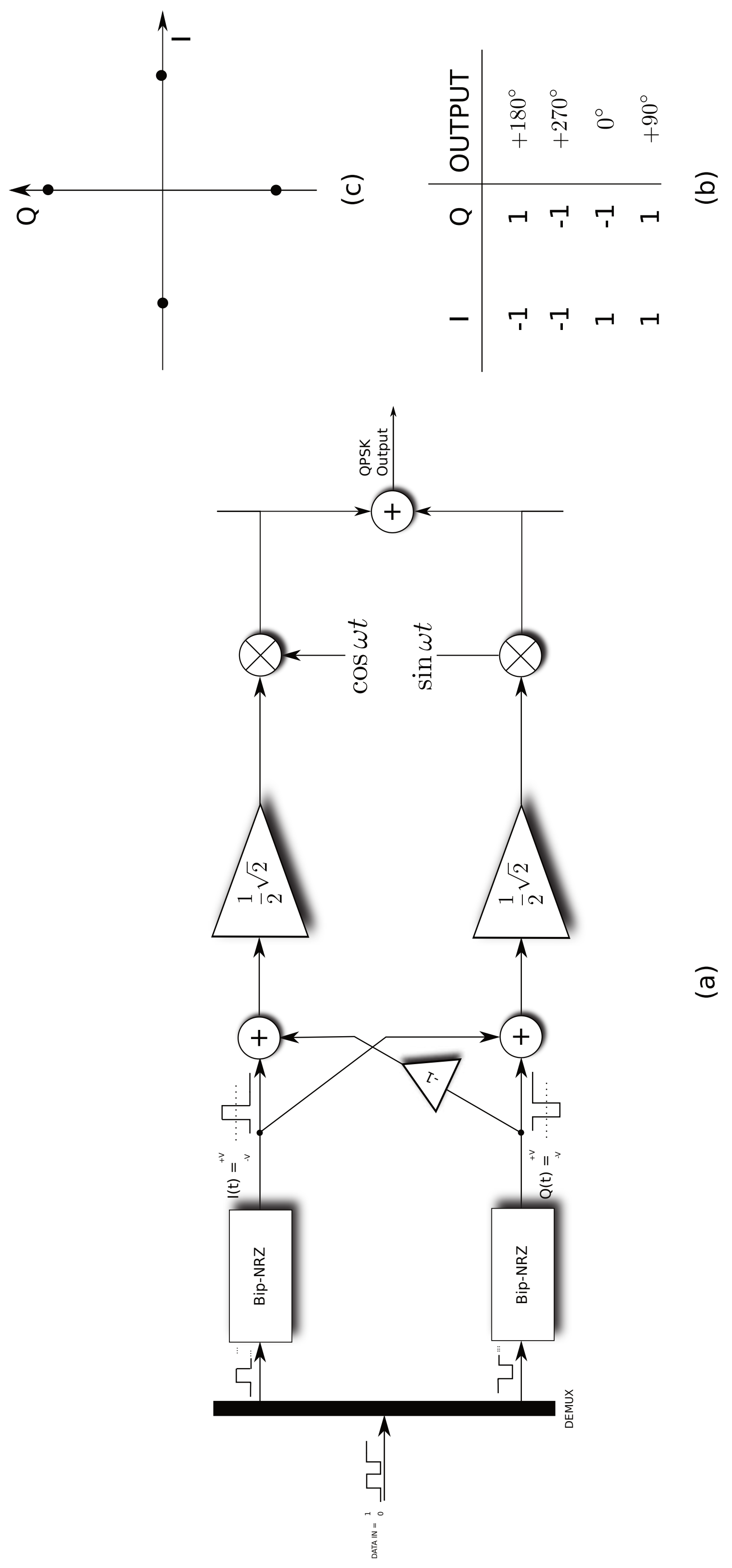


Figure D.6: QPSK demodulator.

D.3. QPSK Modulator with $\frac{\pi}{4}$ phase rotation



I	Q	OUTPUT
-1	1	+180°
-1	-1	+270°
1	-1	0°
1	1	+90°

Figure D.7: QPSK modulator with $\frac{\pi}{4}$ phase rotation.

D.4. QPSK Demodulator with $\frac{\pi}{4}$ phase rotation

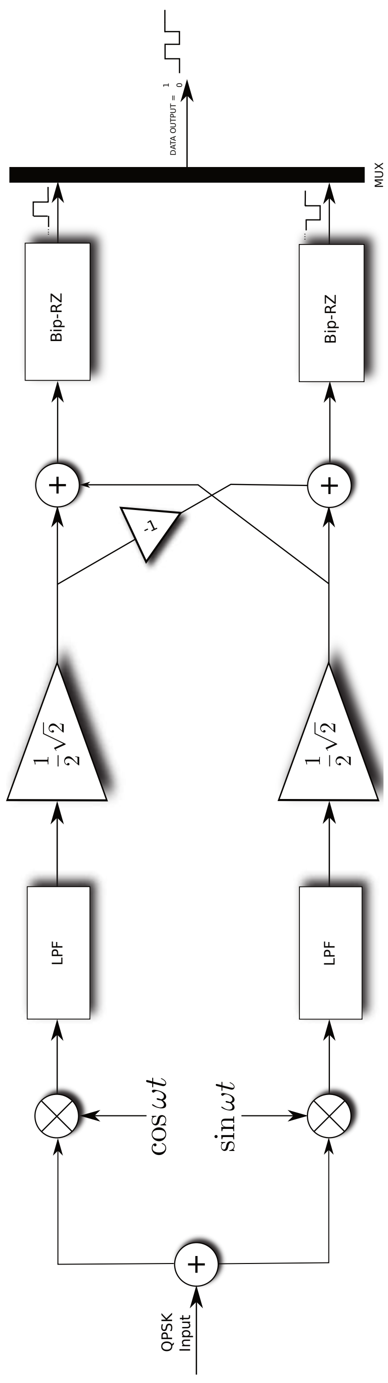


Figure D.8: QPSK demodulator with $\frac{\pi}{4}$ phase rotation.

E

$\frac{\pi}{4}$ DQPSK Modulation and Demodulation

A difficulty in the detection of PSK signals is that the phase relates to the time origin and has no “absolute” meaning. For example, a 90° phase shift in a QPSK waveform converts the constellation to a similar one, but with all the symbols interpreted incorrectly. This is also called phase ambiguity. Thus simple PSK waveforms cannot be detected non-coherently, i.e. there is a need for the demodulator to have a copy of the reference signal (the carrier wave) to determine the exact phase of the received signal (it is a coherent scheme). However if the information lies in the phase change from one bit (or symbol) to the next, then a time origin is not required and non-coherent detection is possible. This is accomplished through "differential" encoding and decoding of the base-band signal before modulation and after demodulation, respectively. But what exactly is differential encoding? In the next section we show an example of differential encoding and decoding using encoder and decoder shown in Figures E.1 and E.2.

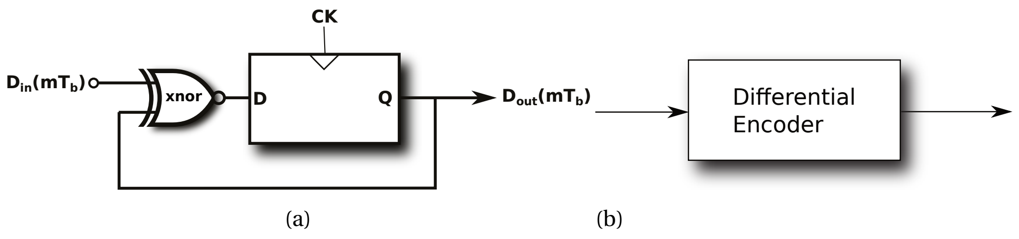


Figure E.1: Differential encoder (a), and its symbol (b)

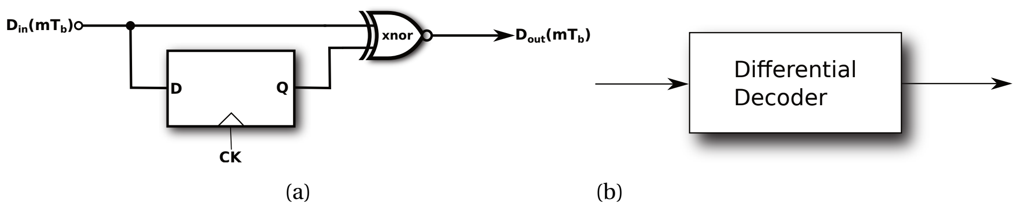


Figure E.2: Differential decoder (a), and its symbol (b)

Example:

This example gives two scenarios:

1. Where the data has been received correctly.
2. Where the data is inverted along the way in the communication channel.

Let us consider binary differential PSK (DPSK). The rule for differential encoding is that if the present input is a ONE, then the output state of the encoder does not change, vice versa.This requires an extra starting bit (of arbitrary value). This extra starting bit, a reference bit, corresponds to the state of the flipflop before the data sequence begins. It can be 0 or 1, it doesn't matter, but lets assume its a 1. The encoding proces is shown in Figure E.3.

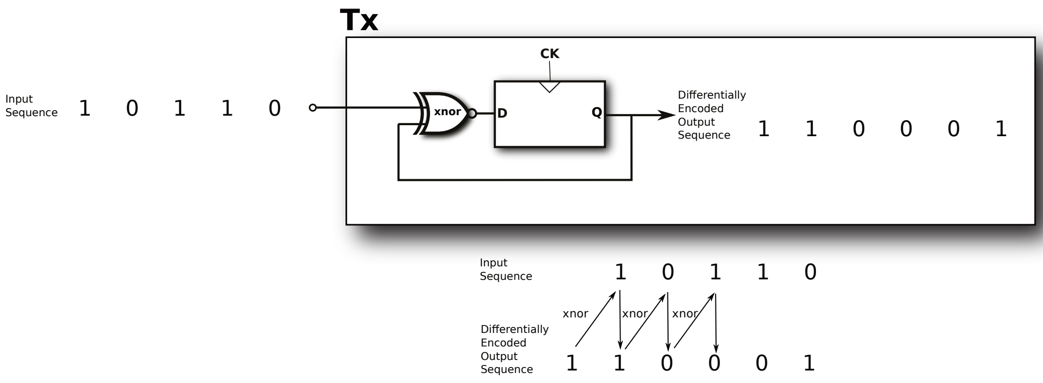

Tx
Input Sequence 1 0 1 1 0
CK
xnor
D
Q
Differentially Encoded Output Sequence 1 1 0 0 0 1
Input Sequence 1 0 1 1 0
xnor
xnor
xnor
Differentially Encoded Output Sequence 1 1 0 0 0 1

Figure E.3: Differential encoding process.

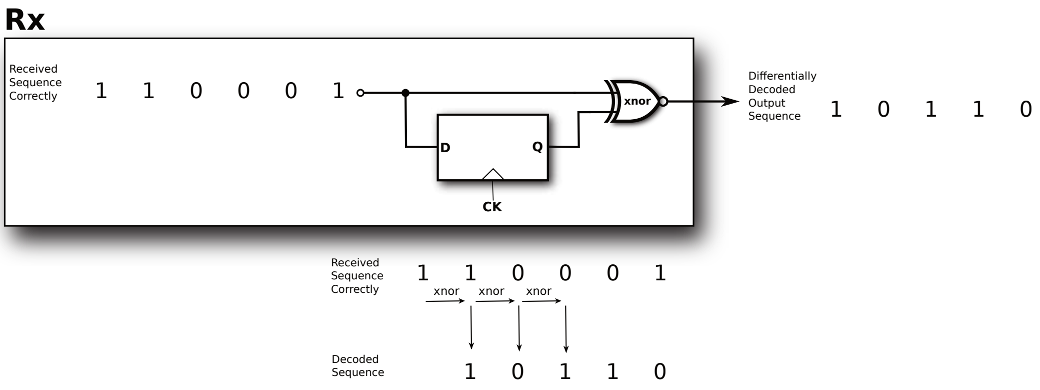

Rx
Received Sequence Correctly 1 1 0 0 0 1
xnor
D
Q
CK
Differentially Decoded Output Sequence 1 0 1 1 0
Received Sequence Correctly 1 1 0 0 0 1
xnor
xnor
xnor
Decoded Sequence 1 0 1 1 0

Figure E.4: Differential decoding process when data has been received correctly.

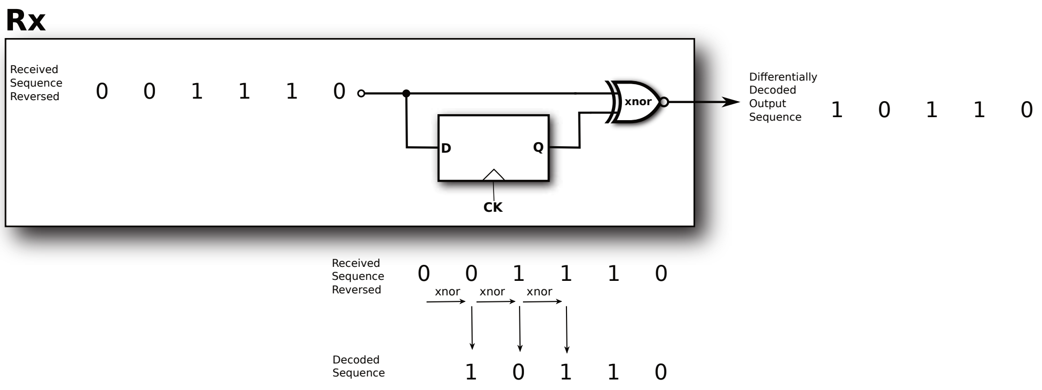

Rx
Received Sequence Reversed 0 0 1 1 1 0
xnor
D
Q
CK
Differentially Decoded Output Sequence 1 0 1 1 0
Received Sequence Reversed 0 0 1 1 1 0
xnor
xnor
xnor
Decoded Sequence 1 0 1 1 0

Figure E.5: Differential decoding process when data has been received incorrectly.

In either case, by the magic of binary numbers, we were able to get the original bit sequence back.

A variant of QPSK that can be differentially encoded is "$\frac{\pi}{4}$-QPSK". In this case, the signal set consists of two QPSK schems, one phase shifted 45° in relation to the other, hence the name "$\frac{\pi}{4}$-DQPSK".

$$x_1(t) = Acos(\omega t + k\frac{\pi}{4}) \quad k\ odd, \tag{E.1}$$

$$x_2(t) = Acos(\omega t + k\frac{\pi}{4}) \quad k\ even. \tag{E.2}$$

The $\frac{\pi}{4}$ modulation is performed by alternately taking the output from each QPSK scheme discussed in appendix C. Figure E.6 shows a concept of $\frac{\pi}{4}$ signal generation.

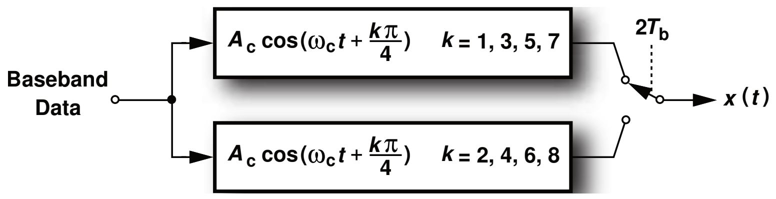
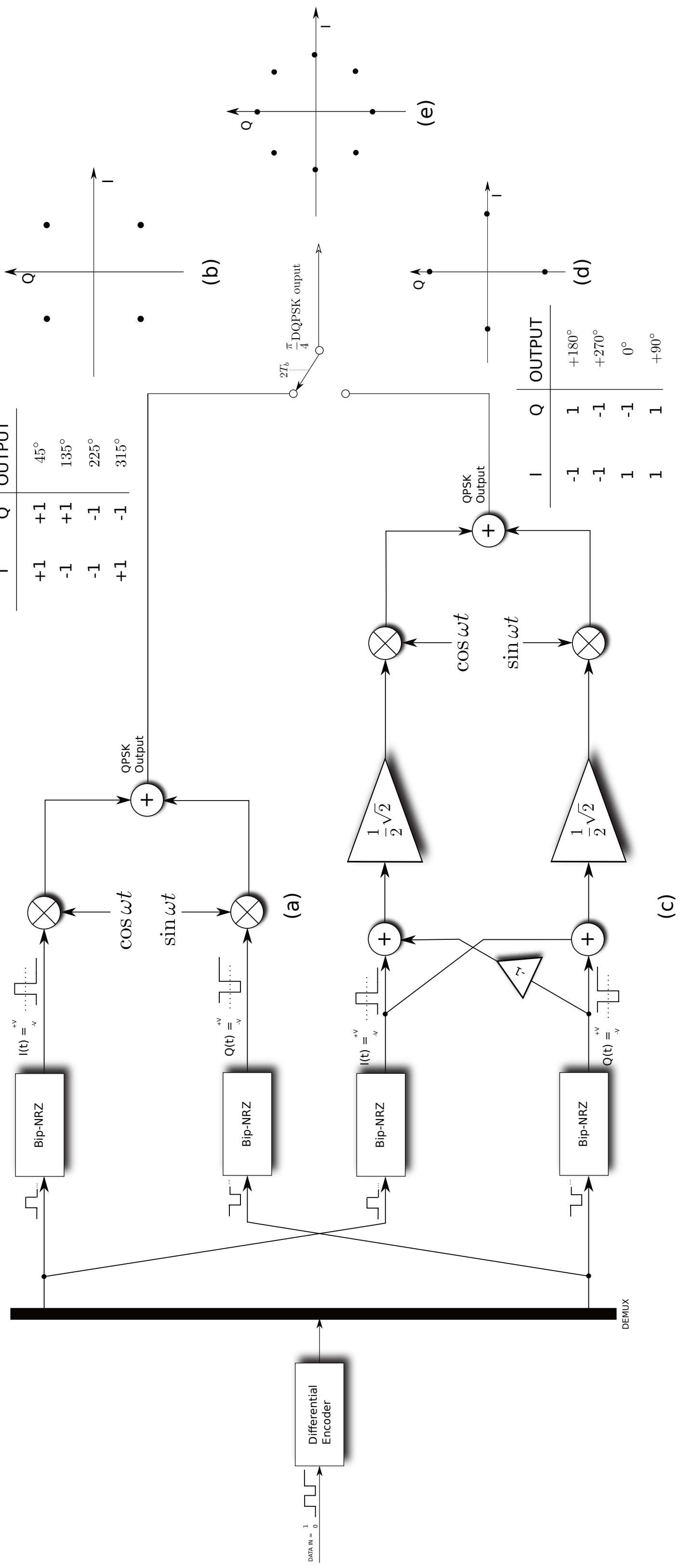

Baseband Data
$A_c \cos(\omega_c t + \frac{k\pi}{4})$ $k = 1, 3, 5, 7$
$A_c \cos(\omega_c t + \frac{k\pi}{4})$ $k = 2, 4, 6, 8$
$2T_b$
$x(t)$

Figure E.6: Conceptual generation of $\frac{\pi}{4}$-QPSK signal.

For further information about $\frac{\pi}{4}$ DQPSK modulation see appendix G subsection G.3.2.

The following subsections show a schematic representation of a $\frac{\pi}{4}$ DQPSK modulator and demodulator.

E.1. $\frac{\pi}{4}$ DQPSK Modulator



I	Q	OUTPUT
+1	+1	45°
-1	+1	135°
-1	-1	225°
+1	-1	315°

I	Q	OUTPUT
-1	1	+180°
-1	-1	+270°
1	-1	0°
1	1	+90°

Figure E.7: $\frac{\pi}{4}$ DQPSK modulator with QPSK modulator (a), its constellation (b), QPSK modulator with 45° phase rotation (c), its constellation (d) constellation of $\frac{\pi}{4}$ DQPSK signal (e).

E.2. $\frac{\pi}{4}$ DQPSK Demodulator

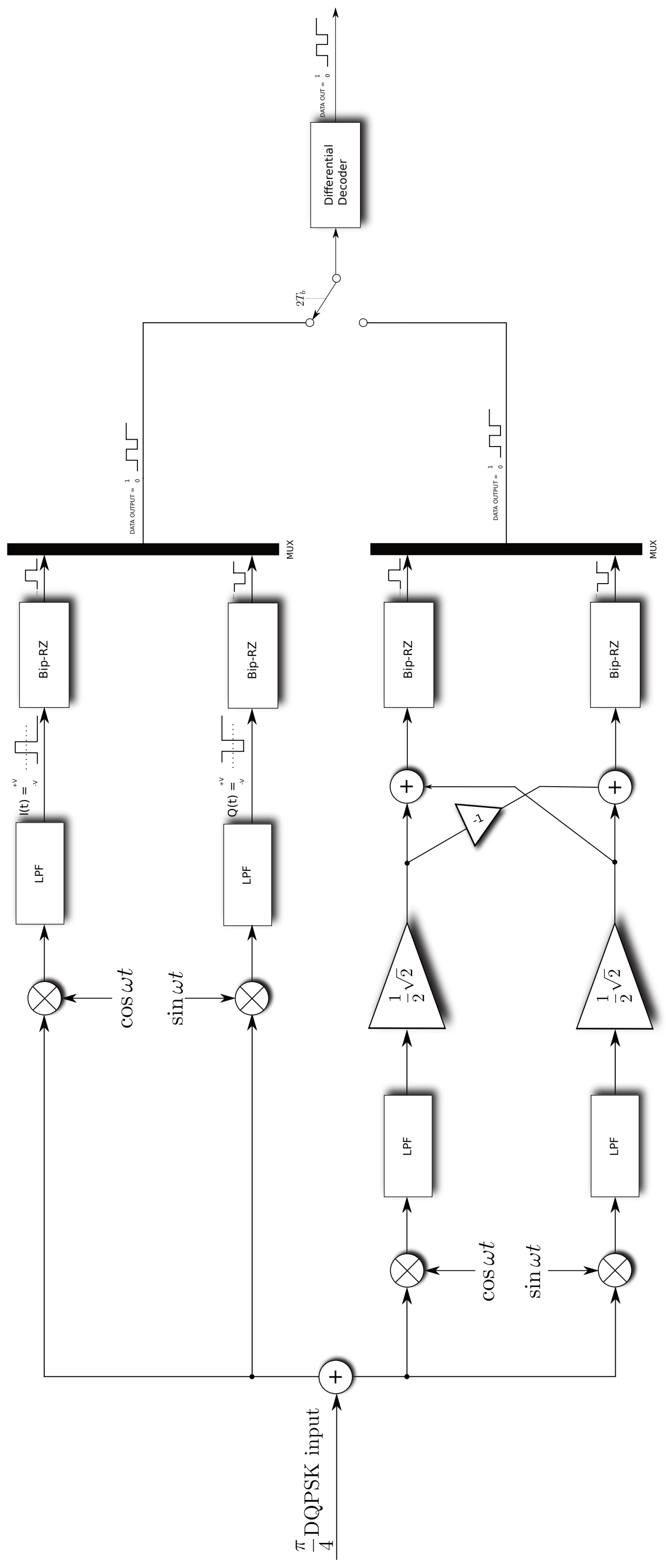
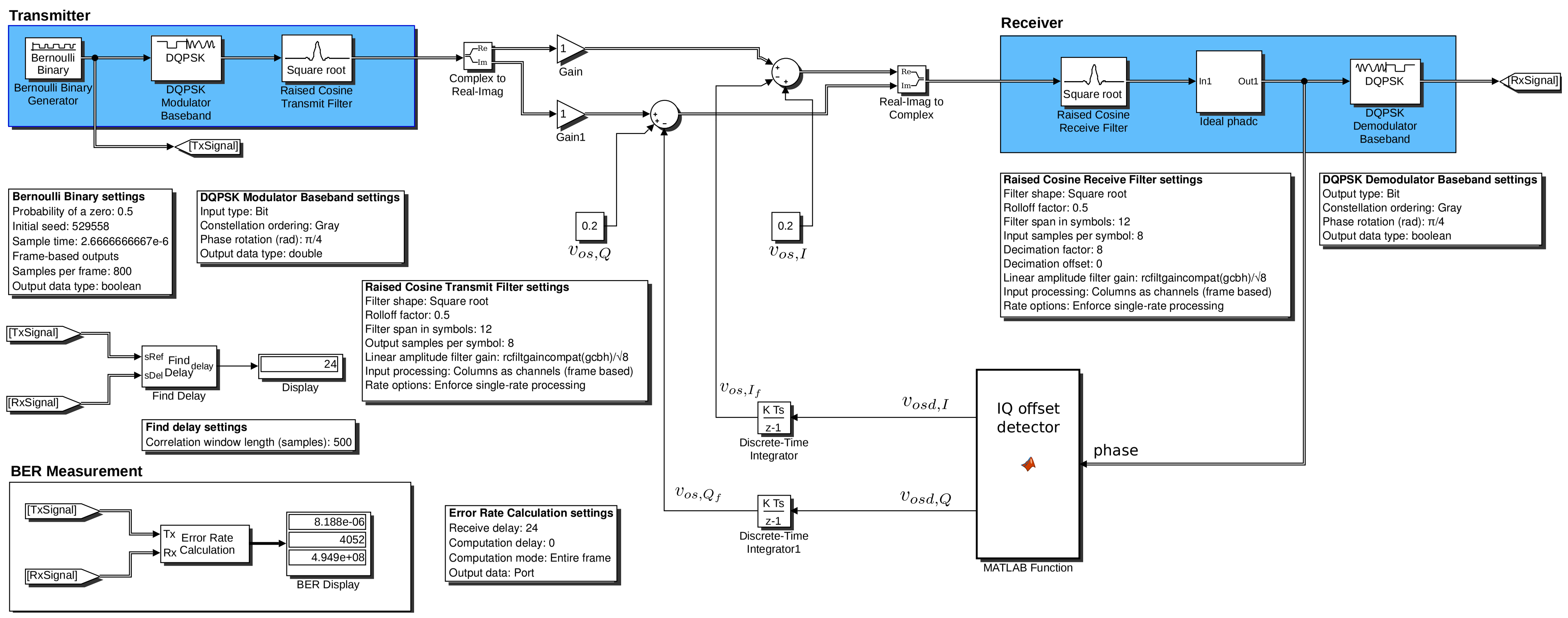


Figure E.8: $\frac{\pi}{4}$ DQPSK demodulator.

F
MATLAB Simulink simulation models for IQ offset compensation

F.1. Model for frame-based simulation



Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

Find delay settings
Correlation window length (samples): 500

Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

Raised Cosine Receive Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Input samples per symbol: 8
Decimation factor: 8
Decimation offset: 0
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

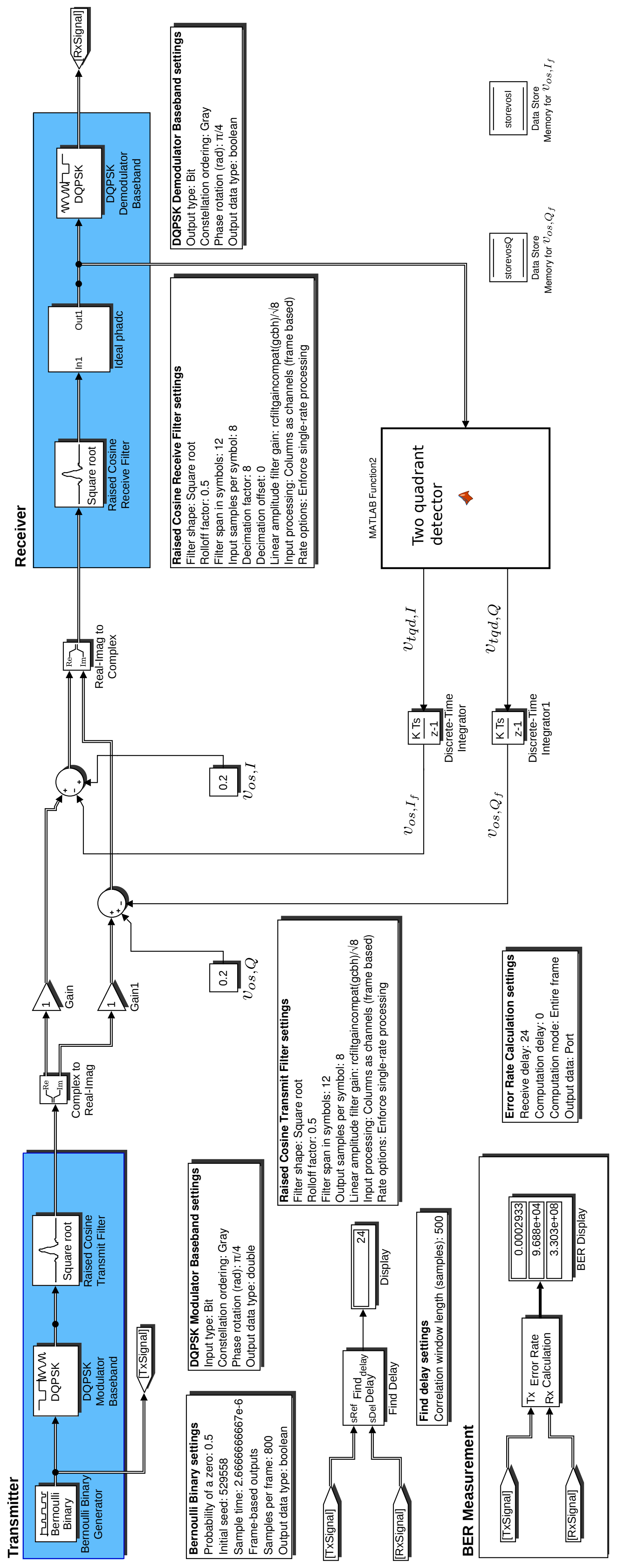
Figure F.1: Simulink IQ offset compensation system model for frame based input-output data processing.

F.1.1. Matlab code for IQ offset detector

```matlab
function [vosd_I, vosd_Q] = OffsetDetector(phase)
%#codegen
N = 800; %frame length
P = 1;
FS = 1;
RHP = 0;
LHP = 0;
BHP = 0;
THP = 0;
datacrop = angle(phase);
N = numel(datacrop);
 for i=1:N
    if datacrop(i) >= -0.5*pi && datacrop(i) < 0.5*pi
      RHP = RHP +1;
    end
    if datacrop(i) >= -pi && datacrop(i) < -0.5*pi || datacrop(i) >= 0.5*pi && datacrop(i) <= pi
      LHP = LHP +1;
    end
    if datacrop(i) >= 0 && datacrop(i) <= pi
      THP = THP +1;
    end
    if datacrop(i) >= -pi && datacrop(i) < 0
      BHP = BHP +1;
    end
 end
IMBI = (RHP-LHP)/N;
IMBQ = (THP-BHP)/N;
vosd_I = IMBI*pi*FS/4;
vosd_Q = IMBQ*pi*FS/4;
```

Figure F.2: Matlab code IQ offset compensation for frame based input-output data processing

F.2. Model for sample-based simulation



Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

Find delay settings
Correlation window length (samples): 500

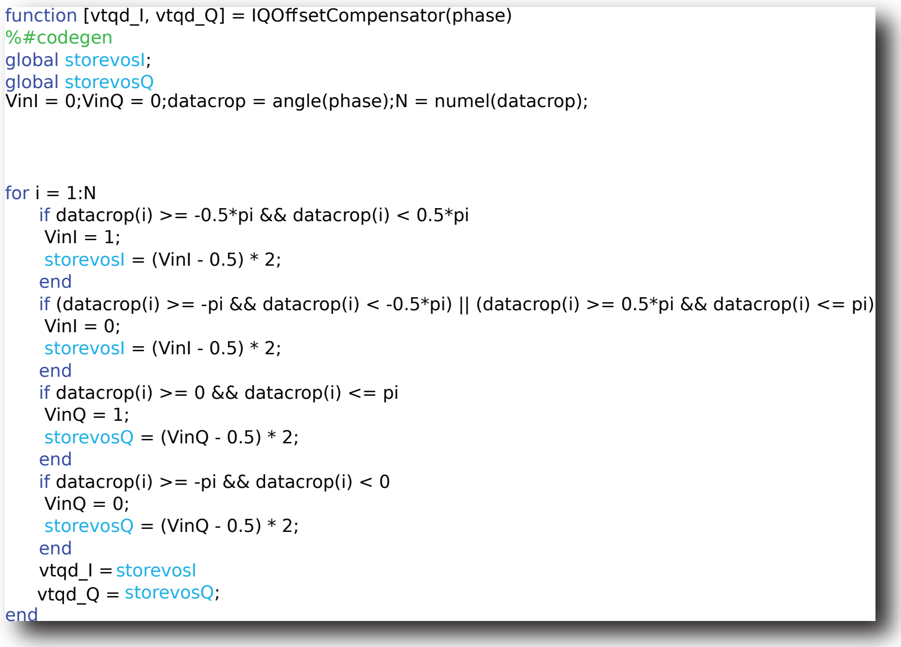
Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

Raised Cosine Receive Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Input samples per symbol: 8
Decimation factor: 8
Decimation offset: 0
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

Figure F.3: Simulink IQ offset compensation system model for sample based input-output data processing

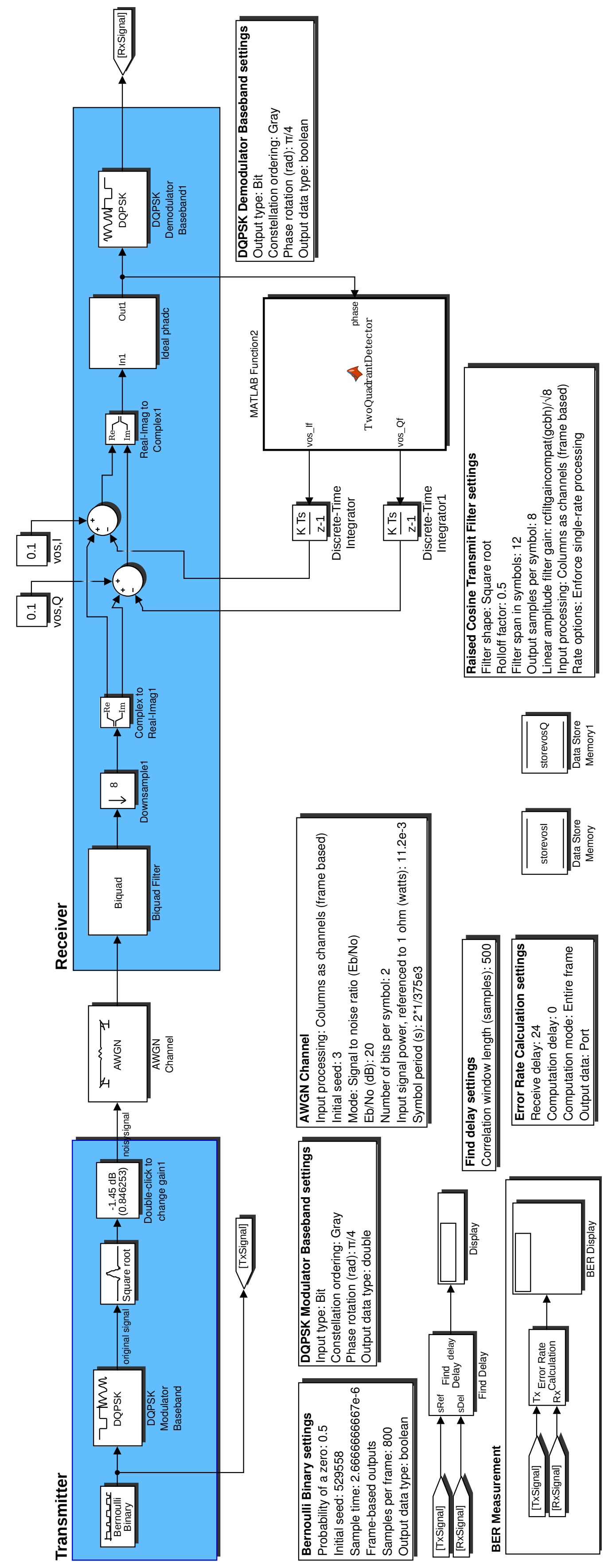
F.2.1. Matlab code for two quadrant detector



```
function [vtqd_I, vtqd_Q] = IQOffsetCompensator(phase)
%#codegen
global storevosI;
global storevosQ
VinI = 0;VinQ = 0;datacrop = angle(phase);N = numel(datacrop);



for i = 1:N
      if datacrop(i) >= -0.5*pi && datacrop(i) < 0.5*pi
       VinI = 1;
       storevosI = (VinI - 0.5) * 2;
      end
      if (datacrop(i) >= -pi && datacrop(i) < -0.5*pi) || (datacrop(i) >= 0.5*pi && datacrop(i) <= pi)
       VinI = 0;
       storevosI = (VinI - 0.5) * 2;
      end
      if datacrop(i) >= 0 && datacrop(i) <= pi
       VinQ = 1;
       storevosQ = (VinQ - 0.5) * 2;
      end
      if datacrop(i) >= -pi && datacrop(i) < 0
       VinQ = 0;
       storevosQ = (VinQ - 0.5) * 2;
      end
      vtqd_I = storevosI
      vtqd_Q = storevosQ;
end
```

Figure F.4: Matlab code IQ offset compensation for sample based input-output data processing

F.3. Model for sample-based simulation including IQ offset, biquad filter, channel and quantization noise



DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

AWGN Channel
Input processing: Columns as channels (frame based)
Initial seed: 3
Mode: Signal to noise ratio (Eb/No)
Eb/No (dB): 20
Number of bits per symbol: 2
Input signal power, referenced to 1 ohm (watts): 11.2e-3
Symbol period (s): 2*1/375e3

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

Find delay settings
Correlation window length (samples): 500

Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

Figure F.5: Simulink IQ offset compensation system model including IQ offset, biqaud filter, channel and quantization noise.

G

Description of the blocks used in Simulink model

G.1. Additive White Guassian Noise

Figure G.1 shows a useful block from Simulink to Add a White Gaussian Noise (AWGN) to the input communication channel. It adds white Gaussian noise to a real or complex input signal.

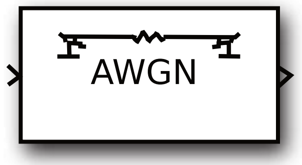


Figure G.1: Simulink AWGN block.

When the input signal is complex, this block adds complex Gaussian noise and produces a complex output signal. This block inherits its sample time from the input signal. The behavior for Real and Complex Input Signals is shown in Figure G.2. The figures illustrate the difference between the real and complex cases by showing the noise power spectral densities N_o of a real bandpass white noise process and its complex low-pass equivalent.

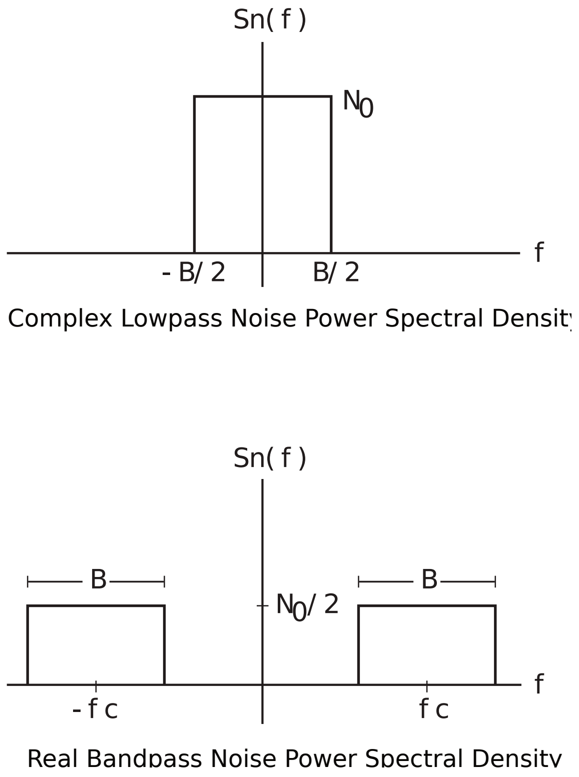
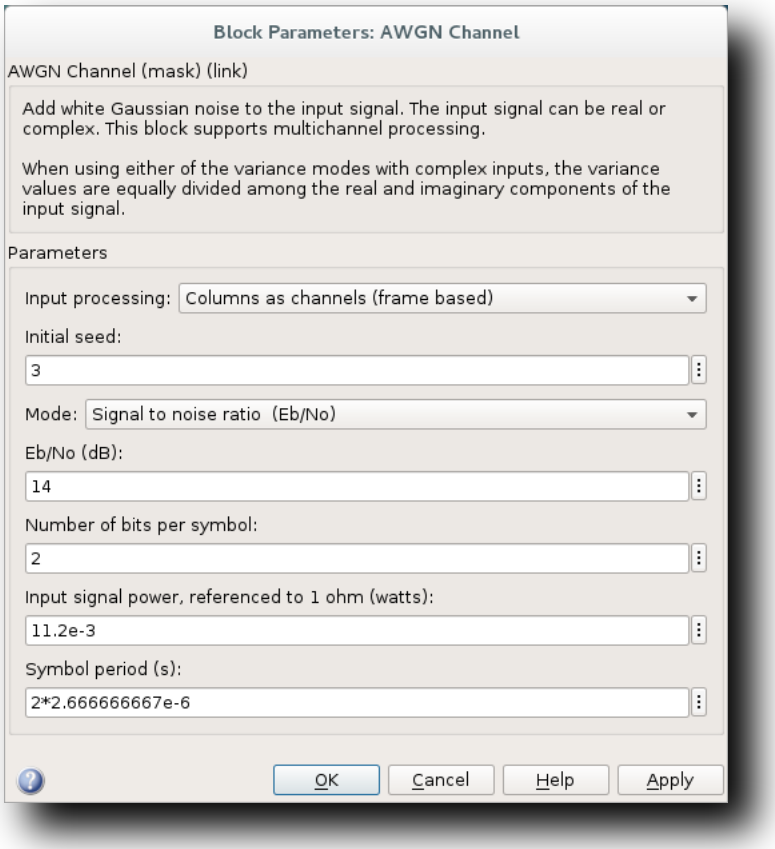

Sn(f)
N0
f
- B/ 2
B/ 2
Complex Lowpass Noise Power Spectral Densit
Sn(f)
B
B
N0/ 2
f
- f c
f c
Real Bandpass Noise Power Spectral Density

Figure G.2: White Gaussian noise behavior for Real and Complex Input Signals.

If you double click on the AWGN Simulink block then a dialog box will open and looks as shown in Figure G.3.


Block Parameters: AWGN Channel

AWGN Channel (mask) (link)

Add white Gaussian noise to the input signal. The input signal can be real or complex. This block supports multichannel processing.

When using either of the variance modes with complex inputs, the variance values are equally divided among the real and imaginary components of the input signal.

Parameters

Input processing: Columns as channels (frame based)

Initial seed: 3

Mode: Signal to noise ratio (Eb/No)

Eb/No (dB): 14

Number of bits per symbol: 2

Input signal power, referenced to 1 ohm (watts): 11.2e-3

Symbol period (s): 2*2.666666667e-6

Figure G.3: AWGN block parameters dialog box.

Here is a description of the Block Parameter settings

Initial seed = is the seed number for the Gaussian noise generator. Can be any arbitrary number. For each seed number the generator generates a different set of random numbers.
Mode = The mode by which you specify the noise variance: Signal to noise ratio E_b/N_o, Signal to noise ratio E_s/N_o, Signal to noise ratio SNR, Variance from mask, or Variance from port.

E_b/N_o (dB)
The ratio of information (i.e., without channel coding) bit energy per symbol to noise power spectral density, in decibels. This field appears only if Mode is set to E_b/N_o.

Es/N_o (dB)
The ratio of information (i.e., without channel coding) energy per symbol to noise power spectral density, in decibels. This field appears only if Mode is set to Es/N_o.

SNR (dB)
The ratio of signal to noise power, in decibels. This field appears only if Mode is set to SNR.

Number of bits per symbol = The number of bits in each input symbol. This field appears only if Mode is set E_b/N_o.
Input signal power, referenced to 1 ohm (watt) = The mean square power of the input symbols (if Mode is E_b/N_o or Es/N_o or input samples (if Mode is SNR), in watts. This field appears only if Mode is set to Eb/N_o , Es/N_o , or SNR.

Symbol period (s) = The duration of an information channel (i.e., without channel coding) symbol, in seconds. This field appears only if Mode is set to Eb/N_o or Es/N_o.

Variance = The variance of white Gaussian noise. This field appears only if Mode is set to Variance from mask

G.2. Bernoulli Binary Generator

The Matlab Simulink Model shown in Figures F.1 and F.3 of Appendix A contains the Binary Bernoulli Generator transmitter source signal. See Figure G.4 for the symbol. This block generates random binary ones and zeros with equal probability. If you double-click on the Bernoulli Binary Generator block, a dialog box will appear that allows you to set various parameters (Figure G.5). One of the parameters is the 'Sample Time'. The Sample Time is simply the reciprocal of the data rate, since the block generates one bit per sample. In other words:

$$T_{samp} = \frac{1}{R} \tag{G.1}$$

where T_{samp} (in seconds per sample) is the sample time and R is the data rate(in bits per second). So all you need to do is to enter the value for $\frac{1}{R}$ in the field for the Sample Time.

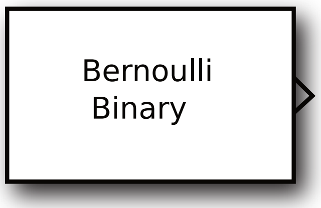
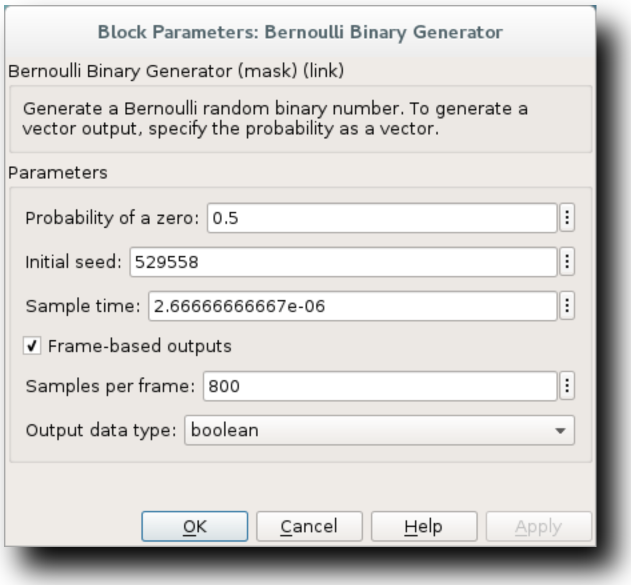


Figure G.4: Simulink Bernoulli Binary Generator.

The output of the Bernoulli Binary Generator block is a single instance of a discrete-time random process, which generates either a 0 or a 1 at a regular periodic time step. It is not a continuous-time signal, and it is not deterministic. You should not expect to see rectangular pulses, because that is not what it generates. You can plot the output as rectangular pulses, or triangular pulses, or pulses of any shape you care to use, but that does not change the fact that it is discrete-time and stochastic. If you want to see rectangular pulses, try using the standard time-based scope block instead of the vector scope. Incidentally, the only "correct" way to plot it is as a stem-and-leaf plot, which shows the signal as it actually is. See Table G.1. With each sample of a $2.666666667.10^{-6}\mu s$ the Bernoulli binary generator will generate a bit. Thus, the bit rate is 375 kBit/s. The output is frame-based and so at every frame 800 bits is sent per $2.133333.10^{-3}s$ which still amounts to a bit rate of 375kBit/s. And also we can say that the duration of a bit is $2.666666667.10^{-6}\mu s$ which is equal to the sampling time T_{samp} . In Figure G.5 the settings are made accordingly.


Block Parameters: Bernoulli Binary Generator

Bernoulli Binary Generator (mask) (link)

Generate a Bernoulli random binary number. To generate a vector output, specify the probability as a vector.

Parameters

Probability of a zero: 0.5

Initial seed: 529558

Sample time: 2.66666666667e-06

[✓] Frame-based outputs

Samples per frame: 800

Output data type: boolean

Figure G.5: Simulink Bernoulli Binary Generator Block Parameters.

Time sec.	Bernoulli Binary Boolean Output
0.0000e+00	0
1.0000e-06	0
2.0000e-06	0
3.0000e-06	0
4.0000e-06	1
5.0000e-06	0
6.0000e-06	0
7.0000e-06	1
8.0000e-06	1
9.0000e-06	1
1.0000e-05	0
1.1000e-05	0
1.2000e-05	1
1.3000e-05	1
1.4000e-05	0
1.5000e-05	1
1.6000e-05	0
1.7000e-05	1
1.8000e-05	1
1.9000e-05	0
2.0000e-05	0
2.1000e-05	1
2.2000e-05	0
2.3000e-05	0
2.4000e-05	1
2.5000e-05	0
2.6000e-05	1
2.7000e-05	1
2.8000e-05	1
2.9000e-05	0
3.0000e-05	0

Table G.1: Table Bernoulli Binary Generator output.

G.3. DQPSK Modulator Baseband

If we continue in the chain of the transmitter shown in Figure F.1 and Figure F.3 of Appendix F we see the next block, namely the DQPSK modulator. Symbol is shown in Figure G.6. The DQPSK Modulator Baseband block modulates using the differential quaternary (M=4) phase shift keying method. The output is a baseband representation of the modulated signal. The input must be a discrete-time signal.

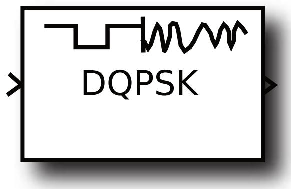
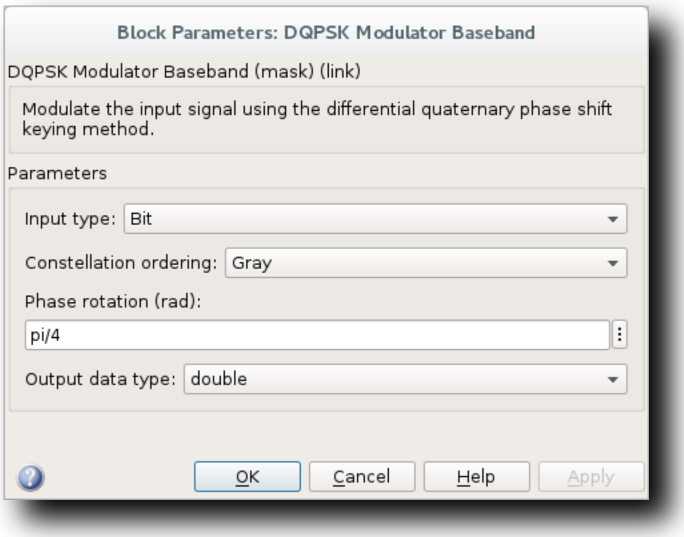


Figure G.6: Simulink DQPSK Modulator Baseband block.

G.3.1. Description

The DQPSK Modulator Baseband block modulates using the differential quaternary (M=4) phase shift keying method. The output is a baseband representation of the modulated signal. The input must be a discrete-time signal. When you double-click the Simulink Modulator Baseband symbol, you will see a dialog box as shown in Figure G.7.


Block Parameters: DQPSK Modulator Baseband

DQPSK Modulator Baseband (mask) (link)

Modulate the input signal using the differential quaternary phase shift keying method.

Parameters

Input type: Bit

Constellation ordering: Gray

Phase rotation (rad):

pi/4

Output data type: double

Figure G.7: Settings for the Simulink DQPSK Modulator Baseband.

G.3.2. Integer-Valued Signals and Binary-Valued Signals

When you set the **Input type** parameter to *Integer*, the valid input values are 0, 1, 2, and 3. In this case, the block accepts a scalar or column vector input signal. If the first input is *m*, then the modulated symbol is

$$e^{j\theta + j\pi m/2} \tag{G.2}$$

where θ represents the **Phase rotation** parameter. If a successive input is m, then the modulated symbol is the previous modulated symbol multiplied by $e^{j\theta + j\pi m/2}$.

When you set the **Input type** parameter to *Bit*, the input contains pairs of binary values. In this case, the block accepts a column vector whose length is an even integer. The following figure shows the complex numbers by which the block multiples the previous symbol to compute the current symbol, depending on whether you set the **Constellation ordering** parameter to *Binary or Gray*. The following Figure G.8 assumes that you set the **Phase rotation** parameter to $\frac{\pi}{4}$ in other cases, the two schematics would be rotated accordingly.

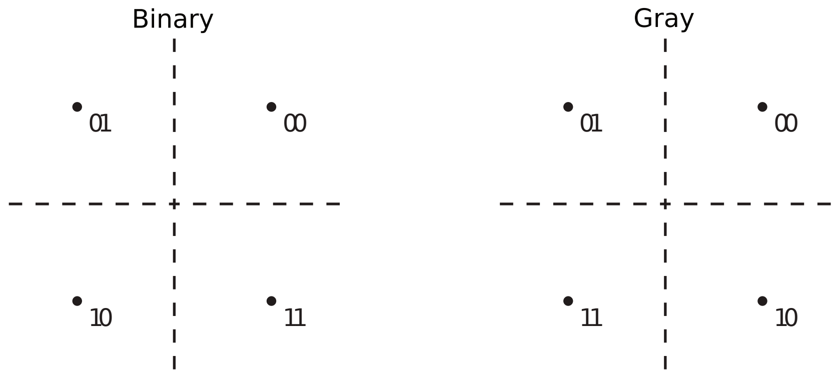


Figure G.8: Binary and Gray constellation ordering.

The constellations shown in Figure G.8 are the constellations used for QPSK (Quadrature Phase Shift Keying). A variant of QPSK that can be differentially encoded (DQPSK, hence the name) is "$\frac{\pi}{4}$ QPSK". In this case, the signal consists of two QPSK schemes, one rotated 45° ($\frac{\pi}{4}$ radians, hence the name) with respect to the other. Figure G.9 below shows the two separate constellations with identical Gray coding but rotated 45° with respect to each other.

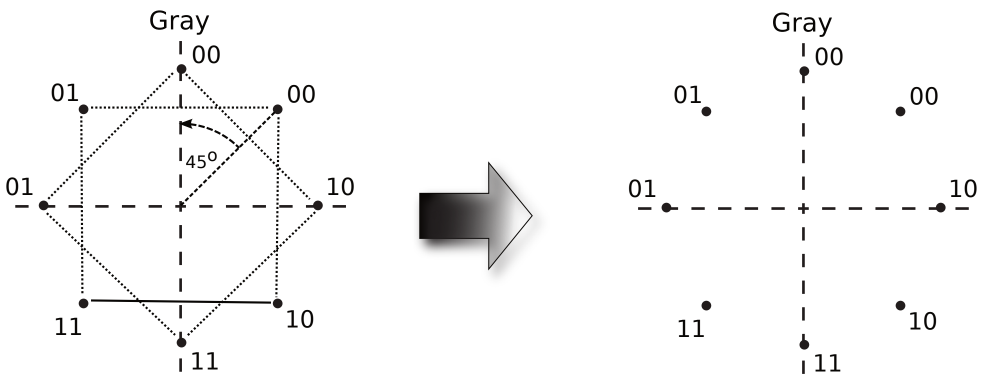


Figure G.9: Dual Constellation Diagrams For $\frac{\pi}{4}$ Gray Coded QPSK Modulation.

The $\frac{\pi}{4}$ modulation is performed by alternately taking the output from each QPSK scheme. An important drawback of QPSK stems from the large phase changes at the end of each symbol.

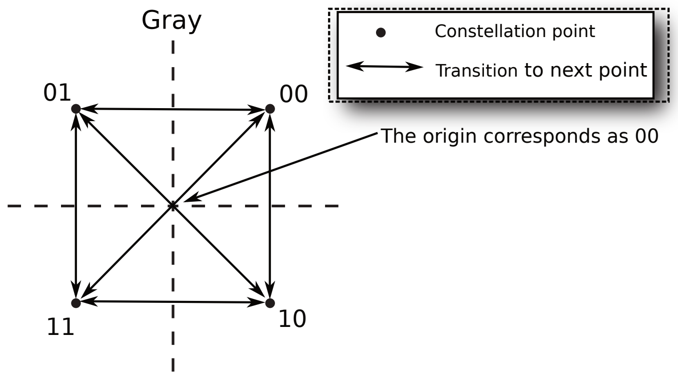


Figure G.10: Possible phase transitions in Gray Coded QPSK Constellation Diagram.

So, the data is read from one constellation to another in either 90° or 180° as shown in the QPSK constellation diagram in Figure G.10. This means signal with QPSK modulation can have a maximum of 180° phase shift when traveling through a multi-path environment. For example, the data is read from 01 to 10 by passing through the origin. So for the receiver, the origin might be mistaken correspond to 00. This will lead to false information decoding at the receiver.

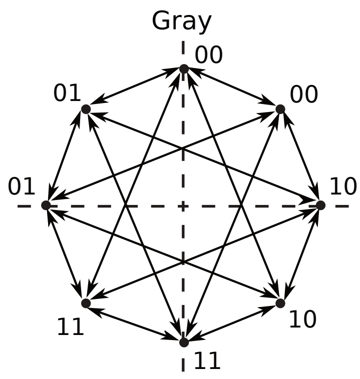


Figure G.11: Possible phase transitions in Gray Coded $\frac{\pi}{4}$ QPSK

If we look at the possible phase transitions for a $\frac{\pi}{4}$ QPSK modulation scheme shown in Figure G.11, we see that there are no phase transitions through the origin and therefore does not have this disadvantage as in QPSK. The key point here is that, since no two consecutive points are from the same constellation, the maximum phase step is 135°, so 45° less than in QPSK.

G.3.3. Differential Phase Shift Keying

A difficulty in the detection of PSK signals is that the phase relates to the time origin and has no “absolute” meaning. For example, a 90° phase shift in a QPSK waveform converts the constellation to a similar one, but with all the symbols interpreted incorrectly. This is also called phase ambiguity. Thus simple PSK waveforms cannot be detected non-coherently, i.e., there is a need for the demodulator to have a copy of the reference signal (the carrier wave) to determine the exact phase of the received signal (it is a coherent scheme). However if the information lies in the phase change from one bit (or symbol) to the next, then a time origin is not required and non-coherent detection is possible. This is accomplished through "differential" encoding and decoding of the base-band signal before modulation and after demodulation, respectively. But what exactly is differential encoding? I will go into in detail in the next section.

G.3.4. Differential Encoding —Is used to provide polarity reversal protection

Bit streams going through the many communications circuits in the channel can be unintentionally inverted. Most signal processing circuits can not tell if the whole stream is inverted. This is also called phase ambiguity.

Differential Encoding is used to protect against this possibility. It is one of the simplest form of error protection coding done on a base-band sequence prior to modulation.

Let us consider binary differential PSK (DPSK). The rule for differential encoding is that if the present input bit is a ONE, then the output state of the encoder does not change, and vice versa. This requires an extra starting bit (of arbitrary value). The concept can be better understood by considering the implementation depicted in Figure G.12. An exclusive-NOR (XNOR) gate compares the present output bit, $D_{out}(mT_b)$, with the present input bit, $D_{in}(mT_b)$, to determine the next output state:

$D_{in}(mT_b)$ = Present Input Bit
$D_{out}(mT_b)$ = Present Output Bit

$$D_{out}[(m+1)T_b] = \overline{D_{in}(mT_b) \oplus D_{out}(mT_b)}$$

implying that if $D_{in}(mT_b) = 1$, then $D_{out}[(m+1)T_b] = D_{out}(mT_b)$, and if $D_{in}(mT_b) = 0$ then $D_{out}[(m+1)T_b] = \overline{D_{out}(mT_b)}$. The extra starting bit mentioned above corresponds to the state of the flipflop before the data sequence begins.

Encoding

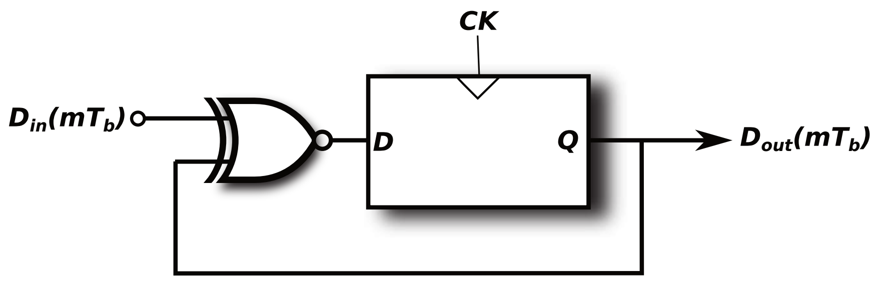


Figure G.12: Differential encoding. (Picture courtesy of Behzad Razavi.)

Here is how it works. Let's take a sequence as shown below in Figure G.13. The Encoding circuit above has a reference bit (it can be 0 or 1, it doesn't matter). The incoming data sequence is XNORed to this reference bit and forms the second bit of the encoded sequence. This bit is then added to the next data bit to continue the process as shown below.

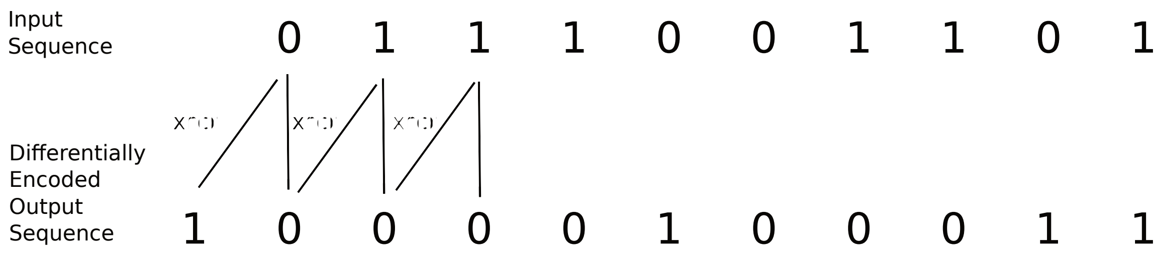


Figure G.13: Encoded data sequence.

Decoding

The decoding process reverses the above. Figure G.14 shows the decoder. The incoming bits are added together to recreate the input data sequence. There are now two possibilities, 1. that the received sequence was not reversed (see Figure G.15), and 2, that it was (see Figure G.16). let's see how the circuit deals with each of these two possibilities.

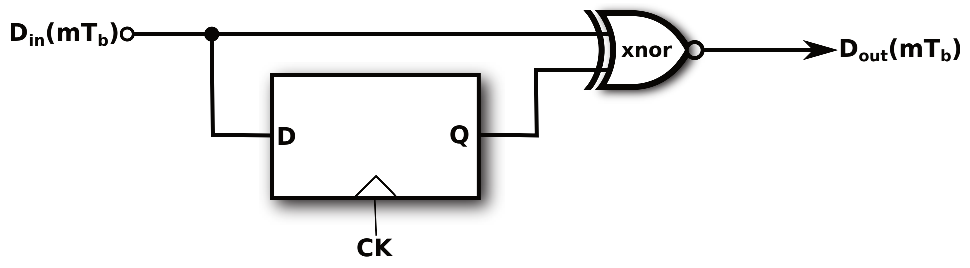


Figure G.14: Differential decoding. (Picture courtesy of Behzad Razavi.)

Bit Sequence Received Correctly

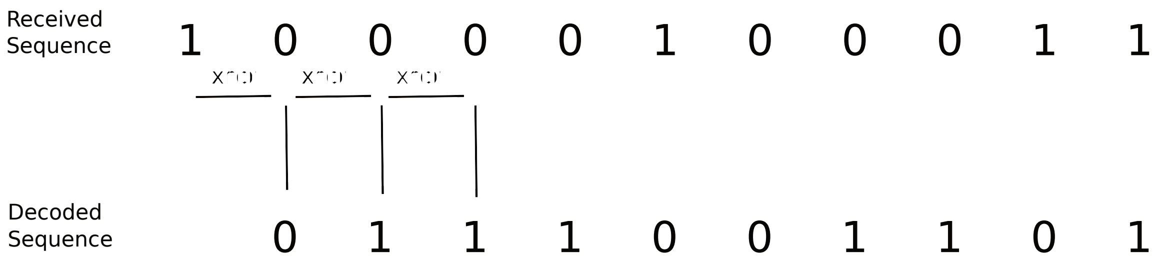


Figure G.15: Decoded data of the sequence received correctly.

Bit Sequence Received Reversed

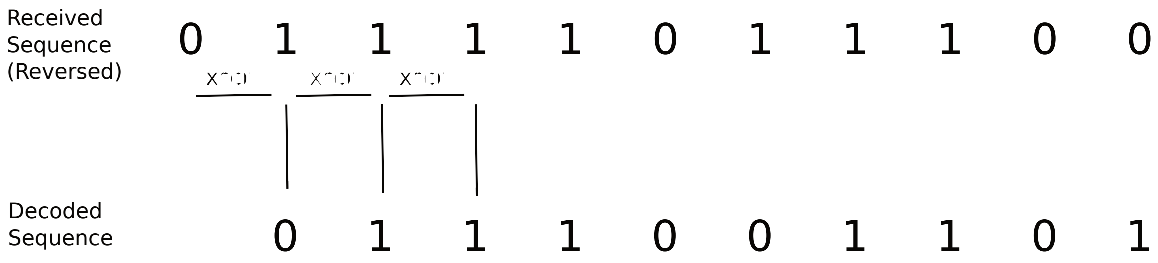


Figure G.16: Decoded data of the sequence received reversed.

In either case, by the magic of binary numbers, we were able to get the original bit sequence back.

Summarized

Main purpose of Differential Encoding is to protect against polarity reversals of input bit sequences. Hence Differentially Encoded data sequences have a slightly superior error performance.

G.4. Raised Cosine Transmit Filter

If we still remain in the treatment of the transmitter shown in Figure F.1 and Figure F.3 of Appendix F, then we see that after the $\frac{\pi}{4}$ DQPSK modulator the Raised Cosine Transmit Filter is inserted. The Simulink symbol is shown below in Figure G.17 .

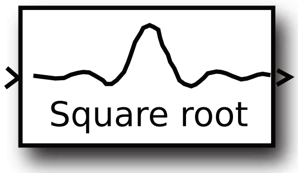


Figure G.17: Simulink Raised Cosine Transmit Filter Block.

The Raised Cosine Transmit Filter is used for pulse shaping the *baseband pulse* by upsampling and filtering the input signal using a normal raised cosine FIR filter or a square root raised cosine FIR filter. This is done for an important reason: the *rectangular baseband pulses* used in the binary sequence contain sharp transitions between ZEROs and ONEs and lead to an unnecessarily wide bandwidth and hence to an increase in integrated noise. Figure G.18 shows a random binary sequence and its $sinc^2$ spectrum.
What happens if this signal is applied to a low pass filter having a narrow bandwidth, e.g., $1/2T_b$? Since the frequency components above $1/2T_b$ are suppressed, the signal experiences substantial "intersymbol interference" (ISI).
Figure G.19 shows the effect of ISI in the time domain. If an ideal periodic rectangular signal is applied to a low-pass filter (LPF), the output exhibits an exponential tail that becomes longer as the filter bandwidth decreases. This occurs fundamentally because a signal cannot be both time-limited and bandwidth-limited: when the time-limited pulse passes through the band-limited system, the output must extend to infinity in the time domain.
Now suppose the output of a digital system consists of a random sequence of ONEs and ZEROs. If this sequence is applied to a LPF, the output can be obtained as the superposition of the responses to each input bit. We note that each bit level is corrupted by decaying tails created by previous bits. Called ISI, this phenomenon leads to a higher error rate because it brings the peak levels of ONEs and ZEROs closer to the detection threshold. We also observe a trade-off between noise and ISI: if the bandwidth is reduced so as to lesson the integrated noise, then ISI increases. In general, any system that removes part of the spectrum of a signal introduces ISI.

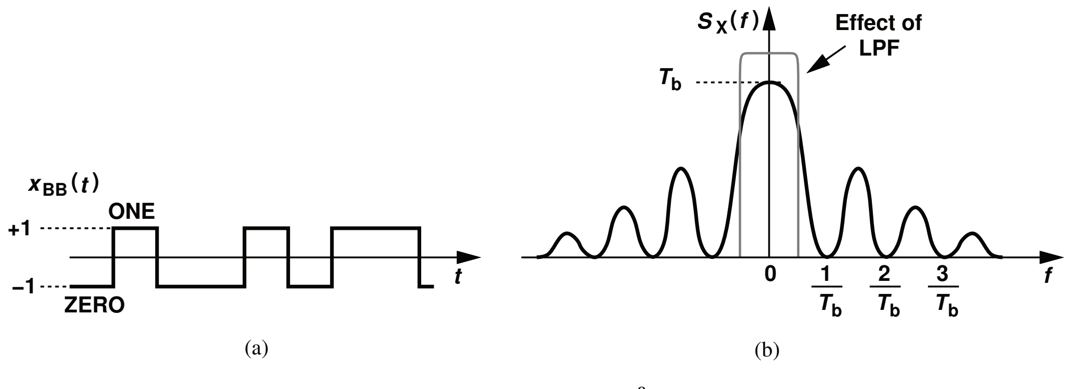


Figure G.18: A random binary sequence toggling between -1 and +1 (a), its $sinc^2$ spectrum. (Pictures courtesy of Behzad Razavi.)

The above analysis suggests that, to reduce the bandwidth of the modulated signal, the baseband pulse must be designed so as to occupy a small bandwidth itself. In this regard, the rectangular pulse used in the binary sequence of Figure G.18(a) is a poor choice.

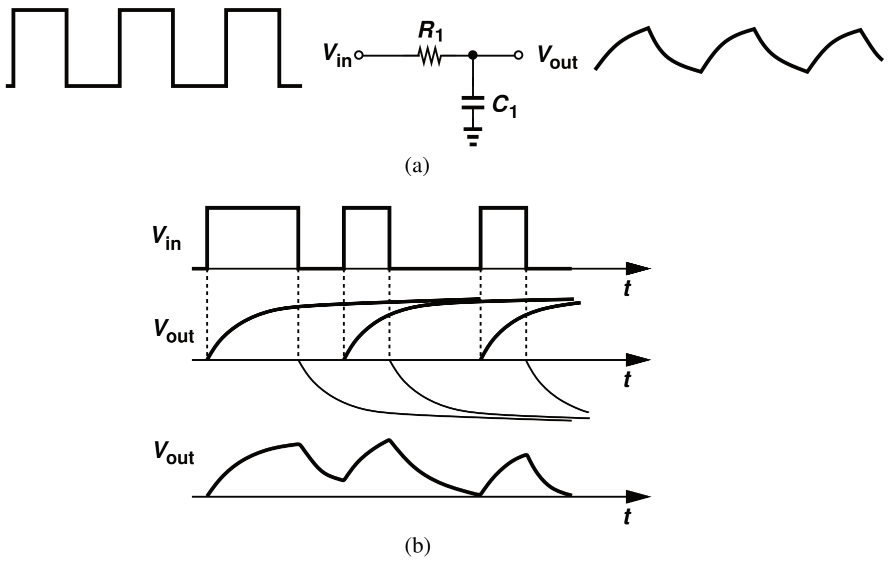


Figure G.19: Effect of low-pass filter on (a) periodic waveform and (b) random sequence. (Pictures courtesy of Behzad Razavi.)

For this reason, the baseband pulses in communication systems are usually "shaped" to reduce their bandwidth. Shown in Figure G.20 is a conceptual example where the basic pulse exhibits smooth transitions, thereby occupying less bandwidth than rectangular pulses.

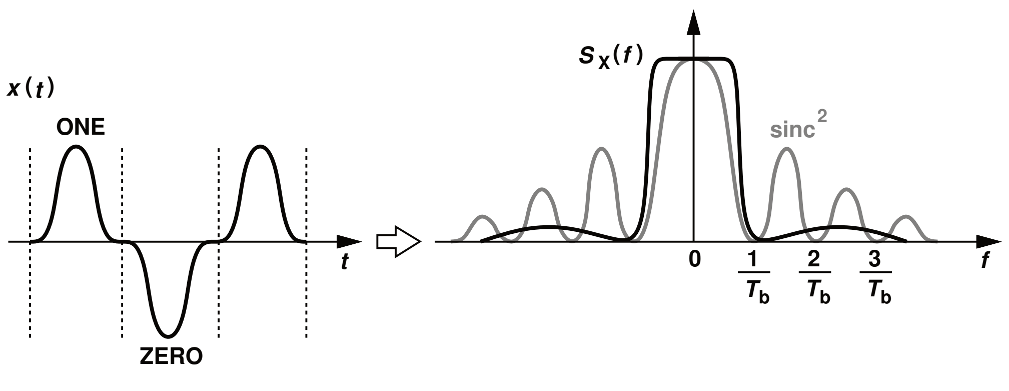
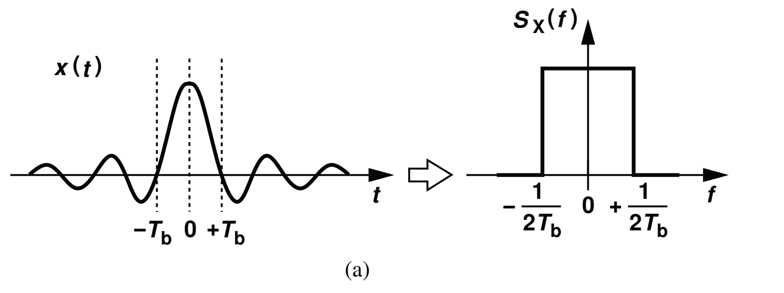


Figure G.20: Effect of smooth data transitions on spectrum. (Pictures courtesy of Behzad Razavi.)

What pulse shape yields the tightest spectrum? Since the spectrum of an ideal rectangular pulse is a sinc, we surmise that a sinc pulse in the time domain gives a rectangular ("brickwall") spectrum. See Figure G.21.



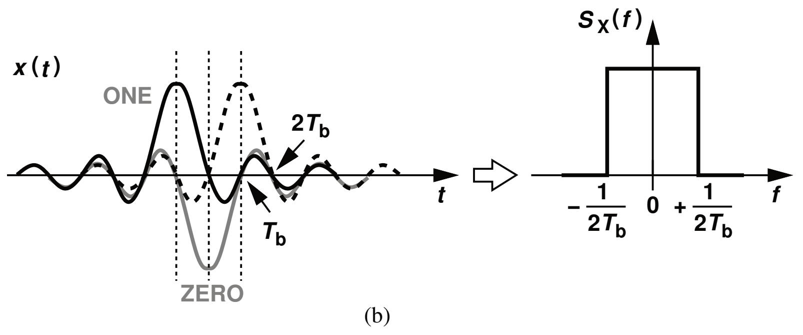
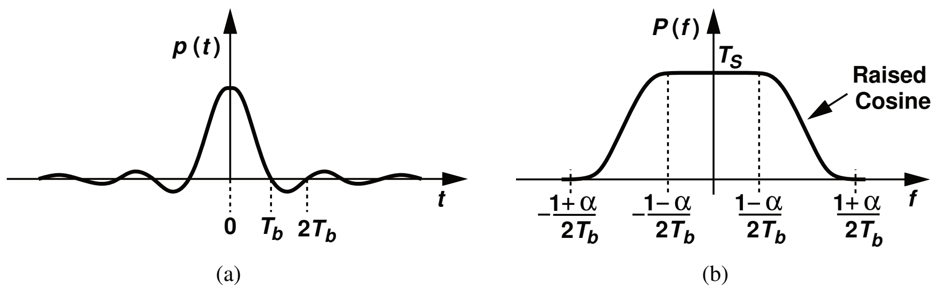


Figure G.21: Sinc pulse and its spectrum (a), random sequence of sinc pulses and its spectrum (b). (Pictures courtesy of Behzad Razavi.)

Now, if a random binary sequence employs such a pulse every T_b seconds, the spectrum still remains a rectangular. See Figure G.21(b). This bandwidth advantage persists after upconversion as well, i.e., when a reference signal (carrier wave) gets modulated by this baseband signal. Do we observe ISI in the waveform of Figure G.21(b)? If the waveform is sampled at exactly multiples of T_b, then ISI is zero because all other pulses go through zero at these points. The use of such overlapping pulses that produce no ISI is called “Nyquist signaling”. In practice sinc pulses are difficult to generate and approximations are used instead. A common pulse shape is shown in Figure G.22(a) and expressed as

$$p(t) = \frac{sinc(\pi/T_s)}{\pi t/T_s}\,\frac{cos(\pi\alpha t/T_s)}{1-4\alpha^2 t^2/T_s^2} \tag{G.3}$$

This pulse exhibits a "raised-cosine" spectrum Figure G.22(b). Called the roll-off factor”, α determines how close p(t) is to a sinc and, hence, the spectrum to a rectangle.



p(t)
0 T_b $2T_b$
t
(a)
P(f)
T_S
Raised Cosine
$-\frac{1+\alpha}{2T_b}$ $-\frac{1-\alpha}{2T_b}$ $\frac{1-\alpha}{2T_b}$ $\frac{1+\alpha}{2T_b}$
f
(b)

Figure G.22: Raised-cosine pulse shaping: (a) basic pulse and (b) corresponding spectrum. (Pictures courtesy of Behzad Razavi.)

And is calculated as

occupied bandwidth=symbol rate $1/T_s(1+\alpha)$ or *occupied bandwidth=symbol rate* $f_s(1+\alpha)$

For $\alpha = 0$, the pulse reduces to a sinc, hence, the spectrum would be a rectangle (brick wall) with sharp transitions and the occupied bandwidth would be

occupied bandwidth=symbol rate $1/T_s(1+0)$=*symbol rate* $1/T_s$

In a perfect world, the occupied bandwidth would be the same as the symbol rate, but this is not practical. An alpha of zero is impossible to implement.

Alpha is sometimes called the "excess bandwidth factor" as it indicates the amount of occupied bandwidth that will be required in excess of the ideal occupied bandwidth (which would be the same as the symbol rate).

At the other extreme, take a broader filter with an $\alpha = 1$, which is easier to implement, the spectrum becomes relatively wide. The occupied bandwidth will be

occupied bandwidth=symbol rate $1/T_s(1+1)$*=2∗ symbolrate* $1/T_s$

An alpha of one uses twice as much bandwidth as an alpha of zero. In practice, it is possible to implement an α below 0.2 and make good, compact, practical radios. Typical values of α are in the range of 0.3 to 0.5. Figure G.23 shows the effect of different values of α on the filter bandwidth.

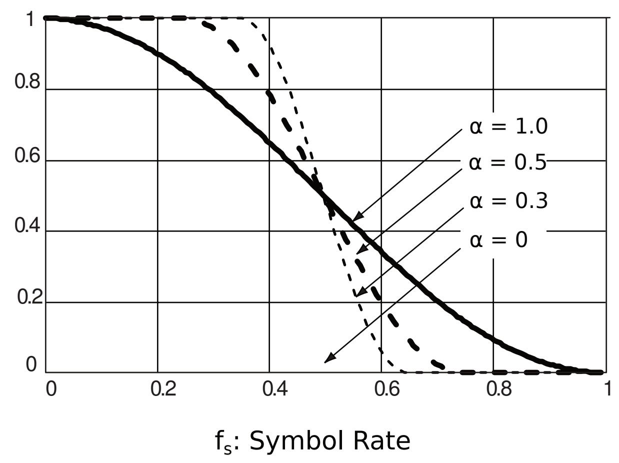
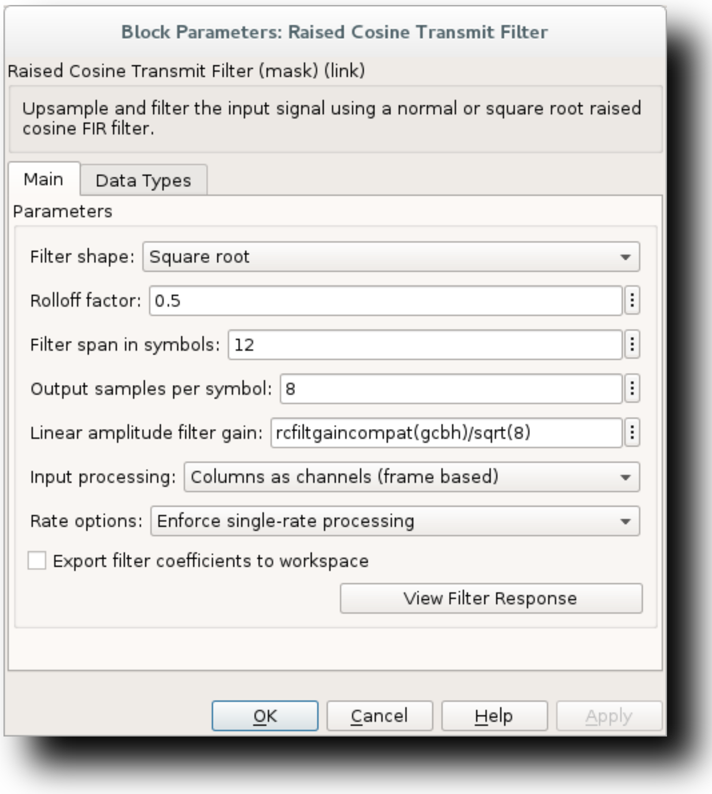


Figure G.23: Filter Bandwidth Parameters "α".

Now that we have gained knowledge about the different characteristics of a Raised Cosine filter we go back to the Simulink Raised Cosine Transmit Filter block to set the parameters correctly. If you double- click this block you get a dialog box as shown in Figure G.24.



Block Parameters: Raised Cosine Transmit Filter

Raised Cosine Transmit Filter (mask) (link)

Upsample and filter the input signal using a normal or square root raised cosine FIR filter.

Main | Data Types

Parameters

Filter shape: Square root

Rolloff factor: 0.5

Filter span in symbols: 12

Output samples per symbol: 8

Linear amplitude filter gain: rcfiltgaincompat(gcbh)/sqrt(8)

Input processing: Columns as channels (frame based)

Rate options: Enforce single-rate processing

[ ] Export filter coefficients to workspace

Figure G.24: Settings for the Simulink Raised Cosine Transmit Filter block.

Filter shape

The Filter shape parameter determines which type of filter the block uses; choices are Normal and Square root. The impulse response of a normal raised cosine filter with rolloff factor α and symbol period T_s is the same as shown in formula G.3.

The impulse response of a square root raised cosine filter with rolloff factor α is

$$p(t) = 4\alpha \frac{cos((1+\alpha)\pi t/T_s) + \frac{sin((1-\alpha)\pi t/T_s)}{4\alpha t/T_s}}{\pi\sqrt{T_s}(1-(4\alpha t/T_s)^2)} \tag{G.4}$$

The impulse response of a square root raised cosine filter convolved with itself is approximately equal to the impulse response of a normal raised cosine filter.

Rolloff Factor

Is the filter's rolloff factor. It must be a real number between 0 and 1. As mentioned before, the rolloff factor determines the excess bandwidth of the filter. For example, a rolloff factor of 0.5 means that the bandwidth of the filter is 1.5 times the input sampling frequency. Which means that the bandwidth of the filter is thus $1.5 * T_s$. We have two bits per symbol for a $\frac{\pi}{4}$ DQPSK modulation. That means that the

bit rate $f_b = 2 * symbol\ rate\ f_s$

And according to [14], so in this case means a bandwidth efficiency of approximately $\eta_{BW} = \frac{2}{1.5} \approx 1.33$ rather than the intended ideal value of 2.

Because the ideal raised cosine filter has an infinite impulse response, the block truncates the impulse response to the number of symbols that the **Filter span in symbols** parameter specifies. The Filter span in symbols, N, and the **Output samples per symbol**, L, determine the length of the filter's impulse response, which is

L∗Filter span in symbols +1

and figure G.25 shows the truncated magnitude response of the square raised cosine filter.

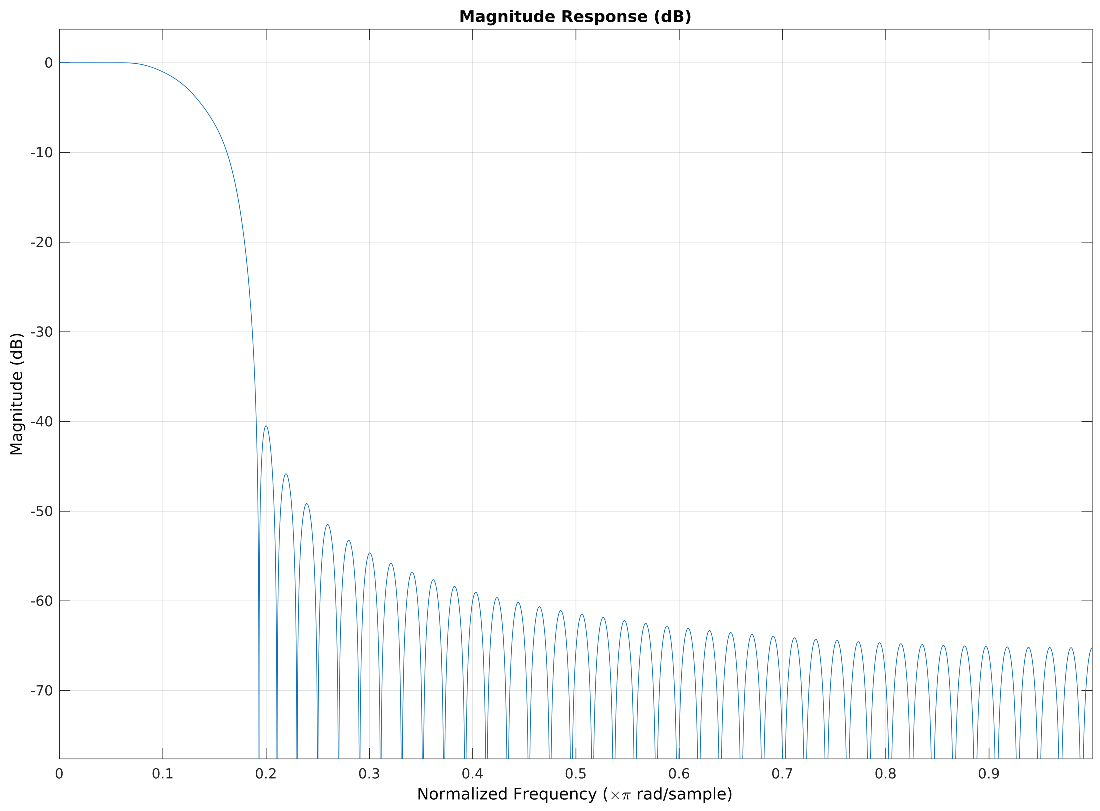


Figure G.25: Frequency response of the square root raised transmit filter.

Linear amplitude filter gain

Specify a positive scalar value that the block uses to scale the filter coefficients. By default, the block normalizes filter coefficients to provide unit energy gain. If you specify a gain other than 1, the block scales the normalized filter coefficients using the gain value you specify.

Input Signals and Output Signals

The input must be a discrete-time signal. This block accepts a column vector or matrix input signal. For information about the data types each block port supports, see the *Matlab Communications System Toolbox Reference* manual [22].

Input processing

Specify how the block processes the input signal. You can set this parameter to one of the following options:

- When you set the Input processing parameter to Columns as channels (frame based), the block treats a $M_i - by - N$ matrix input as N independent channels. The block processes each column of the input over time by keeping the frame size constant ($M_i = M_o$), while making the output frame period (T_{f_o}) L times shorter than the input frame period ($T_{f_o} = T_{f_i}/L$).
- When you set the Input processing parameter to Elements as channels (sample based), the block treats an M-by-L matrix input as M∗N independent channels, and processes each channel over time. The output sample period (T_{s_o}) is L times shorter than the input sample period ($T_{s_i} = T_{s_i}/L$), while the input and output sizes remain identical.

Rate options

Specify the method by which the block should upsample and filter the input signal. You can select one of the following options:

- *Enforce single-rate processing* — When you select this option, the block maintains the input sample rate, and processes the signal by increasing the output frame size by a factor of N. To select this option, you must set the Input processing parameter to Columns as channels (frame based).
- *Allow multi-rate processing* — When you select this option, the block processes the signal such that the output sample rate is N times faster than the input sample rate.

Export filter coefficients to workspace

Select this check box to create a variable in the MATLAB workspace that contains the filter coefficients. Appendix C shows the coefficients of the squared raised cosine transmit filter.

These are the most important settings provided for the filter. The other settings in the Data Types tab of the *Function Block Parameters* dialog box shown in Figure G.24 are left further default. For a detailed discussion of these parameters is beyond the scope of this report and therefore I refer to the Matlab Communications System Toolbox Reference manual [22].

G.5. Raised Cosine Receive Filter

In practice, the raised-cosine filter is decomposed into two sections, one placed in the transmitter and the other in the receiver [15], see Figure G.26. Using a transfer function equal to the square root of $P(f)$ shown in Figure G.22(b), the two sections together allow Nyquist signaling while the section used in the receiver also operates as a matched filter. This technique is described in [7].

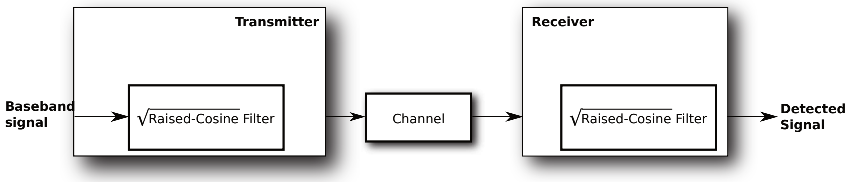


Figure G.26: Decomposition of a raised-cosine filter into two sections.

Having said this, means we also have to place a Raised Cosine Filter at the receiver side, See Figure F.1 and Figure F.3 of Appendix F.

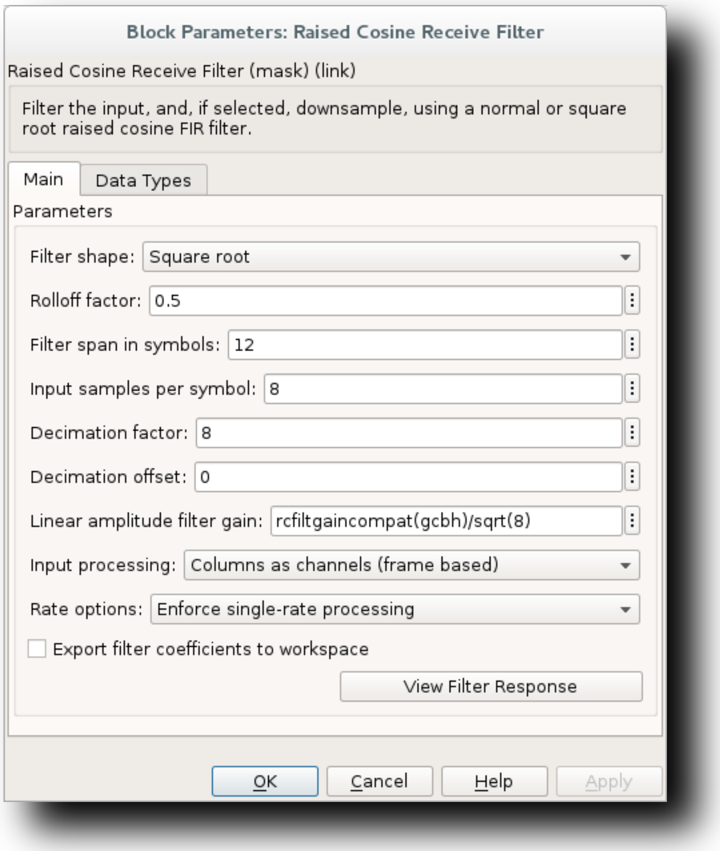
If you double-click the Simulink Raised Cosine Receive Filter, which incidentally has the exact same symbol as the Raised Cosine Transit filter in Figure G.17, you will see a dialog box as shown in Figure G.27.

Description

The *Raised Cosine Receive Filter* block filters the input signal using a normal raised cosine FIR filter or a square root raised cosine FIR filter. It also downsamples the filtered signal if you set the Output mode parameter to Downsampling. The FIR Decimation block implements this functionality. The *Raised Cosine Receive Filter* block's icon shows the filter's impulse response.

Characteristics of Filter

Characteristics of the raised cosine filter are the same as in the *Raised Cosine Transmit Filter* block, except that the length of the filter's input response has a slightly different expression: *L∗Filter span in symbols + 1*, where L is the value of the **Input samples per symbol** parameter (not the Output samples per symbol parameter, as in the case of the *Raised Cosine Transmit Filter block*). The block normalizes the filter coefficients to unit energy. If you specify a Liner amplitude filter gain other than 1, then the block scales the normalized filter coefficients using the gain value you specify.


Block Parameters: Raised Cosine Receive Filter

Raised Cosine Receive Filter (mask) (link)

Filter the input, and, if selected, downsample, using a normal or square root raised cosine FIR filter.

Filter shape:	Square root
Rolloff factor:	0.5
Filter span in symbols:	12
Input samples per symbol:	8
Decimation factor:	8
Decimation offset:	0
Linear amplitude filter gain:	rcfiltgaincompat(gcbh)/sqrt(8)
Input processing:	Columns as channels (frame based)
Rate options:	Enforce single-rate processing

☐ Export filter coefficients to workspace

Figure G.27: Settings for the Simulink Raised Cosine Receive Filter block.

Decimating the Filtered Signal

To have the block decimate the filtered signal, i.e., reducing the sampling rate of a signal, set the *Decimation factor* parameter to a value greater than 1. If K represents the **Decimation factor** parameter value, then the block retains 1/K of the samples, choosing them as follows:

- If the Decimation offset parameter is zero, then the block selects the samples of the filtered signal indexed by 1, K+1, 2∗K+1, 3∗K+1, etc.
- If the **Decimation offset** parameter is a positive integer less than M, then the block initially discards that number of samples from the filtered signal and downsamples the remaining data as in the previous case.

To preserve the entire filtered signal and avoid decimation, set **Decimation factor** to 1. This setting is appropriate, for example, when the output from the filter block forms the input to a timing phase recovery block such as Squaring Timing Recovery. The timing phase recovery block performs the downsampling in that case.

In our case, we want to restore the original signal from the transmitter and therefore the received signal is downsampled with the same factor as the upsampling factor of the Raised Cosine Transmit Filter, which is L=8 .

Input Signals and Output Signals

This block accepts a column vector or matrix input signal. For information about the data types each block port supports, see the *Matlab Communications System Toolbox Reference* manual [22]. If you set **Decimation factor** to 1, then the input and output signals share the same sampling mode, sample time, and vector length. If you set Decimation factor to K, which is greater than 1, then K and the input sampling mode determine characteristics of the output signal:

Single-Rate Processing

When you set the **Rate options** parameter to *Enforce single-rate processing*, the input and output of the block have the same sample rate. To generate the output while maintaining the input sample rate, the block resamples the data in each column of the input such that the frame size of the output (M_o) is $1/K$ times that of the input ($M_o = M_i/K$), In this mode, the input frame size, M_i, must be a multiple of K.

Multirate Processing

When you set the **Rate options** parameter to Allow *multirate processing*, the input and output of the block are the same size, but the sample rate of the output is K times slower than that of the input. When the block is in multirate processing mode, you must also specify a value for the Input processing parameter:

- When you set the **Input processing parameter** to *Elements as channels (sample based)*, the block treats an M-by-N matrix input as M$*$N independent channels, and processes each channel over time. The output sample period (T_{s_o}) is K times longer than the input sample period (T_{s_o}=K$*$ T_{s_i}), and the input and output sizes are identical.
- When you set the **Input processing parameter** to *Columns as channels (frame based)*, the block treats a M_i -by-N matrix input as N independent channels. The block processes each column of the input over time by keeping the frame size constant ($M_i = M_o$), and making the output frame period (T_{f_o}) K times longer than the input frame period (T_{f_o}=K $*$ T_{f_i}).

G.6. DQPSK Demodulator Baseband

Description

The DQPSK Demodulator Baseband block demodulates a signal that was modulated using the differential quaternary phase shift keying method.

The input is a baseband representation of the modulated signal. The input must be a discrete-time complex signal. The output depends on the phase difference between the current symbol and the previous symbol. The first integer (or binary pair, if you set the Output type parameter to Bit) at the block output is the initial condition of zero because there is no previous symbol.

This block accepts either a scalar or column vector input signal. For information about the data types each block port supports, see Matlab Communications System Reference Manual [22].

The Simulink symbol of the DQPSK Demodulator Baseband block is shown in Figure G.28 below.

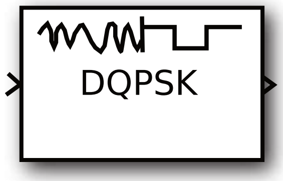


Figure G.28: Simulink DQPSK Demodulator Baseband block.

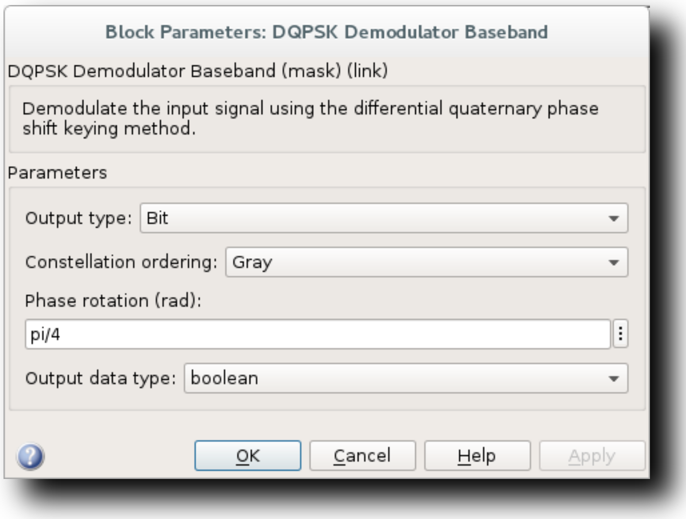
Outputs and Constellation Types

When you set **Output type** parameter to *Integer*, the block maps a phase difference of

$\theta + \pi m/2$

to m, where θ represents the Phase rotation parameter and m is 0, 1, 2, or 3. When you set the **Output type** parameter to *Bit*, then the output contains pairs of binary values. The reference page for the *DQPSK Modulator Baseband* block shows which phase differences map to each binary pair, for the cases when the **Constellation ordering** parameter is either *Binary or Gray*.

When you double-click on the icon then you get to see a dialog box as shown in Figure G.29.


Block Parameters: DQPSK Demodulator Baseband

DQPSK Demodulator Baseband (mask) (link)

Demodulate the input signal using the differential quaternary phase shift keying method.

Parameters

Output type: Bit

Constellation ordering: Gray

Phase rotation (rad):

pi/4

Output data type: boolean

Figure G.29: Settings for the Simulink DQPSK Demodulator Baseband block.

Output type

Determines whether the output consists of integers or pairs of bits.

Constellation ordering

Determines how the block maps each integer to a pair of output bits.

Phase rotation (rad)

This phase difference between the current and previous modulated symbols results in an output of zero.

Output data type

When the parameter is set to *Inherit via internal rule* (default setting), the block will inherit the output data type from the input port. The output data type will be the same as the input data type if the input is of type single or double.
For integer outputs, this block can output the data types int8, uint8, int16, uint16, int32, uint32, single, and double. For bit outputs, output can be int8, uint8, int16, uint16, int32, uint32, boolean, single, or double.
But as you can see the settings for the DQPSK Demodulator Baseband block are exactly the same as the settings for the DQPSK Modulator Baseband block (Figure G.7). Except the fact that the output data type is inherited from the input port.

G.7. Bit Error Rate (BER) measurement

The output from the *Bernoulli Binary Generator* is the transmitted signal *T Signal* and the output from the *DQPSK Demodulator Baseband* is the received signal *R Signal*.
And between the two signals, the delay is determined in the Find Delay block. This is useful when you want to compare a transmitted and received signal to find the bit error rate, but do not know the delay in the received signal. Because you have to find this Delay Before Calculating an Error Rate.

We use this *Receive delay in the Error Rate Calculation* block to compare input data from the Transmitters Bernoulli Binary Generator with input data from the receivers DQPSK Demodulator Baseband. It calculates the error rate as a running statistic, by dividing the total number of unequal pairs of data elements by the total number of input data elements from one source.

Then we use the *Simulink Display* block to show the input data from the *Error Rate Calculation* block. The block output is a three-element vector consisting of the error rate, followed by the number of errors detected and the total number of symbols compared.

See further the bottom left area of Figure F.1 and Figure F.3 of Appendix F how the above mentioned signals and the blocks are connected.

The use of these blocks actually speaks for itself and a detailed explanation is beyond the scope of this report and therefore I refer to the *Matlab Communication Systems Reference* manual [22] for a detailed description and application.

H

Signals from Simulink $\frac{\pi}{4}$ DQPSK transceiver system model

Modulation	$\frac{\pi}{4}$ DQPSK
Channel Bandwidth	300 kHz
Symbol Rate	$187.5 kS/sec$
Bandlimiting Filter	SRRC with rolloff factor $\alpha = 0.5$

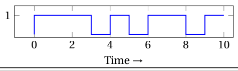
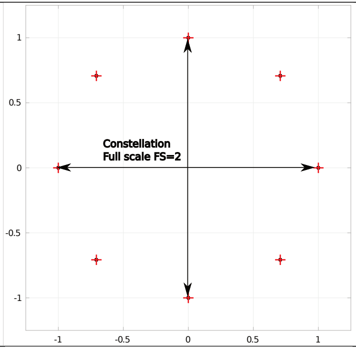
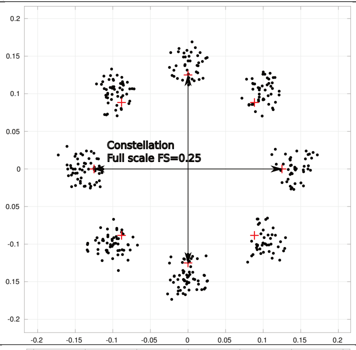
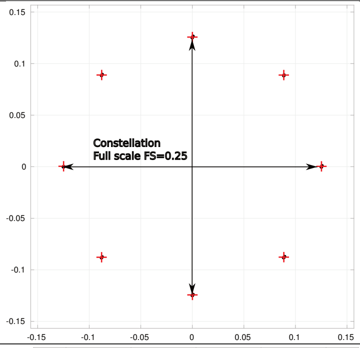
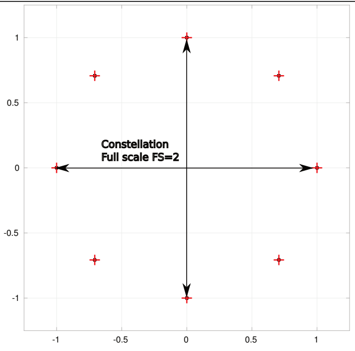
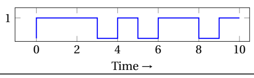
Output signal from device:	**Signal display**	**data type**
Bernoulli Binary Generator		Boolean
DQPSK Modulator		double
SRRC Transmit Filter		double
SRRC Receive Filter		double
Phase-Domain ADC		double
DQPSK Demodulator		Boolean

Table H.1: Table showing signals from Simulink $\frac{\pi}{4}$ DQPSK transceiver system model.

I

Relation between SNR, E_b/N_o and E_s/N_o

I.1. Introduction

There quite seems to be some misunderstanding about the relationship between signal-to-noise ratio SNR, bit-energy-to-noise density E_b/N_o, or for higher dimensions (i.e., multiple bits per symbol) the symbol-energy-to-noise density E_s/N_o. These terms for signal-to-noise ratios are some of the most misused, or at least sloppily used, terms in Digital Wireless Communication (DWC) [13].
The SNR is general to both analog and digital types of communications signals. The second and third term, E_b/N_o and E_s/N_o, is specific to digital communications and is not defined for any type of analog modulation. These relationships will be shown in this Appendix.

I.2. Signal-to-noise ratio (SNR)

SNR is historically defined as a ratio of signal power (P_s) defined in a specified bandwidth, over the locally present noise power (P_n) *measured in the same bandwidth.* Being a ratio of identical units (watts, in this case), SNR is dimensionless. Also, being a ratio of identical units, SNR can be expressed in decibels. Definitions for these terms are described mathematically as

$$snr = \frac{P_s}{P_n}, \text{ and } SNR = 10 * log_{10}(\frac{P_s}{P_n}) \; dB \tag{I.1}$$

I.3. bit-energy-to-noise density $\mathbf{E_b/N_o}$

The fundamental purpose of DWC is to communicate quantized information — bits — from one location to another. It seems natural, then, to have a comparison metric available among DWC signals that directly relates to this fundamental objective. Unlike snr (from section I.2), which is a power ratio and therefore averages the communication signal for possibly infinite time, we instead need a metric that is based upon the quantized, finite intervals central to DWC.
The E_b/N_o measures was developed because it provides this very useful normalization among all DWC signals, allowing meaningful comparison among them. It is a significant ratio because the performance (i.e., bit-error rate) of many common digital communication signals is a function of this ratio. While related to snr – is NOT equal to snr. The use in DWC literature of the term SNR, or even the words "signal-to-noise ratio", when E_b/N_o is actually being used is a major point of confusion and ambiguity. Fortunately, getting these terms confused in reverse order is rarely, if ever, seen. But getting them confused at all is not acceptable.
Let us evaluate this difference. The E_b/N_o measures states that the normalized performance of a data communications system is independent of bandwidth. All that really matters is the input density of noise power

across frequency (assumed to be constant) and the received signal energy per symbol. This is very different from signal-to-noise ratio. As bandwidth is increased, SNR will decrease while E_b/N_o will not. Thus, operation at very low values of snr are definitely possible.

We all know from the fundamental physics that $Energy = Power * Time$. Using SI units, lets verify the following equations.

Energy per bit (E_b) is defined as the ratio of signal power and bit-rate.

$$E_b = \frac{P_s}{f_b}\left[\frac{Watt}{\frac{bits}{second}}\right] = \left[\frac{Watt * second}{bits}\right] = \left[\frac{joule}{bit}\right] \tag{I.2}$$

Where,

f_b = bit rate in bits/second
E_b = Energy per bit in joule/bit
P_s = Total signal Power in Watt

Now, introducing the noise power spectral density $N_o\left[\frac{Watt}{Hz} = joule\right]$ in Equation I.2, which is assumed to be constant across frequency, then the noise power P_n in bandwidth $B\,[Hz = 1/sec]$ is

$$P_n = N_o * B\left[joule/sec = Watt\right] \tag{I.3}$$

where,

N_o = noise power spectral density in Watt/Hz
B = bandwidth required for the signal in Hz
P_n = noise power in the signal bandwidth in Watt

The usual relationship relating E_b/N_o to *snr holds only for uncoded binary signals*, and is the simple expression

$$\frac{E_b}{N_o} = \frac{P_{s*T_s}}{P_n/B} \tag{I.4}$$

for a binary signal, so $T_s = \frac{1}{f_s} = T_b = \frac{1}{f_b}$,

where,

T_s = Symbol time. The time-duration that an information-symbol is mapped onto the signal, which equals the time duration of a signal-state; unit is seconds (usually microseconds)
T_b = Bit time. The time-duration of an input binary bit: unit is seconds (usually microseconds)
State Duration = The time-duration of a physical signal state, equal to the symbol time.
f_s = Symbol-rate: The reciprocal of symbol-time, equal to the number of signal states transmitted per second of time. Unit is baud.*

*The unit *baud* (Bd) is an offical SI unit for symbol rate. Baud is named in honor of J.M. Emile Baudot (1845-1903) who established a five-bits-per-character code for telegraph use which became an international standard (commonly called the Baudet code)

f_b = Bit rate: The reciprocal of bit time, the number of *binary bits* transmitted per symbol: unit is bps (bits per second).

Note: it is important to understand the difference between bit-rate and symbol-rate. The signal bandwidth for the communications channel needed depends on the symbol rate, not on the bit-rate.

$Symbol\ rate\ (T_s) = \frac{bit\ rate(T_b)}{the\ number\ of\ bits\ transmitted\ with\ each\ symbol}$

In general for higher signal dimensions the ratio is

$$\frac{E_s}{N_o} = \frac{P_{s*T_s}}{P_n/B} \tag{I.5}$$

for any signal. And the Energy per symbol (E_s) is defined as the ratio of signal power and symbol rate:

$$E_s = \frac{P_s}{f_s}\left[\frac{Watt}{\frac{symbols}{second}}\right] = \left[\frac{Watt*second}{symbol}\right] = \left[\frac{joule}{symbol}\right] \tag{I.6}$$

Units for the ratio (I.4) are:

$\frac{E_b}{N_o} = \frac{P_s*T_s}{P_n/B} = \frac{[Watt][second]}{[Watt]/[1/second]} = \frac{[Watt][second]}{[Watt]/[second]} = \frac{joule}{joule}$, a dimensionless ratio of energies. Also being of identical units, can be expressed in decibels. Defined mathematically we have

$$\left.\frac{E_b}{N_o}\right|_{dB} = 10*log_{10}\left(\frac{E_b}{N_o}\right)\ dB \tag{I.7}$$

Relating the term $\frac{E_b}{N_o}$ to snr is possible, nothing that the ratio P_s/P_n does exist within its definition (I.4). Manipulating this definition leads to the relation

$$\frac{E_b}{N_o} = \frac{P_s*T_s}{P_n/B} = \frac{P_s}{P_n}*\frac{T_s}{1/B} = snr*\left(\frac{B}{f_s}\right) = \frac{snr}{f_s/B} \tag{I.8}$$

So given the above assumptions, for an uncoded binary signal the value of E_b/N_o is related to the snr by the *bandwidth efficiency* or *spectral efficiency* (see section I.4). E_b/N_o is snr normalized to the *spectral efficiency*. Under the special and very specific case that the signal *bandwidth efficiency* is unity, then and only then will the values of E_b/N_o and snr be numerically equal. Indeed, if a system is operating at the Nyquist bandwidth $B = f_s/2$, then the E_b/N_o value will be one-half of the snr.

Most DWC signals are not binary, but have each symbol representing several information bits. In such cases the calculation is most appropriately done on a per-symbol basis. Applying the definition above then provides a different measure – the symbol energy to noise density ratio (I.5)

$$\frac{E_s}{N_o} = log_2M\left(\frac{E_b}{N_o}\right) = \frac{P_s*(T_s log_2M)}{P_n/B} = \frac{P_s}{P_n}*\frac{T_s log_2M}{1/B} = snr*\frac{T_s}{1/B}log_2M \tag{I.9}$$

M is the number of states available to the signal, which is an extremely important characteristic of an DWC system. On a constellation diagram, it represents the number of constellation points. The relationship between constellation points and number of bits per symbol is

M = 2^n where M = number of constellation points and n = bits/symbol or $n = log_2\,(M)$

I.4. Spectral (Bandwidth) efficiency

Bandwidth efficiency describes how efficiently the allocated bandwidth is utilized or the ability of a modulation scheme to accommodate data, within a limited bandwidth. *Bandwidth efficiency* can be viewed with one of the meanings by considering only the on-air operation. It is actually an effective metric on the modulation technique used, because only *modulation characteristics* (constellation and filtering) matter to this result. There is no differentiation between whether a bit used in the modulation is actual payload information or overhead due the coding, protocol, etc. Bandwidth efficiency as a modulation metric is simply the number of bits per second that are transmitted within the (occupied) spectrum bandwidth.

$$\eta_{BW} = \frac{R}{B} \;\; bits/(sec\ Hz(BW)) \tag{I.10}$$

where R is the actual on-air data rate, and B is the channel bandwidth. If the radio had a perfect (i.e., a brick wall or rectangular in the frequency domain) filter, then the occupied bandwidth could be made equal to the symbol rate, i.e., $B = f_s$. And if we assume the data rate equal to the bit rate, i.e., $R = f_b$ then the *bandwidth efficiency* becomes

$$\eta_{BW} = \frac{f_b}{f_s} \;\; bits/(sec\ Hz(BW)) \tag{I.11}$$

but this is impossible to realize in practical radios since they require perfect modulators, demodulators, filter, and transmission paths.

Lets show a practical example of bandwidth efficiency in radios:

Example:

The TDMA version of the North American Digital Cellular (NADC) system, achieves a 48 Kbits-per-second data rate over a 30kHz bandwidth or 1.6 bits per second per Hz. It is a $\frac{\pi}{4}$ DQPSK (Differential Quadrature Phase Shift Keying) based system and transmits two bits per symbol. The theoretical efficiency would be two bits per second per Hz and in practice it is 1.6 bits per second per Hz.
The table I.1 shows the theoretical bandwidth efficiency limits for the main modulation types.

Modulation format	**Theoretical bandwidth efficiency ($\eta_{\mathbf{BW}}$) limits**
MSK	1 bit/second/Hz
BPSK	1 bit/second/Hz
QPSK	2 bit/second/Hz
8PSK	3 bit/second/Hz
16 QAM	4 bit/second/Hz
32 QAM	5 bit/second/Hz
64 QAM	6 bit/second/Hz
256 QAM	8 bit/second/Hz

Table I.1: Theoretical bandwidth efficiency limits for main modulation types.

Another important example shows why we wouldn't use snr in performance computations.

Example:

Consider comparing two signals S_1 and S_2 with the same bit rate f_b and energy per bit E_b but with varying spectral efficiencies $\eta_{BW_1} = f_b/f_s = 1$ bit/sec /Hz and $\eta_{BW_1} = f_b/f_s = 2$ bit/sec /Hz:

Using equations (I.1), (I.2), (I.3) and (I.11) we can fill in the equations as shown in Table I.2.

Qty	**Signal S$_1$**	**Signal S$_2$**	**Units**
p_s	$E_b * f_b$	$E_b * f_b$	[watt]
B	f_b	$f_b/2$	[Hz]
P_n	$N_o * f_b$	$N_o * f_b/2$	[watts]
snr	E_b/N_o	2*E_b/N_o	[none]
E_b/N_o	E_b/N_o	E_b/N_o	[none]

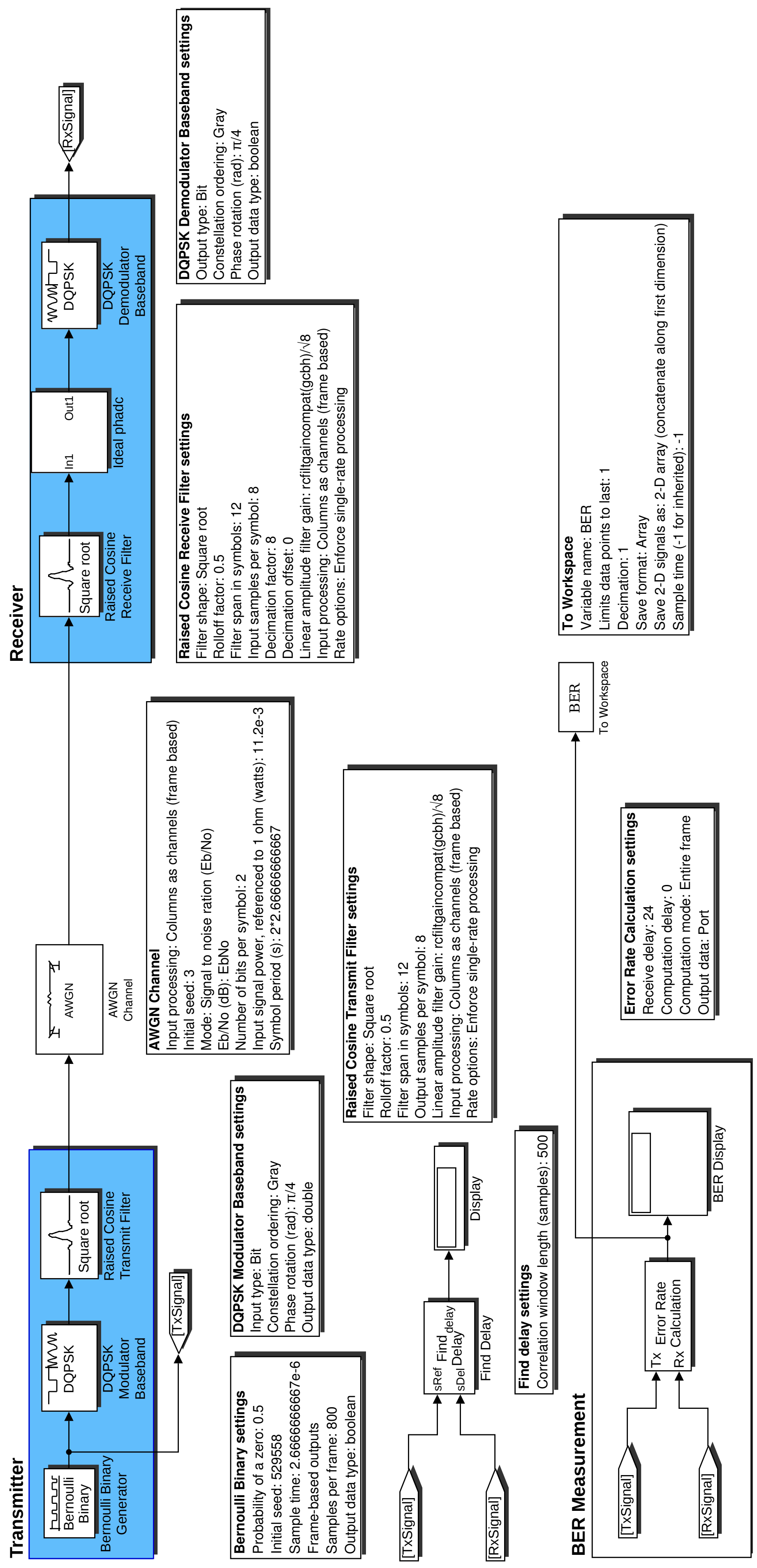
Table I.2: Performance computations of signals S_1 en S_2 with respectively spectral efficiencies of η_{BW_1}=1 and η_{BW_2}=2.

The point here is that the spectral efficiency caused signal S_2 to have a higher snr even though the transmitted power was the same due to the fact that noise bandwidth is half and thus the noise power is half. However, this would not have created any better performance in terms of bit-error rate since its Eb/N_o is the same as signal S_1.

J

MATLAB Simulink transceiver model to simulate BER as a function of E_b/N_o

J.1. Model for frame-based simulation



Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.66666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

AWGN Channel
Input processing: Columns as channels (frame based)
Initial seed: 3
Mode: Signal to noise ration (Eb/No)
Eb/No (dB): EbNo
Number of bits per symbol: 2
Input signal power, referenced to 1 ohm (watts): 11.2e-3
Symbol period (s): 2*2.66666666667

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

Find delay settings
Correlation window length (samples): 500

Raised Cosine Receive Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Input samples per symbol: 8
Decimation factor: 8
Decimation offset: 0
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

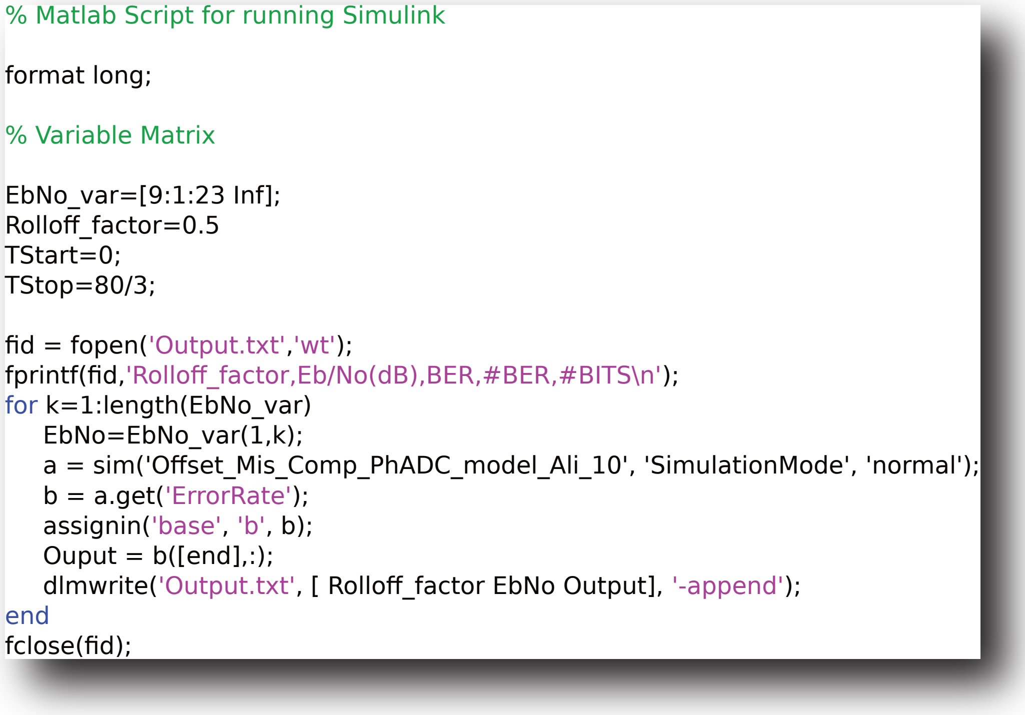
DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

To Workspace
Variable name: BER
Limits data points to last: 1
Decimation: 1
Save format: Array
Save 2-D signals as: 2-D array (concatenate along first dimension)
Sample time (-1 for inherited): -1

Figure J.1: Simulink transceiver model to simulate bit-error rate as a function of E_b/N_o.

J.1.1. Matlab code for BER as a function of E_b/N_o simulation

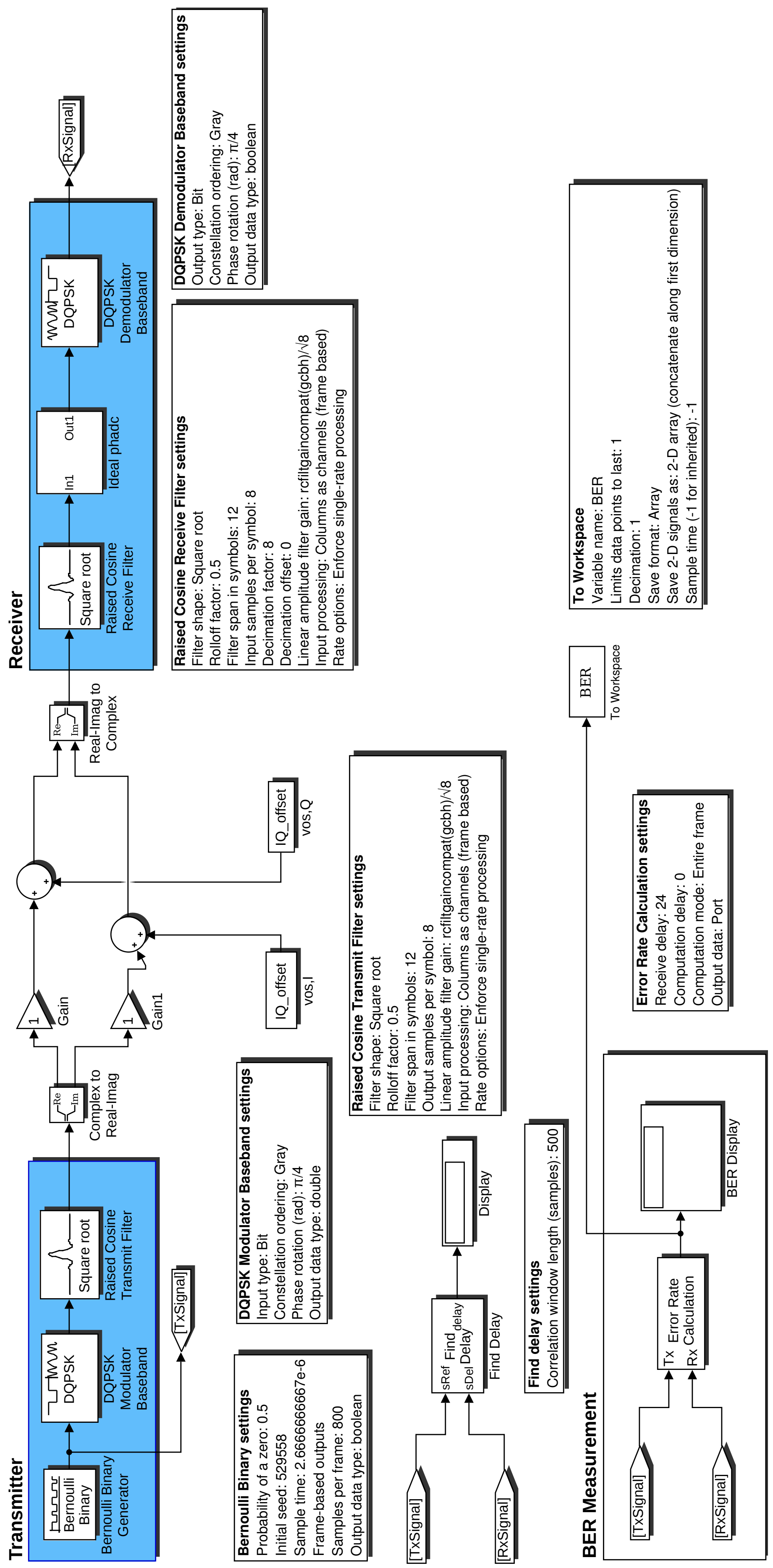


```
% Matlab Script for running Simulink

format long;

% Variable Matrix

EbNo_var=[9:1:23 Inf];
Rolloff_factor=0.5
TStart=0;
TStop=80/3;

fid = fopen('Output.txt','wt');
fprintf(fid,'Rolloff_factor,Eb/No(dB),BER,#BER,#BITS\n');
for k=1:length(EbNo_var)
    EbNo=EbNo_var(1,k);
    a = sim('Offset_Mis_Comp_PhADC_model_Ali_10', 'SimulationMode', 'normal');
    b = a.get('ErrorRate');
    assignin('base', 'b', b);
    Ouput = b([end],:);
    dlmwrite('Output.txt', [ Rolloff_factor EbNo Output], '-append');
end
fclose(fid);
```

Figure J.2: Matlab code to simulate bit-error-rate as a function of channel noise E_b/N_o.

K
MATLAB Simulink transceiver model to simulate BER as a function of IQ offset

K.1. Model for BER as a function of IQ offset simulation



Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

Find delay settings
Correlation window length (samples): 500

Raised Cosine Receive Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Input samples per symbol: 8
Decimation factor: 8
Decimation offset: 0
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

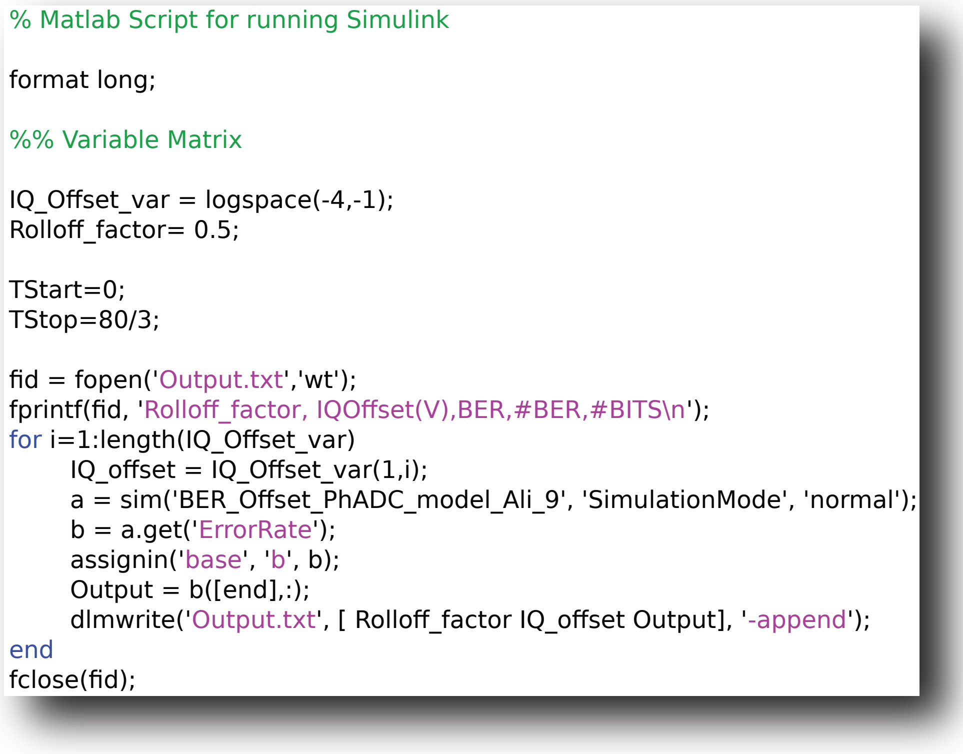
DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

To Workspace
Variable name: BER
Limits data points to last: 1
Decimation: 1
Save format: Array
Save 2-D signals as: 2-D array (concatenate along first dimension)
Sample time (-1 for inherited): -1

Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

Figure K.1: Simulink transceiver model to simulate bit-error rate as a function of IQ offset.

K.1.1. Matlab code for BER as a function of IQ offset simulation



```
% Matlab Script for running Simulink

format long;

%% Variable Matrix

IQ_Offset_var = logspace(-4,-1);
Rolloff_factor= 0.5;

TStart=0;
TStop=80/3;

fid = fopen('Output.txt','wt');
fprintf(fid, 'Rolloff_factor, IQOffset(V),BER,#BER,#BITS\n');
for i=1:length(IQ_Offset_var)
        IQ_offset = IQ_Offset_var(1,i);
        a = sim('BER_Offset_PhADC_model_Ali_9', 'SimulationMode', 'normal');
        b = a.get('ErrorRate');
        assignin('base', 'b', b);
        Output = b([end],:);
        dlmwrite('Output.txt', [ Rolloff_factor IQ_offset Output], '-append');
end
fclose(fid);
```

Figure K.2: Matlab code to simulate bit-error-rate as a function of IQ offset.

L

MATLAB Simulink transceiver model to simulate BER as a function of IQ offset and noise

L.1. Model for BER as a function of IQ offset and noise simulation

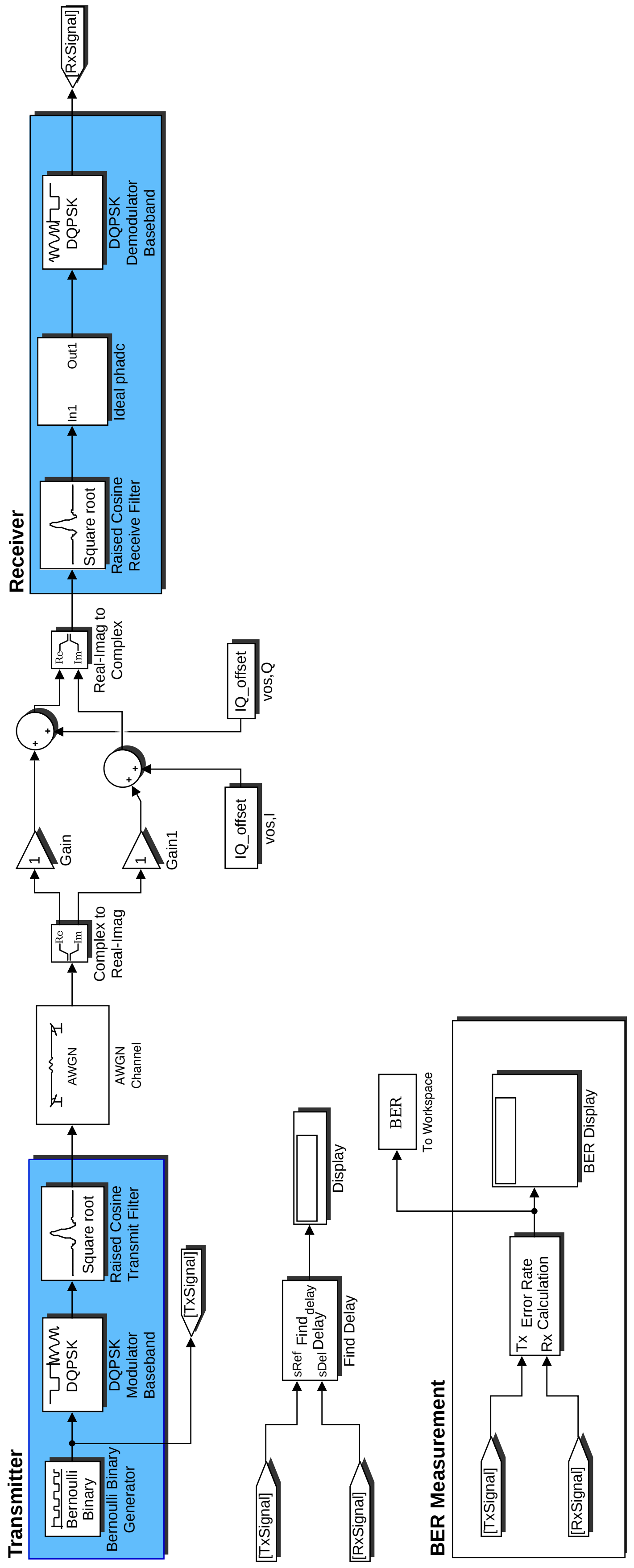
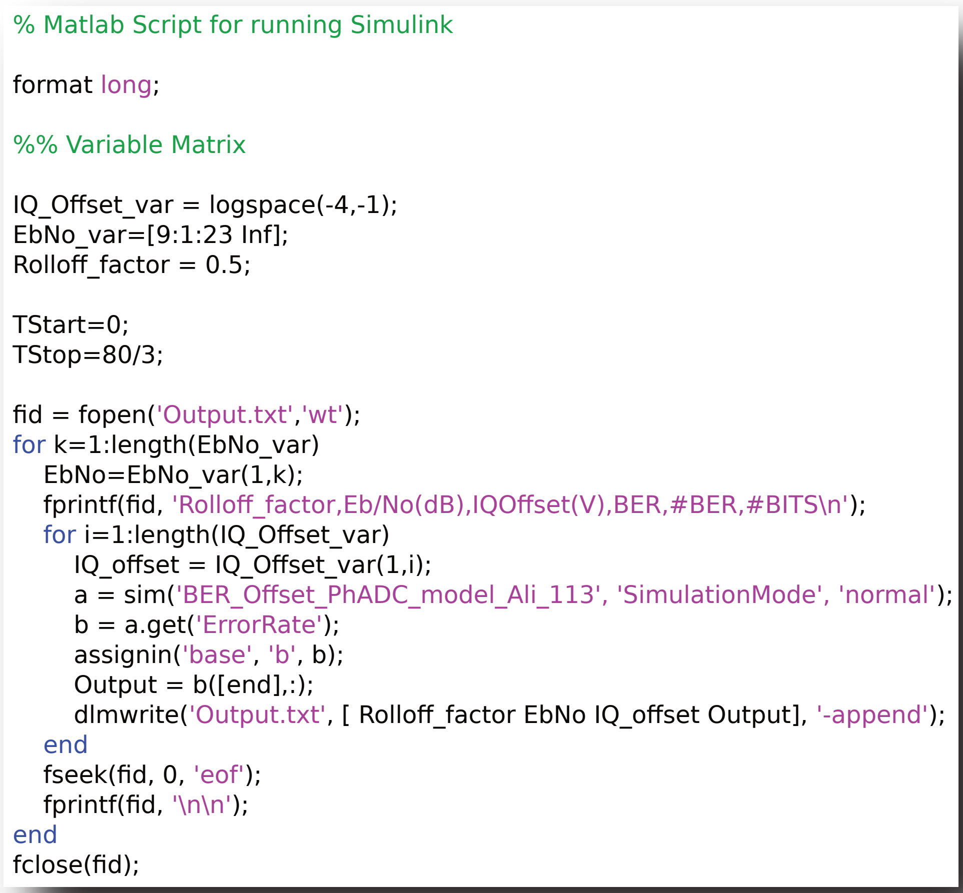


Figure L.1: Simulink transceiver model to simulate bit-error rate as a function of IQ offset and noise.

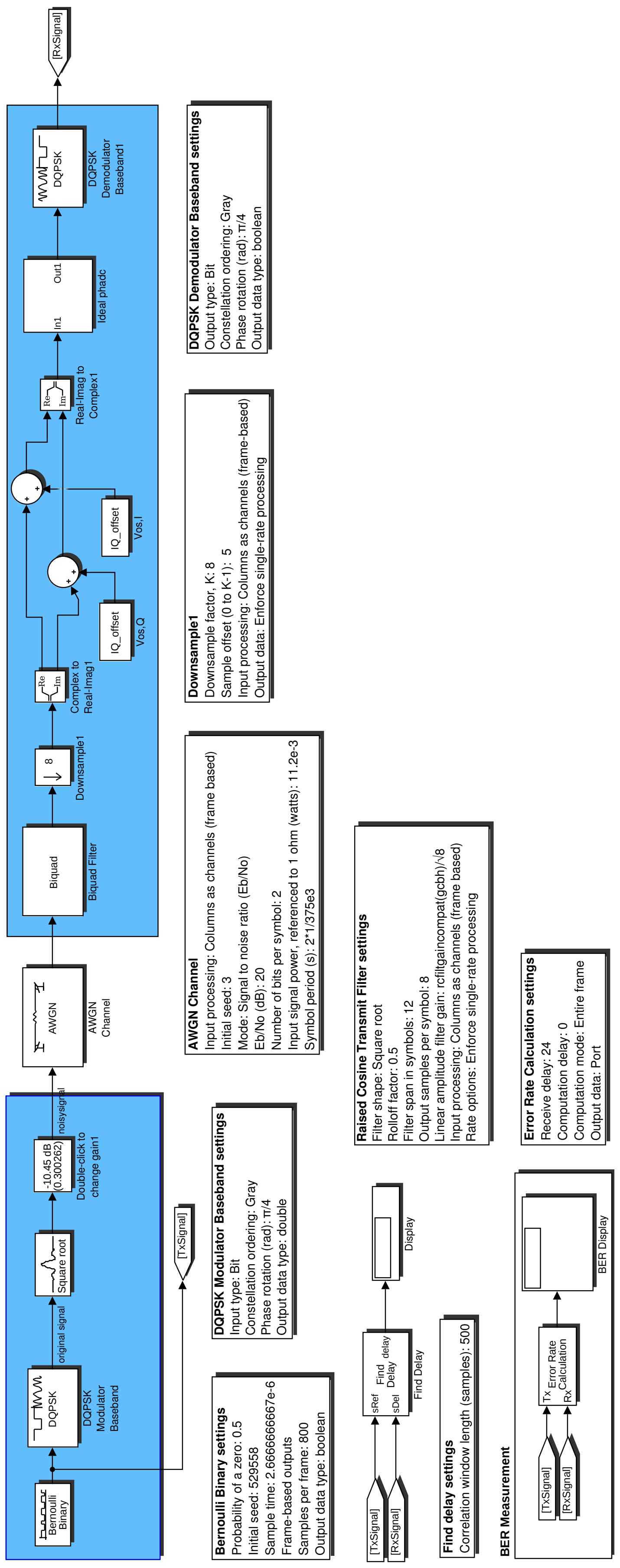
L.1.1. Matlab code for BER as a function of IQ offset noise simulation



```matlab
% Matlab Script for running Simulink

format long;

%% Variable Matrix

IQ_Offset_var = logspace(-4,-1);
EbNo_var=[9:1:23 Inf];
Rolloff_factor = 0.5;

TStart=0;
TStop=80/3;

fid = fopen('Output.txt','wt');
for k=1:length(EbNo_var)
   EbNo=EbNo_var(1,k);
   fprintf(fid, 'Rolloff_factor,Eb/No(dB),IQOffset(V),BER,#BER,#BITS\n');
   for i=1:length(IQ_Offset_var)
      IQ_offset = IQ_Offset_var(1,i);
      a = sim('BER_Offset_PhADC_model_Ali_113', 'SimulationMode', 'normal');
      b = a.get('ErrorRate');
      assignin('base', 'b', b);
      Output = b([end],:);
      dlmwrite('Output.txt', [ Rolloff_factor EbNo IQ_offset Output], '-append');
   end
   fseek(fid, 0, 'eof');
   fprintf(fid, '\n\n');
end
fclose(fid);
```

Figure L.2: Matlab code to simulate bit-error-rate as a function of IQ offset and noise.

L.2. Model for BER as a function of IQ offset and noise simulation using a biquad receive filter



Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

AWGN Channel
Input processing: Columns as channels (frame based)
Initial seed: 3
Mode: Signal to noise ratio (Eb/No)
Eb/No (dB): 20
Number of bits per symbol: 2
Input signal power, referenced to 1 ohm (watts): 11.2e-3
Symbol period (s): 2*1/375e3

Downsample1
Downsample factor, K: 8
Sample offset (0 to K-1): 5
Input processing: Columns as channels (frame-based)
Output data: Enforce single-rate processing

DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

Find delay settings
Correlation window length (samples): 500

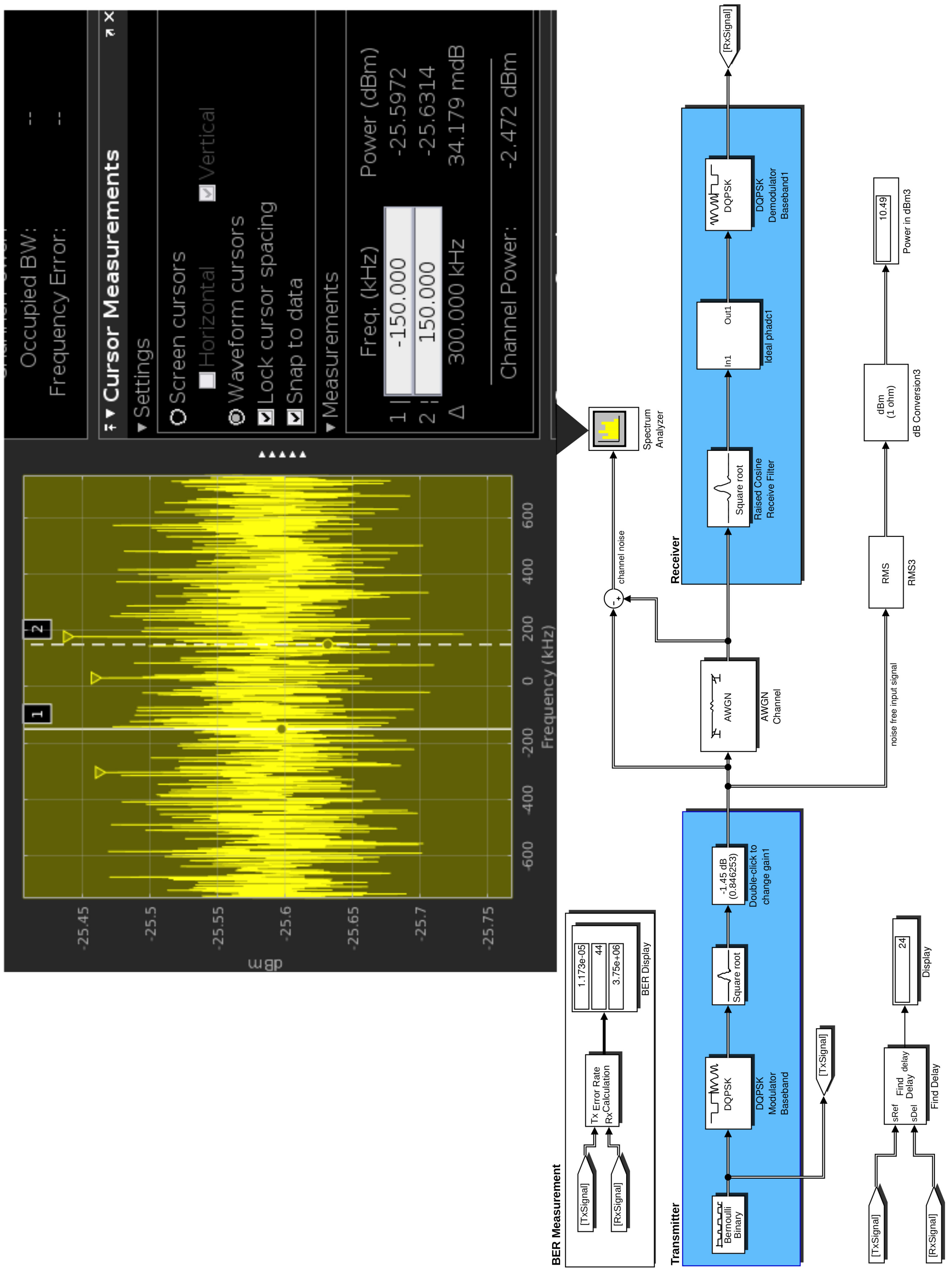
Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

Figure L.3: Simulink transceiver model to simulate bit-error rate as a function of IQ offset and noise using a biquad receive filter.

M

Simulink $\frac{\pi}{4}$ DQPSK transceiver system model for signal to noise ratio simulation

M.1. Model for signal to noise ratio simulation


Occupied BW: --

Frequency Error: --

Cursor Measurements

Settings

- ○ Screen cursors
  - ☐ Horizontal
  - ☑ Vertical
- ● Waveform cursors
- ☑ Lock cursor spacing
- ☑ Snap to data

Measurements

	Freq. (kHz)	Power (dBm)
1	-150.000	-25.5972
2	150.000	-25.6314
Δ	300.000 kHz	34.179 mdB

Channel Power: -2.472 dBm

Figure M.1: Simulink model for simulation of signal to noise ratio.

N Cadence Verilog-AMS system components models

N.1. Verilog-AMS model for the phase-domain ADC

```
// N-bit Phase Analog to Digital Converter (Phadc)
`include "disciplines.vams"
`include "constants.vams"

module phaseadc (Phaseout, Iin, Qin, clk);
  `define bits  5 // resolution (bits)

  //parameter integer bits = 5;   // resolution (bits)
  parameter real fullscale = 1;   // input range is from 0 to fullscale (V)
  parameter real td = 0;          // delay from clock edge to output (s)
  parameter real tt = 1e-9;       // transition time of output (s)
  parameter real AVDD =1.2;       // voltage level of logic 1 (V)
  parameter real thresh = AVDD/2; // logic threshold level (V)
  parameter integer dir = 1 from [-1:1] exclude 0; // 1 for rising edges, −1 for
falling
  parameter real pi = `M_PI;
  input Iin, Qin, clk;
  output [0:`bits-1] Phaseout;
  //voltage Iin, Qin, clk;
  //voltage [0:bits-1] Phaseout;
  electrical Iin, Qin, clk;
  electrical [`bits-1:0] Phaseout;
  real sample_I, sample_Q, midpoint;
  real result[`bits-1:0];
  real phi;
  //reg [4:0] phitob;
  genvar i;


  analog begin
    @(cross(V(clk)-thresh, dir) or initial_step) begin
      sample_I = V(Iin);
      sample_Q = V(Qin);
      midpoint = fullscale/2.0;
      phi = (atan2(sample_Q, sample_I) + pi)/(2*pi);
      for (i = `bits-1; i >= 0; i = i - 1 ) begin
        if (phi > midpoint) begin
          result[i] = AVDD;
          phi = phi - midpoint;
        end else begin
          result[i] = 0.0;
        end
        phi = 2.0*phi;
      end
    end
    for (i = 0; i < `bits; i = i + 1) begin
        V(Phaseout[i]) <+ transition(result[i], td, tt);
    end
    end
endmodule
```

Figure N.1: VerilogAMS code PhADC.

N.2. Verilog-AMS model for the two-quadrant detector

```
//Verilog-AMS HDL for "IQoffsetcompensation", "IQoffsetcompensator2" "verilogams"

// N-bit IQ phase offset compensator

`include "constants.vams"
`include "disciplines.vams"

module TwoQuadrantDetector (out, in, clk);
  `define bits 5 // resolution (bits)
  parameter real fullscale = 1.2; // output range is from 0 to fullscale (V)
  parameter real td = 0; // delay from clock edge to output (s)
  parameter real tt = 1e-9; // transition time of output (s)
  parameter real DVDD = 1.2;  // voltage level of logic 1 (V)
  parameter real thresh = DVDD/2; // logic threshold level (V)
  parameter integer dir = 1 from [-1:1] exclude 0;// 1 for rising edges, −1 for
falling
  parameter real zeropi = 0;
  parameter real halvepi = 0.3;
  parameter real onepi = 0.6;
  parameter real minushalvepi = -0.3;
  parameter real minusonepi = -0.6;
  output [1:0] out; // out[0] used to feedback offset for I and out[1] used to
feedback offset for Q
  input clk;
  input [`bits-1:0] in;
  electrical clk;
  electrical [1:0] out;
  electrical [`bits-1:0] in;
  real aout;
  real result4I;
  real result4Q;
  integer weight;
  genvar i;

  analog begin
    @(cross(V(clk) - thresh, dir) or initial_step) begin
    aout = 0;
    weight = 2;
    for (i = `bits - 1; i >= 0; i = i - 1 ) begin
      if (V(in[i]) > thresh) begin
          aout = aout + fullscale/weight;
      end
      weight = weight*2;
    end
    aout = aout - 0.6;
    if (aout >= minushalvepi && aout < halvepi) begin
        result4I = (DVDD - thresh)*2;
    end
    if (aout >= minusonepi && aout < minushalvepi || aout >= halvepi && aout <=onepi)
begin
       result4I = 0;
    end
    if (aout >= zeropi && aout <= onepi) begin
       result4Q = (DVDD-thresh)*2;
    end
    if (aout >= minusonepi && aout < zeropi) begin
       result4Q = 0;
    end
   end
    V(out[0]) <+ transition(result4I, td, tt);
    V(out[1]) <+ transition(result4Q, td, tt);
  end
```

Figure N.2: VerilogAMS implementation of the Two Quadrant detector.

N.3. Verilog-AMS model for the digital integrator (Up/Down Counter)

```
//Verilog-AMS HDL for "IQoffsetcompensation", "UpDownCounter_3bits" "verilogams"

`include "constants.vams"
`include "disciplines.vams"

module UpDownCounter_8bits (out, clk, in);
`define bits 8
    parameter real td = 0;        // delay from clock edge to output (s)
    parameter real tt = 1e-9;     // transition time of output (s)
    parameter real DVDD = 1.2;    // voltage level of logic 1 (V)
    parameter real thresh = DVDD/2;  // logic threshold level (V)
    parameter integer dir = 1 from [-1:1] exclude 0; // 1 for rising edges, -1 for
falling
    parameter real vhigh = DVDD;//logic high level
    parameter real vlow = 0;//logic low level
    parameter integer setval = 1<<(`bits-1) from [0:(1<<`bits)-1];
    parameter integer outval_max = (1<<`bits)-1;
    parameter integer outval_min = 0;
    input clk;
    input in;
    output [`bits-1:0] out;
    electrical clk, in;
    electrical [`bits-1:0] out;
    integer up; //0=increasing 1=decreasing
    integer outval;
    integer outval2;
    genvar j;


    analog begin
      @(initial_step("static","ac")) outval = setval;
      @(cross(V(clk)-thresh, dir)) begin
       if (V(in) > thresh && outval < outval_max) begin
              up = 1;
              outval = (outval+(+up-!up))%(1<<`bits);
       end
       else if (V(in) < thresh && outval > outval_min) begin
              up = 0;
              outval = (outval+(+up-!up))%(1<<`bits);
       end
      end
      outval2 = !(outval&(1<<(`bits-1))) * (~outval + setval) + !(!(outval&(1<< (`bits-1)))) * outval;
      for (j=`bits-1; j >= 0; j = j-1) begin
        V(out[j]) <+ transition(!(!(outval2&(1<<j)))*vhigh+!(outval2&(1<<j))*vlow,td, tt);
      end
    end
endmodule
```

Figure N.3: VerilogAMS implementation of the Up/Down counter.

N.4. Verilog-AMS models for the Current-steering DAC

N.4.1. Verilog-AMS model for the current source

```
//Verilog-AMS HDL for "IQoffsetcompensation", "current_source" "verilogams"

`include "constants.vams"
`include "disciplines.vams"

// Current Source with Shunt Conductance
//
// Generates DC and AC stimulus.

module current_source(p, n);

inout p, n;
electrical p, n;
parameter real dc=0;
parameter real g=0 from [0:inf);
parameter real mag=0;

analog begin
    I(p,n) <+ g*V(p,n) + dc + ac_stim( "ac", mag );
end
endmodule
```

Figure N.4: VerilogAMS code of current source.

N.4.2. Verilog-AMS model for the switch

```
//Verilog-AMS HDL for "IQoffsetcompensation", "non_ideal_switch" "verilogams"
//Digitally controlled

`include "constants.vams"
`include "disciplines.vams"

`timescale 1ns / 1ps


module non_ideal_switch_p(p, n, s);
    parameter real ron = 10.0 from (0:inf);     // on resistance (ohms)
    parameter real roff = 100.0M from (ron:inf);// off resistance (ohms)
    parameter real td = 0.0;                    // delay time (s)
    parameter real tr = 20n;                    // rise time (on -> off) (s)
    parameter real tf = 20n;                    // fall time (off -> on) (s)
    input s;
    logic s;
    electrical p, n;
    real reff;

    analog begin
        @(posedge s) reff = roff;
        @(negedge s) reff = ron;
        @(initial_step) reff = (s ? roff : ron);
        I(p, n) <+ V(p, n) / transition(reff, td, tr, tf);
    end
endmodule
```

Figure N.5: VerilogAMS code of switch.

N.4.3. Verilog-AMS model for the Binary-to-Thermometer encoder

```
// VerilogA for IQoffsetcompensation, Bin2Thermo, veriloga

`include "constants.vams"
`include "disciplines.vams"

module Bits7_Bin2_Thermo (dout, din, clk );
`define binbits 7 //binary resolution (bits)
`define thermbits 127 //thermometer (bits)

parameter real DVDD = 1.2; // voltage level of logic 1 (V)
parameter real thresh = DVDD/2; // logic threshold level (V)
parameter integer dir = 1 from [-1:1] exclude 0; // 1 for rising edges, —1 for falling
parameter real td = 0;// delay from clock edge to output (s)
parameter real tt = 1e-9;// transition time of output (s)

input [`binbits-1:0] din;
output [`thermbits-1:0] dout;
input clk;
electrical [`binbits-1: 0] din;
electrical [`thermbits-1:0] dout;
electrical clk;
real result[`thermbits-1:0];
integer sum;
genvar i;

analog begin
        @(cross(V(clk) - thresh, dir) or initial_step) begin
        sum = 0;
        //generate i (0,2)
                //sum=sum+((V(din[i])>thresh)?1:0)*pow(2,i);
        for (i=0; i<=`binbits-1; i=i+1)
                sum=sum+((V(din[i])>thresh)?1:0)*pow(2,i);
        end

        for (i=0; i<=`thermbits-1; i=i+1) begin
                if(i<sum) begin
                        result[i]=1;
                end else begin
                        result[i]=0;
                end
        end

        //generate i (7,0)
                //V(dout[i])<+transition(result[i],td,tt);
        for (i=`thermbits-1; i >=0; i=i-1) begin
                V(dout[i])<+transition(result[i],td,tt);
        end
end

endmodule
```

Figure N.6: VerilogAMS code implementation of the Binary-to-Thermometer encoder.

N.5. Verilog-AMS model for the Two Regions Detector

```
//Verilog-AMS HDL for "IQoffsetcompensation", "TwoRegionsDetector" "verilogams"

//This is the Two Regions Detector
//It detects symbols between Regions (1) and (3), (1) and (4), (3) and (2), (3) and
(4)
//The detector will output a logic "1" if a symbol is in Regions (1) or (3)
//The detector will output a logic "0" if a symbol is in Regions (2) or (4)

//Inputs:
// in = 5 bit phase
// clk = operating clock frequency
//Outputs:
//out[0] = out12 signal output a logic "1" if symbol is in region (1) a logic "0" if
in region (2)
//out[1] = out14 signal output a logic "1" if symbol is in region (1) a logic "0" if
in region (4)
//out[2] = out32 signal output a logic "1" if symbol is in region (3) a logic "0" if
in region (2)
//out[3] = out34 signal output a logic "1" if symbol is in region (3) a logic "0" if
in region (4)

`include "constants.vams"
`include "disciplines.vams"

module TwoRegionsDetector (out, in, clk);

 `define bits 5 // resolution (bits)
  parameter real fullscale = 1.2; // output range is from 0 to fullscale (V)
  parameter real td = 0; // delay from clock edge to output (s)
  parameter real tt = 1e-9; // transition time of output (s)
  parameter real vdd = 1.2;  // voltage level of logic 1 (V)
  parameter real thresh = vdd/2; // logic threshold level (V)
  parameter integer dir = 1 from [-1:1] exclude 0;// 1 for rising edges, 1 for falling
  parameter real quarterpi = 0.15;
  parameter real threequarterspi = 0.45;
  parameter real onepi = 0.6;
  parameter real minusquarterpi = -0.15;
  parameter real minusthreequarterspi = -0.45;
  parameter real minusonepi = -0.6;
  output [3:0] out; // out[3:0] are used to feedback amplitude mismatch to Q.
  input clk;
  input [`bits-1:0] in;
  electrical clk;
  electrical [3:0] out;
  electrical [`bits-1:0] in;
  real aout;
  real regions12;
  real regions14;
  real regions32;
  real regions34;
  integer weight;
  genvar i;

  analog begin
    @(cross(V(clk) - thresh, dir) or initial_step) begin
    aout = 0;
    weight = 2;
    //regions12 = 0;
    //regions14 = 0;
    //regions32 = 0;
    //regions34 = 0;
    for (i = `bits - 1; i >= 0; i = i - 1 ) begin
      if (V(in[i]) > thresh) begin
          aout = aout + fullscale/weight;
      end
      weight = weight*2;
    end
    aout = aout - 0.6;
    if (aout >= quarterpi && aout < threequarterspi) begin
        regions12 = (vdd - thresh)*2;
        regions14 = (vdd - thresh)*2;
    end
    if (aout >= threequarterspi && aout <= onepi || aout >= minusonepi && aout <
minusthreequarterspi) begin
       regions12 = 0;
       regions32 = 0;
    end
    if (aout >= minusthreequarterspi && aout < minusquarterpi) begin
       regions32 = (vdd-thresh)*2;
       regions34 = (vdd-thresh)*2;
    end
    if (aout >= minusquarterpi && aout < quarterpi) begin
       regions14 = 0;
       regions34 = 0;
    end
   end
    V(out[0]) <+ transition(regions12, td, tt);
    V(out[1]) <+ transition(regions14, td, tt);
    V(out[0]) <+ transition(regions32, td, tt);
    V(out[0]) <+ transition(regions34, td, tt);
  end
endmodule
```

Figure N.7: VerilogAMS code implementation of the *Two Regions Detector*.

N.6. Verilog-AMS model for the 1:4 MUX

```
//Verilog-AMS HDL for "IQoffsetcompensation", "mux_4to1" "verilogams"
//
//
// model for an n-channel multiplexer
// each channel is simple modelled by a resistor between
// input and output
//
// in[0]----R------|
// in[1]----R------|---- out -----
//
// the value of the channel resistance is controlled by
// the bit pattern at the input, the selected channel
// is ron, all other are roff
// if a new channel is selected, the old one get switched
// of in ttransit_off, the new one is switched on in
// ttransit_on
`include "constants.vams"
`include "disciplines.vams"

module mux_4to1(in, sel, out);

        //parameter integer size = 2 from [1:inf);
        `define size 2

        input [(1 << `size)-1:0] in;
        input [`size-1:0] sel;  // MSB : LSB
        output out;
        electrical [(1 << `size)-1:0] in;
        electrical [`size-1:0] sel; // MSB : LSB
        electrical out;


        parameter real tt_off = 5n from [0:inf); // transit time from ron --> roff
        parameter real tt_on = tt_off from [tt_off:inf); //transit time from roff -->
ron
        parameter real vdd = 1.2; // voltage level of logic 1 (V)
        parameter real thresh = vdd/2; //logic threshold level (V)
        parameter real ron = 1 from (0:inf); // channel on resistance
        parameter real roff = 1M from (ron:inf); //channel off resistance

        integer rch [(1 <<`size)-1:0];
        integer pow2 [`size:0];
        integer channel;
        real iout;
        genvar i, j;

        analog begin
         @(initial_step) begin
                channel = 0;    // initial value for channel
                for (i = `size-1; i >= 0; i = i - 1) begin
                        pow2[i] = 1 << i;       //temporary variables
                        rch[i] = roff; // set all channel to off
                end
                pow2[`size] = 1 << `size;
                rch[0] = ron; // initial channel 0 is on
         end
         for (i = `size-1; i >= 0; i = i - 1)
           @(cross(V(sel[i]) - thresh)) begin
           rch[channel] = roff;         // switch of the old channel
           channel = 0;
           for (j = `size-1; j >= 0; j = j - 1) // calculate new channel
             if (V(sel[j]) > thresh)
                channel = channel + pow2[j];
                rch[channel] = ron;     //switch on the new channel
           end
         iout = 0;
         //for (i = 0; i < pow2[`size]; i = i + 1)
         for (i = (1 << `size)-1; i >= 0 ; i = i - 1)
         iout = iout + V(in[i], out) / transition(rch[i], 0, tt_off, tt_on);
        I(out) <+ iout;
        end
endmodule
```

Figure N.8: VerilogAMS code implementation of the 1:4 MUX.

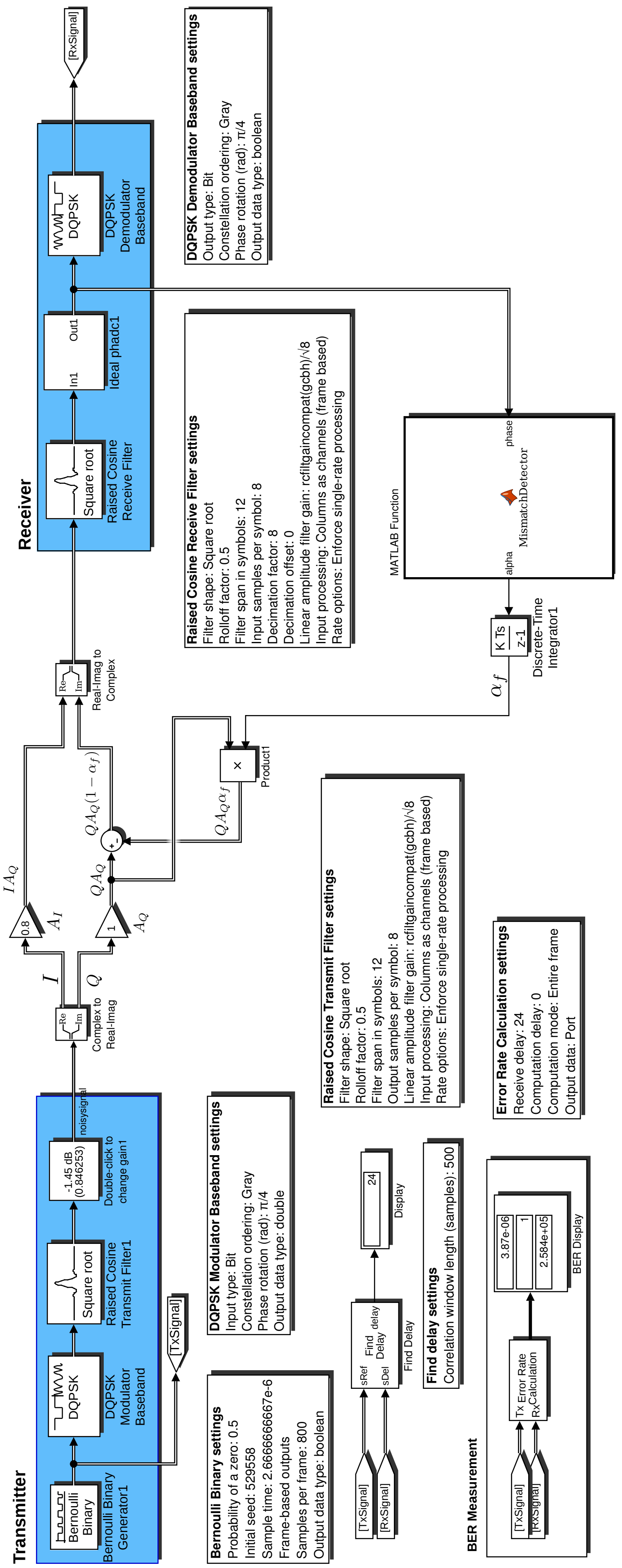
N.7. Verilog-AMS model for the 2 bit counter

```
//Verilog-AMS HDL for "IQoffsetcompensation", "counter" "verilogams"

`include "constants.vams"
`include "disciplines.vams"

module counter2 (out, clk, in);
    `define bits 2
    parameter real td = 0;        // delay from clock edge to output (s)
    parameter real tt = 1e-9;     // transition time of output (s)
    parameter real vdd = 1.2;     // voltage level of logic 1 (V)
    parameter real thresh = vdd/2;  // logic threshold level (V)
    parameter integer dir = 1 from [-1:1] exclude 0; // 1 for rising edges, -1 for falling
    parameter real vhigh = vdd;//logic high level
    parameter real vlow = 0;//logic low level
    parameter integer setval = 0 from [0:(1<<`bits)-1];
    input clk;
    input in;
    output [`bits-1:0] out;
    electrical clk, in;
    electrical [`bits-1:0] out;
    integer up; //0=increasing 1=decreasing
    integer outval;
    genvar j;


    analog begin
      @(initial_step("static","ac")) outval = setval;
      @(cross(V(clk)-thresh, dir)) begin
       if (V(in) > thresh) begin
              up = 1;
              outval = (outval+(+up-!up))%(1<<`bits);
       end
       else if (V(in) < thresh) begin
              up = 0;
              outval = (outval+(+up-!up))%(1<<`bits);
       end
      end
      for (j=`bits-1; j >= 0; j = j-1) begin
        V(out[j]) <+ transition(!(!(outval&(1<<j)))*vhigh+!(outval&(1<<j))*vlow, td, tt);
      end
    end
endmodule
```

Figure N.9: VerilogAMS code implementation of the 2 bit counter.

O
MATLAB Simulink simulation models for IQ gain mismatch compensation

O.1. Model for frame-based simulation



Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

Find delay settings
Correlation window length (samples): 500

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

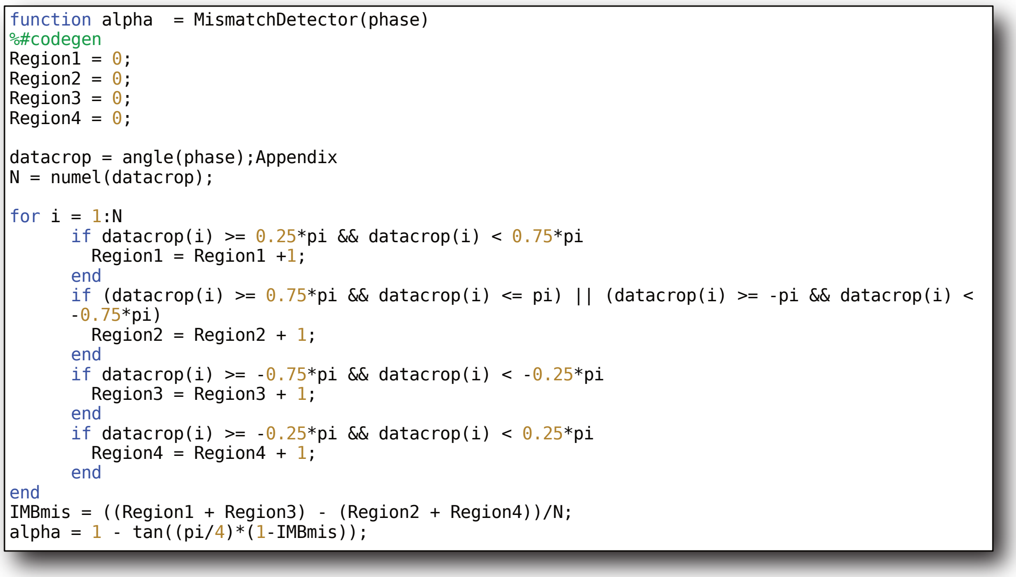
Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

Raised Cosine Receive Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Input samples per symbol: 8
Decimation factor: 8
Decimation offset: 0
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

Figure O.1: Simulink IQ gain mismatch compensation system model for frame based input-output data processing.

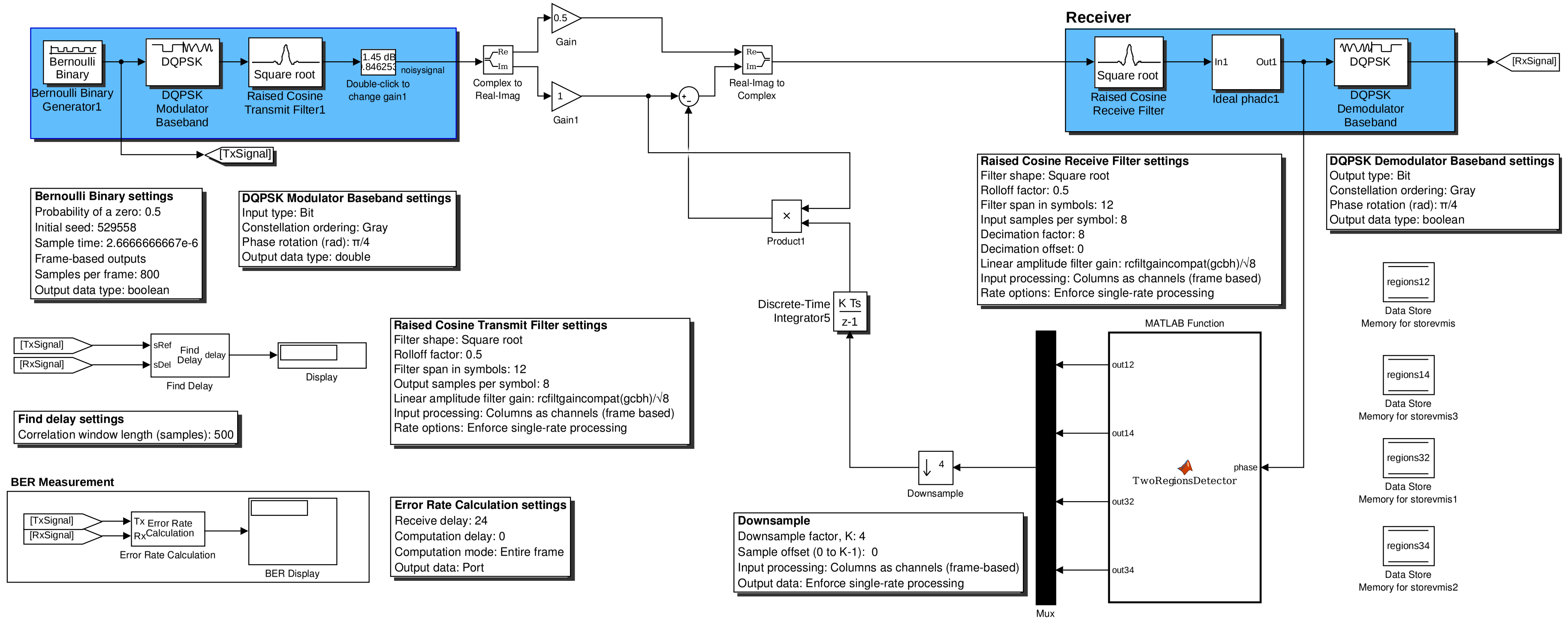
O.1.1. Matlab code for IQ gain mismatch detector



```
function alpha  = MismatchDetector(phase)
%#codegen
Region1 = 0;
Region2 = 0;
Region3 = 0;
Region4 = 0;

datacrop = angle(phase);Appendix
N = numel(datacrop);

for i = 1:N
      if datacrop(i) >= 0.25*pi && datacrop(i) < 0.75*pi
        Region1 = Region1 +1;
      end
      if (datacrop(i) >= 0.75*pi && datacrop(i) <= pi) || (datacrop(i) >= -pi && datacrop(i) <
      -0.75*pi)
        Region2 = Region2 + 1;
      end
      if datacrop(i) >= -0.75*pi && datacrop(i) < -0.25*pi
        Region3 = Region3 + 1;
      end
      if datacrop(i) >= -0.25*pi && datacrop(i) < 0.25*pi
        Region4 = Region4 + 1;
      end
end
IMBmis = ((Region1 + Region3) - (Region2 + Region4))/N;
alpha = 1 - tan((pi/4)*(1-IMBmis));
```

Figure O.2: Matlab code IQ gain mismatch compensation for frame based input-output data processing

O.2. Model for sample-based simulation



Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

Find delay settings
Correlation window length (samples): 500

Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

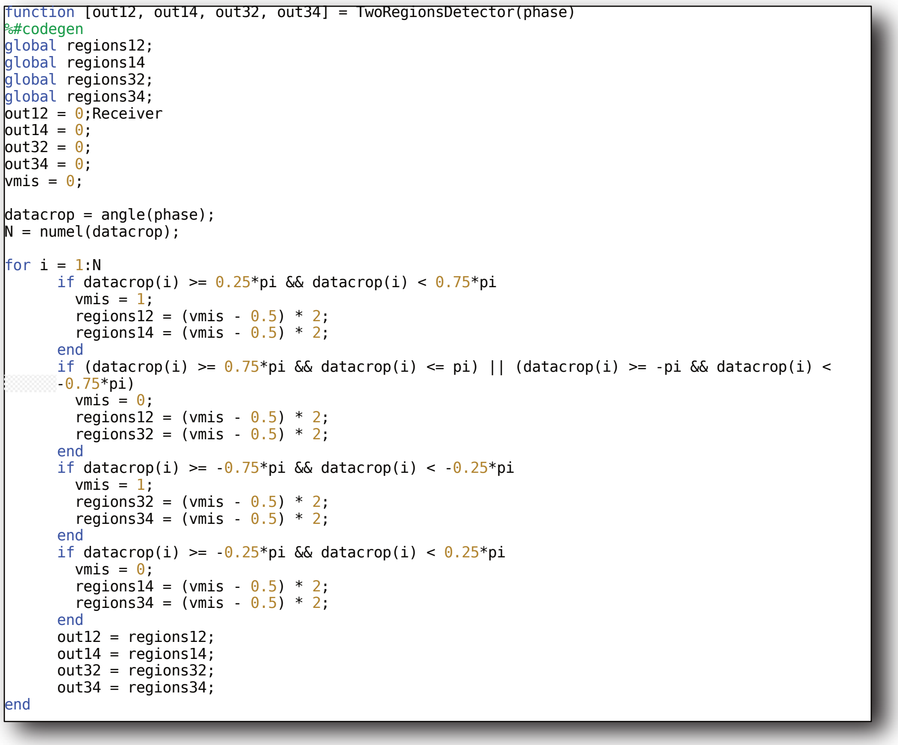
Downsample
Downsample factor, K: 4
Sample offset (0 to K-1): 0
Input processing: Columns as channels (frame-based)
Output data: Enforce single-rate processing

Raised Cosine Receive Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Input samples per symbol: 8
Decimation factor: 8
Decimation offset: 0
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

Figure O.3: Simulink IQ gain mismatch compensation system model for sample based input-output data processing

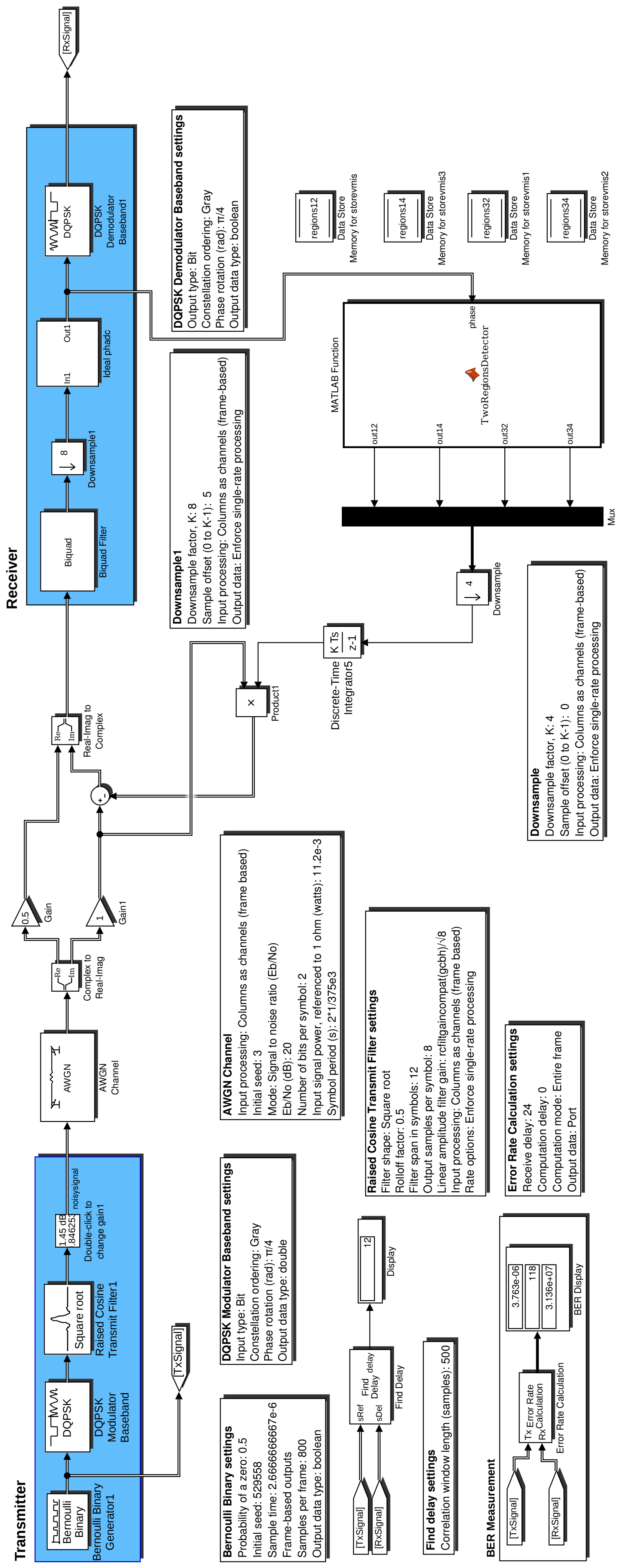
O.2.1. Matlab code for Two Regions Detector



```
function [out12, out14, out32, out34] = TwoRegionsDetector(phase)
%#codegen
global regions12;
global regions14
global regions32;
global regions34;
out12 = 0;Receiver
out14 = 0;
out32 = 0;
out34 = 0;
vmis = 0;

datacrop = angle(phase);
N = numel(datacrop);

for i = 1:N
      if datacrop(i) >= 0.25*pi && datacrop(i) < 0.75*pi
        vmis = 1;
        regions12 = (vmis - 0.5) * 2;
        regions14 = (vmis - 0.5) * 2;
      end
      if (datacrop(i) >= 0.75*pi && datacrop(i) <= pi) || (datacrop(i) >= -pi && datacrop(i) <
      -0.75*pi)
        vmis = 0;
        regions12 = (vmis - 0.5) * 2;
        regions32 = (vmis - 0.5) * 2;
      end
      if datacrop(i) >= -0.75*pi && datacrop(i) < -0.25*pi
        vmis = 1;
        regions32 = (vmis - 0.5) * 2;
        regions34 = (vmis - 0.5) * 2;
      end
      if datacrop(i) >= -0.25*pi && datacrop(i) < 0.25*pi
        vmis = 0;
        regions14 = (vmis - 0.5) * 2;
        regions34 = (vmis - 0.5) * 2;
      end
      out12 = regions12;
      out14 = regions14;
      out32 = regions32;
      out34 = regions34;
end
```

Figure O.4: Matlab code IQ gain mismatch compensation for sample based input-output data processing.

O.3. Model for sample-based simulation including IQ gain mismatch compensation, biqaud filter, channel and quantization noise



DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

Downsample1
Downsample factor, K: 8
Sample offset (0 to K-1): 5
Input processing: Columns as channels (frame-based)
Output data: Enforce single-rate processing

Downsample
Downsample factor, K: 4
Sample offset (0 to K-1): 0
Input processing: Columns as channels (frame-based)
Output data: Enforce single-rate processing

AWGN Channel
Input processing: Columns as channels (frame based)
Initial seed: 3
Mode: Signal to noise ratio (Eb/No)
Eb/No (dB): 20
Number of bits per symbol: 2
Input signal power, referenced to 1 ohm (watts): 11.2e-3
Symbol period (s): 2*1/375e3

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

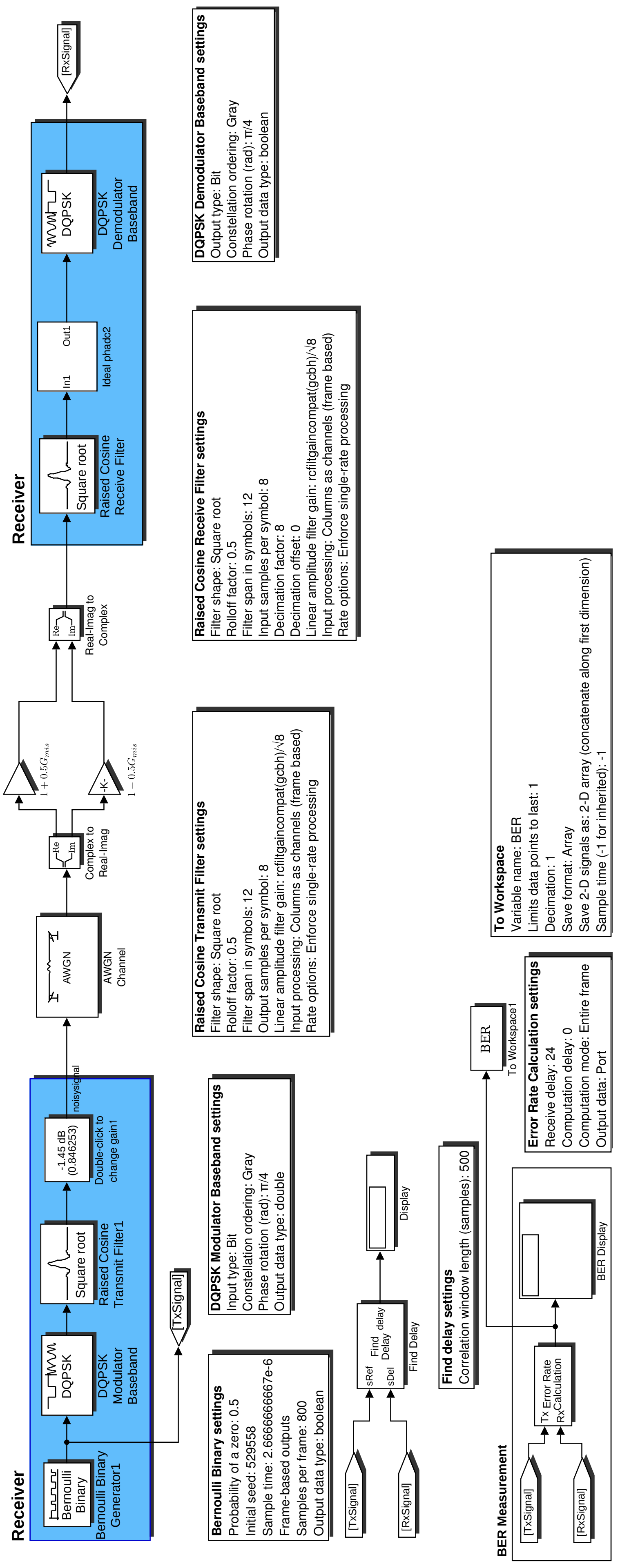
Find delay settings
Correlation window length (samples): 500

Figure O.5: Simulink IQ gain compensation system model including IQ gain mismatch, biqaud filter, channel and quantization noise.

P

MATLAB Simulink transceiver model to simulate BER as a function of IQ gain mismatch

P.1. Model for BER as a function of IQ gain mismatch simulation



Bernoulli Binary settings
Probability of a zero: 0.5
Initial seed: 529558
Sample time: 2.6666666667e-6
Frame-based outputs
Samples per frame: 800
Output data type: boolean

DQPSK Modulator Baseband settings
Input type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: double

Raised Cosine Transmit Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Output samples per symbol: 8
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

Raised Cosine Receive Filter settings
Filter shape: Square root
Rolloff factor: 0.5
Filter span in symbols: 12
Input samples per symbol: 8
Decimation factor: 8
Decimation offset: 0
Linear amplitude filter gain: rcfiltgaincompat(gcbh)/√8
Input processing: Columns as channels (frame based)
Rate options: Enforce single-rate processing

DQPSK Demodulator Baseband settings
Output type: Bit
Constellation ordering: Gray
Phase rotation (rad): π/4
Output data type: boolean

Find delay settings
Correlation window length (samples): 500

Error Rate Calculation settings
Receive delay: 24
Computation delay: 0
Computation mode: Entire frame
Output data: Port

To Workspace
Variable name: BER
Limits data points to last: 1
Decimation: 1
Save format: Array
Save 2-D signals as: 2-D array (concatenate along first dimension)
Sample time (-1 for inherited): -1

Figure P.1: Simulink transceiver model to simulate bit-error rate as a function of IQ gain mismatch.

P.1.1. Matlab code for BER as a function of IQ gain mismatch simulation

```
Gmis_var = 0.5*logspace(-3,2);
Rolloff_factor = [0.5];

TStart=0;
TStop=80/3;

fid = fopen('Output7.txt','a');
fprintf(fid, 'Rolloff_factor,IQAmplitudeMismatch,BER,#BER,#BITS\n');
for i=1:length(Gmis_var)
    Gmis = Gmis_var(1,i);
    a = sim('Gain_Mis_Comp_PhADC_model_Ali_27', 'SimulationMode', 'normal');
    b = a.get('BER');
    assignin('base', 'b', b);
    Output = b([end],:);
    dlmwrite('Output7.txt', [ Rolloff_factor Gmis Output], '-append');
end
fclose(fid);
```

Figure P.2: Matlab code to simulate bit-error-rate as a function of IQ offset.

Q

MATLAB Simulink transceiver model to simulate both IQ offset and gain mismatch compensation

Q.1. Model for sample-based simulation including IQ offset and gain mismatch compensation, biqaud filtering, channel and quantization noise

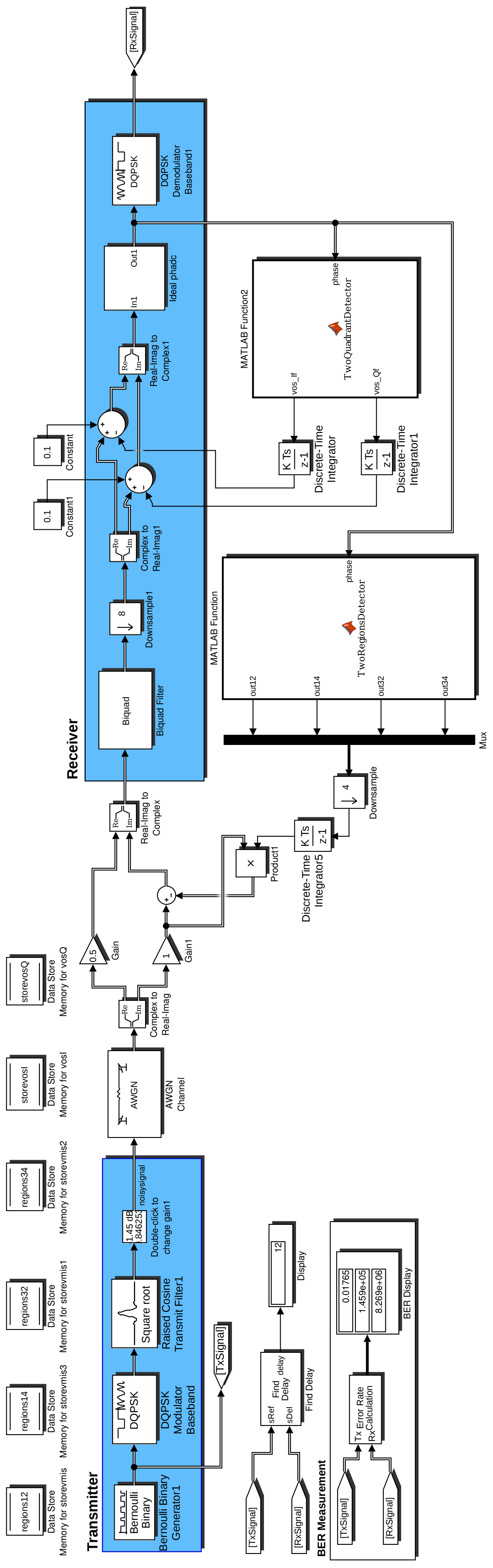


Figure Q.1: Simulink IQ offset and gain compensation system model including IQ offset and gain mismatch, biqaud filter, channel and quantization noise.

Bibliography

[1] AMS AG. Ams foundry support, 2018. URL `http://asic.ams.com/`.

[2] A Astrin. IEEE Standard for Local and metropolitan area networks part 15.6: Wireless Body Area Networks. *IE EE Std 802.15. 6*, 2012.

[3] R Jacob Baker. *CMOS: circuit design, layout, and simulation*, volume 1. John Wiley & Sons, 2008.

[4] Cadence Design Systems. Cadence, 2018. URL `https://cadence.com/`.

[5] Cadence Design Systems. Cadence, 2018. URL `https://www.cadence.com/content/cadence-www/global/en_US/home/training/all-courses/84441.html`.

[6] Belén Calvo, Santiago Celma, Pedro A Martinez, and Maria Teresa Sanz. 1.8 v-100 mhz cmos programmable gain amplifier. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4–pp. IEEE, 2006.

[7] Paul R Gray, Paul Hurst, Robert G Meyer, and Stephen Lewis. *Analysis and design of analog integrated circuits.* Wiley, 2001.

[8] Yao Liu Student Member IEEE, Reza Lotfi Senior Member IEEE, Yonchang Hu Student Member IEEE, and Wouter A. Serdijn Fellow IEEE. A comparative analysis of phase-domain adc and amplitude iq adc. *IEEE Transactions on Circuits and Systems*, 62(3):671–679, March 2015.

[9] Kenneth. Verilog-ams, 2018. URL `https://verilogams.com/`.

[10] Yao Liu. *Analysis and Design of Low-Power Receivers Exploiting Non-50 Ω Antenna Impedance and Phase-Only Quantization.* PhD thesis, Delft University of Technology, 2017.

[11] Yao Liu, Duan Zhao, Yongjia Li, and Wouter A. Serdijn. A 5b 12.9 μW Charge-Redistribution Phase Domain ADC for Low Power FSK/PSK Demodulation. *Proc. IEEE European Solid State Circuits Conference (ESSCIRC)*, pages 275–278, September 2014.

[12] Jens Masuch and Manuel Delgado-Restituto. A 1.1-mW-RX -81.4 -dBm Sensitivity CMOS Transceiver for Bluetooth Low Energy. *IEEE Transactions on Microwave Theory and Techniques*, 61(4):1660–1673, 2013.

[13] Earl McCune. *Practical Digital Wireless Signals.* Cambridge University Press, 2010. ISBN 978-0-521-51630-3.

[14] Agilent Application Note. 1298. digital modulation in communications systems—an introduction. *Hewlett-Packard Company*, 1997.

[15] Behzad Razavi and Razavi Behzad. *RF microelectronics*, volume 1. Prentice Hall New Jersey, 1998.

[16] Shashi Kiran. What are the differences between zero-order hold (ZOH), sample and hold (S&H) and track and hold (T&H) circuits and functions?, 2016. URL `https://www.quora.com/What-are-the-differences-between-zero-order-hold-ZOH-Sample-and-Hold-S-H-and-Track-and-Hold-T-H-circuits-and-functions`.

[17] Inc The MathWorks. Communications system toolbox, 1994-2018. URL `https://nl.mathworks.com/products/communications.html`.

[18] The MathWorks, Inc. Conversion between model types, 1994-2018. URL `https://nl.mathworks.com/help/control/ug/conversion-between-model-types.html`.

[19] The MathWorks, Inc. bertool, 1994-2018. URL `https://nl.mathworks.com/help/comm/ref/bertool.html`.

[20] The MathWorks, Inc. Evm measurement, 1994-2018. URL `https://nl.mathworks.com/help/comm/ref/evmmeasurement.html`.

[21] The MathWorks, Inc. Frequency response data (frd) models, 2018. URL `https://nl.mathworks.com/help/control/ug/frequency-response-data-frd-models.html`.

[22] The MathWorks, Inc. Communications system toolbox, 2018. URL `https://nl.mathworks.com/help/comm/index.html`.

[23] Kenneth Martin Tony Chan Carusone, David Johns. *Analog Integrated Circuit Design*. Wiley, 2015.

[24] Adrianus Wilhelmus Maria van den Enden and NAM Verhoeckx. *Digitale signaalbewerking*. Delta press, 1998.