

University of New Hampshire

University of New Hampshire Scholars' Repository

Master's Theses and Capstones

Student Scholarship

Winter 2007

Software-defined radio using LabVIEW and the PC sound card: A teaching platform for digital communications

Jose Carlos Lanzoni

University of New Hampshire, Durham

Follow this and additional works at: <https://scholars.unh.edu/thesis>

Recommended Citation

Lanzoni, Jose Carlos, "Software-defined radio using LabVIEW and the PC sound card: A teaching platform for digital communications" (2007). *Master's Theses and Capstones*. 334.
<https://scholars.unh.edu/thesis/334>

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact Scholarly.Communication@unh.edu.

**SOFTWARE-DEFINED RADIO USING LABVIEW AND THE PC SOUND CARD:
A TEACHING PLATFORM FOR DIGITAL COMMUNICATIONS**

BY

JOSE CARLOS LANZONI

B.S.E.E., University of New Hampshire, 2002

THESIS

Submitted to the University of New Hampshire

In Partial Fulfillment of

The Requirements for the Degree of

Master of Science

in

Electrical Engineering

December, 2007

UMI Number: 1449592

Copyright 2007 by
Lanzoni, Jose Carlos

All rights reserved.

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform 1449592

Copyright 2008 by ProQuest LLC.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346

ALL RIGHTS RESERVED

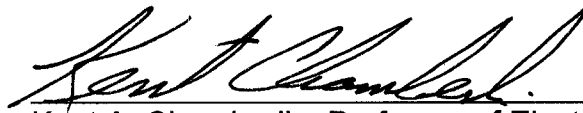
© 2007

Jose Carlos Lanzoni

This thesis has been examined and approved.



Thesis Advisor, Michael J. Carter, Associate
Professor of Electrical and Computer Engineering



Kent A. Chamberlin, Professor of Electrical and
Computer Engineering



W. Thomas Miller, Professor of Electrical and
Computer Engineering

December, 11, 2007
Date

DEDICATION

This work is dedicated to my wife, Neide. Thank you for all the support you provided me to accomplish this project. I also dedicate this to my parents.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis advisor, Dr. Michael J. Carter, for the guidance, encouragement, and the assistance on the development of this thesis. I am also grateful to my dissertation committee: Dr. Kent Chamberlin and Dr. Thomas Miller. They provided valuable comments and suggestions to improve this work.

I would like acknowledge Jonathan Beckwith and Mathew Plante. They supplied me with many insights from their work. Also, I would like to acknowledge Robert Cinq-Mars for his friendly assistance with LabVIEW at the University of New Hampshire.

Finally, I would like to thank my family and my friends for their support during these years (Analia, Heloisa, Erik, Adilson, Rafaela, Matteo, Giulia, Dal, Maurik, Daniel, Bel, Lena, Inês, Daniel, Ricardo, Wladia, Mari, Humberto, Roberta, Cassiano, Leticia, Valentina, Elias, Claudia, Evilene, Carina, James, Tyrso, Heloisa, Fabricio, Rosangela, Edward, Hudson, Luciano, Isabel). The friendship built during my time at UNH has proven to be an invaluable joy in my life. In special, I would like to thank Edinaldo Tebaldi, Fernando Santo, and Sergio dos Santos, for sharing their supportive comments and discussions in so many circumstances.

TABLE OF CONTENTS

DEDICATION.....	iv
ACKNOWLEDGEMENTS.....	v
LIST OF FIGURES.....	xi
ABSTRACT.....	xv

CHAPTER	PAGE
INTRODUCTION.....	1
1. TEACHING DIGITAL COMMUNICATION: THE LABORATORY.....	2
1.1 How Students Learn: Learning Styles.....	3
1.2 Issues in a Communications Laboratory.....	3
1.3 Objectives and Proposed System.....	4
1.4 Document Organization.....	5
2. SOFTWARE-DEFINED RADIO USING LABVIEW AND THE PC'S SOUND CARD AS A DIGITAL COMMUNICATION TEACHING SYSTEM.....	7
2.1 Digital Modulations.....	8
2.2 Software-Defined Radios.....	13
2.3 The LabVIEW Graphical Environment.....	14
2.4 SDR Using LabVIEW.....	17
2.5 The Flex-Radio Systems SDR-1000 Platform.....	22

2.6 Laboratory Experiments	24
2.7 Conclusion.....	26
3. DIGITAL COMMUNICATION SYSTEMS AND THE SDR.....	27
3.1 The Typical Digital Communication System Structure.....	27
3.1.1 Source Coding	29
3.1.2 Channel Coding.....	29
3.1.3 Modulation.....	29
3.1.4 Up Conversion.....	29
3.1.5 Channel	30
3.1.6 Down Conversion	30
3.1.7 Demodulation	30
3.1.8 Channel Decoding	30
3.1.9 Source Decoding	30
3.1.10 Tools Used in a Digital Communication System	31
3.2 SDR in a Digital Communication System	31
3.3 Remarks.....	34
4. THE 4 QAM SDR TRANSCEIVER: USER INTERFACE AND	
OPERATION.....	35
4.1 Front Panel and System Description	36
4.1.1 Execution Control	38
4.1.2 Transmitter/Receiver Mode Tabs.....	38
4.1.3 ON Led Indicator.....	38
4.1.4 Stop Button.....	39

4.1.5	Modulation Parameters in Transmitter Mode	39
4.1.6	Sound Card Parameters	40
4.1.7	Graph Display Tabs in Transmitter Mode	41
4.1.8	Frame Marker Bits in Transmitter	42
4.1.9	Apply Noise	43
4.1.10	Modulation Parameters in Receiver Mode	43
4.1.11	Graph Display Tabs in Receiver Mode	44
4.1.12	Symbol Timing – DLL	46
4.1.13	RX Status	46
4.1.14	Apply Time Delay Control	46
4.2	The 4 QAM SDR Transceiver Operation	47
5.	THE 4 QAM SDR TRANSCEIVER: THEORY AND LABVIEW CODE	
	IMPLEMENTATION	50
5.1	Quadrature Signals	50
5.2	Transmitter Structure	53
5.2.1	Frame Generation	54
5.2.2	Differential Encoding	56
5.2.3	Bits to Symbol Mapping	59
5.2.4	Pulse Shaping	60
5.2.4.1	How the Pulse Shaping Filter Works	61
5.2.4.2	Intersymbol Interference	63
5.2.4.3	Pulse Shaping and the Eye Diagram	64

5.2.4.4	Pulse Shaping Implementation in the 4 QAM	
	SDR Transceiver	66
5.2.5	Conversion to IF	67
5.2.6	Sound Card Output.....	68
5.3	Receiver Structure.....	70
5.3.1	Sound Card Input.....	71
5.3.2	Automatic Gain Control (AGC) and High Pass Filtering (HPF).....	72
5.3.3	Carrier Synchronization and Down Conversion	72
5.3.3.1	The Costas Loop for 4 QAM	74
5.3.3.2	Phase Ambiguity Resolution	80
5.3.3.3	Carrier Synchronization and Down Conversion Code Implementation	81
5.3.4	Matched Filter.....	88
5.3.5	Symbol Synchronization	89
5.3.5.1	The Delay-Locked Loop (DLL) Algorithm.....	90
5.3.6	Waveform Decimation	94
5.3.7	Symbol to Bits Mapping	97
5.3.8	Differential Decoding	99
5.3.9	Frame Synchronization and Alignment.....	100
5.3.10	Message Retrieving.....	104
5.4	Remarks.....	107
6.	CONCLUSION.....	108

LIST OF REFERENCES	112
APPENDICES	114
APPENDIX A: LABORATORY EXPERIMENTS	115
APPENDIX B: 4 QAM SDR TRANSCEIVER CODE	163

LIST OF FIGURES

Figure 2-1:	Modulation	8
Figure 2-2:	Amplitude Modulation (AM)	9
Figure 2-3:	Frequency Modulation (FM)	9
Figure 2-4:	Basic digital modulation schemes	11
Figure 2-5:	Constellation diagram examples	12
Figure 2-6:	Block diagram of a SDR communication system	14
Figure 2-7:	Components of a VI	16
Figure 2-8:	SDR system using LabVIEW	18
Figure 2-9:	Signal path in the SDR system	19
Figure 2-10:	SDR system using RF front end hardware stage	23
Figure 3-1:	Typical digital communication system block diagram	28
Figure 3-2:	Software and hardware implementation in a digital communication system	32
Figure 4-1:	4 QAM SDR Transceiver front panel	37
Figure 4-2:	VI's execution control	38
Figure 4-3:	Transmitter modulation parameters	39
Figure 4-4:	Sound card parameters	40
Figure 4-5:	Graph display tabs for transmitter mode	41
Figure 4-6:	Apply noise control	43
Figure 4-7:	Modulation parameters in receiver mode	44
Figure 4-8:	Graph display tabs for receiver mode	45
Figure 4-9:	RX status display	46
Figure 4-10:	Time delay control	47
Figure 5-1:	Quadrature modulation and demodulation	51
Figure 5-2:	Transmitter structure	53
Figure 5-3:	Frame structure	54
Figure 5-4:	PN generator with a six-stage linear feedback shift register structure	55
Figure 5-5:	64 Sample Period PN Sequence Generator.vi	57
Figure 5-6:	Frame generation	58
Figure 5-7:	Differential Encoder.vi	58
Figure 5-8:	4 QAM constellation diagram	59
Figure 5-9:	4 QAM Bits to Symbol Mapping.vi	60
Figure 5-10:	Some pulse shapes in the time and frequency domains	62
Figure 5-11:	Intersymbol interference (ISI)	63
Figure 5-12:	Ideal root raised cosine response	64
Figure 5-13:	Pulse shaping example	65
Figure 5-14:	Eye diagram	66
Figure 5-15:	Upconvert to IF.vi	69
Figure 5-16:	Receiver structure	71
Figure 5-17:	AGC and HPF.vi	73
Figure 5-18:	The 4 QAM Costas Loop algorithm	79
Figure 5-19:	Phase update in the presence of frequency offset	80
Figure 5-20:	Carrier Synchronization and Downconversion.vi front panel	82

Figure 5-21: Carrier Synchronization and Downconversion.vi block diagram	83
Figure 5-22: Carrier frequency and phase recovery part of Carrier Synchronization and Downconversion.vi	84
Figure 5-23: Down conversion part of Carrier Synchronization and Downconversion.vi.....	84
Figure 5-24: Phase ambiguity resolution of Carrier Synchronization and Downconversion.vi.....	85
Figure 5-25: Quadriphase Costas Loop Carrier Recovery.vi	86
Figure 5-26: 1st Order Butterworth LPF 7k.vi	87
Figure 5-27: Symbol synchronization.....	89
Figure 5-28: DLL sampling.....	90
Figure 5-29: DLL block diagram.....	90
Figure 5-30: Symbol Timing Recovery.vi block diagram	92
Figure 5-31: Symbol Timing Recovery.vi block diagram: detail of case structures.....	93
Figure 5-32: Symbol Timing Recovery.vi front panel	93
Figure 5-33: Down Sampler.vi.....	96
Figure 5-34: Symbol Timing and Decimation.vi.....	97
Figure 5-35: 4 QAM Symbol to Bits.vi	98
Figure 5-36: Differential Decoder.vi	100
Figure 5-37: Frame synchronization	101
Figure 5-38: Frame Sync and Alignment.vi block diagram.....	102
Figure 5-39: Frame Sync and Alignment.vi block diagram: case structures details	103
Figure 5-40: Frame Sync and Alignment.vi front panel	103
Figure 5-41: Message Extraction.vi.....	105
Figure 5-42: BER.vi.....	106
Figure A-1: Components of a VI.....	117
Figure A-2: PN generator with a six-stage linear feedback shift register structure.....	118
Figure A-3: While Loop.....	119
Figure A-4: Creating stop button	119
Figure A-5: Running the VI.....	120
Figure A-6: Adding shift register	120
Figure A-7: Creating array control shell.....	121
Figure A-8: Initializing the shift register	121
Figure A-9: Wiring the rotate 1D array function	122
Figure A-10: Inserting index array	122
Figure A-11: Wiring XOR function	123
Figure A-12: Inserting replace array subset	123
Figure A-13: After initial state data type change.....	124
Figure A-14: Inserting PN sequence indicator.....	124
Figure A-15: Running the VI with the waveform chart	125
Figure A-16: Moving waveform chart outside the loop	126
Figure A-17: VI after enabling indexing	126
Figure A-18: For loop	127

Figure A-19: Generation of 63 PN samples.....	127
Figure A-20: Final BD.....	128
Figure A-21: Icon editor window.....	128
Figure A-22: VI connector pane.....	129
Figure A-23: Documenting the VI.....	129
Figure A-24: Context Help window.....	130
Figure A-25: Digital modulation schemes.....	132
Figure A-26: Transmitter structure.....	133
Figure A-27: Receiver structure.....	134
Figure A-28: Frame structure.....	134
Figure A-29: 4 QAM constellation diagram.....	135
Figure A-30: Ideal root raised cosine frequency response.....	136
Figure A-31: Frame synchronization.....	140
Figure A-32: Some pulse shapes in the time and frequency domains.....	145
Figure A-33: Intersymbol interference (ISI).....	146
Figure A-34: Ideal root raised cosine response.....	146
Figure A-35: Pulse shaping example.....	147
Figure A-36: Eye diagram.....	147
Figure A-37: Up and down conversion.....	151
Figure A-38: Quadrature conversion to RF.....	152
Figure A-39: Carrier recovery and down conversion.....	152
Figure A-40: 4 QAM Costas Loop.....	154
Figure A-41: Phase update in the presence of frequency offset.....	154
Figure A-42: DLL sampling.....	159
Figure A-43: Delay-Locked Loop block diagram.....	159
Figure B-1: 4 QAM SDR Transceiver front panel: transmitter mode.....	164
Figure B-2: 4 QAM SDR Transceiver front panel: receiver mode.....	165
Figure B-3: 4 QAM SDR Transceiver block diagram, transmitter mode: part 1.....	166
Figure B-4: 4 QAM SDR Transceiver block diagram, transmitter mode: part 2.....	167
Figure B-5: 4 QAM SDR Transceiver.vi BD, transmitter mode: Frame Bits case structures.....	168
Figure B-6: 4 QAM SDR Transceiver.vi BD, transmitter mode: Symbol Mapping case structures.....	168
Figure B-7: 4 QAM SDR Transceiver.vi BD, transmitter mode: Pulse Shaping case structures.....	169
Figure B-8: 4 QAM SDR Transceiver.vi BD, transmitter mode: Constellation and Eye diagrams structures.....	169
Figure B-9: 4 QAM SDR Transceiver.vi BD, transmitter mode: Output Signal and FFT Output Signal case structures.....	170
Figure B-10: 4 QAM SDR Transceiver.vi BD, receiver mode: part 1.....	171
Figure B-11: 4 QAM SDR Transceiver.vi BD, receiver mode: part 2.....	172
Figure B-12: 4 QAM SDR Transceiver.vi block diagram, receiver mode: false case structure.....	173

Figure B-13: 4 QAM SDR Transceiver.vi BD, receiver mode: Baseband conversion case structures	174
Figure B-14: 4 QAM SDR Transceiver.vi BD, receiver mode: Matched Filter case structures	174
Figure B-15: 4 QAM SDR Transceiver.vi BD, receiver mode: Decimation case structures	175
Figure B-16: 4 QAM SDR Transceiver.vi BD, receiver mode: Received Message case structures	175
Figure B-17: 4 QAM SDR Transceiver.vi BD, receiver mode: FFT Input Signal, Phase Update, and Symbol Timing case structures	176
Figure B-18: 4 QAM SDR Transceiver.vi BD, receiver mode: Constellation Diagram, Eye Diagram, and SOF Index case structures	176
Figure B-19: 4 QAM SDR Transceiver.vi BD, receiver mode: Received Frame, Message Error, and BER case structures	177
Figure B-20: 64 sample period PN Sequence Generator.vi icon and description	177
Figure B-21: Differential Encoder.vi icon and description	177
Figure B-22: 4 QAM Bits to Symbol Mapping.vi icon and description	177
Figure B-23: Upconvert to IF.vi icon and description	177
Figure B-24: Apply Noise.vi	178
Figure B-25: Input Scaling and Energy Detection.vi	179
Figure B-26: AGC and HPF.vi icon and description	180
Figure B-27: Carrier Synchronization and Downconversion.vi icon and description	180
Figure B-28: Quadrphase Costas Loop Carrier Recover.vi icon and description	181
Figure B-29: 1st Order Butterworth LPF 7k.vi icon and description	181
Figure B-30: Symbol Timing and Decimation.vi icon and description	181
Figure B-31: Symbol Timing Recovery.vi icon and description	182
Figure B-32: Down Sampler.vi icon and description	182
Figure B-33: 4 QAM Symbol to Bits Mapping.vi icon and description	182
Figure B-34: Differential Decoder.vi icon and description	182
Figure B-35: Frame Synchronization and Alignment.vi icon and description	183
Figure B-36: Message Extraction.vi icon and description	183
Figure B-37: BER.vi icon and description	184
Figure B-38: 4 QAM SDR Transceiver_maxeye.vi block diagram, receiver mode: part 1	185
Figure B-39: 4 QAM SDR Transceiver_maxeye.vi block diagram, receiver mode: part 2	186
Figure B-40: 4 QAM SDR Transceiver_maxeye.vi block diagram, receiver mode: false case structure	187
Figure B-41: 4 QAM SDR Transceiver_maxeye block diagram, receiver mode: sample rate update case structure 1 detail	188
Figure B-42: 4 QAM SDR Transceiver_maxeye block diagram, receiver mode: sample rate update case structure 1 detail	188

ABSTRACT

SOFTWARE DEFINED RADIO USING LABVIEW AND THE PC SOUND CARD:

A TEACHING PLATFORM FOR DIGITAL COMMUNICATIONS

BY

JOSE CARLOS LANZONI

University of New Hampshire, December, 2007

Different modulation techniques and protocols require a standard communications laboratory for engineering courses to be equipped with a broad set of equipment, tools and accessories. However, the high costs involved in a hardware-based laboratory can become prohibitively expensive for many institutions. Software simulations alone can replicate most real-world applications with much lower costs. Nevertheless, they do not replace the real-world feeling provided by hardware-based systems, which can produce and receive physical signals to and from the exterior media.

Advances in computer technology are allowing software-defined radio (SDR) concepts to be applied in many areas of communications. In this type of system, the baseband processing is performed completely in software while an analog RF front end hardware can be used for RF processing. The use of a software-defined radio platform in a digital communications laboratory can offer the benefits of software simulations coupled with the enthusiasm presented by hardware-based systems.

A low-cost software-defined radio teaching platform implemented in LabVIEW using the personal computer sound card was developed for a digital communications laboratory along with a set of exercises to help students assimilate the concepts involved in communications theory and system implementation. This system allows for the generation, reception, processing, and analysis of signals in a 4 QAM (quadrature amplitude modulation) transceiver using the personal computer sound card to transmit and receive modulated signals.

This teaching platform provides the means necessary to explore the theoretical concepts of digital communication systems in a laboratory environment. National Instruments' LabVIEW graphical programming environment allows a more intuitive way of coding, which helps students to spend more time learning the relevant theory concepts and less time coding the applications. Being a flexible and modular system, modifications can be made for optimization and use with different and/or more complex techniques.

INTRODUCTION

Laboratory practice is a very important activity in engineering and physical sciences. It helps students to understand theoretical concepts learned in the classroom and it stimulates their interest and creativity by showing them how these concepts can be applied in the real world. The laboratory must be equipped with the necessary tools to perform experiments designed to reach specific learning goals. These tools can sometimes be very expensive and susceptible to damage due to accidents and/or misuse. Also, different types of tools and equipment may be available, and the choice of the apparatus must take into account the respective costs and performance for each one. Considering the typical high costs involved in a standard digital communication laboratory, a set of low-cost learning tools is proposed in this thesis. This thesis is an extension of the work done by Plante [7] and Beckwith [8], where a software defined-radio using MATLAB platform was presented as a teaching tool for digital communications.

CHAPTER 1

TEACHING DIGITAL COMMUNICATIONS: THE LABORATORY

In order to provide students with the conditions to accomplish their experiments, a digital communication laboratory must have a broad set of equipment and tools such as oscilloscopes, spectrum analyzers, vector signal analyzers, logic analyzers, function generators, amplifiers, and filters, among others. Even though this equipment is very helpful in many applications, it can be prohibitively expensive when considering the need to have multiple lab stations to support several students in the same laboratory session. The limitations imposed by these hardware components regarding to their flexibility to accommodate structural changes (modularity), and vulnerability to bad wire connections that can cause equipment damage, must also be considered.

An alternative to lower the laboratory equipment costs and damage risks is to use pure computer simulations. However, these simulations typically do not replace the feeling associated with using real-world equipment.

Another approach to fill this gap is the subject of this thesis: software-defined radio using LabVIEW and the personal computer's sound card. It is a system composed of hardware and software components which gives great flexibility and robustness with relatively low-cost equipment.

1.1 – How Students Learn: Learning Styles

In order to choose a teaching platform, it is necessary to determine the goals that must be reached to teach properly the subject of interest and also to understand how students learn. Students learn best by different methods and the choice of the teaching platform must accommodate fundamental differences in learning styles.

As discussed in [8] and shown in [26], there are three main learning styles: auditory, visual and kinesthetic. Students who fall in the auditory style learn better by hearing. For the ones that fall in the visual style, they can learn better when they read or look at a drawing that depicts the subject of interest. The ones that fall in the kinesthetic style can better learn when they touch and move objects. For every student, these three learning methods can coexist, although one of them is usually predominant. Therefore, it is crucial to integrate these three learning styles into the teaching method to better attend the overall student's need.

1.2 – Issues in a Communications Laboratory

The digital communication laboratory must have equipment capable of generating and visualizing signals in the time, frequency and I/Q domains. As discussed earlier and in [8], there are issues involving high costs, damage risks, flexibility, and robustness of the equipment used. Also, it must be considered that

students have a better feeling for the theoretical concepts when they work with real equipment, changing parameters through the use of knobs and switches. Therefore, a purely hardware or purely software solution would not completely address these issues.

The experiments in a hardware-based communication laboratory often divide the whole process into stages. These are discrete blocks responsible for specific tasks that can be designed and analyzed separately from the other stages. Thus, it is desirable to have a system that could provide this modular approach, allowing more flexibility to include different techniques.

Another issue of consideration is the “idealized” environment in which the experiments are done. The tools and techniques often suffer from the lack of interferences found in the real world. The system used in the learning process must also account for imperfections usually found in practical implementations.

1.3 – Objectives and Proposed System

The choice of a teaching platform to be used in the laboratory must consider the main objectives of teaching digital communications: low cost, low risk of equipment damage due to accidental bad wiring (more robustness), combination of hardware and software, modularity for more flexibility, possibility to integrate real world issues faced by implementers, modifiable for new techniques, incorporation of techniques that encapsulate the three fundamental

learning styles and excite students to learn more. In an attempt to satisfy these requirements, a system using software-defined radio concepts is presented here.

The use of software-defined radio (SDR) is a good alternative to address the issues and the teaching objectives in digital communications laboratory. With the technological advances in faster processors, it is now possible to implement a complete digital communication system using very few hardware components. In such a system, the majority of the tasks are done in software, giving more flexibility to implement different techniques and allowing visualization at several nodes of the system diagram, while lowering costs and equipment damage risks.

1.4 – Document Organization

This document presents a teaching platform to be used in a digital communication laboratory. Many theoretical and practical aspects involved in digital communications are discussed along with the proposed system.

Chapter 2 starts with a discussion on modulation schemes and the tools and equipment used in digital communication systems. Next, it presents the software-defined radio (SDR) key concepts and an introduction to the National Instruments' LabVIEW graphical environment. The proposed SDR teaching system is presented with some detail, followed by a discussion on the RF front end hardware, the Flex-Radio SDR-1000. The set of laboratory experiments included in Appendix A is also discussed in this chapter.

Chapter 3 describes a complete digital communication system structure and how SDR concepts can be incorporated into it.

Chapter 4 presents the 4 QAM SDR Transceiver.vi user interface and operation. This LabVIEW code implementation provides the interface for the SDR teaching system. The user interface and the radio operation are explained in detail in this chapter.

Chapter 5 starts with an introduction to quadrature signals. Later on, it discusses the pertinent theory behind each stage of the transmitter and receiver structures of the 4 QAM SDR Transceiver.vi, also presenting the LabVIEW code implementation of these stages.

Appendix A contains the set of proposed laboratory experiments, while Appendix B contains the entire LabVIEW code of the 4 QAM SDR Transceiver.vi.

CHAPTER 2

SOFTWARE-DEFINED RADIO USING LABVIEW AND THE PC'S SOUND CARD AS A DIGITAL COMMUNICATION TEACHING SYSTEM

Chapter 1 explored the motivation and criteria to find a platform for teaching digital communications in a laboratory setting. Considering the need to maximize hardware flexibility and modularity while minimizing equipment costs and vulnerability to damage due to accidents, and still promoting means to explore the three learning styles discussed in section 1.1, a platform consisting in the use of software-defined radio concepts using LabVIEW and the PC's sound card to teach digital communications is proposed and discussed in this chapter.

This chapter starts with a discussion on modulation schemes and the tools and equipment necessary for generation and visualization of signals used in a digital communication lab. An introduction to LabVIEW, the graphical software environment in which the SDR program code was implemented, is also presented here followed by the proposed system and a proposed set of lab experiments.

2.1 – Digital Modulations

A signal can be processed for use in a communication system by analog or digital modulation. The modulated signal has two components: the information (modulating) signal and the carrier signal.

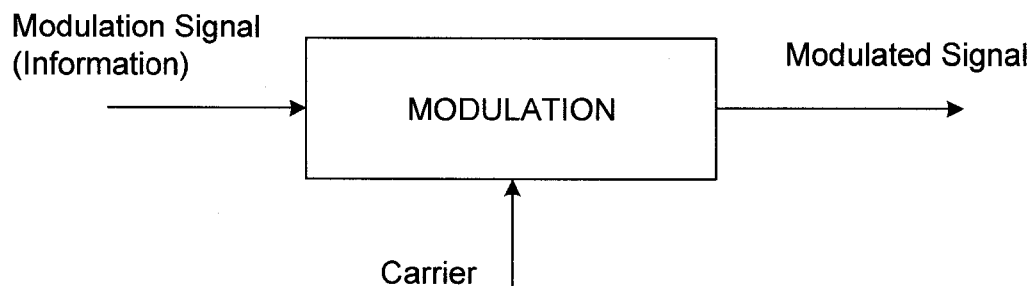


Figure 2-1 – Modulation

In analog modulation, there are three different ways to modulate the signal: Amplitude Modulation (AM), Frequency Modulation (FM), and Phase Modulation (PM). They are said to be analog modulation types because the original (modulating) signal and the carrier signal are analog. Each one of these schemes has advantages and disadvantages, with FM being the one with relative lower immunity to noise interference.

In Amplitude Modulation (AM), the modulating signal is mixed with a sinusoidal carrier signal with a specific frequency and the resulting signal will have its amplitude varied proportionally to the amplitude of the modulating signal.

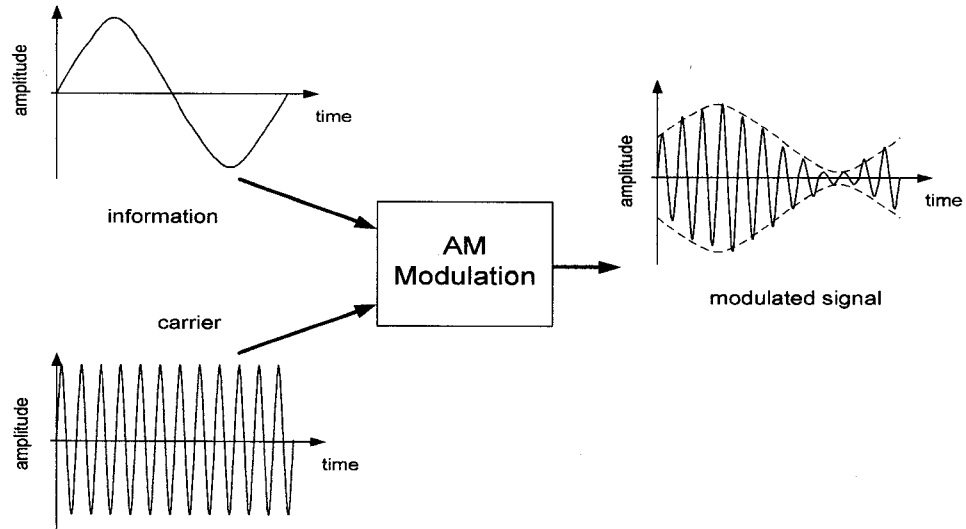


Figure 2-2 – Amplitude Modulation (AM)

In Frequency Modulation (FM), the frequency of the carrier is varied in proportion to the amplitude of the modulating signal. The amplitude of the resulting modulated signal is constant.

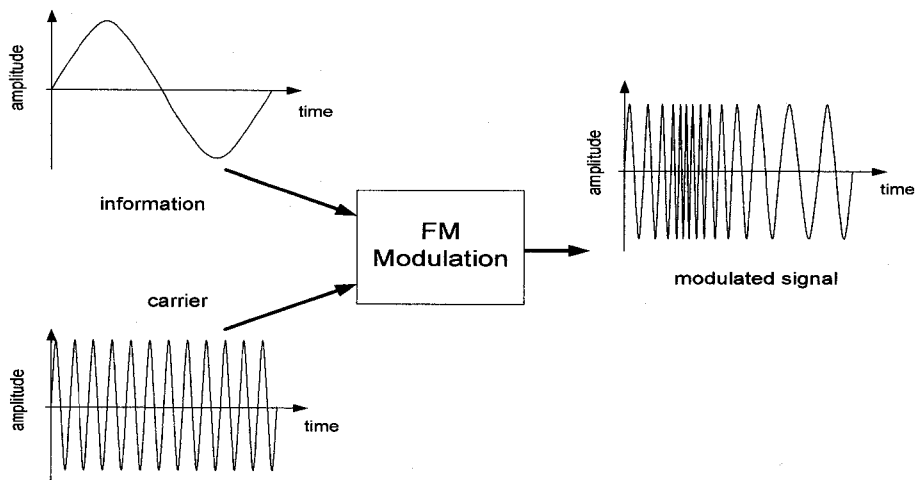


Figure 2-3 – Frequency modulation (FM)

Phase Modulation (PM) is similar to FM. The phase of the carrier is shifted proportionally to the modulating signal. As in FM, the resulting modulated signal has constant amplitude.

The need for more reliable and secure transmission of information brought us the digital modulation schemes. By using digital modulation, the information is defined in bits before it is modulated, making it possible to apply encoding methods that can add security layers and better error correction features. In order to understand the received digital message it is necessary to demodulate and decode the received signal by using the same coding scheme used in the transmitter. Anyone who attempts to understand the information after the demodulation process will need to know how it was encoded; otherwise the received bit sequence will be unintelligible to the receiver.

There are three basic digital modulation types: amplitude shift keying (ASK), frequency shift keying (FSK), and phase shift keying (PSK). These three techniques modulate the carrier signal by varying one of the carrier's parameters: amplitude, frequency, or phase according to the value of each information symbol to be transmitted, respectively. Each symbol is defined by the modulation scheme and it can convey one or more bits of information. For example, Binary Phase Shift Keying (BPSK) conveys only one bit per symbol while Quadrature Phase Shift Keying (QPSK) conveys two bits per a symbol. Figure 2-4 shows these basic digital modulation schemes.

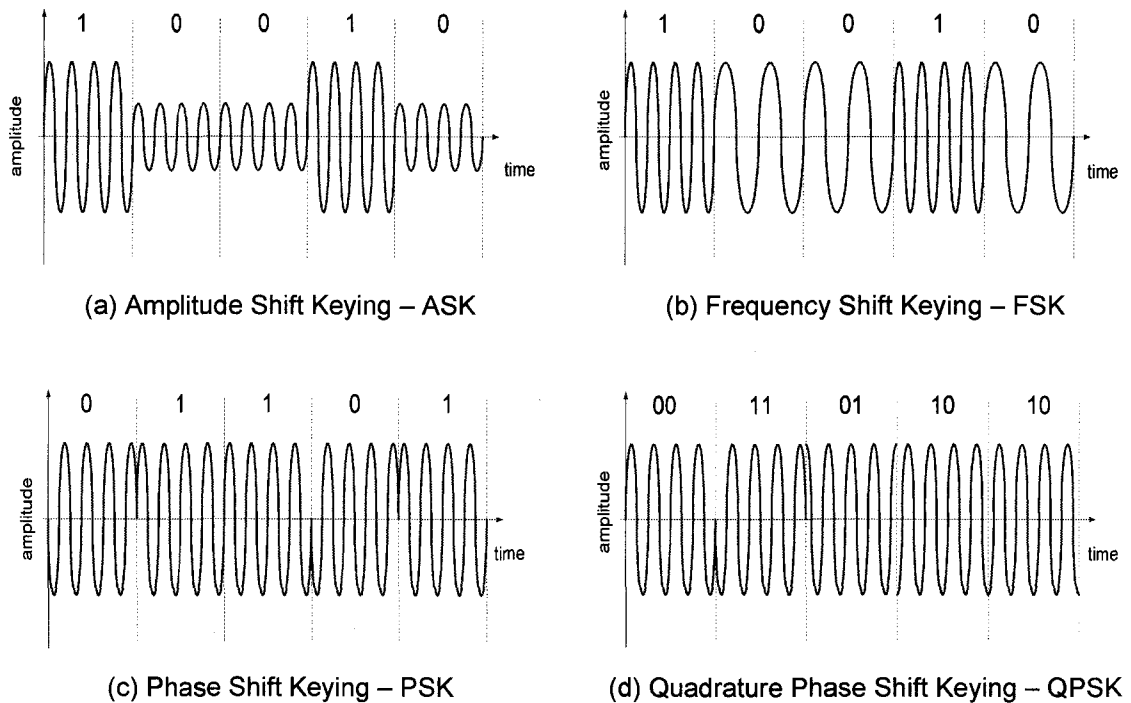


Figure 2-4 – Basic digital modulation schemes

Arbitrary digital signals may be formed by simultaneously modulating quadrature carriers and combining them. The I/Q modulator uses the basis set of sine and cosine functions as carriers. Since these two functions are orthogonal, any signal can be written as a vector sum of these two functions. A quadrature signal is represented by a complex value, where the real part is defined as the In-phase (I) component and the imaginary part is defined as the Quadrature (Q) component of the signal. The information bits to be modulated are converted to symbols defined in a complex plane called constellation diagram (I/Q domain).

At the receiver side, the I and Q components of the signal can be mapped back into a constellation diagram, which gives a better understanding of how well

the signal is being received. Constellation diagrams are shown by figure 2-5. (For detailed information about quadrature signals see [4])

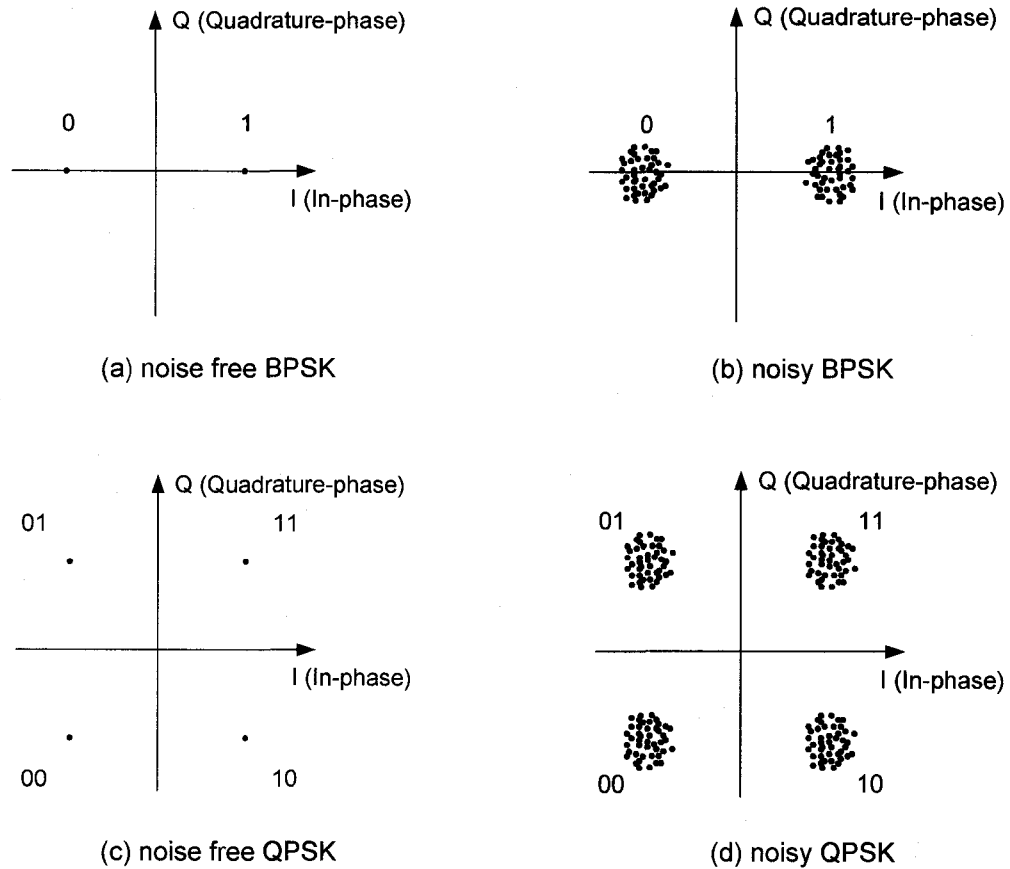


Figure 2-5 – Constellation diagram examples

In a laboratory teaching environment, it is expected to have a set of tools and equipment necessary to provide signal generation, signal processing, and signal measurement. Measuring equipments should allow observations of a given signal in the time, frequency, and I/Q domains.

2.2 – Software-Defined Radios

Software-defined radio is a programmable communication system with its key elements defined in software. The software is responsible for all baseband processing, which includes functions like source coding, channel coding, modulation, and equalization. Any change in the system parameters, such as the modulation scheme, can be done by just changing blocks of software code. This reduces the system cost and increases its flexibility, since there is no need to change any hardware if one decides to change some operational capabilities of the radio.

The use of SDR systems is becoming more popular as better processors with faster processing speeds are becoming available. These systems can be found in military and commercial applications, like cell phone base and amateur radio stations. There is a great expectation that SDR will be the dominant technology in radio communication in the near future.

Normally high-performance digital signal processors and/or FPGAs are used to process the baseband signal due to the intense computations performed by DSP algorithms. Therefore, it is expected to have some limitations when using personal computers as the core of a SDR system. Some of the LabVIEW routines may not respond as fast as required for a continuous process due to the nature of the programming environment. Operational system delays and processor speed may also be subjects of concern here.

In a typical SDR communication system, the information to be sent is baseband-processed in software and sent to an up-conversion hardware system (controlled by the software) responsible to convert the baseband signal into an RF (radio-frequency) signal. This RF signal travels through the communication channel where it may suffer from impairments. The RF signal reaches the receiver that will pass it through the down-conversion hardware (also controlled by software) to convert it into its baseband form. The received signal in its baseband form is then processed in software and the information is extracted. A block diagram of this system is shown by figure 2-6.

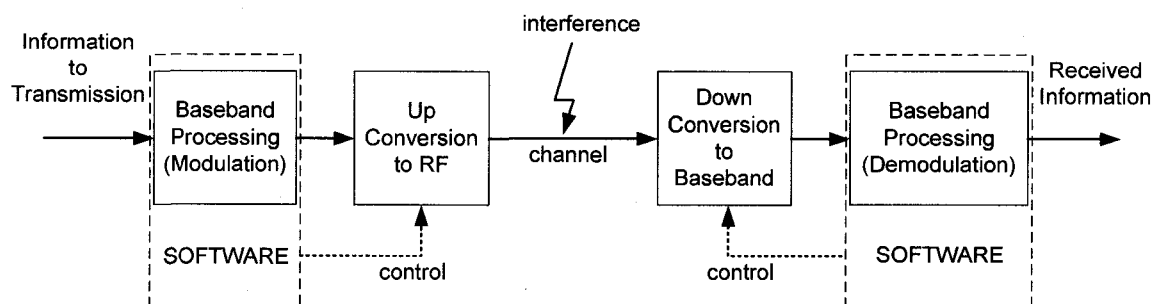


Figure 2-6 – Block diagram of a SDR communication system

2.3 – The LabVIEW Graphical Environment

National Instruments' LabVIEW is a graphical programming environment that works on a data flow model (data flows from data source to data sink). It is a powerful and flexible software package widely used in industry and academia for

data acquisition, instrument control, and analysis. LabVIEW can run in many different platforms like Windows, MacOS, Linux, and Solaris. Instead of using lines of code, LabVIEW provides a graphical programming environment that makes coding a lot easier compared to text based languages. Its graphical language makes programming more intuitive, and therefore, easier to write code and understand how a piece of code works.

With its graphical programming language (called G) LabVIEW can reduce the programming time by orders of magnitude. It has an Analysis library which contains functions for many different tasks like signal generation, signal processing, filters, statistics, array arithmetic, and linear algebra. There is also an extensive library of examples that can be applied in different areas of interest. LabVIEW can also incorporate m-files (code used in MATLAB) using its built-in function called Mathscript.

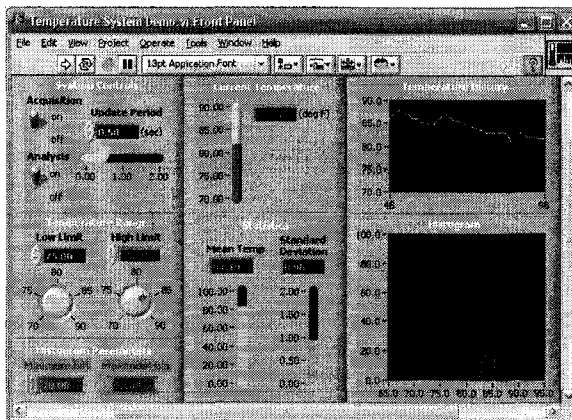
A LabVIEW program, called a virtual instrument (VI), has three main parts: the front panel (FP), the block diagram (BD), and the icon/connector pane. VIs are hierarchical and modular. A VI used within another VI is called subVI (similar to a subroutine).

The front panel is the interactive user interface. It can contain graphical objects used as controls or indicators, like push buttons, switches, knobs, and graphs.

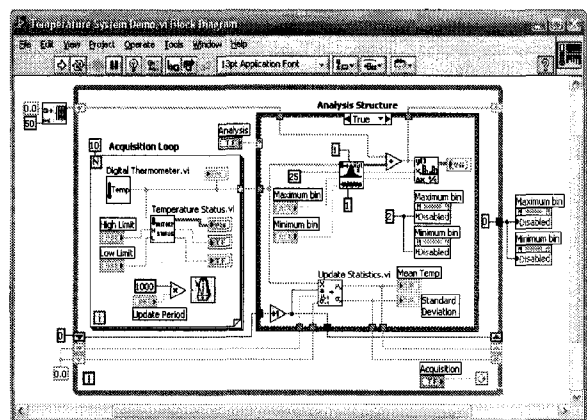
The block diagram is where the VI's source code (executable) resides. It contains sources, sinks, subVIs (lower-level VIs), and execution control structures. These objects are connected by wires to define the program logic.

Front panel objects have corresponding terminals on the block diagram in order to pass data between the user and the program.

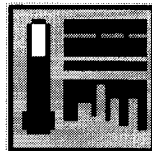
The VI icon is the graphical representation of the VI. The VI icon is displayed when it is inserted in a block diagram of another VI. The connector pane is the mechanism to connect wires from the block diagram to the subVI, defining its inputs (controls) and outputs (indicators).



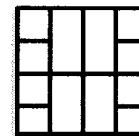
(a) Front Panel - FP



(b) Block Diagram - BD



(c) Icon



(d) Connector Pane

Figure 2-7 – Components of a VI

Many resources are available for LabVIEW users. Besides its extensive list of examples found in its library and its help files, there are many books destined to beginners, intermediate, and advanced users. Also, other great resources like tutorials and user forums can be easily found in the internet.

2.4 – SDR Using LabVIEW

The system proposed in this document as a teaching platform for the digital communications laboratory has in its core the LabVIEW graphical programming environment. It uses a personal computer with the LabVIEW software installed and the computer's sound card to send and/or receive the modulated signal. By using this system, the student can practice with a real working radio where it is possible to easily change key parameters and observe the system behavior virtually at any node of the system. Any desired change can be done in software, giving more flexibility while restricting drastically chances of accidental equipment damage.

The computer may work as a transmitter or a receiver (transceiver). When set to work as a transmitter, a binary sequence is used as the information to be sent through the communication channel. This binary sequence passes through the necessary stages, from frame encapsulation to the final intermediate frequency (IF) modulated signal, which is sent to the communication channel using the sound card output. When working as a receiver, the receiving computer acquires the incoming IF modulated signal using its sound card which is passed through the necessary steps to retrieve the information sent by the transmitter. A simple audio cable is used as the communication medium.

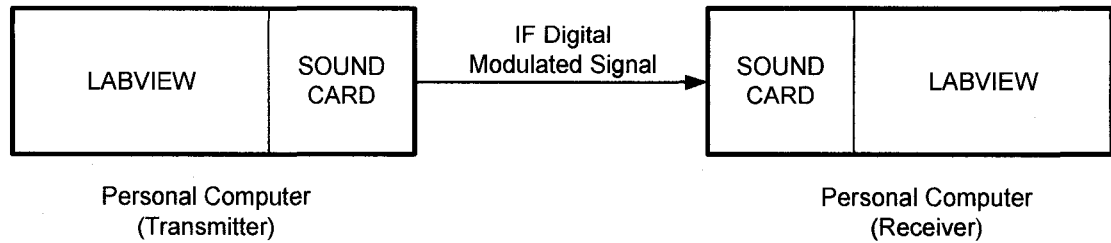
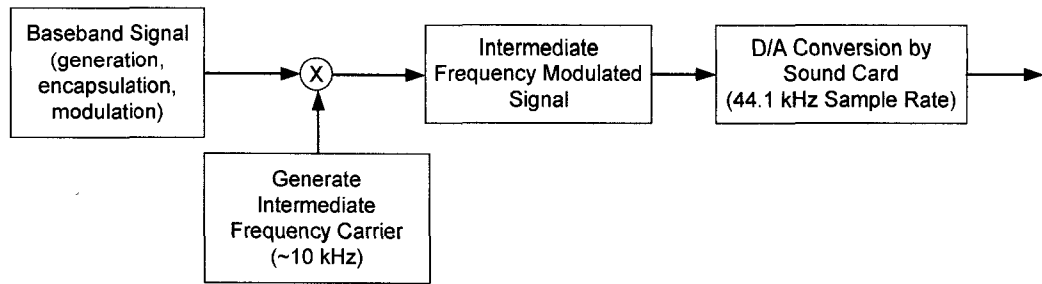
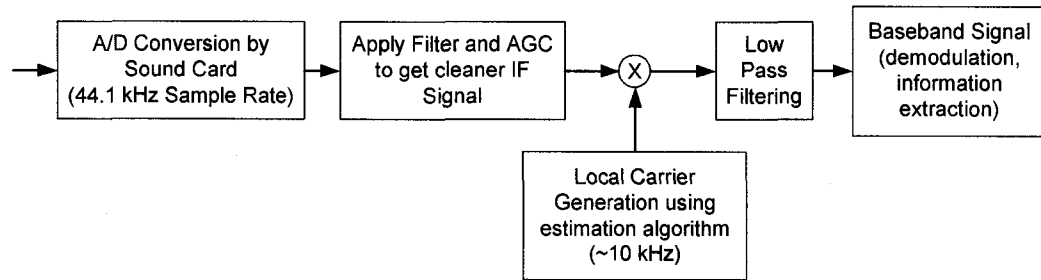


Figure 2-8 – SDR system using LabVIEW

This system was chosen to simplify the laboratory experiments, since the final stage at the transmitter up converts the modulated signal only to an intermediate frequency (IF) which was originally set to 10 kHz. This value was chosen taking into account the sound card sample rate capabilities (the sound card sample rate was set to 44.1 kHz) and the possibility to send the modulated signal to a radio frequency (RF) stage, like the one used by the FlexRadio Systems SDR-1000 Software Radio, which uses an intermediate frequency signal at 11.025 kHz. The SDR-1000 is a software-defined radio platform originally designed for amateur radio operators. It was the chosen platform adapted for use with MATLAB in Plante and Beckwith' work [7, 8]. A discussion on the SDR-1000 platform is presented in the next section of this chapter. Figure 2.9 shows the signal path in the proposed SDR system.



(a) Transmitter



(b) Receiver

Figure 2-9 – Signal path in the SDR system

In the transmitter, the information (a binary sequence) is generated and encapsulated into a binary frame before modulation. After modulation, the signal is up-converted to an IF signal by mixing it with a carrier of 10 kHz (the frequency of the carrier can be adjusted by the user). This IF modulated signal is then sent to the sound card for transmission. The sound card is set to work at a sample rate of 44.1 kHz and converts the digital signal into an analog signal (D/A conversion).

In the receiver, the modulated IF signal is captured by the sound card, which is also set to work with a sample rate of 44.1 kHz. The analog signal is converted into a digital signal (A/D conversion) and stored in the sound card

buffers. The signal is then filtered and an automatic gain control (AGC) is applied to get a cleaner signal with a proper dynamic range. Later, the IF carrier is extracted by digitally mixing the incoming signal with a local carrier generated in the receiver using a carrier synchronization algorithm and low pass filtering (LPF). The baseband signal is then demodulated and the information is extracted.

There are many toolkits available to be added to LabVIEW, like control design, advanced signal processing, order analysis, system identification, and modulation toolkits, to name a few. The modulation toolkit has a vast variety of modules and functions that can be applied in communications systems. However, these modules were originally designed to work in computer simulations and therefore many of them are not well suited for use with communication systems working with external physical signals in real-time. In order to make a real-time working SDR system able to send and receive signal to and from outside the computer, it is necessary to develop other functions and modules.

Since the proposed SDR system uses a Windows-based personal computer and LabVIEW, it is expected to have some limitations in its use. These limitations are due to the complexity of the LabVIEW algorithms used in the system, the computer's processing speed, the interruptions often made as the operating system is working (Windows is not a real-time operating system), and the computer's sound card specifications. As discussed before, a SDR system requires a lot of computation done by the DSP algorithms. When the system is

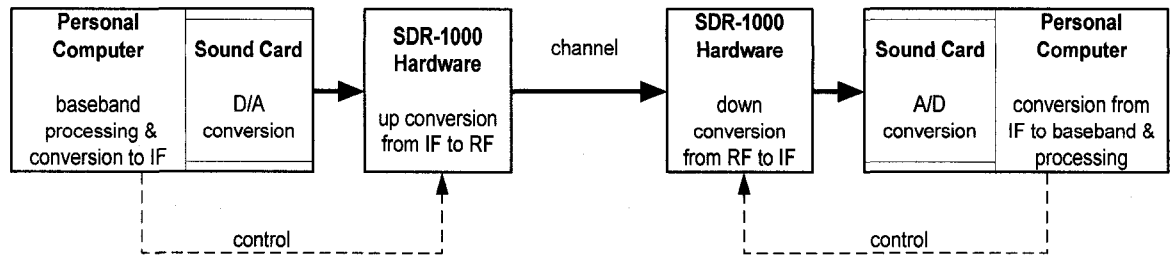
not fast enough (processing takes too much time), information sent or received by the computer is likely to be lost. In order to reduce this problem, the number of sound card buffers can be increased in the receiver. This procedure gives more time for processing before the overall sound card buffer exceeds its capacity (buffer overflow). When the transmitter or receiver is in an idle state (no heavy computations are being done), there is no problem of buffer overflow. However, buffer overflows can occur at some point if the system processes heavy computations continuously.

LabVIEW has a sound card interface that can be used to send and receive signals to and from the computer's sound card. It uses the `lvsound.dll` library provided by National Instruments. However, this interface has some limitations in terms of sample rates, resolution, number of buffers, and number of channels. For this reason, another interface was chosen to manipulate these parameters: the Wave I/O Sound Card Interface designed by Zeitnitz [12]. By using this interface, the number and size of the sound card buffers can be readily adjusted, as well as the sound card sample rate.

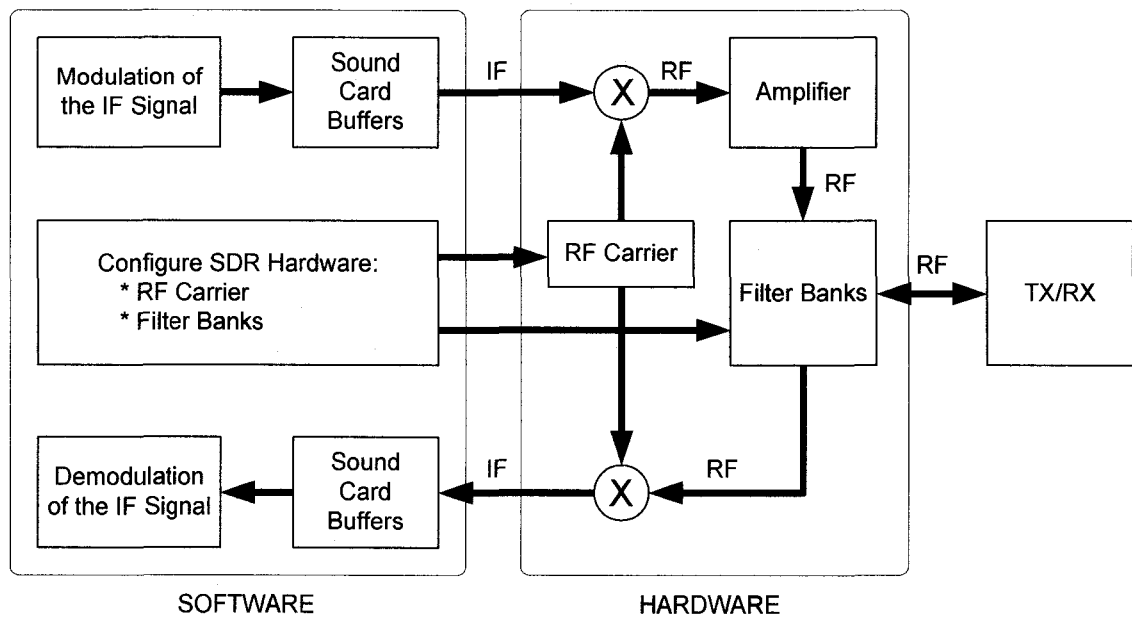
The proposed SDR teaching system using LabVIEW and the computer's sound card was developed for a 4 QAM modulation (4 QAM SDR Transceiver.vi). A set of laboratory experiments was designed as a guide to explore the entire SDR system. Students can easily change parameters and/or blocks of code and observe the behavior of the system at several different nodes. The 4 QAM SDR Transceiver is discussed in more detail in chapters 4 and 5.

2.5 – The Flex-Radio Systems SDR-1000 Platform

The FlexRadio Systems SDR-1000 Software Radio was developed by Gerald Youngblood originally for use in amateur radio communications. It started using Visual Basic as the software platform and later it has migrated to Microsoft C#. The software runs in a Windows-based personal computer using its sound card to send or receive an intermediate frequency (IF) modulated signal. This system uses hardware specially designed to convert the IF signal to the radio frequency (RF) range and vice-versa. The SDR-1000 hardware can be adapted to work with the designed LabVIEW based SDR learning system, as it was done by Plante [7] and Beckwith [8] in their SDR system using MATLAB. Actually, by using Mathscript (one of the LabVIEW built-in functions), it is possible to incorporate the same hardware control code used by Plante and Beckwith into LabVIEW to control the SDR-1000 hardware interface. Since the focus of this thesis is on developing the LabVIEW user interface, the incorporation of hardware control code was deferred to a later time. Figure 2.10 shows the structure of an SDR system using the RF stage implemented in hardware.



(a) Block diagram



(b) Signal flow diagram

Figure 2-10 – SDR system using RF front end hardware stage

The SDR-1000 hardware operation is controlled via the computer's parallel port. When the system is set to transmit, the parallel port sends the proper controls to specify the carrier frequency of the transmitted RF signal and to select the proper filter to send the signal to the antenna or connecting cable. When it is set to receive, the signal comes from the antenna and the computer's

parallel port sends the appropriate controls to the filter bank to select a bandpass filter and also to generate an RF carrier to be mixed with the signal coming from the filter bank. After this processing, the signal arrives at the computer's sound card in the IF band to be processed by the software. For more detail on the SDR-1000 hardware see [7, 8, 18, 19, 20, and 21].

2.6 – Laboratory Experiments

A set of laboratory experiments is also proposed as a guide for the digital communication teaching platform using LabVIEW. These experiments were designed to expose all of the working parts of a real digital communication system, from the binary information generation to transmission and from signal reception to the binary information extraction at the receiver.

The first proposed experiment (Lab #1) introduces the LabVIEW graphical environment to the student via a tutorial in the pseudo-random binary sequences theory. The student can become familiar with the basic functions and structures used in LabVIEW and explore its capabilities by designing a pseudo-random binary sequence generator.

The second experiment (Lab #2) introduces the SDR system using LabVIEW and the computer's sound card. In this experiment, students will become familiar with a real SDR system and its block diagram. They will be able to observe the frequency spectrum of the modulated signal as changes in the carrier frequency are applied. Students can also verify the existence of aliasing in

the modulated signal when the symbol rate is set too high. In addition, the sound card's sample frequency offset can be observed, since a sound card is likely to sample at a slightly different sample rate from another one.

The third experiment (Lab #3) deals with pulse shaping. With this experiment students will explore the task of pulse shaping in communication systems. By using different pulse shapes, they will be able to compare the baseband modulated signal using constellation and eye diagrams. Pulse shape parameters can also be varied to observe the signal behavior at points of interest.

Lab #4 covers the carrier recovery process. The SDR system works with a 4 QAM modulation scheme and uses the 4 QAM Costas Loop algorithm to recover the IF modulated signal to its baseband form. Students will become familiar with the down-conversion of the IF signal to its baseband form by exploring this algorithm.

Symbol timing recovery is covered in Lab #5. The Delay-Locked Loop (DLL) and the Max-Eye algorithms are presented and students can explore their performance in the system.

These proposed laboratory experiments are included in Appendix A of this document.

2.7 – Conclusion

Considering the requisites for a digital communication system teaching platform, a SDR using LabVIEW was presented in this chapter. This system provides flexibility and modularity, making it possible to replace, modify, add, and/or eliminate blocks of code to attend specific goals. It has a relatively low cost, since it uses only a personal computer with LabVIEW software installed. Also, since the only wired connection is accomplished with an audio cable to connect the sound cards of the transmitter and the receiver computers, the risk of accidental damage becomes essentially negligible. In addition, the three main learning styles can be explored by using this system: students can actuate switches and buttons, modify/create blocks of code, hear the sound of the modulated signals from the computer's sound card, and visualize the signal's behavior in several nodes of the system.

A set of laboratory experiments is proposed to be used with this SDR system. These experiments can be modified to attend different tasks.

In conclusion, a combination of hardware and software that implements the characteristics presented above provides a robust platform for conveying the fundamental, practical concepts of digital communications.

CHAPTER 3

DIGITAL COMMUNICATION SYSTEMS AND THE SDR

A communication system is typically made of a transmitter, a communication channel and a receiver. There are many operational building blocks responsible for different tasks inside this structure. This chapter discusses the working blocks of a typical digital communication system and how software-defined radio (SDR) concepts can be incorporated into it. First, the block diagram of a classic system is presented followed by a description of each block. A discussion on how software and hardware can fit into this system follows next: the SDR implementation, its tasks, applications, advantages, and limitations.

3.1 – The Typical Digital Communication System Structure

The transmitter, the receiver, and the communication channel form the basic composition of a communication system. The transmitter and receiver, however, may have several different stages where the information signal needs to be processed. A general block diagram is depicted by figure 3-1.

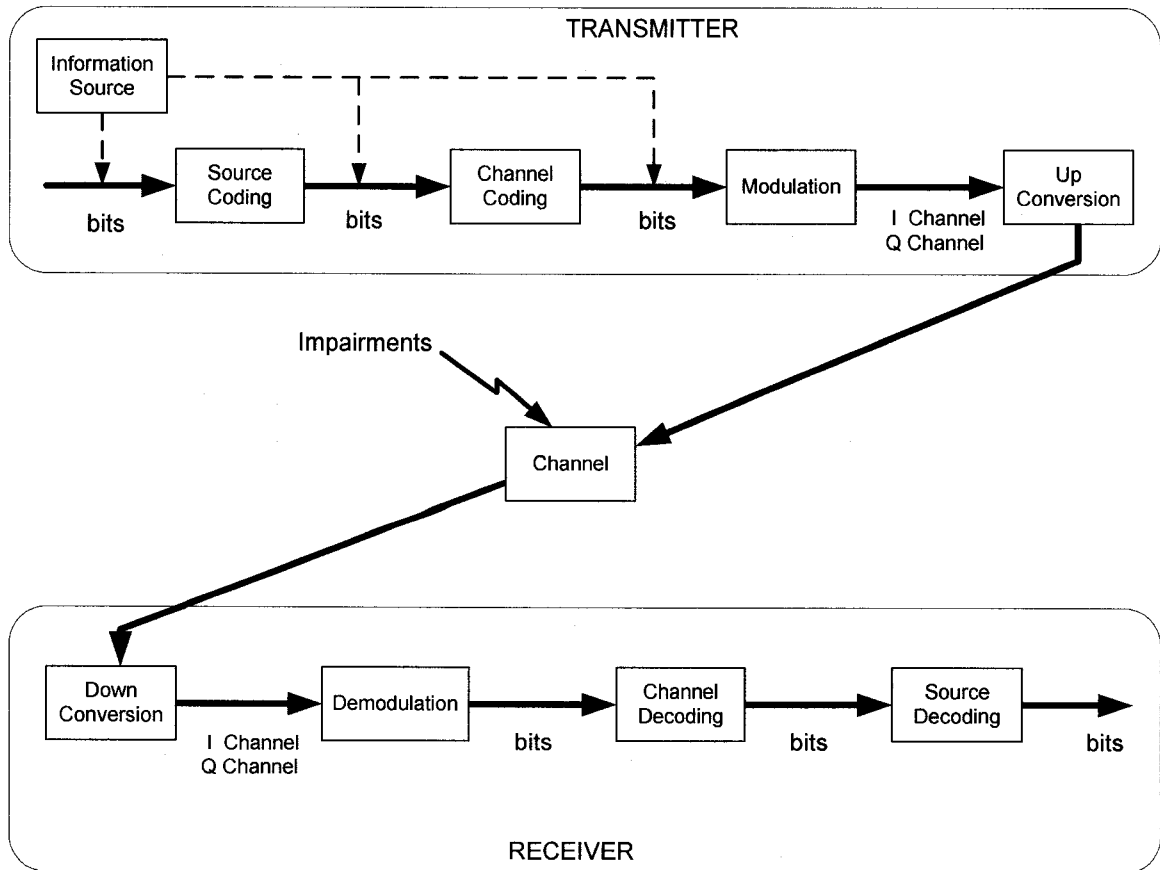


Figure 3-1 – Typical digital communication system block diagram

The information source generates the signal for transmission. This signal can be analog, such as voice, or digital, such as a binary sequence. An analog to digital (A/D) conversion is necessary if the information signal is analog. The information, generally a baseband signal, can be directly processed by the modulation block, or passed through a source coding process and/or a channel coding stage. A description of each block is given next. Many, but not all, of these blocks appear in the LabVIEW implementation of the 4 QAM SDR Transceiver.

3.1.1 – Source Coding

Data compression algorithms are generally applied in the source coding stage to minimize transmission bandwidth. Examples of source coding are JPEG and MPEG.

3.1.2 – Channel Coding

The channel coding stage is used to increase immunity to noise and interference at the receiver by using error correction algorithms. This is done by adding redundant data bits to the data sequence. Examples of error-correcting codes are Convolutional, Hamming, Golay, and Reed-Solomon.

3.1.3 – Modulation

The modulation stage is responsible for applying the desired modulation scheme on the incoming signal and splitting it into quadrature components: the in-phase (I) and the quadrature-phase (Q) components. In this stage, pulse-shaping is applied to the data bit stream in order to reduce intersymbol interference and bandwidth usage.

3.1.4 – Up-Conversion

The up-conversion stage takes the incoming complex I/Q baseband signal and converts it into a real passband radio frequency (RF) signal to be transmitted through the communication channel.

3.1.5 – Channel

The transmitted signal is passed through the communication channel where it may suffer from impairments due to noise, interferences, and fading.

3.1.6 – Down-Conversion

The down-conversion stage is responsible to convert the incoming RF signal into its original complex I/Q baseband form. In order to do this, a local, phase coherent carrier is generated in the receiver and mixed with the RF signal. The resultant signal is passed through a low pass filter to recover the complex I/Q baseband signal.

3.1.7 – Demodulation

The demodulation stage processes the complex I/Q baseband signal to extract the information. This process involves matched filtering, symbol timing extraction, symbol synchronization, frequency offset correction, and equalization.

3.1.8 – Channel Decoding

The channel decoding stage is responsible for correcting the incoming information signal using the specified decoding algorithm for the error-correcting code used by the transmitter.

3.1.9 – Source Decoding

The source decoding stage uses the equivalent decompression algorithm for the data compression method used by the transmitter to retrieve the original information.

3.1.10 – Tools Used in a Digital Communication System

Different tools are used to investigate a digital communication system. Normally, the bit error rate (BER) is a measure to identify how well a system performs. Other measurements like I/Q Gain Imbalance, Quadrature Skew, Magnitude Error, Phase Error, Error Vector Magnitude (EVM), and Rho (ρ) are also common to determine the system performance. Among the most common tools for visualization are the Eye diagram, the Constellation diagram, the Trellis diagram, and the XY graph.

3.2 – SDR in a Digital Communication System

The use of software-defined radio in communication systems has become more popular with the advances in the computer technology. The intense computations required by the DSP algorithms can be handled by fast processors running under real-time operating systems. Thus, software can be used to implement the key stages of an entire communication system, and thereby reducing the need for protocol-specific hardware elements.

Figure 3-2 shows a block diagram of a digital communication system highlighting the parts implemented in software and the ones implemented in hardware.

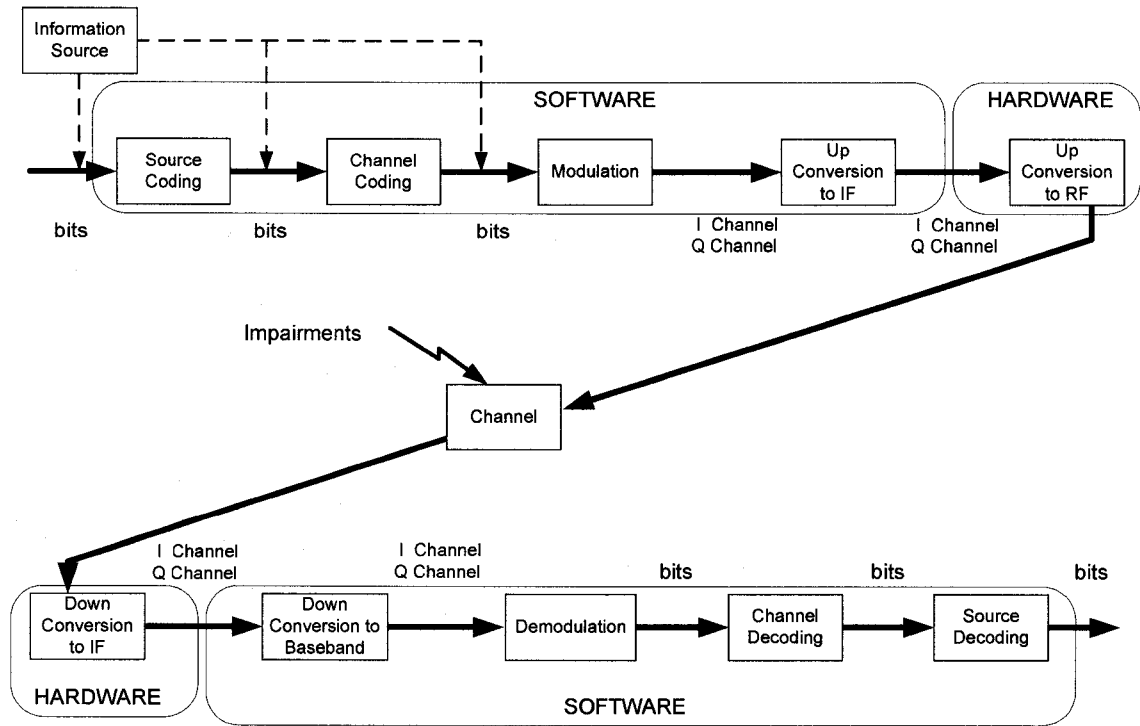


Figure 3-2 – Software and hardware implementation in a digital communication system

As stated earlier, software is responsible for all the key tasks where important operations within the system are defined. It is within the software that the choices of coding, modulation schemes, and frequency band are specified. The hardware used by SDR systems consists of a superheterodyne RF front end responsible for converting intermediate frequency (IF) signals to radio frequency (RF) signals during transmission and vice-versa during reception. This hardware

is also controlled by software, which determines the desired frequency band to be used. An analog to digital (A/D) converter is also necessary to convert analog IF signals into a digital form for processing within the software and vice-versa.

The use of software-defined radio has gained attention for its flexibility and modularity. Software-defined radio devices can operate using various communication protocols in different frequency bands. These devices can be reconfigured on-the-fly and can incorporate enhanced features with software upgrades. In addition, SDR concepts can be used for building radio devices (cognitive radios) capable of optimizing their own performance by looking at the RF spectrum in their neighborhood and reconfiguring their settings accordingly.

When using a personal computer running under a non-real time operating system like Windows and a programming environment like LabVIEW, there are some limitations to which one must pay attention when working with a SDR system. The intense computations required by the DSP algorithms can take a lot of the processing time, particularly in the receiver part of the radio. Even when allocating a large number of sound card buffers, buffer overflow may occur during the demodulation process. Besides increasing the number of sound card buffers, it is necessary to keep the code optimized and clean of unnecessary operations. Also, some software platforms are better than others when dealing with the DSP algorithms, since certain code languages take more time than others to process a specific task.

3.3 – Summary

Software-defined radios can be used in a complete digital communication system providing more flexibility and modularity. They can offer the use of different communication protocols, modulation types, operating frequency band, and allow changes of configuration on-the-fly. In addition, upgrades to enhance their performances can be easily added to the system.

For these advantages, the SDR has gained a lot of attention in the radio communications scenario. It has already being used in military applications and cell phone base stations, and has a great potential for several other types of applications, like the teaching platform proposed by this document.

By compensating for the limitations when used in a computer running under a non-real time operating system with a particular software programming language, an SDR can be used satisfactorily to provide a low-cost alternative to a digital communication system teaching platform.

CHAPTER 4

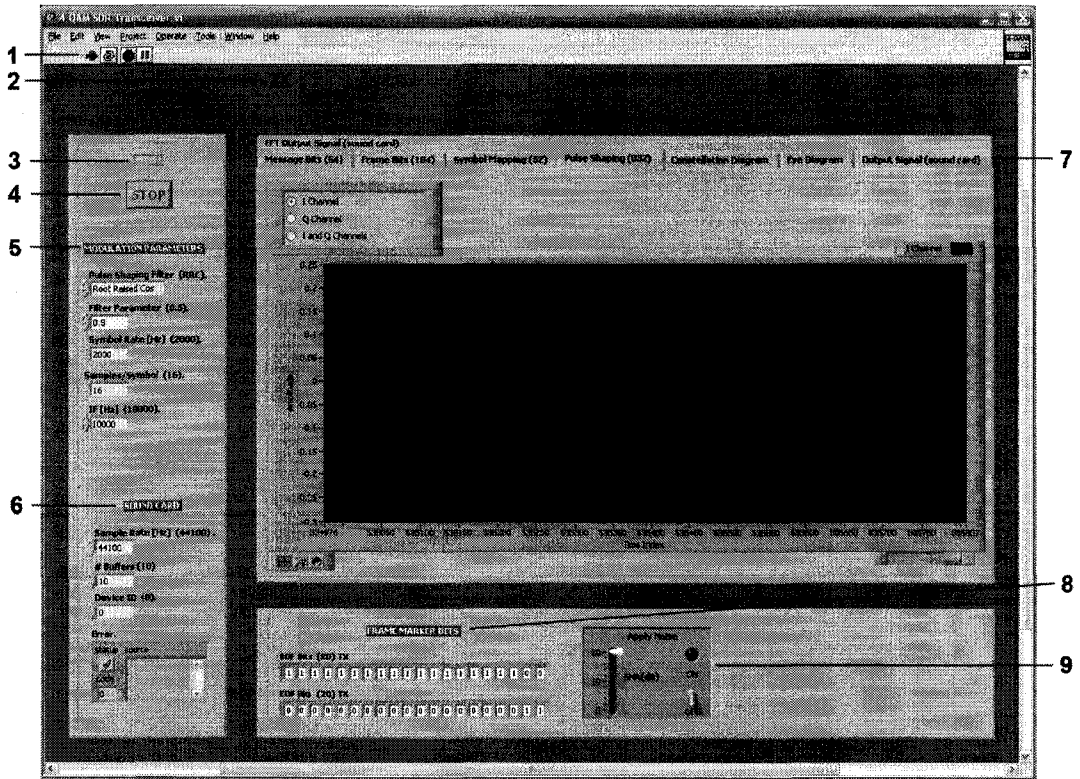
THE 4 QAM SDR TRANSCEIVER: USER INTERFACE AND OPERATION

Quadrature Amplitude Modulation (QAM) is a technique widely used in digital communication systems. Applications like high speed cable, FAX modem, multi-tone wireless, and satellite channels make use of this type of modulation scheme. 4 QAM was the modulation technique chosen to implement the SDR teaching platform using LabVIEW for its simplicity, as compared to more complex QAM schemes like 64 QAM and 256 QAM used by digital cable television and cable modems. This chapter introduces the 4 QAM SDR Transceiver implemented in LabVIEW using the personal computer's sound card. The user interface (the VI's front panel) is discussed in detail in section 4.1, while the system operation is presented in section 4.2. The code that implements the SDR is presented in chapter 5, in which is discussed the pertinent digital communications theory and the LabVIEW implementation of each block of the system. Figures 5-2 and 5-16 show the block diagram of the transmitter and the receiver structures, respectively.

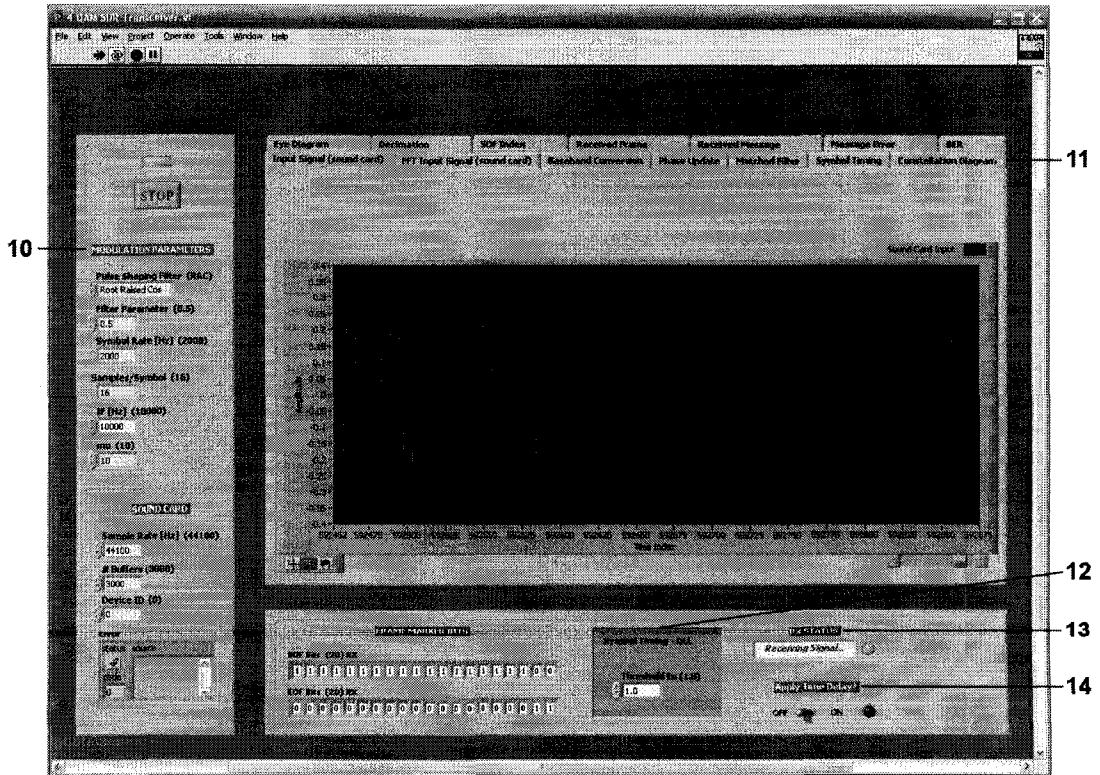
4.1 – Front Panel and System Description

The 4 QAM SDR Transceiver was designed to integrate the basic working stages of a digital communication system and to offer easy access for operation control, system parameter changes, and signal observation for use as a teaching tool in a digital communication laboratory. The user can control the modulation parameters like pulse shape filter type, symbol rate, and carrier frequency and observe the system behavior as a consequence of those changes. Sound card parameters like sample rate and number of buffers can also be manipulated by the user. This system provides graphs for visualization of signals in the time, frequency, and I/Q domains in several points of interest within the circuitry of the SDR. In addition, it allows modifications in the code to add more features to the system.

The front panel of the 4 QAM SDR Transceiver.vi is shown in figure 4-1; figure 4-1 (a) shows the front panel when the radio is set to the transmitter mode and figure 4-1 (b) shows it in the receiver mode. A description of each part of the front panel is given next with reference to figure 4-1, which helps to understand how the radio can be operated. It is relevant to note that the default values for the controls are displayed in parentheses in the control panel. More details regarding the operation of the 4 QAM SDR Transceiver are given in section 4.2.



(a) Transmitter mode



(b) Receiver mode

Figure 4-1 – 4 QAM SDR Transceiver front panel

4.1.1 – Execution Control

These icons (1) control the execution of the VI (virtual instrument). A description of the execution icons in reference to figure 4-2 is given next.

a – Run button: it starts the execution once.

b – Run Continuously button: it starts the execution once and then every time the VI is stopped.

c – Abort button: it aborts the VI's execution. Its use is not recommended for operation of this VI, since it can cause problems when the VI starts running again.

d – Pause button: it pauses the VI's execution. This is particularly useful for printing graphs.

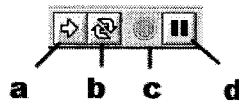


Figure 4-2 – VI's execution control

4.1.2 – Transmitter/Receiver Mode Tabs

These tabs (2) are used to choose the desired operating mode of the transceiver: TX for transmitter or RX for receiver.

4.1.3 – ON Indicator

When the VI is running, this indicator (3) will display a green light. It turns off as the VI is stopped.

4.1.4 – Stop Button

This is the button (4) that must be used to stop the VI's execution.

4.1.5 – Modulation Parameters in Transmitter Mode

Figure 4-3 shows the modulation parameters (3) defined in the control panel. In reference to this figure, these parameters are:

a – Pulse Shaping Filter: it selects the desired pulse shaping filter type. The options are raised cosine (RC), root raised cosine (RRC), and none. The default value is RRC.

b – Filter Parameter: this is the value used for the filter roll-off factor (α). Its default value is 0.5.

c – Symbol Rate: It defines the symbol rate for the information transmission in symbols per second or Hz (Hertz). The default value is 2000 Hz.

d – Samples/Symbol: it defines the number of samples per symbol used in the pulse shaping filter. The default value is 16.

e – IF: it is the intermediate frequency value used in the carrier in Hz. The default value is 10 kHz.

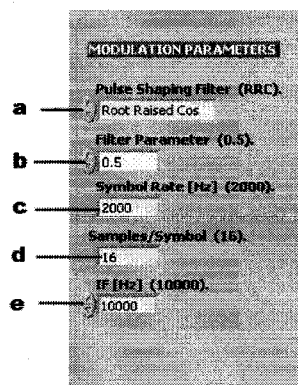


Figure 4-3 – Transmitter modulation parameters

4.1.6 – Sound Card Parameters

The sound card parameters (6) are explained as follows, according to figure 4-4:

a – Sample Rate: it defines the sound card sample rate. The default value is 44.1 kHz.

b - # of Buffers: it defines the number of buffers used by the sound card. The default value is 10 when the VI is in transmitter mode and 3000 when it is in the receiver mode.

c – Device ID: it is the identification number of the device used as sound output while in the transmitter mode or sound input when in receiver mode.

d – Error: it displays the sound card error status and the source of the error.

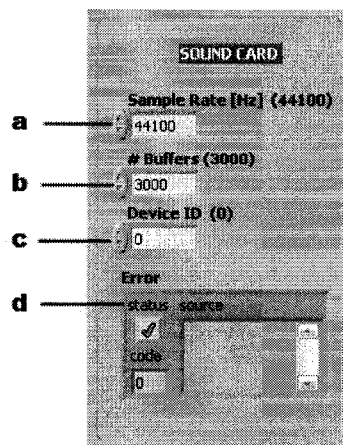


Figure 4-4 – Sound card parameters

4.1.7 – Graph Display Tabs in Transmitter Mode

The system allows the user to observe waveforms at several points of interest by clicking in the desired graph display tab (7). Figure 4-5 shows the transmitter graph display tabs. They are explained as follows.

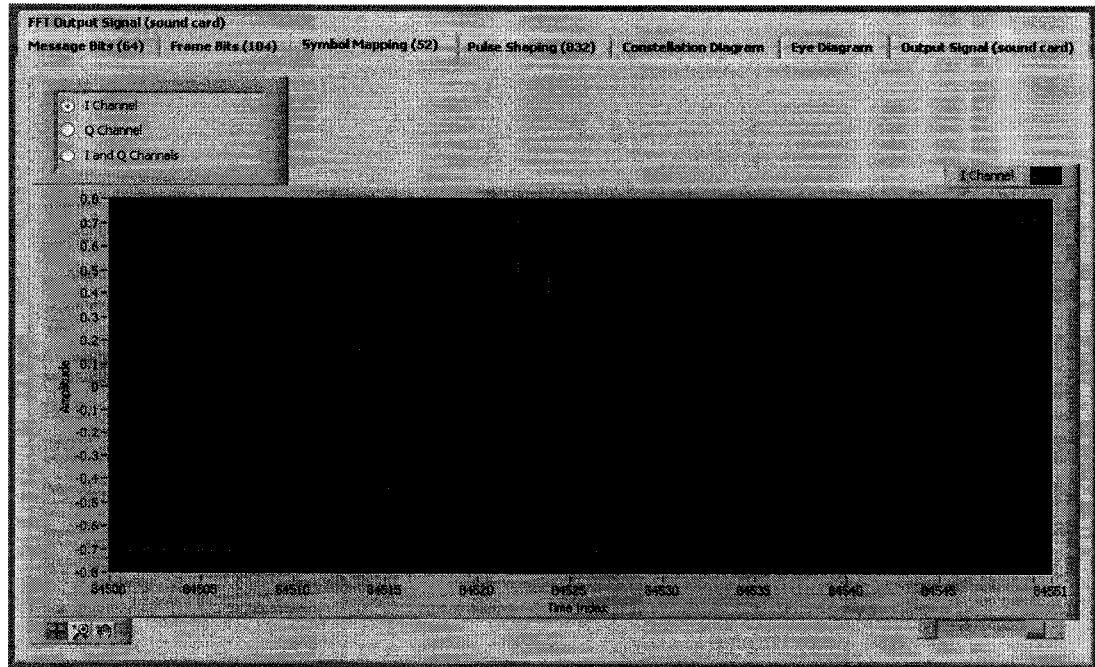


Figure 4-5 – Graph display tabs for transmitter mode

Message Bits: the binary original information data. This VI uses a pseudo random binary sequence generator to create 64 binary digits.

Frame Bits: each block of information to be sent by the transmitter is encapsulated into a frame for synchronization purposes. The frame structure starts with a predefined 20-bit sequence of a defined bit pattern called SOF (start of frame) bits. The message bits are appended next and are followed by another

20-bit sequence of another predefined pattern called EOF (end of frame) bits.

The total frame size is 104.

Symbol Mapping: it displays the in-phase and quadrature-phase signals after the frame bits are converted into complex symbols. A symbol is defined as a point determined by the complex I/Q plane. The 104 frame bits are divided into the in-phase (I) and quadrature-phase (Q) channels with 52 samples each.

Pulse Shaping: it shows the up-sampled and pulse-shaped signals in the I and Q channels. In each channel, the signal is up-sampled by the samples per symbol value specified in the Samples/Symbol control. For its default value (16), there will be 832 samples per frame in each channel.

Constellation Diagram: it displays the constellation diagram of the transmitted signal.

Eye Diagram: it shows the eye diagram of the transmitted signal.

Output Signal (sound card): it displays the signal in the time domain after it is up-converted by the IF carrier and it is ready to be sent to the sound card for transmission. The output signal amplitude is adjusted by a simple automatic gain control to get a better dynamic range in the sound card.

FFT Output Signal (sound card): it displays the signal sent to the sound card for transmission in the frequency domain.

4.1.8 – Frame Marker Bits in Transmitter

In these controls (8) the SOF and EOF bit patterns are defined.

4.1.9 – Apply Noise

The user can add white Gaussian noise to the transmitted signal to verify the performance of the receiver at different levels of noise in the transmission path. This can be done using the Apply Noise controls (9) on the front panel, as shown in detail in figure 4-6. According to this figure, the controls and the indicator are as follows.

a – SNR (dB) Slide Control: it is the amount of signal to noise ratio (SNR) in decibels.

b – On/Off Switch: it turns on and off the addition of noise at the transmitted signal.

c – Noise Indicator: a green light indicates the noise addition is on; otherwise, the indicator is turned off.

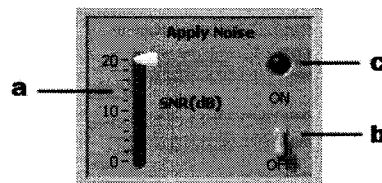


Figure 4-6 – Apply noise control

4.1.10 – Modulation Parameters in Receiver Mode

In the receiver mode, the modulation parameters (10 in figure 4-1, and **a**, **b**, **c**, **d**, and **e** in figure 4-7) must match the ones set in the transmitter. Besides having all of the section controls of the transmitter, the receiver has the control μ (μ), indicated in figure 4-7 by the letter **f**. This is the adaptation constant value used by the phase synchronization algorithm. Its default value is 10.

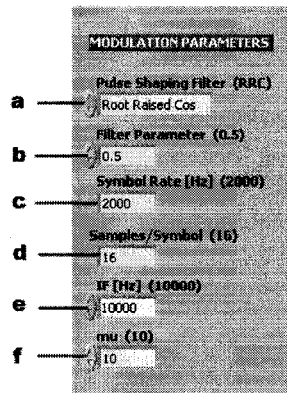


Figure 4-7 –Modulation parameters in receiver mode

4.1.11 – Graph Display Tabs in Receiver Mode

Figure 4-8 shows the graph display tabs of the receiver mode (11). They are described as follows.

Input Signal (sound card): it displays the signal received by the sound card in the time domain.

FFT Input Signal (sound card): it displays the signal received by the sound card in the frequency domain.

Baseband Conversion: it shows the signals in the I and Q channels after carrier synchronization and down-conversion to baseband.

Phase Update: it displays how the phase is updated by the carrier synchronization algorithm.

Matched Filter: it displays the signals at the output of the matched filter.

Symbol Timing: it shows the symbol clock recovered by the symbol clock recovery algorithm.

Constellation Diagram: it displays the constellation diagram of the received signal after demodulation.

Eye Diagram: it shows the eye diagram of the received signal in the I and Q channels.

Decimation: it displays the output of the decimation block, when the signal is down-sampled by the same amount it was up sampled in the transmitter.

SOF Index: it shows the index of the first start of frame (SOF) bit.

Received Frame: it displays the 104-bit frame received from the demodulation process.

Received Message: it shows the 64-bit received message.

Message Error: it displays the error in the received message.

BER: it displays the bit error rate.

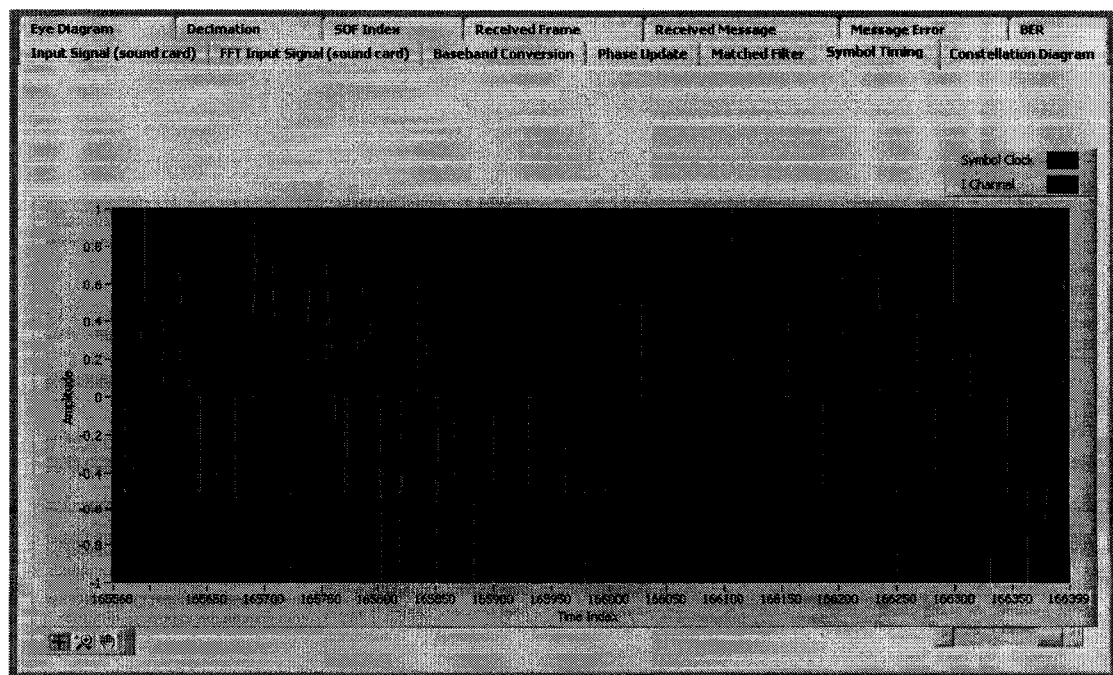


Figure 4-8 – Graph display tabs for receiver mode

4.1.12 – Symbol Timing – DLL

In this section (12), the value of the threshold E_0 used in the symbol timing recovery algorithm is defined. Its default value is 1.0.

4.1.13 – RX Status

This indicator (13) displays the reception status. Figure 4-9 shows the RX Status display. A description of the contents of this display is given next, according to the figure:

a – Alphanumeric Display: it displays the message “Receiving Signal” when processing a received signal. Otherwise, it displays “No Signal”.

b – Reception Indicator: when receiving a transmitted signal, it lights up. If the signal is too weak to be processed, the indicator turns off.



Figure 4-9 – RX status display

4.1.14 – Apply Time Delay Control

This section (14) allows the user to apply a processing delay of 2 seconds for each iteration of the main processing loop of the code. This gives to the user the capability of slowing down the demodulation process in order to better understand how it is being done when looking at the graphs. Figure 4-10 depicts this control in the front panel. It is described as follows:

a – On/Off Switch: it turns on or off the time delay.

b – Delay Indicator: it turns on when the time delay is applied to the system.

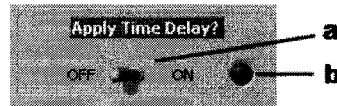


Figure 4-10 – Time delay control

4.2 – The 4 QAM SDR Transceiver Operation

The 4-QAM SDR Transceiver system works with two personal computers with the LabVIEW software and the 4-QAM SDR Transceiver.vi installed on both computers. Each computer has the ability to work as transmitter or receiver, but to make things simple we are considering one of them will be the transmitter while the other computer will work as the receiver. Both transmitter and receiver need to agree on the modulation parameters for proper communication between them. The sound card output of the transmitter is connected via an audio cable to the sound card input of the receiver.

To run the receiver, one must click on the start button of the VI (see figure 4-2). When the transmitter is not working, i.e., no signal is being transmitted, the receiver may get a very weak signal which will not be recognized as an information signal to be decoded, and the signal is discarded before any more processing. The receiver does that by analyzing the incoming signal power and comparing it to a pre-specified value. When the transmitter is working, the signal

arrives at the receiver with a larger power. When this power is large enough to overcome that pre-specified value, then the demodulation process starts.

If the user desires to stop the transmitter or the receiver, he/she needs to do that by clicking on the stop button in the control panel. The user should not use the abort button located in the VI's execution control shown in figure 4-2. Using the abort button can cause memory problems when the VI is set to run again. If this is done accidentally, it is recommended to close the VI and open it again to clear the memory usage.

The VI execution can be paused anytime by pressing the pause button. This is a convenient resource to copy a graph to another application like a word processor. To copy a graph to another application, the user can right-click on the graph when the VI is paused and choose the desired option from the pop up menu.

The user can try several changes in parameters by just changing the control values in the control panel. If he/she wants to get back the default values for these controls, he/she can click on the "Edit" menu and choose "Reinitialize Value to Default."

Some of the graph tabs offer the choice of displaying one or two graphs. The user can select the desired graph by clicking on the respective radio button located in the tab.

The demodulation process in the receiver takes a lot more processing time than the modulation in the transmitter. Buffer overflow at the receiver sound card may occur at some point when the receiver is working in a continuous

demodulation process. This problem can be reduced by increasing the number of sound card buffers. Typically, with the default value of 3000 sound card buffers the receiver can work in the demodulation process for about 3 minutes. After a buffer overflow condition, it is required to reinitialize the receiver. The receiver checks the presence of a possible incoming signal for demodulation. The demodulation process only starts after verifying the presence of an incoming energy signal at the sound card buffer. This helps to avoid unnecessary processing and buffer overflows. The sound card error display indicator notifies the user when a sound card related error occurs.

It is important to notice that this system runs under Windows, which is not a real-time operating system. Interruptions made in the computing environment can cause strange behavior in the SDR system. Sometimes it is recommended that a shut down and reboot of the entire system be done in order to get the SDR system working properly again.

CHAPTER 5

THE 4 QAM SDR TRANSCEIVER: THEORY AND LABVIEW CODE IMPLEMENTATION

The user interface and operation of the 4 QAM SDR Transceiver were explained in chapter 4. This chapter starts with an introduction to the quadrature modulation theory. Next, it discusses the 4 QAM SDR Transceiver structure, starting with the transmitter and later the receiver. The relevant theory behind each stage of the system is also presented, as well as the corresponding LabVIEW code implementation. The entire code, the subVIs icons, and some other VI implementations are presented in appendix B.

5.1 – Quadrature Signals

Quadrature signal modulation is a technique where two sinusoids out of phase with each other by 90° are used as carriers. The information to be sent is divided into two components: in-phase ($I(t)$) and quadrature-phase ($Q(t)$). The $I(t)$ and $Q(t)$ signals are put into the in-phase and quadrature-phase channels, respectively, and mixed with the sinusoids of frequency f_c and phase θ . The transmitted signal $s(t)$ is given by the form:

$$s(t) = I(t) \cos(2\pi f_c t + \theta) - Q(t) \sin(2\pi f_c t + \theta) \quad (5-1)$$

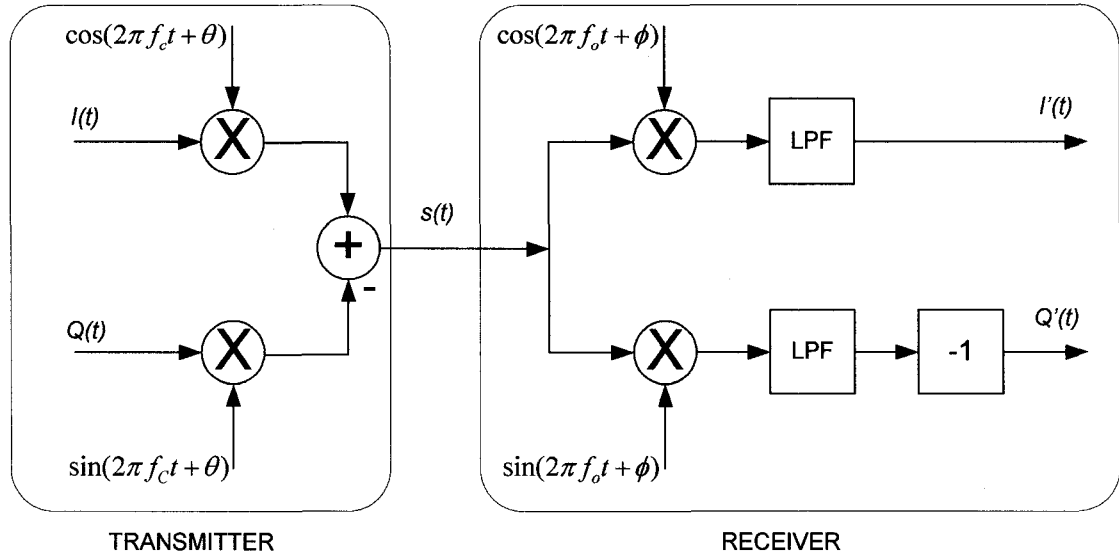


Figure 5-1 – Quadrature modulation and demodulation

The received quadrature signals $I'(t)$ and $Q'(t)$ are given by:

$$\begin{aligned}
 I'(t) &= LPF \{s(t) \cos(2\pi f_o t + \phi)\} \\
 &= LPF \{(I(t) \cos(2\pi f_c t + \theta) - Q(t) \sin(2\pi f_c t + \theta)) \cos(2\pi f_o t + \phi)\} \\
 &= LPF \{I(t) \cos(2\pi f_c t + \theta) \cos(2\pi f_o t + \phi) - Q(t) \sin(2\pi f_c t + \theta) \cos(2\pi f_o t + \phi)\}
 \end{aligned} \tag{5-2}$$

$$\begin{aligned}
 Q'(t) &= -LPF \{s(t) \sin(2\pi f_o t + \phi)\} \\
 &= -LPF \{(I(t) \cos(2\pi f_c t + \theta) - Q(t) \sin(2\pi f_c t + \theta)) \sin(2\pi f_o t + \phi)\} \\
 &= -LPF \{I(t) \cos(2\pi f_c t + \theta) \sin(2\pi f_o t + \phi) - Q(t) \sin(2\pi f_c t + \theta) \sin(2\pi f_o t + \phi)\}
 \end{aligned} \tag{5-3}$$

where f_o and ϕ are the locally generated carrier frequency and phase, respectively.

Considering the relationships:

$$\sin(x) \cos(y) = \frac{1}{2} [\sin(x-y) + \sin(x+y)] \quad (5-4)$$

$$\cos(x) \cos(y) = \frac{1}{2} [\cos(x-y) + \cos(x+y)] \quad (5-5)$$

$$\sin(x) \sin(y) = \frac{1}{2} [\cos(x-y) - \cos(x+y)] \quad (5-6)$$

$$\sin(-x) = -\sin(x) \quad (5-7)$$

and the frequencies f_c and f_o with very close values, which makes $f_c + f_o \approx 2 f_o$,

the signals $I'(t)$ and $Q'(t)$ become:

$$I'(t) = \frac{1}{2} I(t) \cos[2\pi(f_c - f_o)t + (\theta - \phi)] - \frac{1}{2} Q(t) \sin[2\pi(f_c - f_o)t + (\theta - \phi)] \quad (5-8)$$

$$Q'(t) = \frac{1}{2} I(t) \sin[2\pi(f_c - f_o)t + (\theta - \phi)] + \frac{1}{2} Q(t) \cos[2\pi(f_c - f_o)t + (\theta - \phi)] \quad (5-9)$$

If the receiver retrieves the proper carrier frequency and phase, i.e., $f_o = f_c$

and $\theta = \phi$, the signals $I'(t)$ and $Q'(t)$ become

$$I'(t) = \frac{1}{2} I(t) \quad (5-10)$$

$$Q'(t) = \frac{1}{2} Q(t) \quad (5-11)$$

which are scaled versions of the original $I(t)$ and $Q(t)$ signals. In order to generate the proper carrier frequency and phase, the receiver needs to have a

carrier frequency and phase synchronization device. See [9] for more details on quadrature modulation.

5.2 – Transmitter Structure

In the 4 QAM SDR Transceiver system, the transmitter takes a binary message and encapsulates it into a frame for synchronization purposes. This frame, also a binary sequence, passes through a differential encoding process to help solving the phase ambiguity problem in the carrier synchronization stage of the demodulation stage. The differential encoded sequence bits are then mapped into symbols in the complex I/Q plane. In this stage the sequence is split bitwise into the in-phase (I) and quadrature-phase channels (Q). Later, pulse shaping is applied on both I and Q channels. These signals are mixed with a carrier for up-conversion to the desired intermediate frequency (IF). The quadrature modulated signal is scaled for better use of the sound card dynamic range and sent for transmission by the sound card buffers. Figure 5-2 shows the transmitter structure. The details of each of these blocks are presented next.

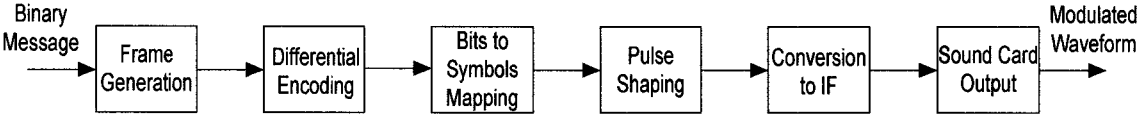


Figure 5-2 – Transmitter structure

5.2.1 – Frame Generation

The binary message is encapsulated into a frame pattern to be transmitted, as shown in figure 5-3. The frame structure starts with 20 bits of a predefined pattern, called SOF (start of frame) bits. After the SOF bits, the message bits are appended. The message bits used in this VI comprise a 64-bit sequence generated by the 64 PN Sequence Generator.vi. Another 20-bit sequence of a different predefined pattern, called EOF (end of frame) bits, is appended after the message bits. In the receiver, the SOF and EOF bits are correlated with the received bit sequence in order to extract the correct frame. They are also used for resolving the phase ambiguity of even multiples of $\pi/2$ radians at the carrier synchronization stage. These bit patterns can be seen and modified in the front panel.



Figure 5-3 – Frame structure

The 64 PN Sequence Generator.vi is a pseudo-noise sequence generator that produces 64 1-digit binary periodic samples. Pseudo noise (PN) sequence generators can be built using linear feedback shift registers. In these binary sequences, also called pseudo-random, maximal length, or m-sequences, the elements (0s and 1s) have a probability very close to 0.5.

The PN sequence generator used in the transmitter has a six-stage linear feedback shift register structure. Figure 5-4 shows this structure.

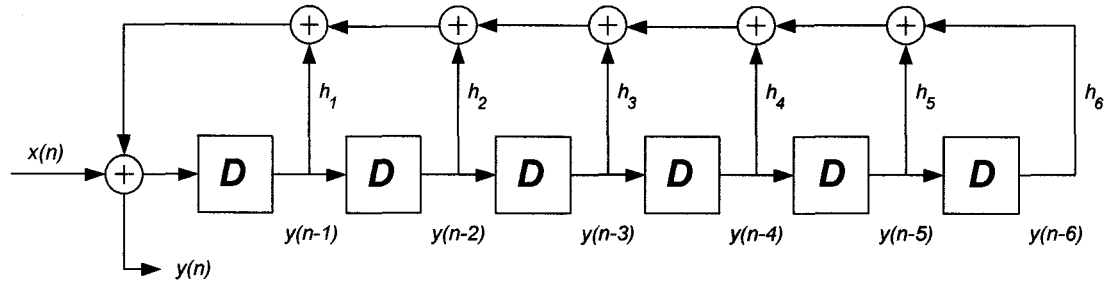


Figure 5-4 – PN generator with a six-stage linear feedback shift register structure

The connection polynomial used to build the PN generator is given by:

$$h(D) = 1 + D + D^6 \quad (5-12)$$

where D is the delay and the summations represent modulo 2 additions. The 64 Sample Period PN Sequence Generator.vi is implemented using this connection polynomial, which allows the sequence generator to have a period of 63 ($2^6 - 1 = 63$). In order to have a 64-bit periodic sequence (even number of bits to be combined into symbols in the I and Q channels), for each cycle one bit is added to the end of the generated sequence. Figure 5-5 shows the LabVIEW code that implements this sequence generator. Figure 5-6 depicts the frame generation stage used in the main code. Refer to [2] for more detail on PN sequences.

The message bits and the frame bits can be observed from the Message Bits and Frame Bits tabs in the front panel, respectively, when the VI is running as a transmitter.

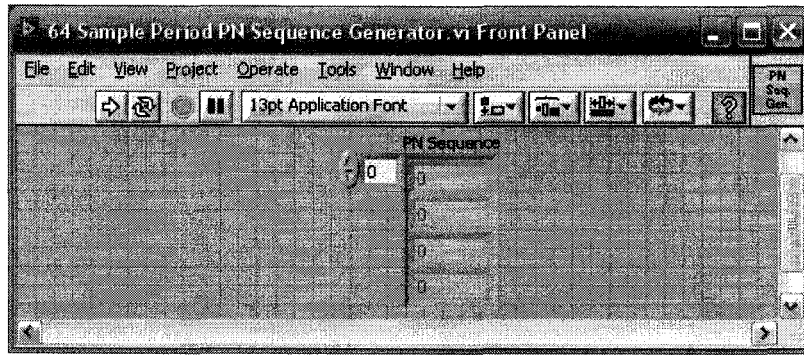
5.2.2 – Differential Encoding

Differential encoding is applied to eliminate the phase ambiguity of a multiple of π radians at the carrier synchronization block. The frame bits are encoded using a Boolean XOR operation bitwise, according to equation 5-13:

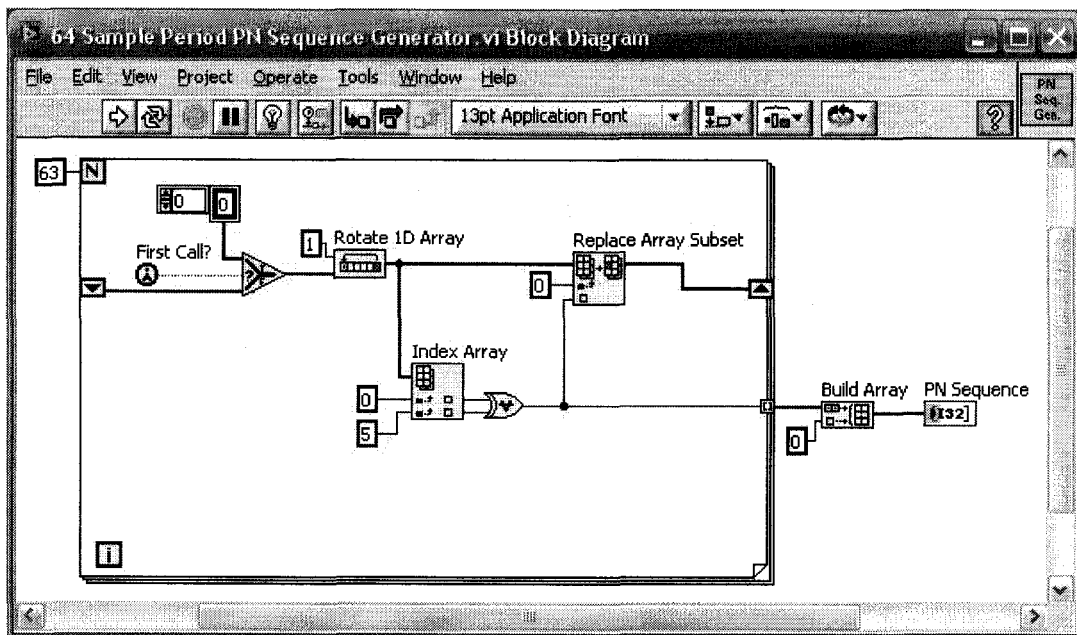
$$y_n = x_n \oplus y_{n-1} \quad (5-13)$$

where y_n is the encoded sequence, x_n is the input sequence, and n is the sequence index.

The subVI Differential Encoder.vi implements this operation. The encoded frame bits can be observed from the Frame Bits tab in the front panel, choosing the proper radio button in the tab. Figure 5-7 shows the code implementation of the Differential Encoder.vi.



(a) Front panel



(b) Block diagram

Figure 5-5 – 64 Sample Period PN Sequence Generator.vi

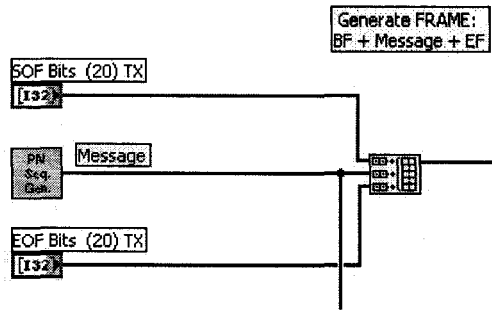
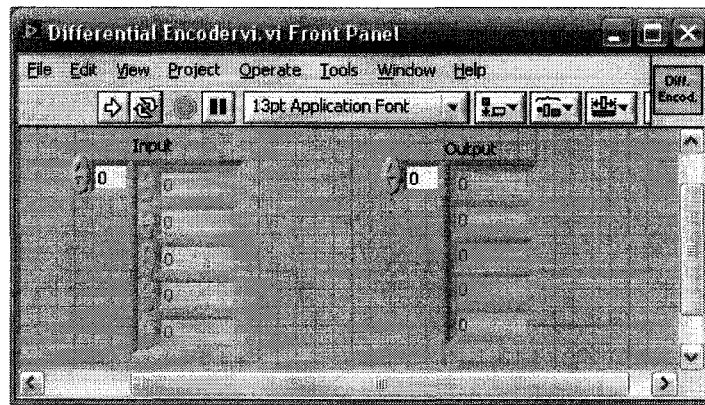
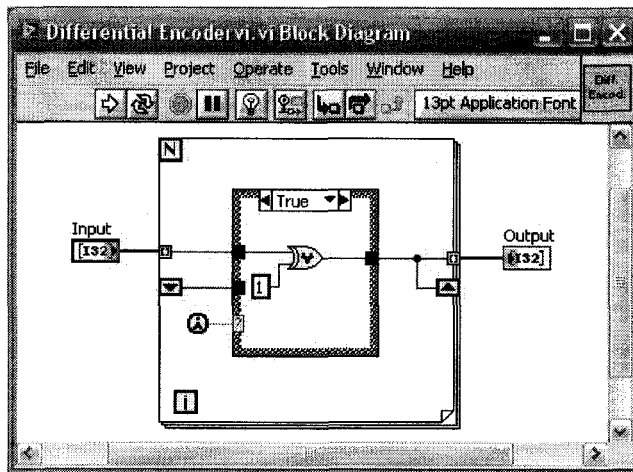


Figure 5-6 – Frame generation



(a) Control panel



(b) Block diagram

Figure 5-7 – Differential Encoder.vi

5.2.3 – Bits to Symbols Mapping

The frame bits are mapped into the in-phase (I) and quadrature-phase (Q) components to form the symbols by the 4 QAM Bits to Symbol Mapping.vi. Each symbol is represented by a complex value. Later, the real and imaginary parts of the symbols are separated into the I and Q channels. In this 4 QAM scheme, each pair of bits is mapped into a point in the constellation diagram according to the figure 5-8. These points can be observed from the Constellation Diagram chart in the front panel of the 4 QAM SDR Transceiver.vi. Figure 5-9 shows the code implementation of the 4 QAM Bits to Symbol Mapping.vi.

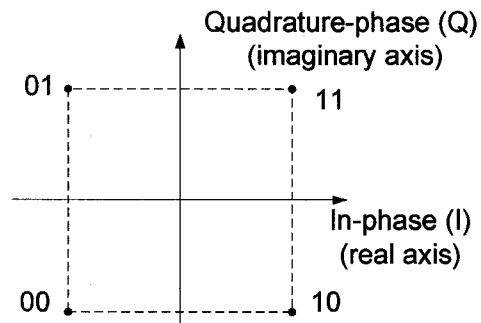
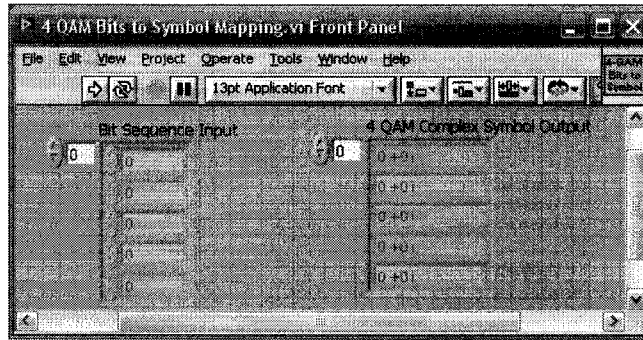
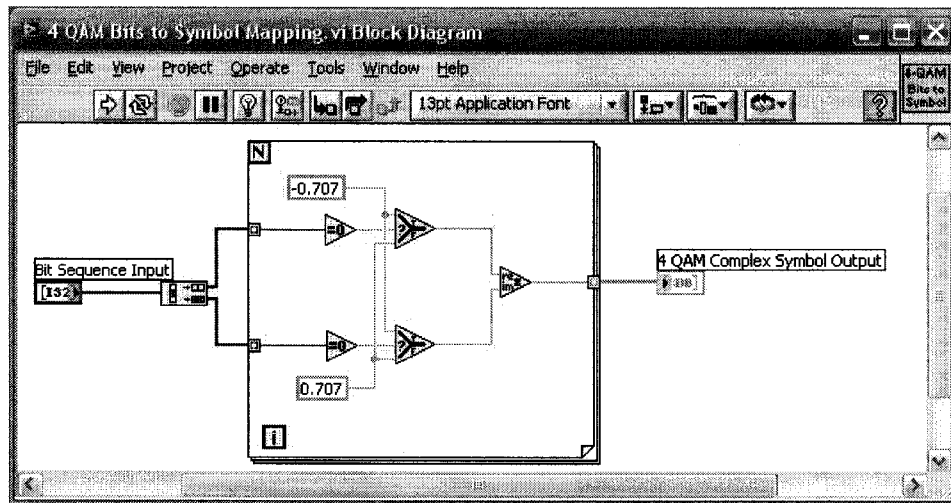


Figure 5-8 – 4 QAM constellation diagram



(a) Front panel



(b) Block diagram

Figure 5-9 – 4 QAM Bits to Symbol Mapping.vi

5.2.4 – Pulse Shaping

During the transmission process, the digital message needs to be converted into an analog signal. The pulse shaping filter converts each digital symbol into a corresponding analog pulse by up-sampling the original signal into a shape that helps to overcome the problem of intersymbol interference (ISI), limiting the signal bandwidth, and improves the signal to noise ratio (SNR) at the receiver. Different types of filters can be applied, where each filter type will

determine specific characteristics of the transmitted signal in the time and frequency domains.

5.2.4.1 – How the Pulse Shaping Filter Works

Consider an analog signal $w_a(t)$ passing through a filter with impulse response $p(t)$. In the time domain, the output of the filter is given by the convolution.

$$x(t) = w_a(t) * p(t) \quad (5-14)$$

Considering $W_a(f)$ and $P(f)$ as the Fourier transform of $w_a(t)$ and $p(t)$, respectively, the Fourier transform of the output signal is given by the product

$$X(f) = W_a(f) P(f) \quad (5-15)$$

By looking at this last equation (5-15), it can be observed that the filter response scales the output signal and limits its bandwidth ($X(f)$ is zero at frequencies where $P(f)$ is zero).

When choosing a particular pulse shaping, one must consider the tradeoff between the advantages and disadvantages in the time and frequency domains. For example, a rectangular pulse in the time domain is the best form of sampling the signal. However, in the frequency domain, the corresponding frequency response is a sinc function, which causes infinite bandwidth. On the other hand, the sinc pulse in the time domain has a corresponding rectangular form in the

frequency spectrum, which limits the bandwidth of the incoming signal, but is not physically realizable. Figure 5-10 shows some pulse shape examples in time and frequency domains.

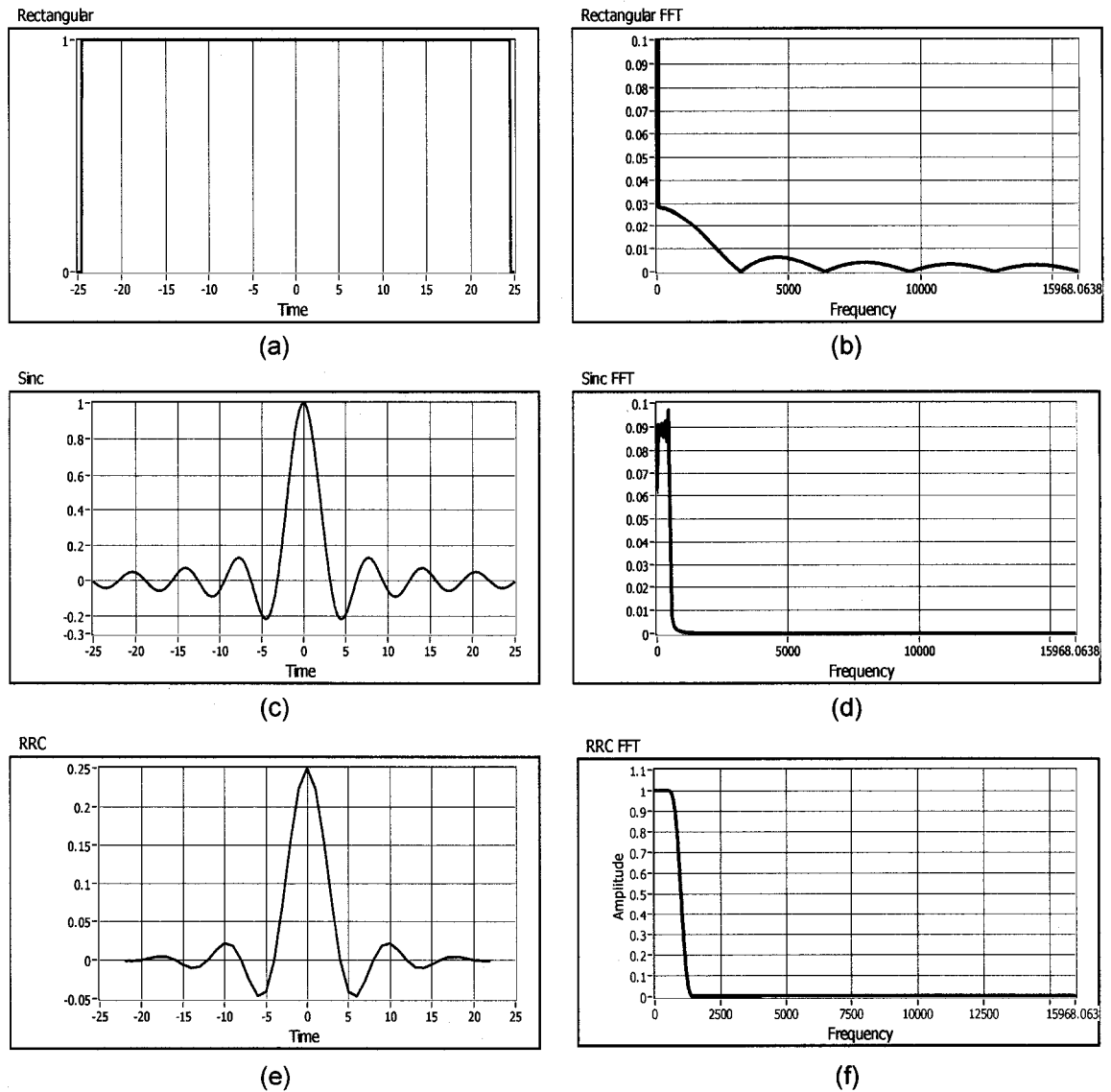


Figure 5-10 – Some pulse shapes in the time and frequency domains

5.2.4.2 – Intersymbol Interference

Intersymbol interference (ISI) is a phenomenon that occurs when adjacent symbols interfere with each other. It can occur when the pulse shape is wider than a symbol interval and when the channel causes distortions on neighboring pulses. Pulse shapes that are wider in time occupy narrower bandwidth.

However, as the pulse shape becomes wider, more ISI will degrade the signal.

Figure 5-11 portrays this phenomenon.

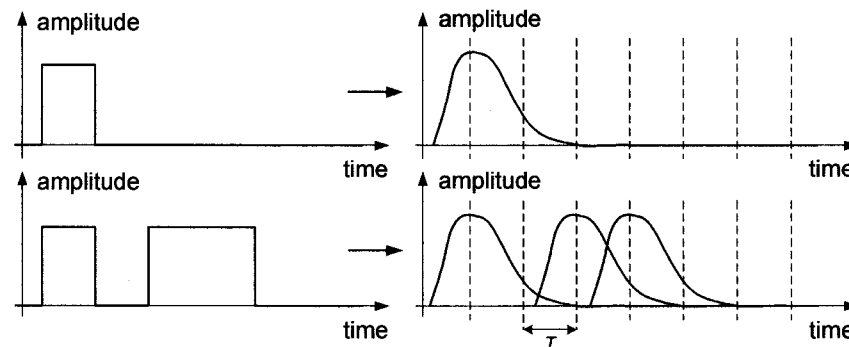


Figure 5-11 – Intersymbol interference (ISI)

A pulse shape with a longer duration in time and satisfying the Nyquist condition, which must be zero at all integer multiples of the symbol period T but one, provides a narrow bandwidth and avoids ISI. A pulse $h(t)$ is said to be a Nyquist pulse if it obeys the following equation:

$$h(kT + \tau) = \begin{cases} c & k = 0 \\ 0 & k \neq 0 \end{cases} \quad (5-16)$$

for some τ , where k is an integer and c is a constant. Raised cosine (RC) and root raised cosine (RRC) pulse shapes are commonly used in communication systems for having the properties of zero crossing at desired times, sloped band edges in the frequency domain, and a considerable fast decay in the time domain while preserving a narrow bandwidth.

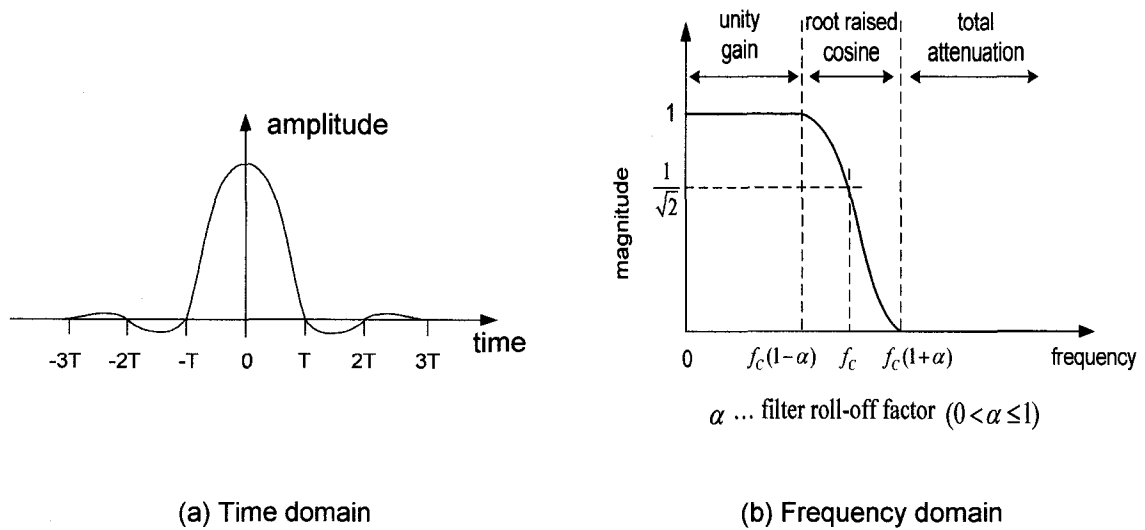
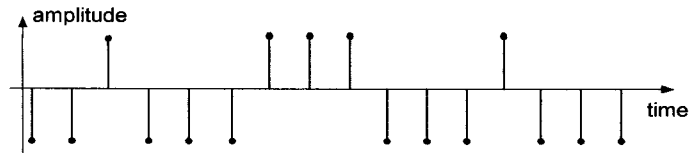


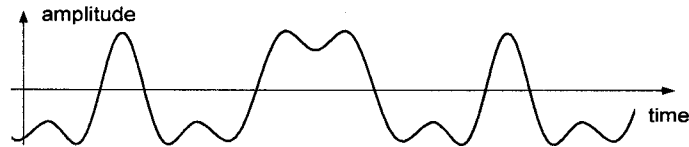
Figure 5-12 – Ideal root raised cosine response

5.2.4.3 – Pulse Shaping and the Eye Diagram

The pulse shaping operation involves the convolution of the incoming signal with the pulse shape. Figure 5-13 provides an example of the root raised cosine pulse shape.



(a) Unshaped signal



(b) Corresponding pulse shaped signal

Figure 5-13 – Pulse shaping example

The eye diagram (figure 5-14) is a visualization tool used for inspecting the pulse-shaped signal. Several traces of the waveform are superimposed in the same picture, starting at integer multiples of the symbol time. Ideally, the original message should be recovered by sampling the pulse-shaped signal exactly in the middle of each symbol. In this diagram, this point is said to be in the center of the eye. As the sample point deviates from the center of the eye, the recovery process becomes more susceptible to errors. Therefore, the eye diagram illustrates the limitation for sampling time errors. Also, if the signal distortion is extremely severe, the diagram will have the eye closed, showing that it is impossible to recover the original message.

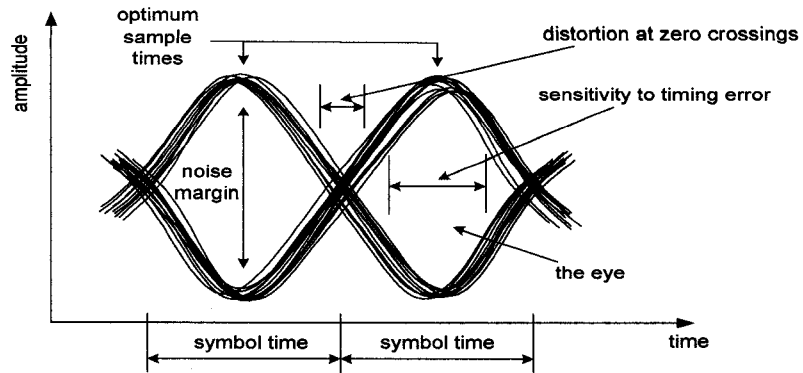


Figure 5-14 – Eye diagram

5.2.4.4 – Pulse Shaping Implementation in the 4 QAM SDR Transceiver

The pulse shape stage is implemented in the 4 QAM SDR Transceiver.vi by the MT Pulse Shaping Filter.vi found in the Modulation toolkit in LabVIEW (Addons → Modulation → Digital → Utilities). Each symbol is up sampled by the value specified in the Samples/Symbol control in the front panel. The default value is 16. The filter coefficients are generated by the Generate Filter Coefficients.vi, also found in the same LabVIEW library.

The type of pulse shaping filter used is defined in the Pulse Shaping Filter control in the front panel of the 4 QAM SDR Transceiver.vi. It is possible to use the raised cosine (RC), the root raised cosine (RRC), or the rectangular (none) pulse shaping filter. The default type is root raised cosine (RRC). The filter parameter alpha (roll-off factor) is defined in the Filter Parameter control. Its default value is 0.5.

This block is also responsible for the desired transmitted symbol rate. It can be set in the Symbol Rate control. The default value is 2000 Hz. The

waveforms of the in-phase and quadrature-phase baseband signals can be observed by clicking on the Pulse Shaping tab and choosing the proper radio button in the tab.

5.2.5 – Conversion to IF

The complex baseband signal needs to be mixed with a complex intermediate frequency (IF) carrier signal before it is sent to the sound card for transmission. Consider the baseband complex signal $m(t)$ and the complex carrier $c(t)$:

$$m(t) = I(t) + jQ(t) \quad (5-17)$$

$$c(t) = \cos(2\pi f_c t + \theta) + j \sin(2\pi f_c t + \theta) \quad (5-18)$$

where $m(t)$ is the output of the pulse shape filter stage with $I(t)$ the in-phase component and $Q(t)$ the quadrature component, f_c is the carrier frequency, and θ is the carrier phase. Their product will be:

$$s_c(t) = m(t) \cdot c(t) \quad (5-19)$$

$$= \begin{aligned} & [I(t) \cos(2\pi f_c t + \theta) - Q(t) \sin(2\pi f_c t + \theta)] \\ & - j [I(t) \sin(2\pi f_c t + \theta) + Q(t) \cos(2\pi f_c t + \theta)] \end{aligned} \quad (5-20)$$

Note that $s(t)$ is the real part of the complex signal $s_c(t)$ from equation 5-1. Even though $s(t)$ is a real number, it represents two signals in quadrature. This signal is then scaled to optimize the sound card dynamic range use and sent to the sound card buffers for transmission.

The value of the carrier frequency can be adjusted in the control panel using the IF control. Its default value is 10 kHz. This block is also responsible for the actual sample rate generated by the code. The symbol rate and the number of samples per symbol determine the sample rate of the modulating carrier according to equation 5-21:

$$F_s = (\text{Symbol Rate}) \cdot (\# \text{ of Samples per Symbol}) \quad (5-21)$$

where F_s is the sample rate in hertz, while the symbol rate is given in symbols per second. The effective sample rate of the output signal is defined by the sound card parameters, as explained next.

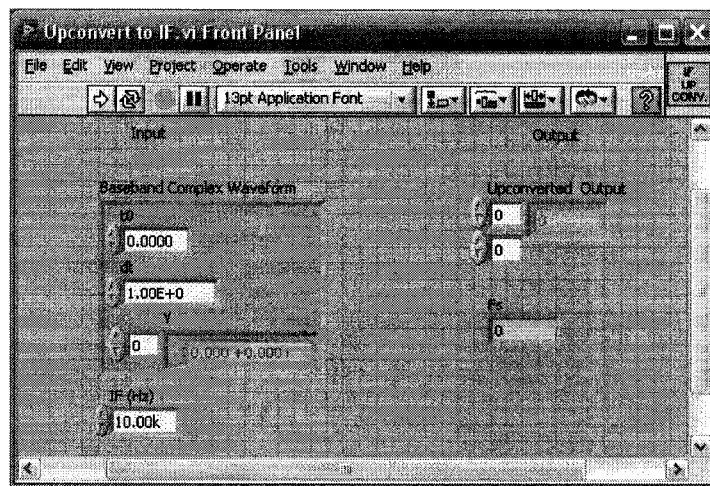
The subVI responsible for the up-conversion to IF stage is the Upconvert to IF.vi. Figure 5-15 shows its front panel and block diagram.

5.2.6 – Sound Card Output

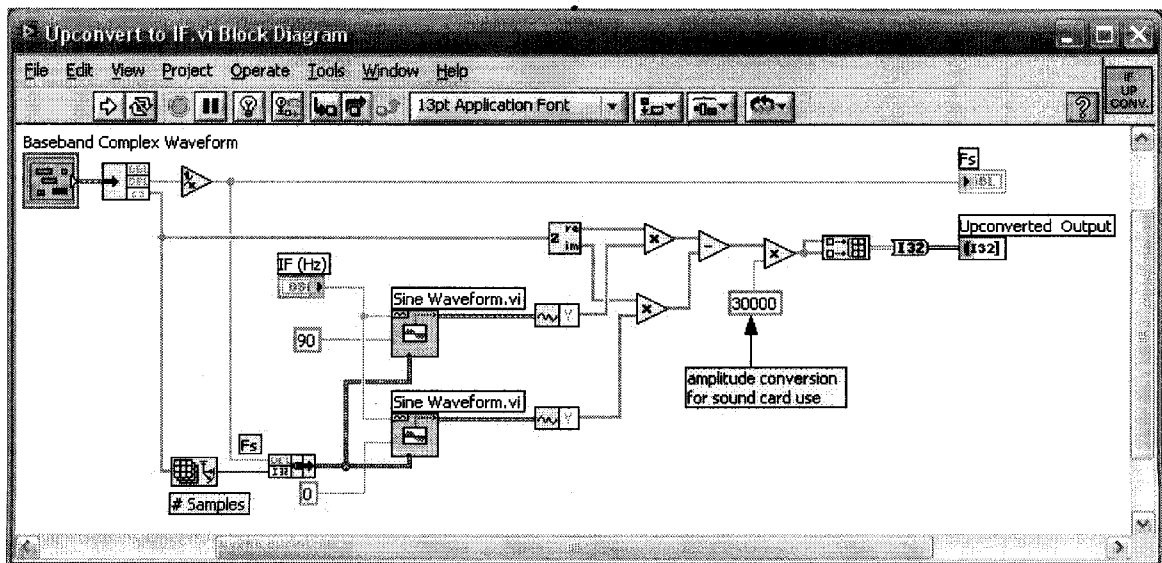
In the final stage of the transmitter, the signal is output by the sound card at a sample rate defined in the Sample Rate control in the front panel. Its default value is 44100 Hz. This sample rate can be set in 1 Hz steps.

Buffers are used to transmit data between the SDR code (in transmitter and receiver modes) and the Windows API (the core set of application programming interfaces). When the time between calls to the Windows API is in excess of the total buffer time per channel, buffer over-runs on the receiver or under-runs on the transmitter occur. In order to avoid this problem, the number of

buffers must be at least two, but a higher number is more reliable. The number of buffers can be set in the # of Buffers control in the control panel. The default value is 10 for transmitter mode. The buffer size is defined as the size of the bit frame structure. The sound card is controlled by the WavelO set of VIs described in Chapter 4. See [12] for more detail about the WavelO set of VIs.



(a) Front panel



(b) Block diagram

Figure 5-15 – Upconvert to IF.vi

The Device ID control in the control panel specifies the output device used by the SDR system (in transmitter and receiver modes). If the computer has more than one sound card device, it is possible to choose which one to use. The default device ID is 0.

5.3 – Receiver Structure

On the receiver side, the modulated waveform is captured by the sound card and placed into its buffers. The signal is passed from the sound card buffers and scaled to better match to the amplitude range assumed by the code. An automatic gain control (AGC) algorithm is then applied to bring the signal to an appropriate amplitude level followed by a high pass filter to clean up low frequency interference like 60 Hz from the power line and others added by the PC's sound card. Carrier synchronization is performed to bring the signal to its baseband form. A corresponding matched filter is applied next to help eliminate intersymbol interference (ISI) and increase the signal to noise ratio (SNR) of the signal. The signal is then passed through the symbol synchronization and decimation stages to recover the symbols. The symbols are mapped back from the I and Q channels into the frame bits. The sequence of bits is differentially decoded to resolve phase ambiguity from the carrier synchronization stage. The frame synchronization stage retrieves the complete frame and the message is recovered by the message retrieving block. Figure 5-16 depicts the receiver structure.

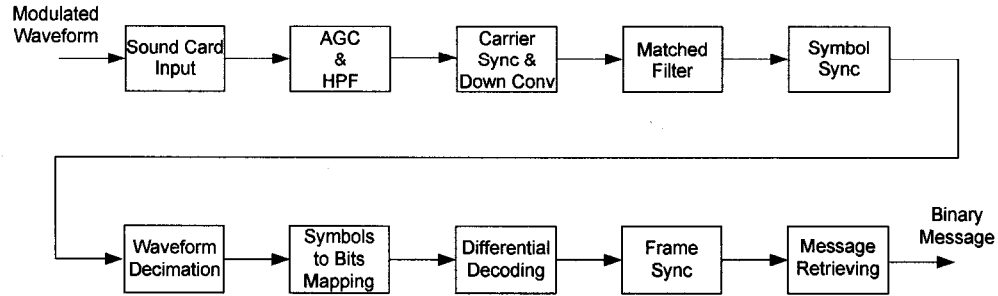


Figure 5-16 – Receiver structure

5.3.1 – Sound Card Input

This stage is responsible for the acquisition of the transmitted signal. As described for the sound card output stage, the WaveIO set of VIs are responsible for this operation. Here, the sound card sample and the number of buffers are defined in the control panel. The default value for the sample rate is also 44100 Hz. The default value for the number of buffers is set at 3000, a much higher value than the one of the transmitter, in an attempt to avoid buffer overflow during the demodulation process. The number of buffers can be increased to guarantee more demodulation time without buffer overflow errors.

Ideally, the sound cards' sample rates should have the same value for the transmitter and the receiver computers. However, small discrepancies are likely to occur and they can cause synchronization problems in the demodulation process, depending on which algorithms are used.

5.3.2 – Automatic Gain Control (AGC) and High Pass Filtering (HPF)

The amplitude of the incoming signal must be adjusted in order to make the demodulation process work properly. Amplitude variations in the received signal can cause erroneous behavior of subsequent processing stages. The automatic gain control (AGC) provides a constant average signal level applied to the demodulation stages. It works by measuring the RMS (root mean square) value of the input signal from the sound card and adjusting the signal to a predefined RMS level, according to equation 5-22.

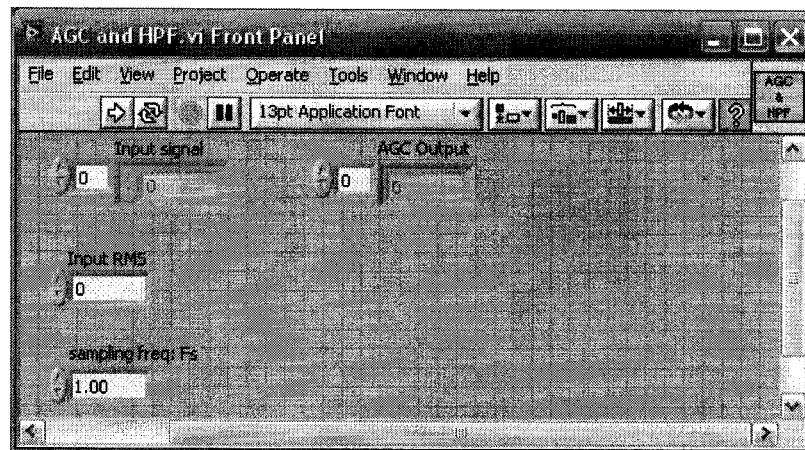
$$output\ signal = \left(\frac{desired\ RMS}{RMS\ of\ input\ signal} \right) \cdot input\ signal \quad (5-22)$$

Also in this stage, a high pass filter (HPF) is applied to clean the incoming signal from the low frequency noise added by the sound cards and other interference sources. The cutoff frequency is set to 2000 Hz. This stage is implemented by the AGC and HPF.vi. Figure 5-17 shows its front panel and block diagram.

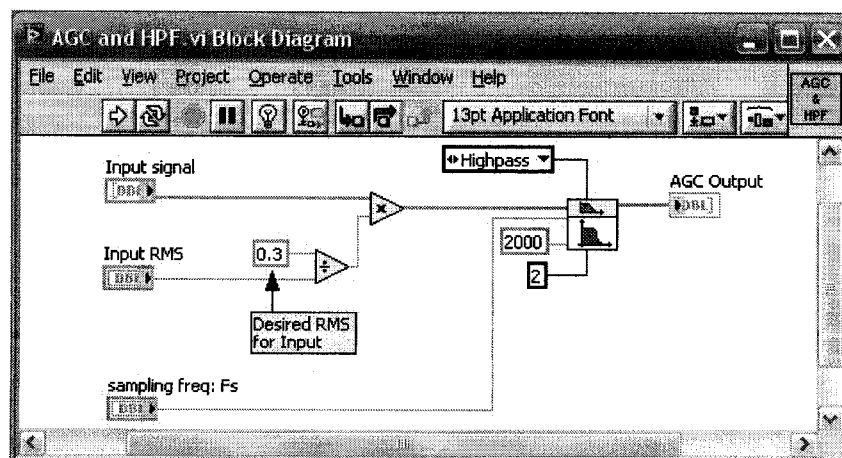
5.3.3 – Carrier Synchronization and Down-Conversion

The incoming signal needs to be down-converted back to its baseband form in order to recover the transmitted message, as explained in section 5.1. The down conversion to baseband process is done by mixing the incoming signal with a sinusoid with the same phase and frequency as the carrier, and this mixing output passes through a low pass filter with a cutoff frequency value close or

lower to the intermediate frequency (IF) value, depending on the bandwidth of the modulating signal. Figure 5-1 depicts this scheme. Even though the transmitter and the receiver are set to work with the same carrier frequency, slightly frequency offsets are expected to happen. Therefore, the receiver needs to find both frequency and phase of the carrier sinusoid to recover the original signal.



(a) Front panel



(b) Block diagram

Figure 5-17 – AGC and HPF.vi

There are different techniques that can be used in the carrier recovery process, each one with different requirements and performance depending on the modulation scheme. In the 4 QAM SDR Transceiver.vi, the 4 QAM Costas Loop was employed as the carrier recovery algorithm. It is explained next.

5.3.3.1 – The Costas Loop for 4 QAM

The Costas Loop is an adaptation algorithm widely used in communication systems. It operates directly on the received signal and uses the “hill climbing” adaptive method to maximize the cost function J_C , which for the 4 QAM scheme is:

$$J_C = \cos^2(2(\theta - \phi)) \quad (5-23)$$

where θ is the actual phase of the carrier and ϕ is the estimated one. The cost function J_C will have a maximum of one when $\cos[4(\theta - \phi)] = 1$ or $4(\theta - \phi) = 0, 2\pi, 4\pi, \dots$, which is equivalent to

$$\phi = \theta + \rho (\pi/2) \quad (5-24)$$

for any integer ρ . As it can be seen from equation 5-24, finding the maximum of the cost function J_C will lead to find an estimated phase ϕ with the value of the actual carrier frequency and possible phase ambiguity of multiple of $\pi/2$ radians.

The phase estimate algorithm searches for values of ϕ that approach the carrier phase θ (for maximizing J_C) except for the noted phase ambiguity. It starts with some initial value for ϕ and follows a path defined by the gradient of

the cost function J_C evaluated at the current point using an adaptation constant μ (μ has a positive value). The adaptation constant μ determines the convergence of the algorithm described by equation 5-25.

$$\phi[k+1] = \phi[k] + \mu \left. \frac{\partial J_C}{\partial \phi} \right|_{\phi=\phi[k]} \quad (5-25)$$

In order to implement this algorithm we need to determine how to implement the derivative $\frac{\partial J_C}{\partial \phi}$ evaluated at $\phi = \phi[k]$.

With some manipulation, the derivative $\frac{\partial J_C}{\partial \phi}$ can be written as

$$\frac{\partial J_C}{\partial \phi} = 4 \cos[2(\theta - \phi)] \sin[2(\theta - \phi)] \quad (5-26)$$

which lead us to find a signal proportional to the product of the sine and cosine of $2(\theta - \phi)$.

In order to find how to generate such a signal, we define the four signals

$$x_1(t) = LPF\{s(t) \cos(2\pi f_0 t + \phi)\} \quad (5-27)$$

$$x_2(t) = LPF\{s(t) \cos(2\pi f_0 t + \phi + \pi/4)\} \quad (5-28)$$

$$x_3(t) = LPF\{s(t) \cos(2\pi f_0 t + \phi + \pi/2)\} \quad (5-29)$$

$$x_4(t) = LPF\{s(t) \cos(2\pi f_0 t + \phi + 3\pi/4)\} \quad (5-30)$$

where $s(t)$ is the received IF signal and f_0 is the carrier frequency estimate.

Using the relationships described by equations 5-4 and 5-5, and considering the local generated frequency f_o equal to the carrier frequency f_c and the low pass filter (LPF) with cutoff frequency around f_o , $x_1(t)$ becomes

$$\begin{aligned} x_1(t) &= LPF \{ I(t) \cos(2\pi f_c t + \theta) \cos(2\pi f_o t + \phi) \} \\ &= \frac{1}{2} [I(t) \cos(\theta - \phi) - Q(t) \sin(\theta - \phi)] \end{aligned} \quad (5-31)$$

In a similar way,

$$x_2(t) = \frac{1}{2} \left\{ I(t) \cos \left[\theta - \phi - \left(\frac{\pi}{4} \right) \right] - Q(t) \sin \left[\theta - \phi - \left(\frac{\pi}{4} \right) \right] \right\} \quad (5-32)$$

$$x_3(t) = \frac{1}{2} \left\{ I(t) \cos \left[\theta - \phi - \left(\frac{\pi}{2} \right) \right] - Q(t) \sin \left[\theta - \phi - \left(\frac{\pi}{2} \right) \right] \right\} \quad (5-33)$$

$$x_4(t) = \frac{1}{2} \left\{ I(t) \cos \left[\theta - \phi - \left(\frac{3\pi}{4} \right) \right] - Q(t) \sin \left[\theta - \phi - \left(\frac{3\pi}{4} \right) \right] \right\} \quad (5-34)$$

Using the relationships

$$\sin(x) = \cos(x - \pi/2) \quad (5-33)$$

$$\cos(x) = -\sin(x - \pi/2) \quad (5-34)$$

$$\sin(x - \pi) = -\sin(x) \quad (5-35)$$

$$\cos(x - \pi) = -\cos(x) \quad (5-36)$$

the products $x_1(t) x_3(t)$ and $x_2(t) x_4(t)$ become

$$x_1(t) x_3(t) = \frac{1}{8} \left\{ [I^2(t) - Q^2(t)] \sin[2(\theta - \phi)] + 2 I(t) Q(t) \cos[2(\theta - \phi)] \right\} \quad (5-37)$$

$$x_2(t) x_4(t) = \frac{1}{8} \left\{ -[I^2(t) - Q^2(t)] \cos[2(\theta - \phi)] + 2 I(t) Q(t) \sin[2(\theta - \phi)] \right\} \quad (5-38)$$

and the product $x_1(t) x_2(t) x_3(t) x_4(t)$ will be

$$x_1(t) x_2(t) x_3(t) x_4(t) = \frac{1}{64} \left\{ [-I^2(t) - Q^2(t)]^2 + 4 I^2(t) Q^2(t) \right\} \sin(2(\theta - \phi)) \cos(2(\theta - \phi)) \\ + 2 I(t) Q(t) (I^2(t) - Q^2(t)) (\sin^2(2(\theta - \phi)) - \cos^2(2(\theta - \phi))) \quad (5-39)$$

Using

$$-(I^2(t) - Q^2(t))^2 + 4 I^2(t) Q^2(t) = -I^4(t) - 2 I^2(t) Q^2(t) - Q^4(t) \\ = -(I^2(t) + Q^2(t))^2 \quad (5-40)$$

and

$$\cos(2x) = \cos^2(x) - \sin^2(x) \quad (5-41)$$

the four-term product $x_1(t) x_2(t) x_3(t) x_4(t)$ becomes

$$x_1(t) x_2(t) x_3(t) x_4(t) = [8 I^2(t) Q^2(t) - (I^2(t) + Q^2(t))^2] \sin(2(\theta - \phi)) \cos(2(\theta - \phi)) \\ - 2 I(t) Q(t) (I^2(t) - Q^2(t)) \cos(4(\theta - \phi)) \quad (5-42)$$

It can be seen from equation 5-42 that the product of all four signals has the signal we desire to generate (a signal proportional to the product of the sine and cosine of $2(\theta - \phi)$).

Consider a pulse shape $p(x)$ that is time-limited to one symbol interval T . A pulse shaped signal in one of the quadrature channels ($I(t)$ or $Q(t)$) can be written as

$$I(t) = \sum_k s_I[k] p(t - kT) \quad (5-43)$$

where $s_I[k]$ can assume values of 1 or -1. For the 4 QAM scheme

$$I^2(t) = \sum_k s_I^2[k] p^2(t - kT) = \eta(t) \quad (5-44)$$

where $\eta(t)$ is periodic with period T . Using this relationship, we can write

$$I^2(t) - Q^2(t) = 0 \quad (5-45)$$

$$I^2(t) Q^2(t) = \eta^2(t) \quad (5-46)$$

$$I^2(t) + Q^2(t) = 2\eta(t) \quad (5-47)$$

Using the relationships 5-45, 5-46, and 5-47 in equation 5-42, the four-term product $x_1(t) x_2(t) x_3(t) x_4(t)$ becomes

$$x_1(t) x_2(t) x_3(t) x_4(t) = 4\eta^2(t) \sin[2(\theta - \phi)] \cos[2(\theta - \phi)] \quad (5-48)$$

which is proportional to the product of the sine and cosine of $2(\theta - \phi)$ by a non-negative factor. Thus, we can use the four term product $x_1(t) x_2(t) x_3(t) x_4(t)$ to

implement the derivative $\frac{\partial J_c}{\partial \phi}$.

The 4 QAM Costas Loop algorithm becomes

$$\phi[k+1] = \phi[k] + \mu x_1(t) x_2(t) x_3(t) x_4(t) \Big|_{t=kT_s, \phi=\phi[k]} \quad (5-49)$$

where T_s is the sampling period of the signal. Figure 5-18 shows the block diagram for the 4 QAM Costas Loop algorithm.

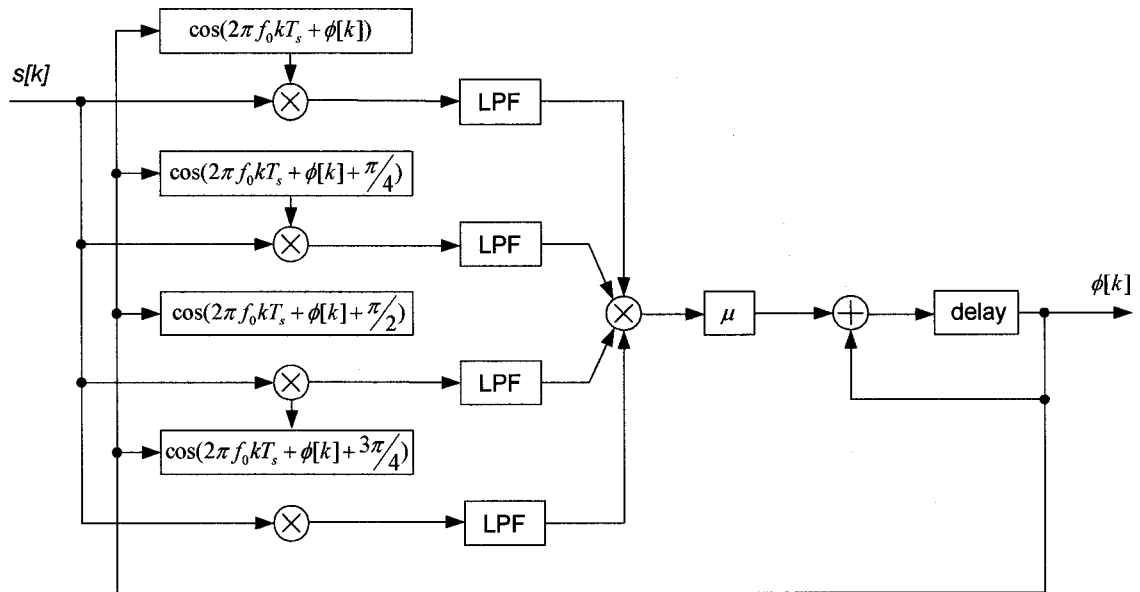


Figure 5-18 – The 4 QAM Costas Loop algorithm

This scheme will provide a phase estimate ϕ that will converge to the phase of the carrier plus an integer multiple of $\pi/2$ radians. In the presence of a frequency offset between the carrier frequency and the expected value at the receiver, the phase update will not converge to a constant value. Instead, it will converge to a line with slope m which represents the frequency offset f_{offset} , as shown in figure 5-19.

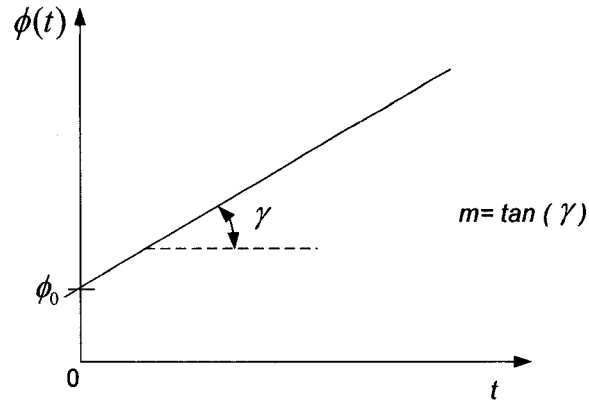


Figure 5-19 – Phase update in the presence of frequency offset

In accordance with figure 5-19, the phase update is given by the following equation:

$$\phi(t) = mt + \phi_0 \quad (5-50)$$

Consider now equation 5-51:

$$\phi(t) = 2\pi f_{\text{offset}} t + \phi_0 \quad (5-51)$$

Comparing these two equations will lead us to

$$f_{\text{offset}} = \frac{\tan(\gamma)}{2\pi} \quad (5-52)$$

5.3.3.2 – Phase Ambiguity Resolution

The phase ambiguity can be resolved by one of the following methods:

- a. correlating the down-converter output with a known training sequence, or

- b. using a differential encoding technique, or
- c. using a trained equalizer.

The 4 QAM SDR Transceiver.vi uses differential encoding and bit pattern error to eliminate the phase ambiguity problem. The in-phase and quadrature-phase channels are encoded separately in the transmitter and decoded separately in the receiver. This takes care of phase ambiguities of π radians. Phase ambiguities of $\pi/2$ and $3\pi/2$ radians are resolved by checking the start and end of bit patterns (SOF and EOF bits). If an error is detected, the phase is changed by a $\pi/2$ radian rotation. See [4] for more detail in 4 QAM down-conversion techniques.

5.3.3.3 – Carrier Synchronization and Down-Conversion Code Implementation

The carrier synchronization and down conversion stage is performed in the 4 QAM SDR Transceiver.vi by the subVI Carrier Synchronization and Downconversion.vi. In this subVI, three tasks are performed: carrier phase and frequency synchronization, down-conversion to baseband, and phase ambiguity resolution. The control panel and block diagram of this VI is shown in figures 5-20, and 5-21 respectively. Figures 5-22, 5-23, and 5-24 show the parts corresponding to the carrier phase and frequency synchronization, down-conversion to baseband, and phase ambiguity resolution, respectively.

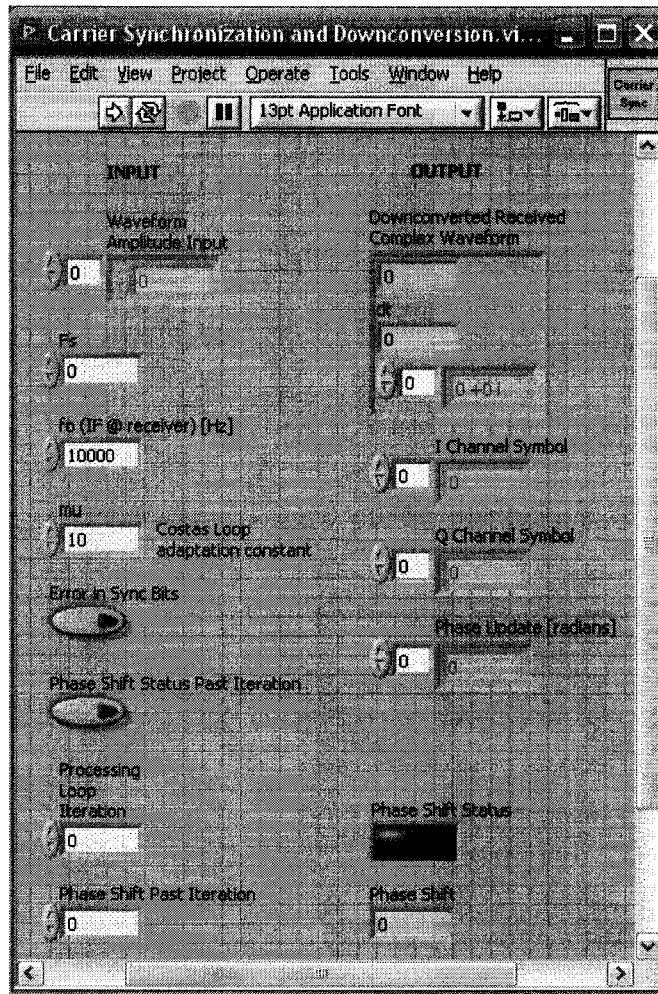


Figure 5-20 – Carrier Synchronization and Downconversion.vi front panel

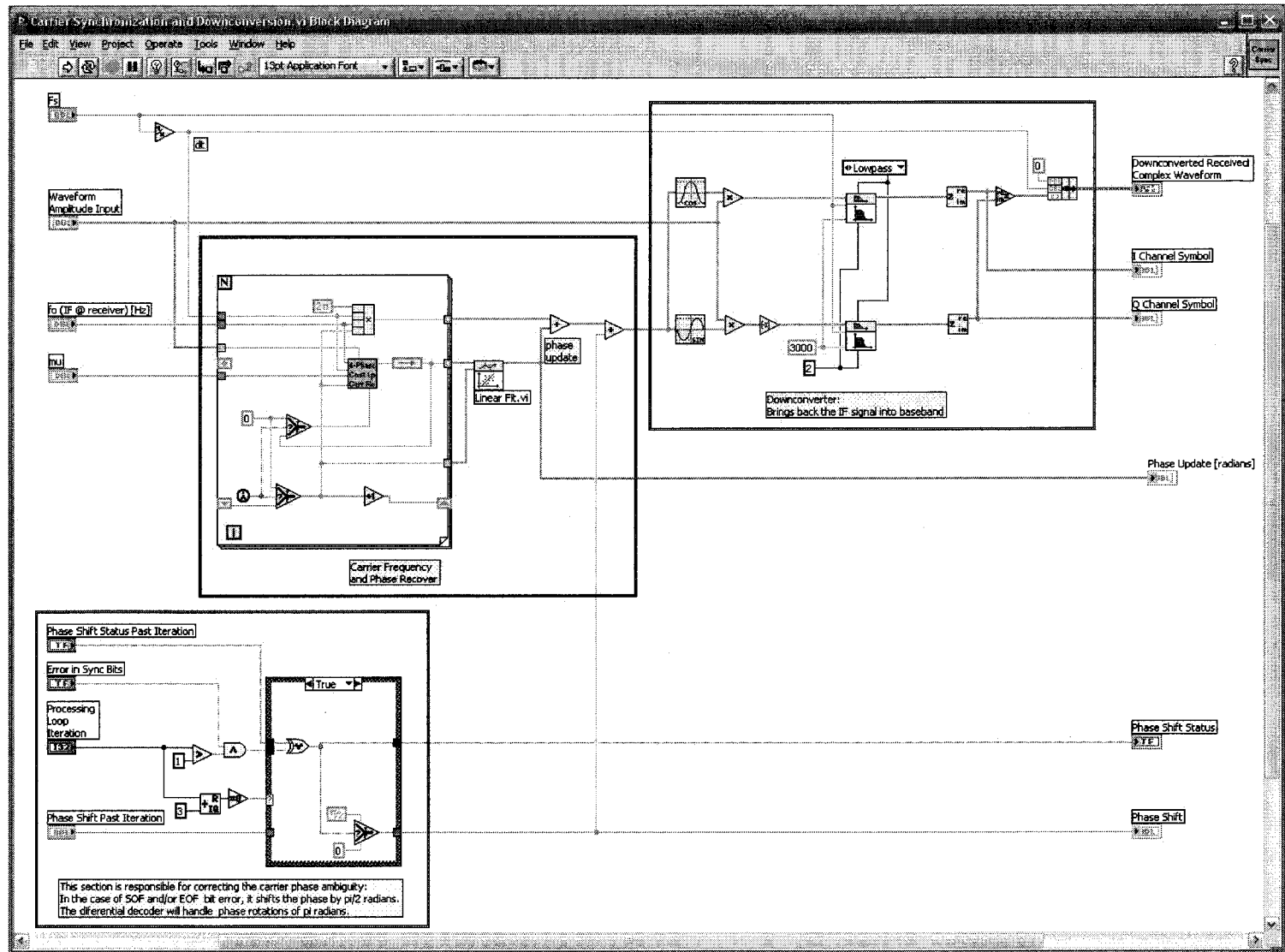


Figure 5-21 – Carrier Synchronization and Downconversion.vi block diagram

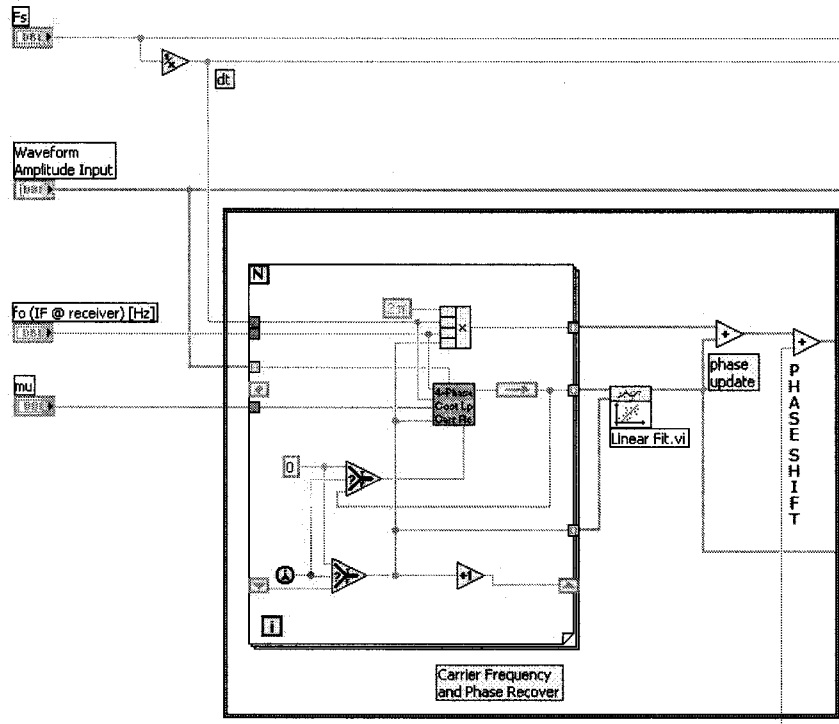


Figure 5-22 – Carrier frequency and phase recovery part of Carrier Synchronization and Downconversion.vi

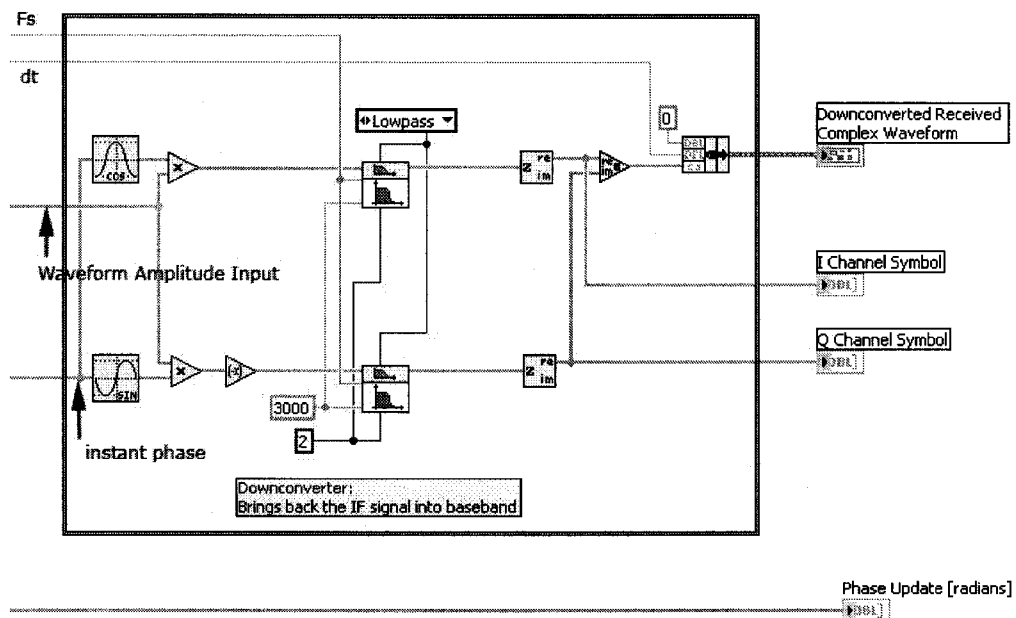
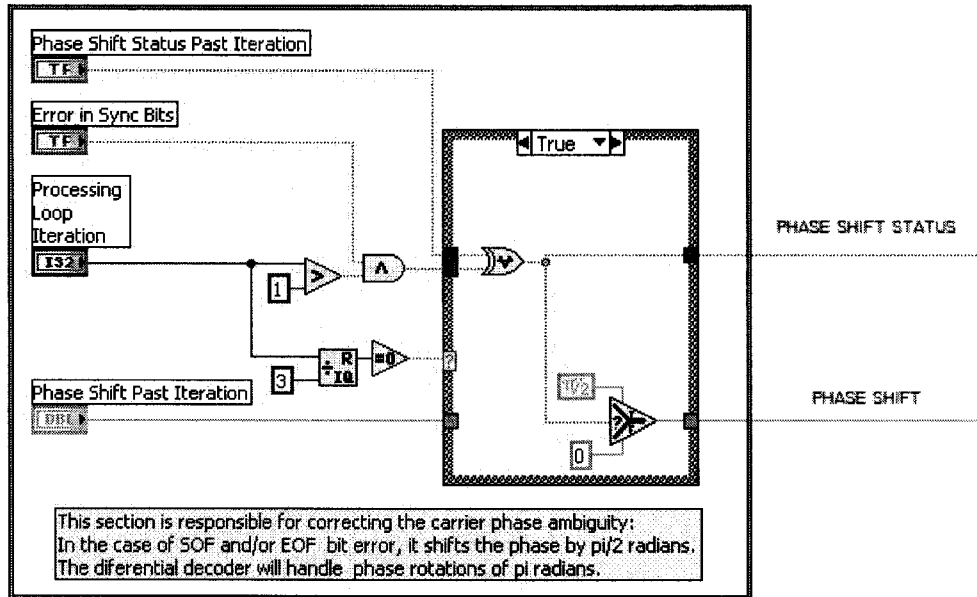
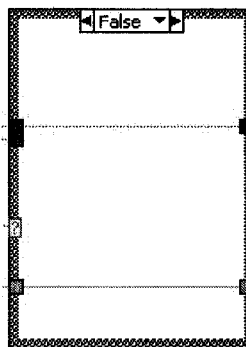


Figure 5-23 – Down-conversion part of Carrier Synchronization and Downconversion.vi



(a) True case

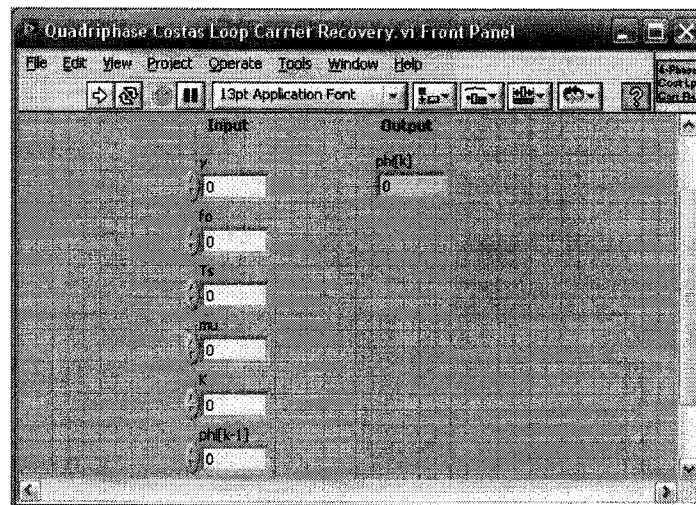


(b) False case

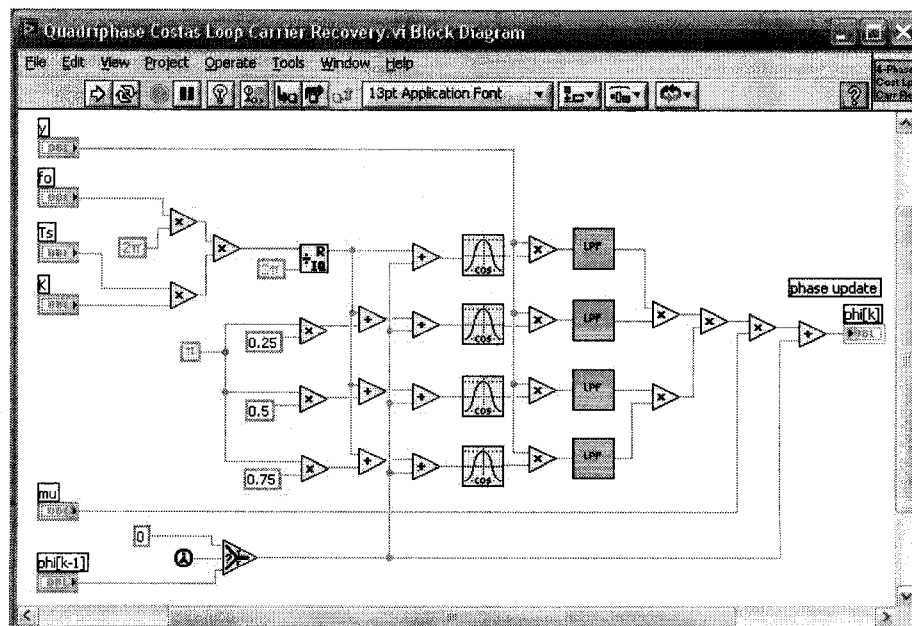
Figure 5-24 – Phase ambiguity resolution part of Carrier Synchronization and Downconversion.vi

The Quadrphase Costas Loop algorithm described before in this section is implemented in the Carrier Synchronization and Downconversion.vi by the subVI called Quadrphase Costas Loop Carrier Recovery.vi. Figure 5-25 shows the front panel and block diagram of this VI. In order to optimize the processing

speed, the low pass filter used by this algorithm was implemented by the subVI 1st Order Butterworth LPF 7k.vi, which is an infinite impulse response (IIR) filter with cutoff frequency of 7 kHz. Figure 5-26 shows the front panel and block diagram of the 1st Order Butterworth LPF 7k.vi.

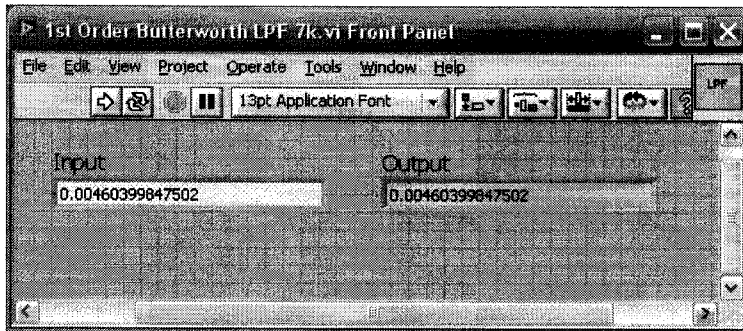


(a) Front panel

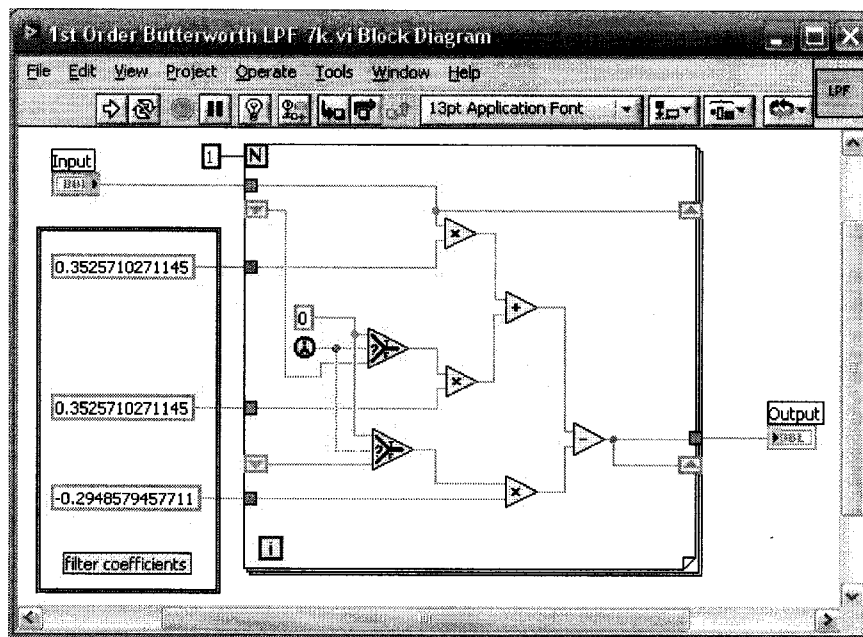


(b) Block diagram

Figure 5-25 – Quadrphase Costas Loop Carrier Recovery.vi



(a) Front panel



(b) Block diagram

Figure 5-26 – 1st Order Butterworth LPF 7k.vi

The Costas loop algorithm uses an adaptation constant, here called μ (mu), the value of which depends on how large the frequency offset is for proper convergence of the algorithm. Larger frequency offsets require larger values of μ to make the Costas loop acquire the right frequency and phase from the incoming signal. The default value of μ is 10. The outputs of the carrier

synchronization and down-conversion stage block are the I (in-phase) and the Q (quadrature-phase) baseband signals, which can be observed from the Baseband Conversion tab in the control panel. The I and Q components can be selected by clicking on the corresponding radio button in that tab. The phase update output of the Costas loop algorithm can be observed by clicking the Phase Update tab in the front panel.

5.3.4 – Matched Filter

After the incoming signal is returned to its baseband form, it needs to be passed through a filter that will reduce the intersymbol interference (ISI) and maximize the signal-to-noise (SNR) ratio. In order to do this, the receive filter should be matched to the pulse shape filter created in the transmitter (its impulse response must be a scaled time reversal of the pulse shape). It is called a matched filter for this reason. This stage is implemented by the MT Matched Filter.vi found in the Modulation Toolkit in LabVIEW (Addons → Modulation → Digital → Utilities). The filter coefficients are again generated by the Generate Filter Coefficients.vi found in the same LabVIEW library.

The outputs of the matched filter stage can be observed by clicking on the Matched Filter tab in the front panel. The I and Q components can be selected by clicking on the corresponding radio button in that tab.

5.3.5 – Symbol Synchronization

This block is responsible for finding the first ideal symbol time instant from the matched filter output. Its output is a symbol time aligned complex waveform, where the first sample corresponds to the optimal symbol instant. This is done to prepare the waveform for the decimation process, where one sample of each symbol must be extracted at an optimal time. Figure 5-27 depicts an up-sampled waveform and the ideal time instant samples for symbol synchronization.

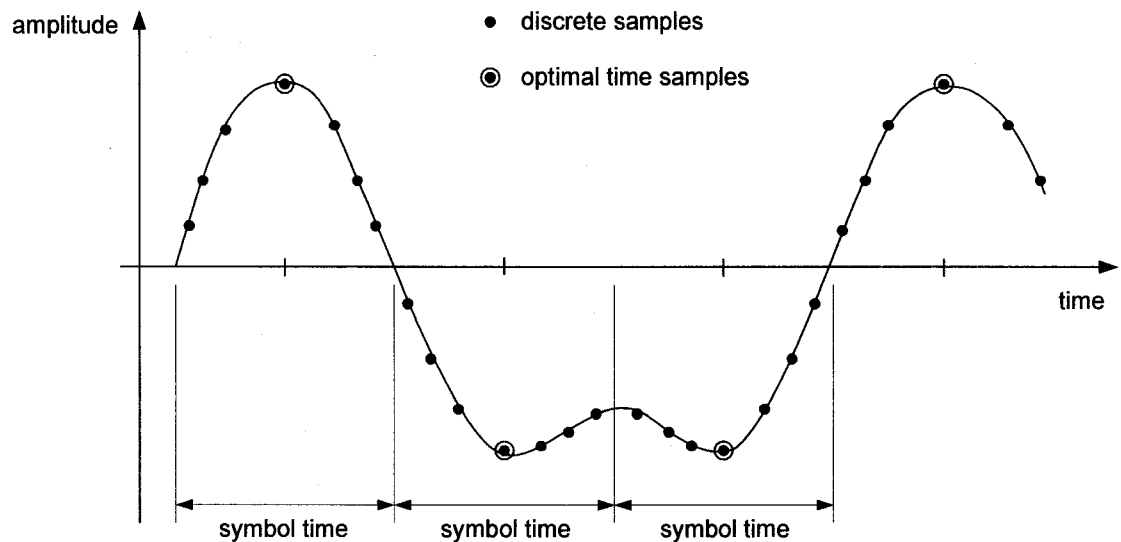


Figure 5-27 – Symbol synchronization

There are different techniques that can be applied for symbol synchronization. The Delay-Locked Loop (DLL) algorithm, also called early-late sampling, is the technique used in the 4-QAM SDR Transceiver.vi.

5.3.5.1 – The Delay-Locked Loop (DLL) Algorithm

The delay-locked loop (DLL) is a simple technique used to recover the symbol timing. In quadrature modulation, the DLL algorithm works on one of the two components of the quadrature signal. For instance, it takes the in-phase component and tries to find the samples located at the peaks to ensure better performance when noise is present. The sample obtained at a peak of the incoming signal is said to be an on-time sample. If the sample is not at the peak, then it is said to be a too early or too late sample. Figure 5-28 depicts the three possible cases in this scheme, while figure 5-29 shows the DLL block diagram.

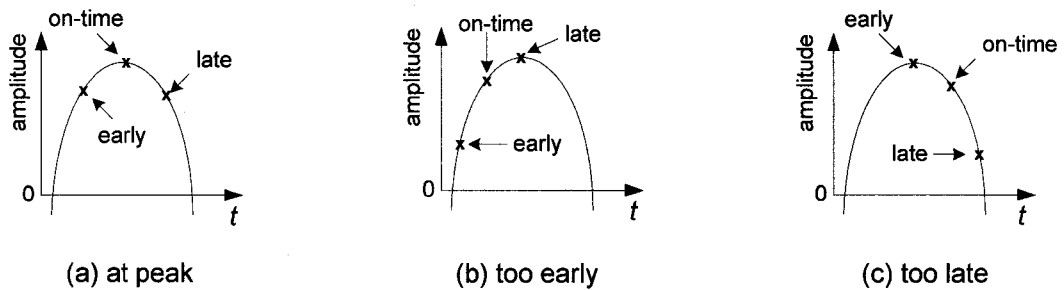


Figure 5-28 – DLL sampling

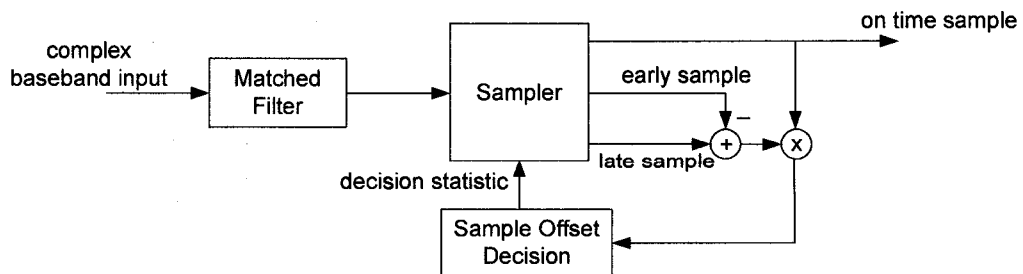


Figure 5-29 – DLL block diagram

This algorithm takes three sequential samples and compares their values to verify which one has the larger value in magnitude by using the decision statistics input at the sample offset decision block. Since the number of samples per symbol N is known by the decision block, it adjusts the next on-time sample index according to the decision statistics input:

- a. If the on-time sample is at the peak, the next on-time sample will be at the next N sample index.
- b. If the on-time sample is too early, the next on-time sample will be at the next $N+1$ sample index.
- c. If the on-time sample is too late, the next on-time sample will be at the next $N-1$ sample index.

In the presence of noise, instead of using only the decision statistics as the parameter for updating the on-time sample, a more reliable approach is taken. For each DLL iteration, the offset value from the decision statistics input is added to the previous one. The next on-time sample is taken every other N sample index until the sum exceeds a threshold value. After that, the next on-time sample takes into account the value of the sum: if it is negative, the next on-time sample is at the next $N-1$ sample index; otherwise (if the sum is positive), the next on-time sample is taken at the next $N+1$ sample index.

The subVI Symbol Timing Recovery.vi is responsible for the symbol synchronization stage in the 4 QAM SDR Transceiver.vi. It is embedded into the subVI called Symbol Timing and Decimation.vi. Figures 5-30 and 5-31 show its block diagram, while figure 5-32 shows its front panel.

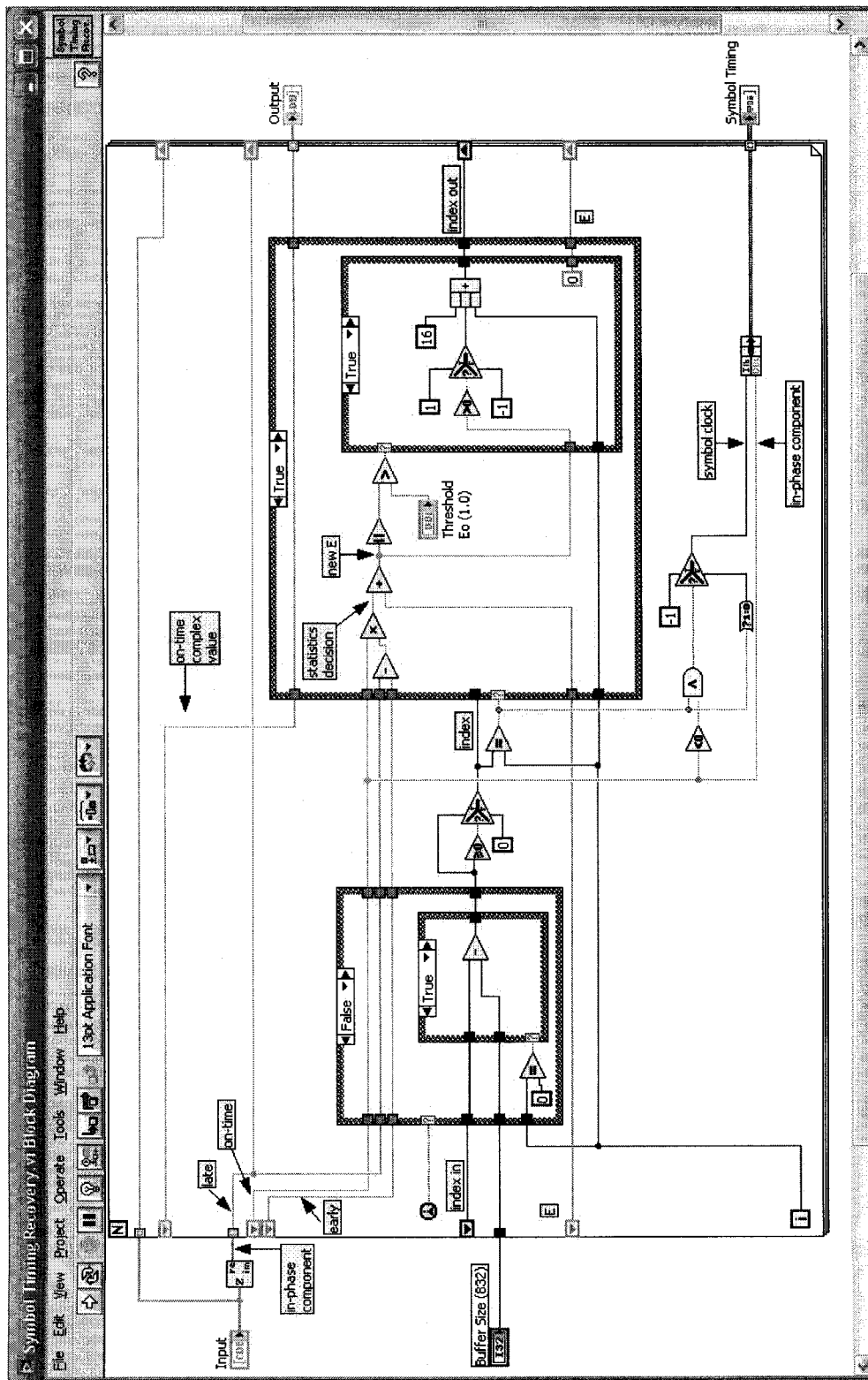


Figure 5-30 -- Symbol Timing Recovery.vi block diagram

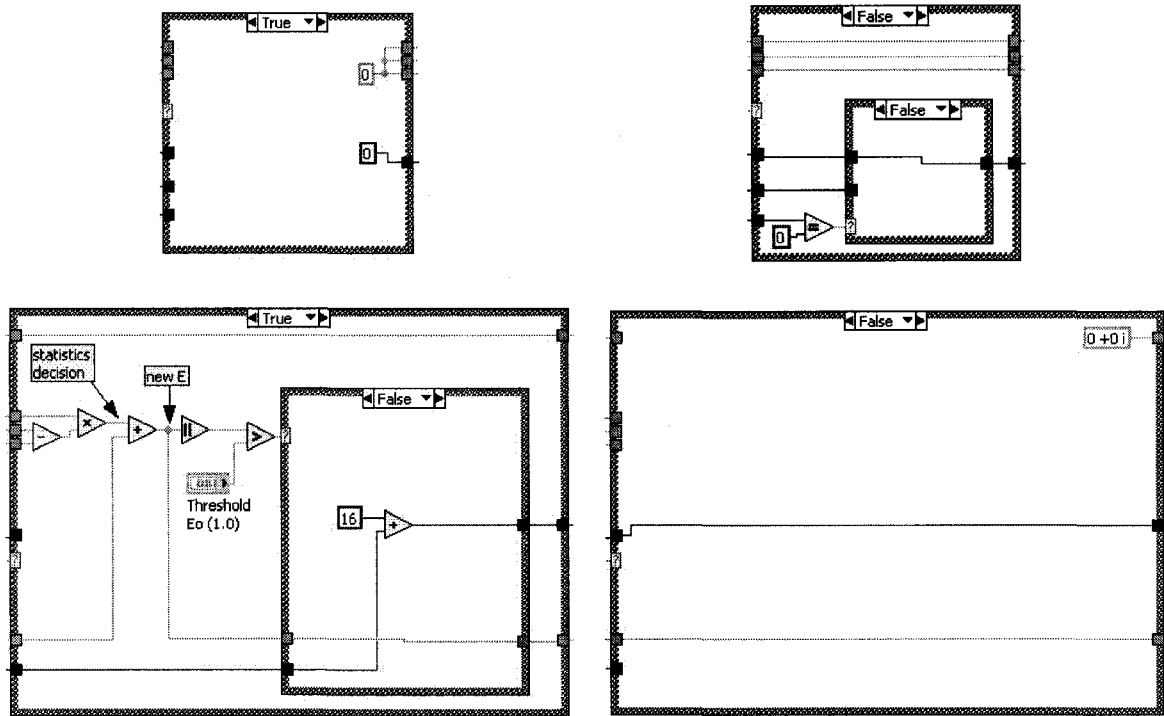


Figure 5-31 – Symbol Timing Recovery.vi block diagram: detail of case structures

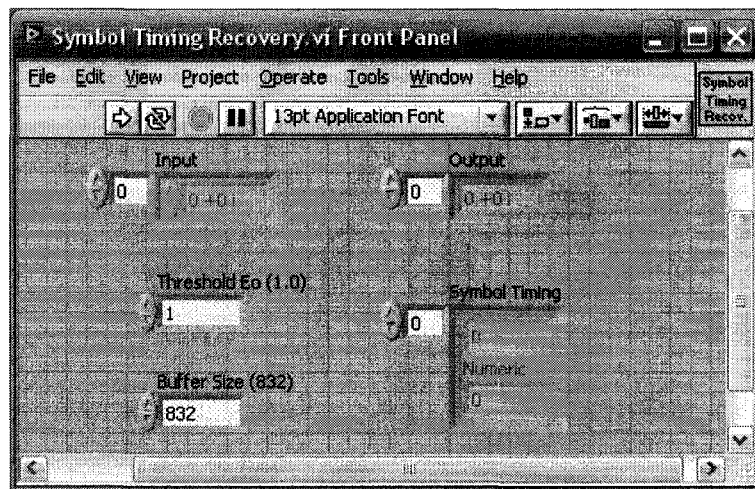


Figure 5-32 – Symbol Timing Recovery.vi front panel

The inputs of the Symbol Timing Recovery.vi are the complex array containing the quadrature symbol values from matched filter stage, the threshold E_o , and the buffer size. The threshold E_o contains the successive decision statistics sum value needed to correct symbol timing in the DLL algorithm. If this sum is larger than, or equal to the threshold E_o , the algorithm corrects the symbol clock. Its default value is 1.0. The buffer size is the size of the sound card buffer. Its default value is 832.

The outputs of the Symbol Timing Recovery.vi are the complex array containing the quadrature symbol values corresponding to optimal sampling times and zero values for other sampling times, and the recovered symbol clock signal bundled with the in-phase component of the input.

The symbol timing recovery process can be observed in the 4 QAM SDR Transceiver.vi by clicking on the Symbol Timing tab in the front panel, which displays the in-phase signal and the recovered symbol clock.

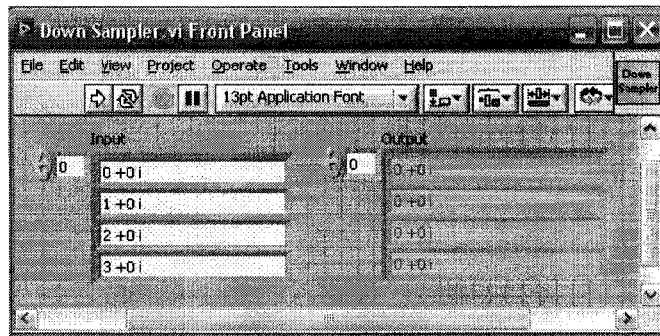
5.3.6 – Waveform Decimation

The waveform decimation stage down-samples the incoming waveform by the same amount it was up-sampled during the modulation process in the transmitter, after the symbol timing recovery stage.

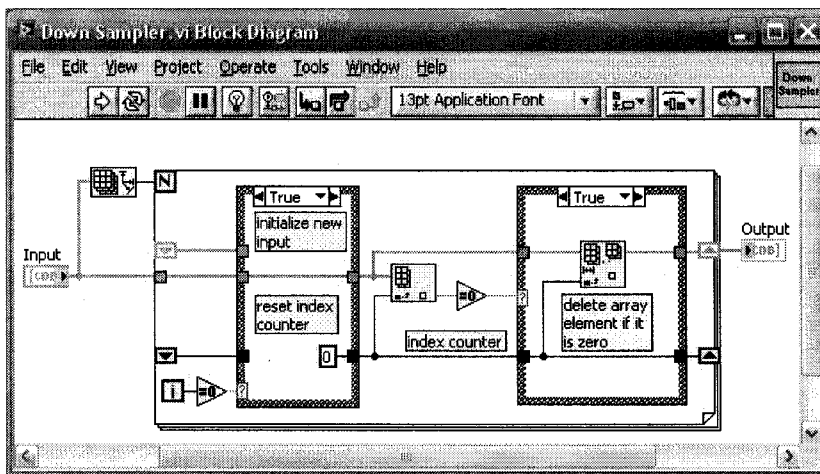
During the decimation process, the waveform is sampled at integer multiples of the symbol time. The symbol timing recovery stage is responsible for acquiring the proper symbol clock, which will determine the proper times to sample the incoming waveform. The samples are taken from the center of eye in

the eye diagram, as explained in section 5.2.4. The center of the eye is considered the optimum time to sample the signal because of its lower susceptibility to symbol errors (a signal showing a lot of smearing in its eye diagram is more susceptible to symbol errors). The eye diagram of the received waveform components can be observed from the Eye Diagram tab and choosing the corresponding component (I or Q). The eye diagram is implemented by the MT Format Eye Diagram (complex).vi found in the LabVIEW library (Addons → Modulation → Digital → Visualization).

The subVI Down Sampler.vi is responsible for the waveform decimation stage. It is embedded in the subVI Symbol Timing and Decimation.vi. Figure 5-33 shows its front panel and block diagram. The decimated I and Q components can be observed from the decimation tabs by clicking on the corresponding radio button.



(a) Front panel



(b) Block diagram

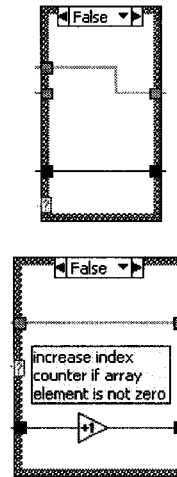
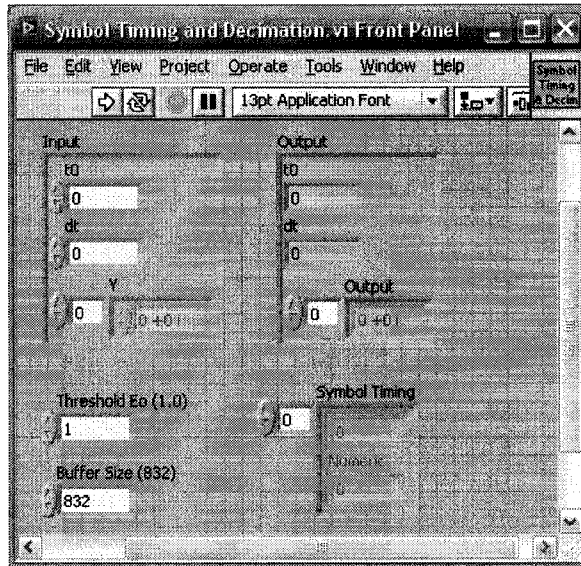


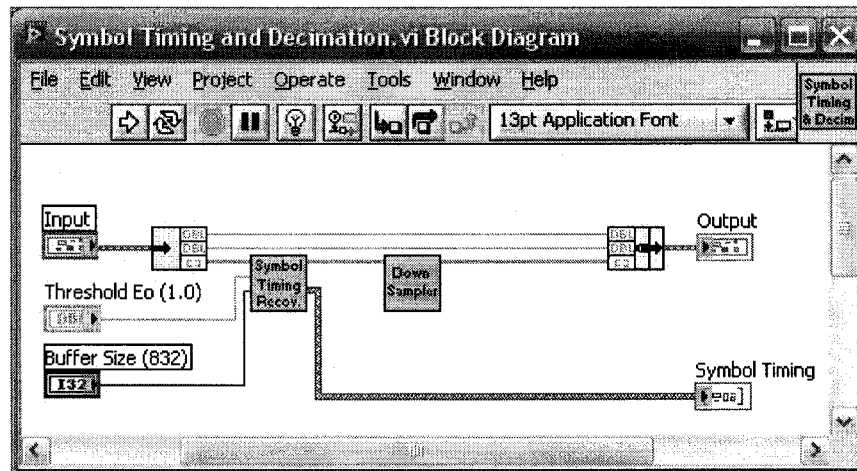
Figure 5-33 – Down Sampler.vi

The subVIs Symbol Timing Recovery.vi and Down Sampler.vi are encapsulated into the subVI called Symbol Timing and Decimation.vi. Figure 5-34 shows the front panel and the block diagram of this subVI.

The constellation diagram can be observed in the Constellation Diagram tab in the front panel of the 4 QAM SDR Transceiver.vi. This graph displays the received 4 QAM symbols in the complex I/Q plane after the decimation process.



(a) Front panel



(b) Block diagram

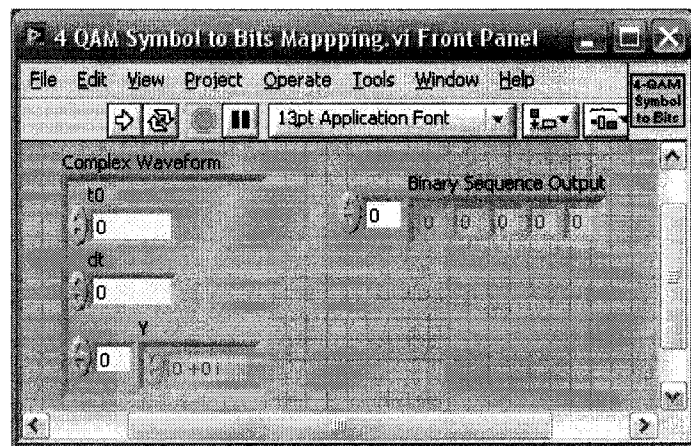
Figure 5-34 – Symbol Timing and Decimation.vi

5.3.7 – Symbol to Bits Mapping

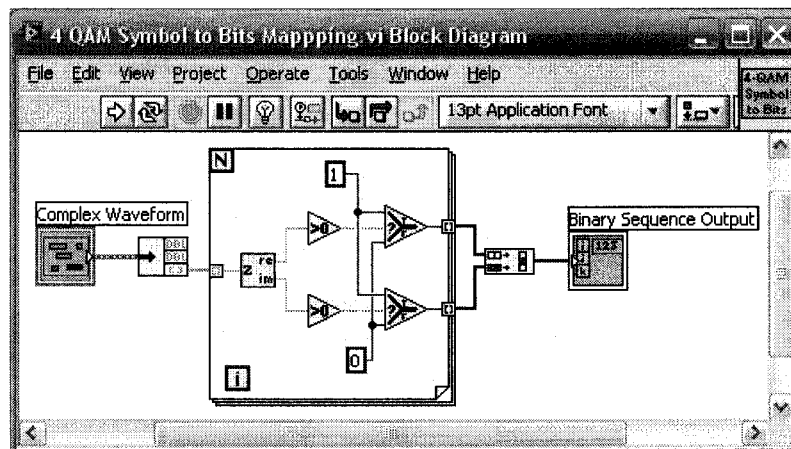
This stage is responsible for recovering the transmitted frame bits from the in-phase (I) and quadrature-phase (Q) waveform components of the decimated

waveform. Each symbol is converted back to the corresponding pair of bits according to the mapping scheme used in the transmitter (see figure 5-8).

The subVI 4 QAM Symbol to Bits.vi is responsible for this stage. Figure 5-35 shows its front panel and block diagram.



(a) Front panel



(b) Block diagram

Figure 5-35 – 4 QAM Symbol to Bits.vi

5.3.8 – Differential Decoding

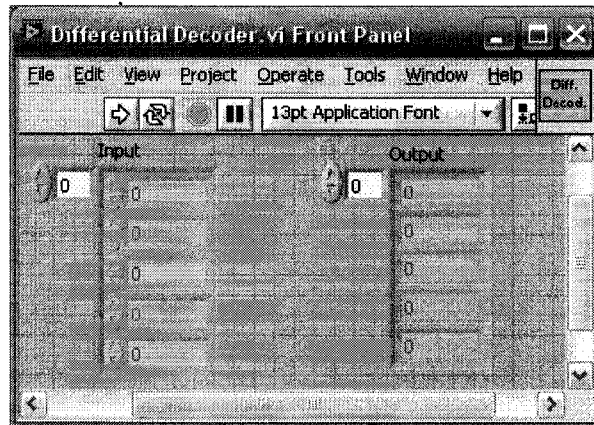
Differential encoding is applied to the original frame bits to help the receiver resolve the phase ambiguity problem caused by the carrier synchronization algorithm. Differential decoding is applied in the receiver to recover the frame bits. The received sequence bits from the symbol to bits mapping stage is decoded using a Boolean XOR operation bitwise, according to equation 5-53:

$$x_n = y_n \oplus y_{n-1} \quad (5-53)$$

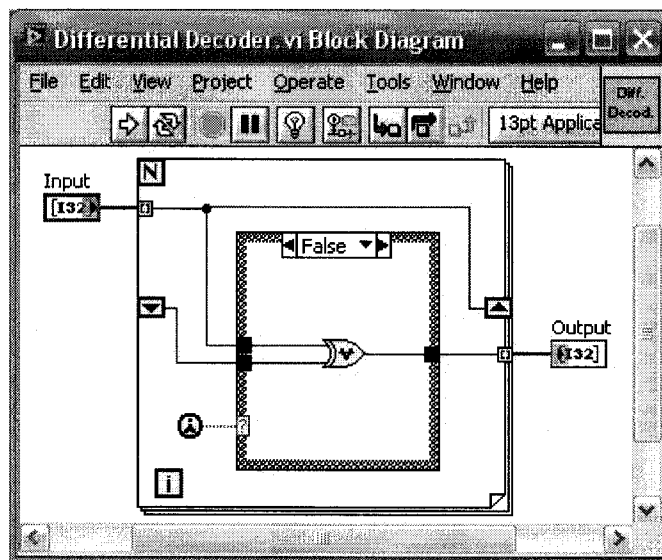
where y_n is the encoded received bit sequence, x_n is the output bit sequence (frame bits), and n is the bit sequence index.

The subVI Differential Decoder.vi is used for this decoding operation.

Figure 5-36 shows its front panel and block diagram.



(a) Front panel



(b) Block diagram

Figure 5-36 – Differential Decoder.vi

5.3.9 – Frame Synchronization and Alignment

When the modulated signal arrives at the receiver, the corresponding start of frame (SOF) bits can be anywhere inside the receive buffer. Frames are recovered by identifying the start-of-frame (SOF) bits inside the received bit sequence and manipulating consecutive bit sequences. Figure 5-37 provides an

example of how this process is done when the sound card buffer has the same size as the frame structure. The SOF bit structure is first identified by correlating the incoming bit sequence with the original SOF bit pattern in the receiver. After the SOF bit pattern identification, the bit sequence corresponding to the current buffer is divided in two parts: one starting with the SOF bits and the remaining part of the sequence. During the current process iteration, the first part of the sequence is saved in order to be concatenated with the remaining sequence part in the next process iteration. The SOF bit pattern correlation may have a problem when the SOF bits are divided between buffers. In this case, the end-of-frame EOF bit pattern is used in the correlation process in a similar way.

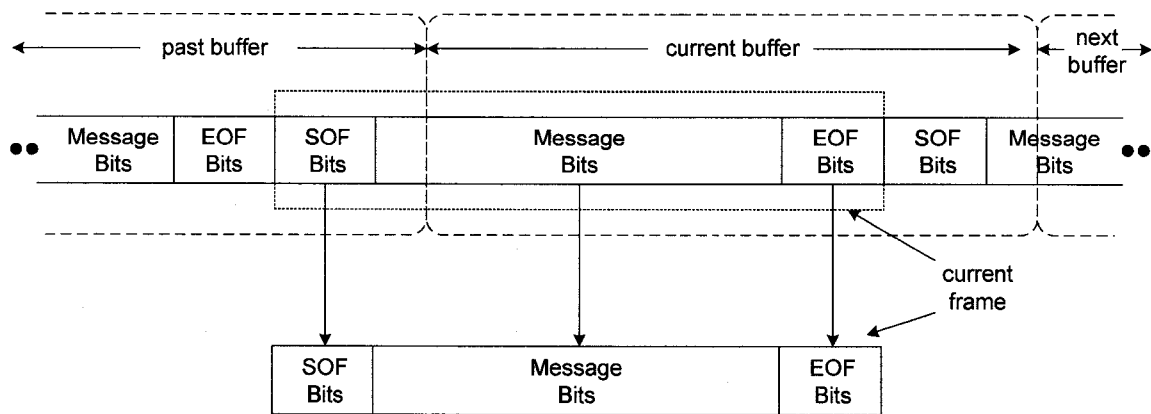


Figure 5-37 – Frame synchronization

The subVI Frame Sync and Alignment.vi implements this stage. Figures 5-38 and 5-39 show its block diagram, while figure 5-40 shows its front panel.

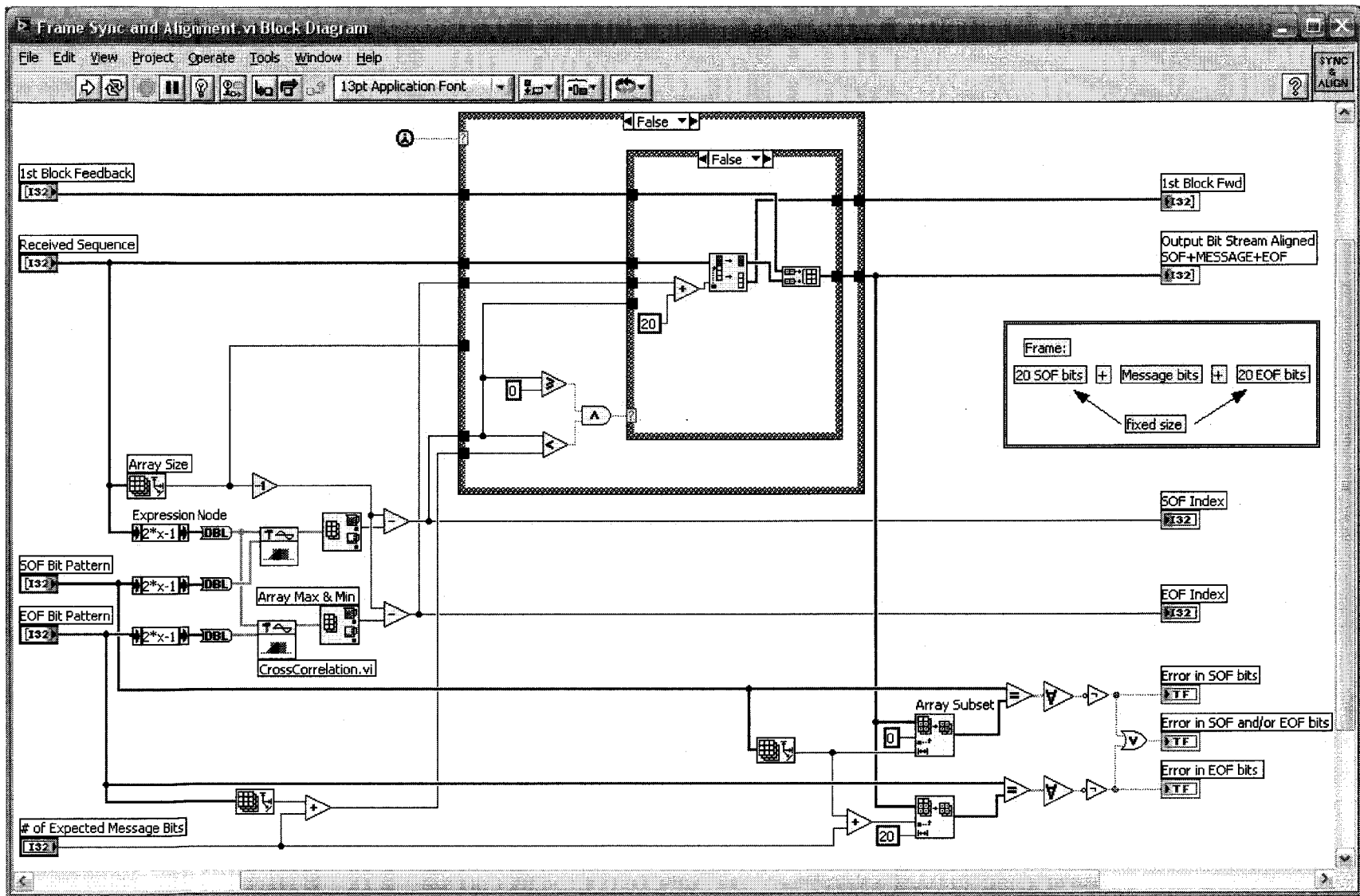


Figure 5-38– Frame Sync and Alignment.vi block diagram

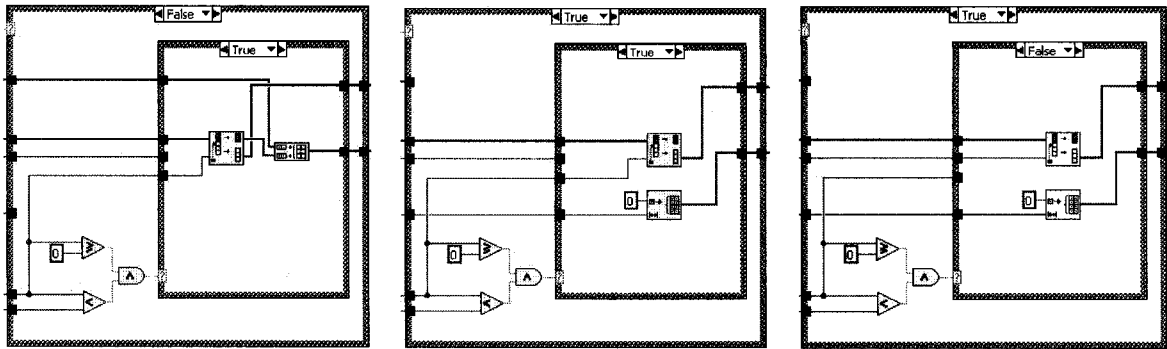


Figure 5-39 – Frame Sync and Alignment.vi block diagram: case structures details

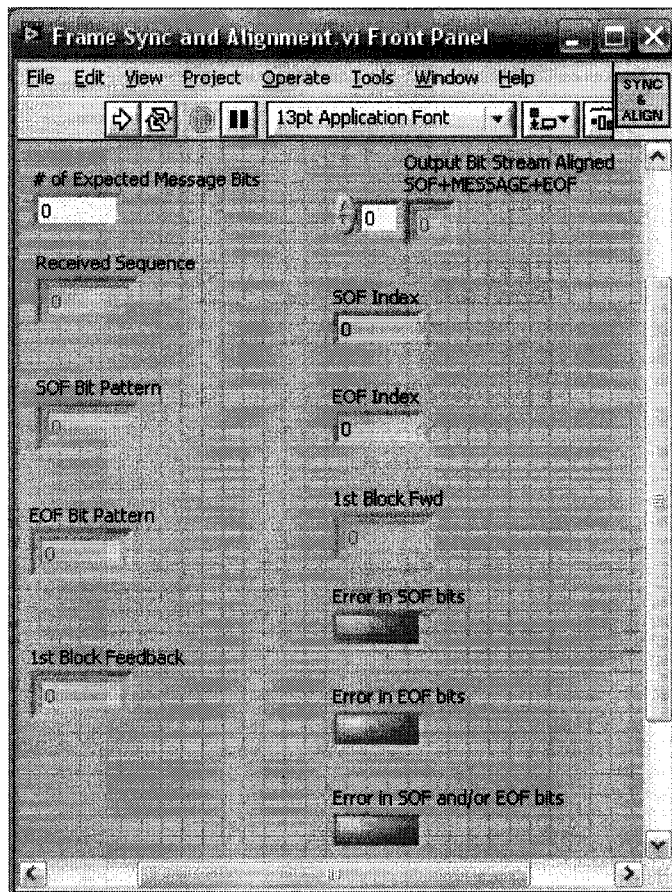


Figure 5-40 – Frame Sync and Alignment.vi front panel

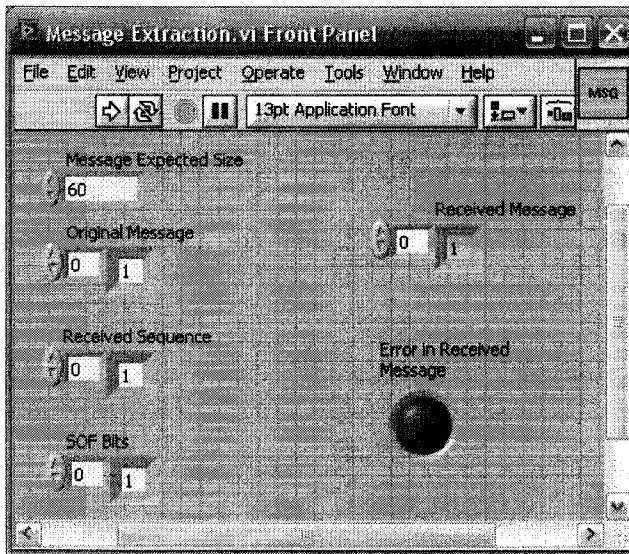
The received frame can be observed from the Received Frame tab in the front panel of the 4 QAM SDR Transceiver.vi. The SOF bit pattern index can also be observed from the SOF Index tab. This can be useful to observe how the receiver sound card sample rate differs from the one at the transmitter.

5.3.10 – Message Retrieving

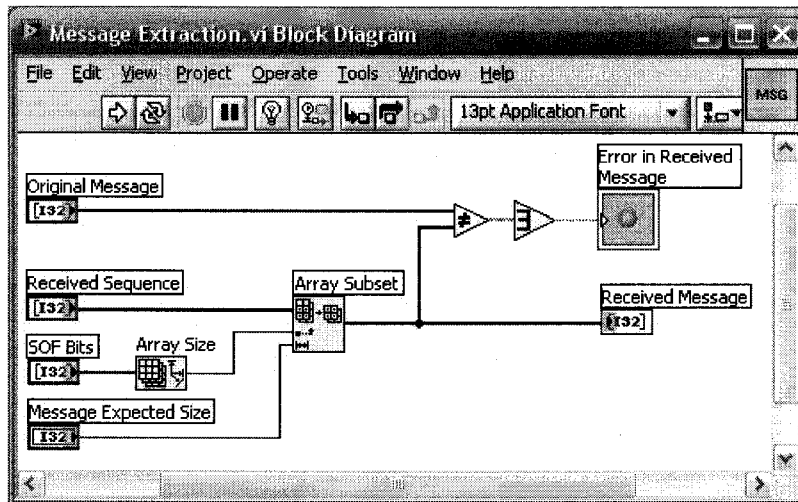
In the final stage of the system the message is separated from the SOF and EOF bits. The subVI Message Extraction.vi is responsible for this operation. It also compares the received extracted message with a copy of the original one. A message error is flagged if there is error in any received message bit. This subVI outputs the received message bits and the message error signal to be used in the display charts of the 4 QAM SDR Transceiver.vi.

The Received Message tab in the receiver's front panel displays the received and original messages, according to the radio button selection. The Message Error tab shows the message error chart for each frame. Figure 5-41 shows the front panel and block diagram of the Message Extraction.vi.

Also in this stage, a subVI, called BER.vi and responsible for the bit error rate (BER) calculation, was implemented. Its output is passed to the main code and the BER tab displays the bit error rate in the front panel. Figure 5-42 shows its front panel and block diagram.

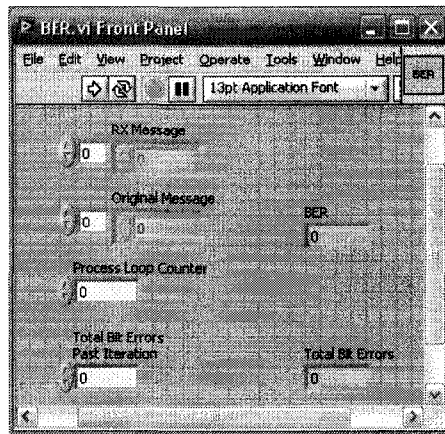


(a) Front panel

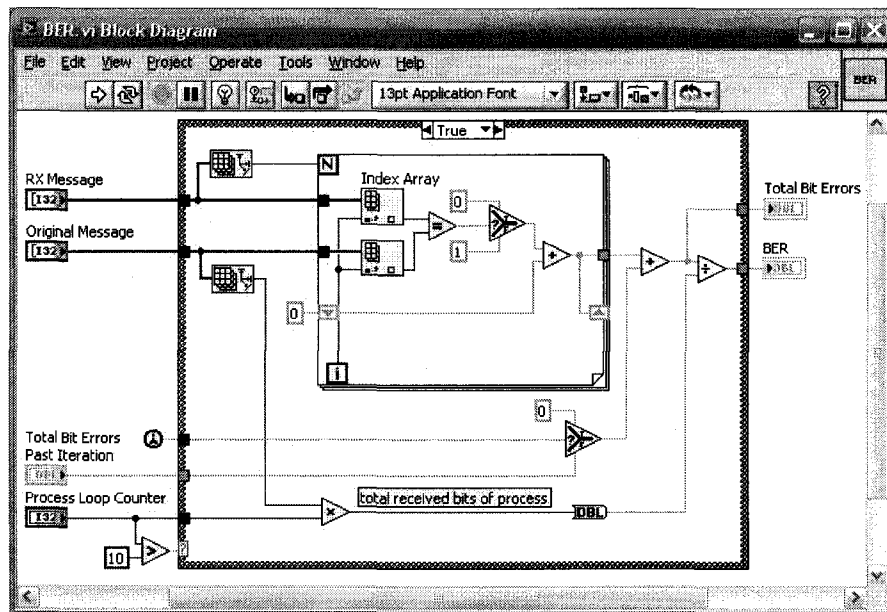


(b) Block diagram

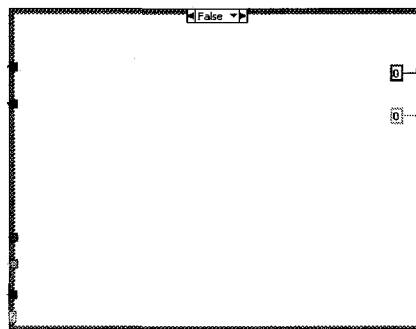
Figure 5-41 – Message Extraction.vi



(a) Front panel



(b) Block diagram



(c) Block diagram: detail of case structure

Figure 5-42 – BER.vi

5.4 – Remarks

Several different techniques are available to implement the stages of a digital communication system. This chapter presented the implementation of the 4 QAM SDR Transceiver.vi built-in stages using some of these techniques along with the relevant theory behind each process in the system.

The 4 QAM SDR Transceiver.vi was built using many resources provided by the LabVIEW libraries. Even though there are many other resources available for use in digital communication applications in the LabVIEW's library, some of these applications are not well suited for use in real-time operations as required by the 4 QAM SDR Transceiver.vi. Many stages inside the main code needed another implementation approach for this reason.

LabVIEW graphical programming proves to be more intuitive for programmers to create and understand parts of the code as compared to text-based programming languages. The modularity offered by the LabVIEW graphical environment helps to create blocks of code, called subVIs, which can be integrated into other VIs and subVIs to give a cleaner aspect to the overall code program. This modularity leaves the code open for modifications, if one desires to apply other technique in any stage and/or enhancements to the system.

CHAPTER 6

CONCLUSION

The SDR teaching system using LabVIEW and the personal computer's sound card offers the means necessary to explore the theoretical concepts of digital communication systems in a laboratory environment. It provides low cost, robustness, flexibility, and modularity, allowing additions and modifications for optimization and use with different techniques.

Software-defined radios employ the concepts learned in communications theory. Fundamentally, any technique learned in a digital communications course can be applied in a SDR system. Being not just a computer simulation, this type of system provides the means for the student to apply the theoretical concepts in a laboratory setting, dealing with a real-time system that actually produces physical transmitted signals. This proves to be more attractive and stimulating to students, since they can change system parameters and/or blocks of code to observe the system behavior by looking at the display charts and listening to the transmitted sound signal.

The SDR system described by this document uses the National Instruments' LabVIEW graphical environment. LabVIEW is becoming widely adopted in industry and academia for being a powerful and flexible software with

uses in data acquisition, instrument control, and analysis. Its graphical language makes programming more intuitive, allowing shorter time for coding and easier understanding of code segments.

This LabVIEW SDR learning platform was tested under real working conditions and it was concluded that it successfully works, providing the means to accommodate the main objectives in a digital communications laboratory setting, with a relatively low-cost alternative compared to traditional hardware-based systems. It is also a more robust system, being less susceptible to damage due to accidents and bad wire connections. For its graphical user interface, graphical coding environment, and the use of a personal computer's sound card, the proposed SDR teaching system using LabVIEW reaches the three main learning styles objectives discussed in chapter 1. Moreover, software applications are assuming the main role in many hardware-based systems due to the technological advances in electronics and signal processing techniques. Therefore, it is expected that electrical engineers become more familiar with software programming in view of this increasing tendency. The use of a software-defined radio platform in a digital communications laboratory environment can help prepare students for this new trend.

This study shows that this teaching platform is a flexible and modular system, which can be easily adapted for modifications and improvements. In addition to the features offered by this system, some future research topics that could be included in this system are suggested below.

a – Incorporation of RF Front End Hardware

This system operates by modulation and demodulation of an intermediate frequency signal in order to use a personal computer's sound card as the input/output device for transmission. In order to generate a RF signal, a RF front end hardware needs to be added to the system. This work was done by Plante [7] and Beckwith [8] in their SDR teaching platform using MATLAB. Since LabVIEW can use its function called Mathscript to use MATLAB code within LabVIEW code, a simple code adaptation of their routines interfacing the SDR-1000 hardware and the computer via the parallel port would be necessary.

b – Source and Channel Coding

Source and channel coding schemes can be added for bandwidth minimization and to increase the robustness of the system regarding message errors. LabVIEW has in its library many routines for implementation of coding schemes. Nevertheless, the possibilities are open for the one who desires to implement their own code.

c – Other Modulation Schemes

Other modulation schemes may be used in the SDR teaching platform, like 16 QAM and OFDM modulations. However, the appropriate routines must be developed to make the system work properly with the specific chosen modulation scheme, and the increased complexity of demodulation may exacerbate the buffer overflow problem.

d – Different Carrier Phase Recovery Techniques

The carrier frequency and phase recovery stage is a crucial process inside the receiver. Different techniques like Phase-Locked Loop (PLL) could also be attempted in this process in order to show other options to students.

e – Equalization

Equalization is a topic not covered by the proposed platform, since the system works with a short audio cable connecting the transmitter to the receiver and an equalizer was not necessary in the demodulation process. However, if one desires to implement an equalizer, there are also some predefined equalization blocks in the LabVIEW library. Also, one may attempt to use a different equalization technique implementation and incorporate it into the main code.

LIST OF REFERENCES

- [1] Jeffrey Travis, Jim Kring, "LabVIEW for Everyone: Graphical Programming Made Easy and Fun," Prentice Hall PTR, New Jersey, 2006
- [2] Nasser Kehtarnavaz, Namjin Kim, "Digital Signal Processing System Level Design Using LabVIEW," Newnes, Burlington, 2005
- [3] C. Richard Johnson Jr., William A. Sethares, "Telecommunication Breakdown," Pearson Prentice Hall, Upper Saddle River, New Jersey, 2004
- [4] Johnson, C. Richard Jr., "A Digital Quadrature Amplitude Modulation (QAM) Radio," Prepared for inclusion on CD-ROM accompanying Johnson and Sethares, Telecommunication Breakdown [3]
- [5] Steven A. Tretter, "Communication System Design Using DSP Algorithms," Kluwer Academic/Plenum Publishers, New York, 2003
- [6] Leon W. Couch, II, "Digital and Analog Communication Systems," Sixth Edition, Prentice Hall, Upper Saddle River, New Jersey, 2000
- [7] Plante, Matthew, "A Software Defined Radio Communications Laboratory Experience" Master thesis, University of New Hampshire, Durham, New Hampshire, 2006
- [8] Beckwith, Jonathan, "A Matlab and Software Defined Radio Approach to Teaching Digital Communications" Master thesis, University of New Hampshire, Durham, New Hampshire, 2005
- [9] Richard Lyons, *Quadrature signals: Complex, but not complicated*, [Online], Available: <http://www.dspguru.com/info/tutor/quadsig.htm>
- [10] Richard Lyons, "Understanding Digital Signal Processing," Second Edition, Prentice Hall PTR, Upper Saddle River, New Jersey, 2004
- [11] Connexions, *LabVIEW Graphical Programming Course*, [Online], Available: <http://cnx.org/content/col10241/latest/>
- [12] C. Zeitnitz, WaveIO: A Soundcard Interface for LabVIEW, [Online], Available: <http://www.zeitnitz.de/Christian/WaveIO/waveio.html>

[13] National Instruments, *Pulse Shape Filtering in Communication Systems*, [Online], Available: <http://zone.ni.com/devzone/cda/tut/p/id/3876>

[14] National Instruments, *Pulse Shaping to Improve Spectral Efficiency*, [Online], Available: <http://zone.ni.com/devzone/cda/ph/p/id/200>

[15] National Instruments, "LabVIEW User Manual," Part Number 320999E-01, 2003

[16] LAVA, *LabVIEW Advanced Virtual Architects Forums*, [Online], Available: <http://forums.lavag.org/forums.html>

[17] Flex-Radio Systems [Online], Available: <http://www.flex-radio.com>

[18] Gerald Youngblood, "A Software Defined Radio for the Masses, Part 1," QEX (including Communications Quarterly), July/Aug 2002

[19] Gerald Youngblood, "A Software Defined Radio for the Masses, Part 2," QEX (including Communications Quarterly), Sept/Oct 2002

[20] Gerald Youngblood, "A Software Defined Radio for the Masses, Part 3," QEX (including Communications Quarterly), Nov/Dec 2002

[21] Gerald Youngblood, "A Software Defined Radio for the Masses, Part 4," QEX (including Communications Quarterly), Mar/Apr 2003

[22] University of Illinois ECE 463 – Digital Communications Laboratory, [Online], Available: <http://courses.ece.uiuc.edu/ece463/>

[23] John G. Proakis, "Digital Communications," WCB McGraw-Hill, 3rd. ed., New York, 1995

[24] Mahesh L. Chugani, Abhay R. Samant, Michael Cerna, "LabVIEW Signal Processing," Prentice Hall PTR, Upper Saddle River, New Jersey, 1998

[25] Intuitive Guide to Principles of Communications, [Online], Available: <http://www.complextoreal.com/>

[26] Learnativity, Learning Theory/Styles [Online], Available: <http://www.learnativity.com/learningstyles.html>

[27] National Instruments, "Linear Systems in LabVIEW," Application Note 039, February 1993

APPENDICES

APPENDIX A

LABORATORY EXPERIMENTS

Lab #1 - Introduction to the LabVIEW Environment / PN Sequences

Lab #2 - Introduction to the Software-Defined Radio System Using LabVIEW and the PC Sound Card

Lab #3 - Pulse Shaping

Lab #4 - Carrier Recovery

Lab #5 - Symbol Timing Recovery

LAB #1 - Introduction to the LabVIEW Environment / PN Sequences

Abstract:

The Digital Communications Laboratory has a software-based format, using software-defined radio (SDR) as the teaching platform. Programs codes are designed in LabVIEW, which uses block-based code that allow more intuitive and shorter coding times compared to text-based programming languages. Although the course goal is not specifically to teach students how be LabVIEW programmers, it requires that they become familiar with the syntax of LabVIEW in order to understand the SDR design code and be able to write their own code when required.

References:

- [1] LabVIEW Graphical Programming Course, available online at <http://cnx.org/content/col10241/latest/>
- [2] Steven A. Tretter, "Communication System Design Using DSP Algorithms," Kluwer Academic/Plenum Publishers, New York, 2003
- [3] Jeffrey Travis, Jim Kring, "LabVIEW for Everyone: Graphical Programming Made Easy and Fun," Prentice Hall PTR, New Jersey, 2006

Required Equipment:

1. (1) Personal computer with LabVIEW software installed.

1. Background Information

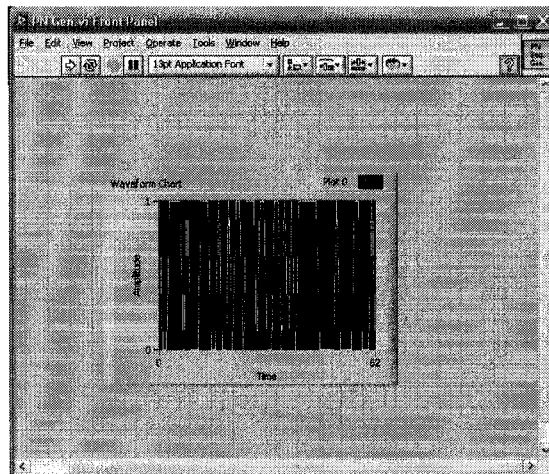
1.1. LabVIEW Graphical Environment

LabVIEW is a graphical programming environment that works on a data flow model (data flows from data source to data sink). A LabVIEW program, called a virtual instrument (VI), has three main parts: the front panel (FP), the block diagram (BD), and the icon/connector pane. A VI used within another VI is called a subVI (similar to a subroutine).

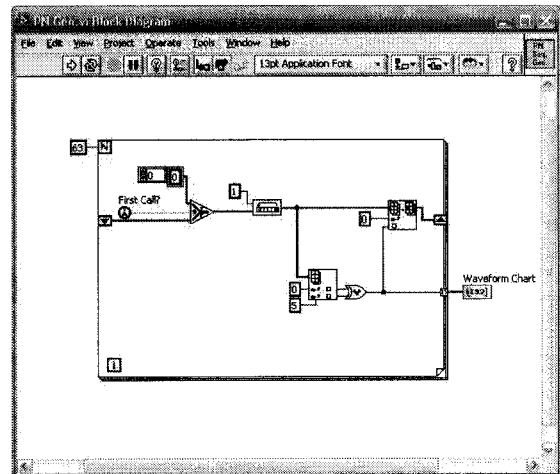
The front panel is the interactive user interface. It can contain graphical objects used as controls or indicators, like push buttons, switches, knobs, and graphs.

The block diagram is where the VI's source code (executable) resides. It contains sources, sinks, subVIs (lower-level VIs), and execution control structures. These objects are connected by wires to define the program logic. Front panel objects have corresponding terminals on the block diagram in order to pass data between the user and the program.

The VI icon is the graphical representation of the VI. The VI icon is displayed when it is inserted in a block diagram of another VI. The connector pane is the mechanism to connect wires from the block diagram to the subVI, defining its inputs (controls) and outputs (indicators).



(a) Front panel



(b) Block diagram



(c) Icon



(d) Connector pane

Figure A-1 – Components of a VI

In order to help you familiarize yourself with the LabVIEW environment, you will be guided to build a pseudo-noise sequence generator.

1.2. Pseudo Noise (PN) Sequences

Pseudo-noise (PN) sequences generators can be built using linear feedback shift registers. In these binary sequences, also called pseudo-random, maximal length, or m-sequences, the elements (0s and 1s) have a probability very close to 0.5.

In this experiment, a pseudo noise generator will be built using a six-stage linear feedback shift register structure, as shown by figure A-2.

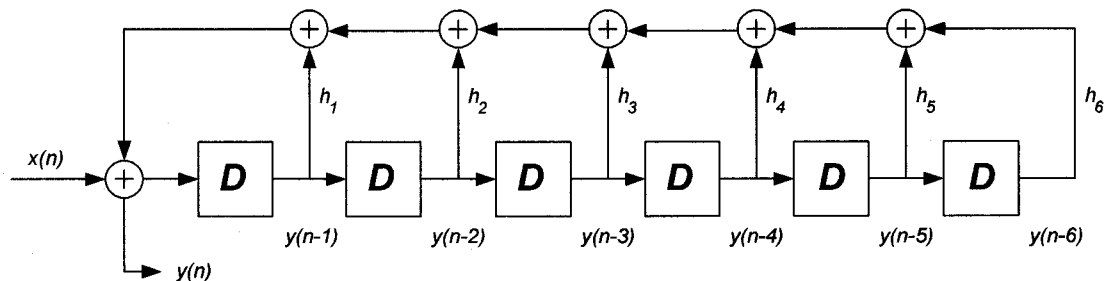


Figure A-2 – PN generator with a six-stage linear feedback shift register structure

The connection polynomial we will use is given by:

$$h(D) = 1 + D + D^6 \quad (\text{A-1})$$

where D is the delay and the summations represent modulo 2 additions.

This sequence generator has a period of 63 ($2^6 - 1 = 63$). Refer to [2] for more details about PN sequences.

2. Procedure

2.1. Open LabVIEW.

2.2. Open a new blank VI. The FP and BD window will open.

- 2.3. Click CTRL-T to tile the FP and BD windows. On the BD left-click on Help menu and choose Show Context Help. The Context Help window shows the description of the icons.
- 2.4. In the BD window, right-click to open the Functions Palette and choose Structures>>While Loop. Use the left mouse button to create the while loop box.

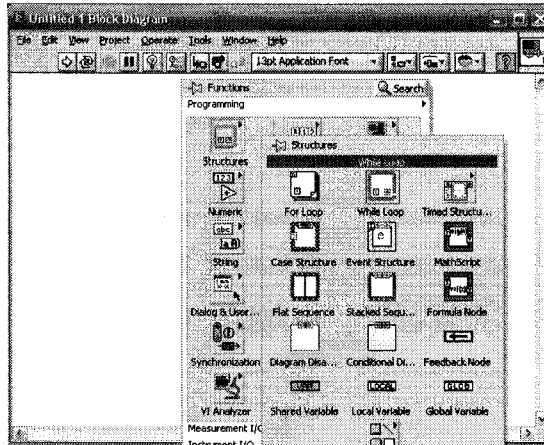



Figure A-3 – While Loop

- 2.5. Place the mouse cursor over the stop sign. You will see the cursor showing the wiring tool . Create a button to control the loop execution by right-clicking on the stop sign and choosing Create Control. A stop button will automatically appear on the FP and its icon on the BD will be wired to the loop stop sign. Click CTRL-E to change between the BD and the FP windows.

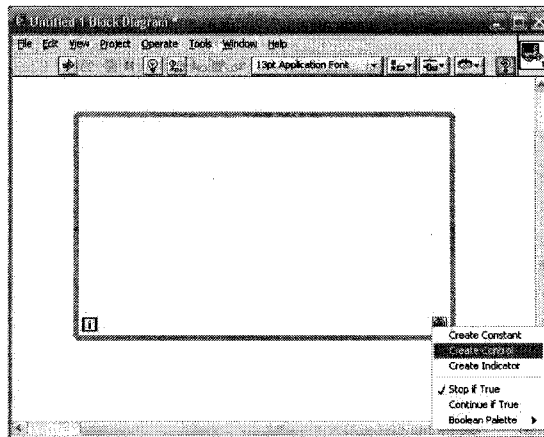


Figure A-4 – Creating stop button

- 2.6. Press the run arrow to start running the VI. Stop the VI using the stop button. Remember: Always stop the VI by using your stop button. Use the LabVIEW toolbar stop button only as a last resort.



Figure A-5 – Running the VI

- 2.7. On the BD, right-click on the left or right edge of the loop and choose Add Shift Register. You will see a pair of terminals directly opposed to each other on the vertical sides of the loop. The terminal of the right side passes the value wired to it at the end of each loop iteration to the terminal at the left side. Shift registers adapt automatically to the data type of the objects wired to it.

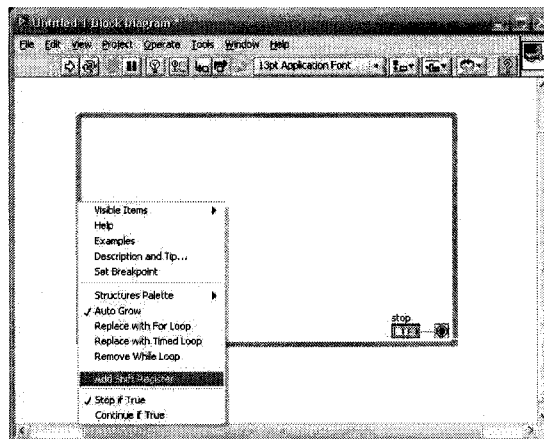


Figure A-6 – Adding shift register

- 2.8. Right-click on the FP to open the Controls Palette. Choose Array, Matrix & Cluster>>Array. An array shell is created on the FP. Now right-click again

on the FP and choose Numeric>>Numeric Control on the Controls Palette to create a numeric control and place it into the array shell. Resize the array shell to get six numeric controls inside it. Double-click on the label and change it to Initial State. Left-click inside the numeric controls to insert the values shown on the figure A-7.

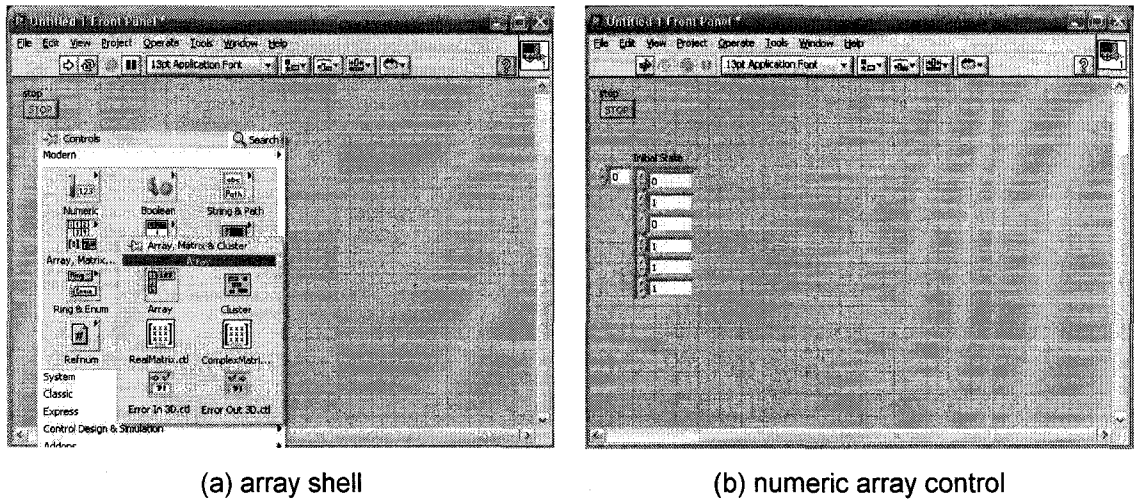


Figure A-7 – Creating array control shell

- 2.9. On the BD, left-click on the array Initial State icon to drag it to the left side of the loop and wire it to the shift register, as shown by figure A-8. The Initial State array will initialize the shift registers with its values.

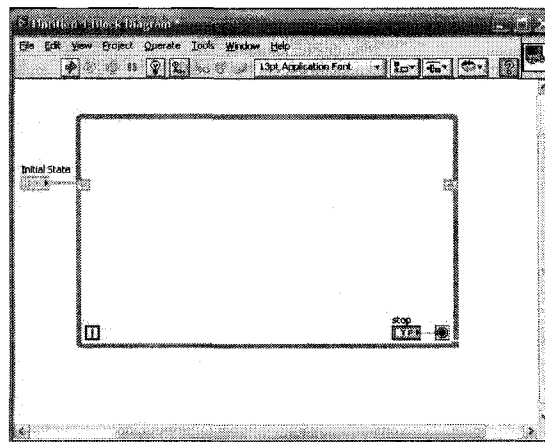


Figure A-8 – Initializing the shift register

- 2.10. Right-click on the BD and choose Array>>Rotate 1D Array. Place it inside the loop and wire it to the shift registers according to figure A-9. Right-click on its upper left terminal and choose Create>>Constant to create a constant. Change the value of this constant to 1 (we want to rotate the array elements by only one place). Note that the wire connecting this constant is thinner. This is because this constant is a scalar (arrays have thicker wires).

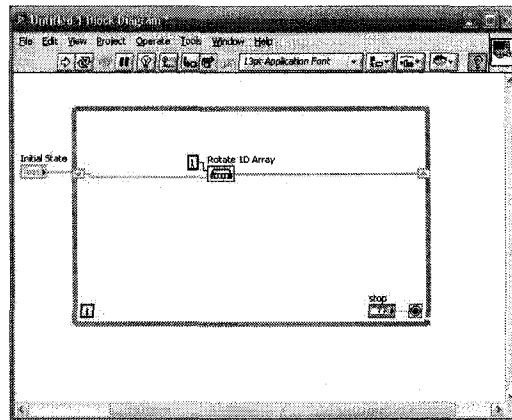


Figure A-9 – Wiring the rotate 1D array function

- 2.11. Right-click on the BD and choose Array>>Index Array. Place it inside the loop. The Index Array function allows you to access a particular element of an array. In our case, we want to access the elements of index 0 and 5 (the first and sixth elements). Resize the Index Array icon to show two index terminals. Create the constants, modify its values, and wire according to the figure A-10.

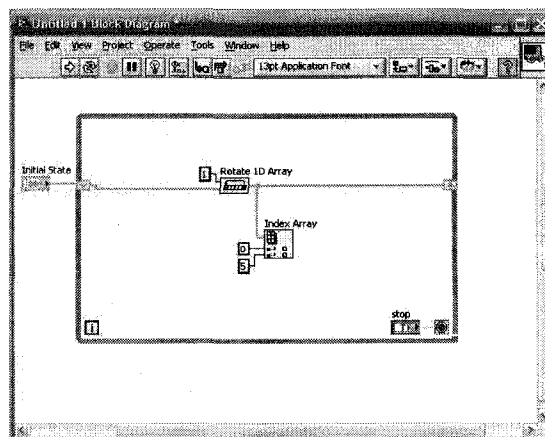


Figure A-10 – Inserting index array

- 2.11. We need to XOR the values of the first and sixth shift registers. Place the XOR function on the BD by right-clicking on it and choosing Boolean>>Exclusive OR and wire according to the figure A-11. The XOR output will be the PN sequence generator output.

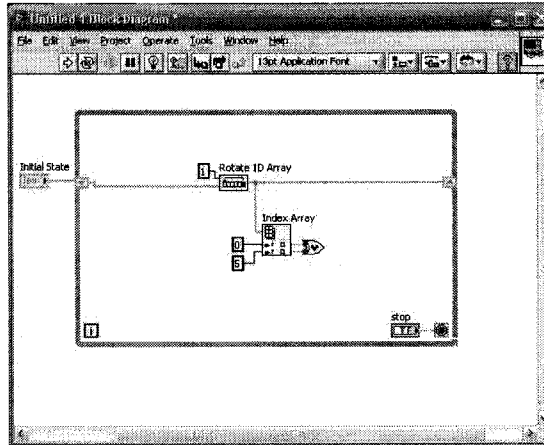


Figure A-11 – Wiring XOR function

- 2.12. Now the index 0 element of the array needs to be replaced by the output value of the XOR operation. Right-click on the BD and choose Array>>Replace Array Subset. Right-click on its icon and create a constant of value 0 for the index terminal. Delete the wire connecting the right side shift register (left-click on it to mark and backspace). Wire according to the figure A-12.

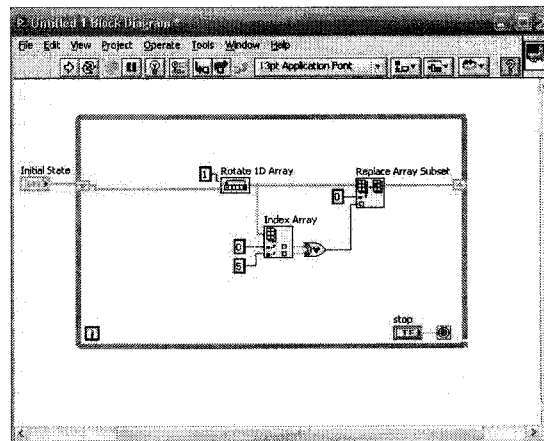


Figure A-12 – Inserting replace array subset

- 2.13. Do you see red dots on some of the icon terminals? They are called Coercion Dots. They are done automatically in order to match the data type of different data type objects. When possible, we want to avoid coercion dots to save memory (LabVIEW uses memory to store a copy of the data when using coercion dots). Let's fix this. Right-click on the Initial State array icon and choose Representation>>I32. The data type change makes all the coercion dots vanish.

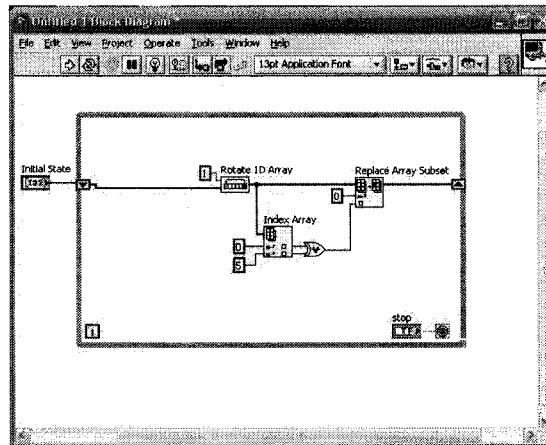
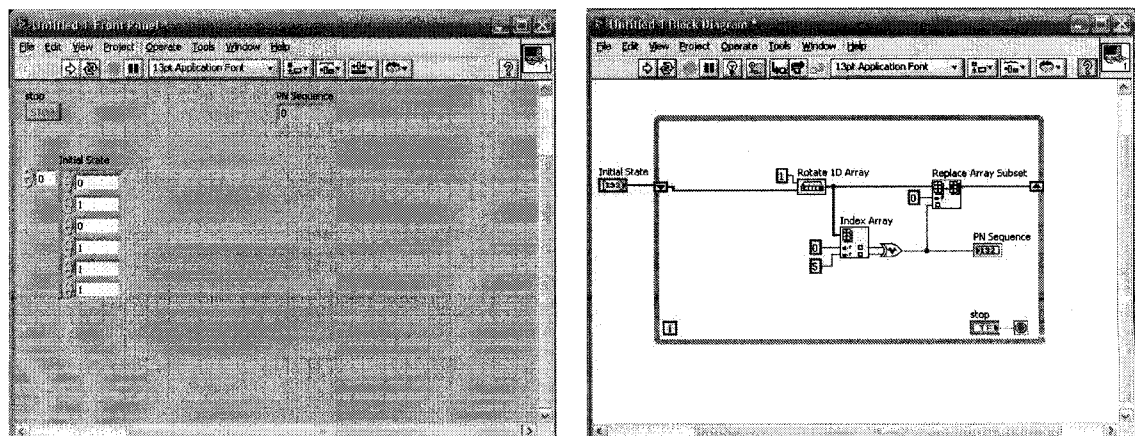


Figure A-13 – After initial state data type change

- 2.14. Create an indicator to display the output of the PN sequence generator. Move the cursor over the output of the XOR function to see the wiring tool. Right-click and choose Create>>Indicator. Double-click on its label and change it to PN Sequence.

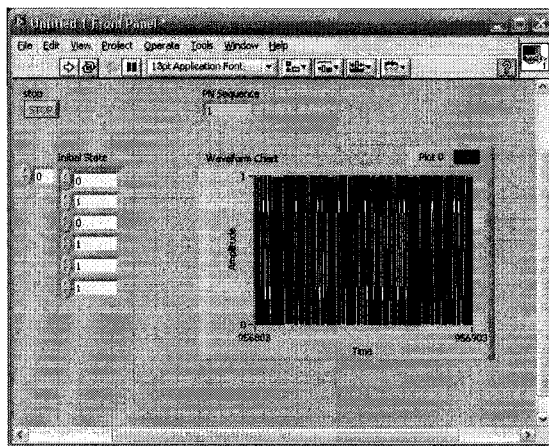


(a) FP

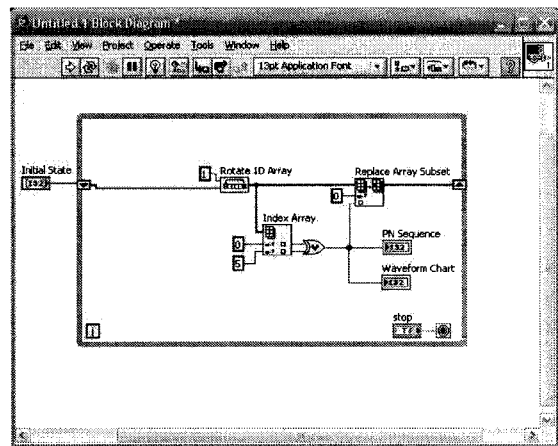
(b) BD

Figure A-14 – Inserting PN sequence indicator

- 2.15. On the FP, click the run button on the toolbar and see how the PN Sequence indicator changes rapidly. Stop the VI by clicking on your stop button.
- 2.16. Now let's make the output display more interesting. Create a waveform chart by right-clicking on the FP and choosing Graph>>Waveform Chart. Go to the BD (CTRL-E to switch back to the BD) and place its icon inside the loop. Wire it to the XOR output. Run the VI and observe the waveform. Stop the VI. Can you see the periodic characteristic of the waveform? What is the period of the generated sequence in samples? Clear the chart by right-clicking on the Waveform Chart and choosing Data Operations>>Clear Chart.



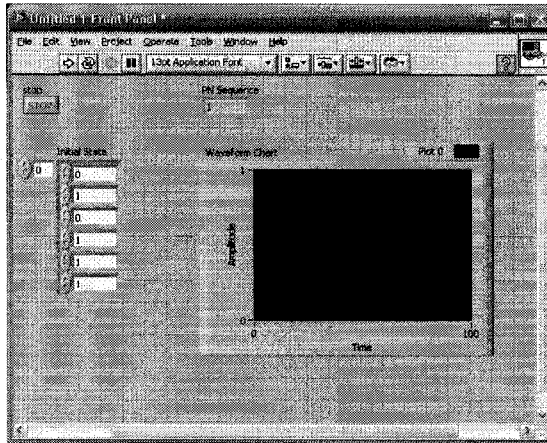
(a) FP



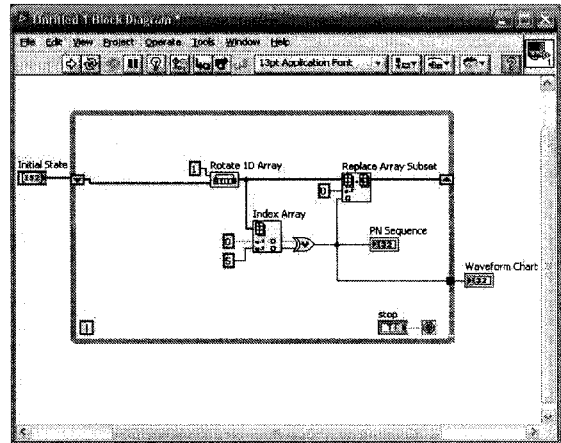
(b) BD

Figure A-15 – Running the VI with the waveform chart

- 2.17. Now go to the BD and move the Waveform Chart icon outside the loop (left-click and drag it). Choose Edit>>Remove Broken Wires from the toolbar. Connect the Waveform Chart to the output of the XOR function. Notice now that you have created a loop tunnel (the little square on the loop border). Its purpose is to pass data out of the loop when the loop stops executing. Run the VI and observe the chart. Stop the VI.



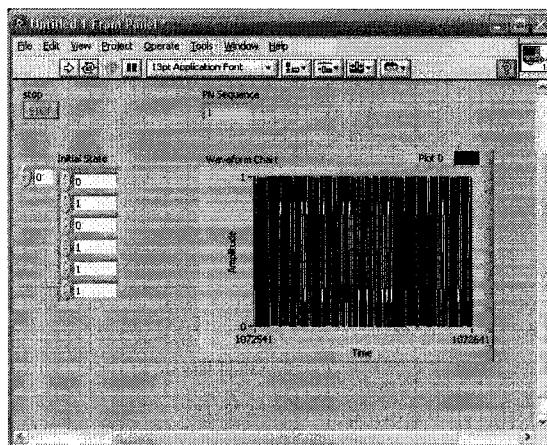
(a) FP



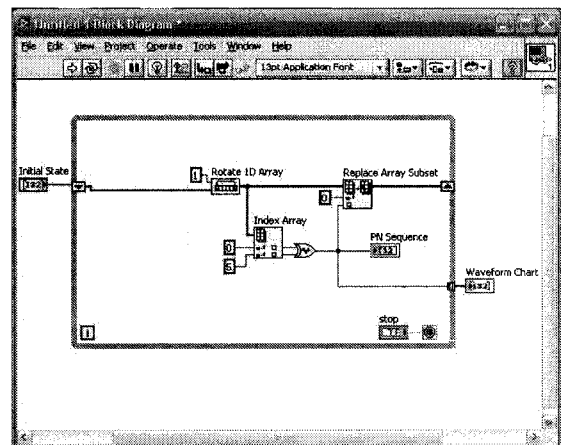
(b) BD

Figure A-16 – Moving waveform chart outside the loop

2.18. Did you notice that the chart displayed only one value at the time the loop stops? You need to enable indexing on the loop tunnel in order to have the output data of every loop iteration accumulated. Right-click on the loop tunnel and choose Enable Indexing. Now when the loop stops, the output data of every loop iteration will pass to the chart. Run the VI and observe the difference. Stop the VI.



(a) FP



(b) BD

Figure A-17 – VI after enabling indexing

2.19. Now let's prepare this VI to generate only 63 periodic samples every time it is called. Substitute the while loop with a for loop by right-clicking on the while loop border and choosing Replace with For Loop. Delete the stop button (left-click it to mark and backspace).

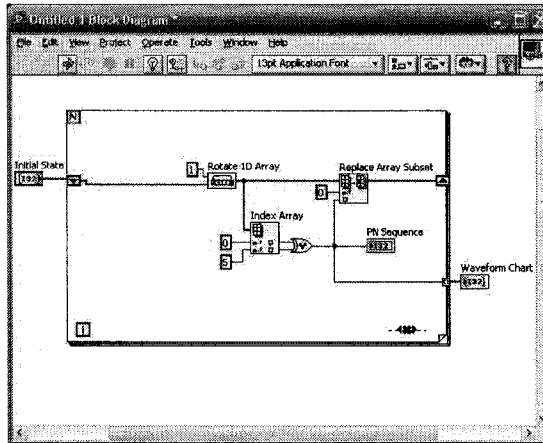


Figure A-18 – For loop

- 2.20. Right-click on the Count terminal of the loop and choose Create Constant. Change the constant value to 63 (the period of the PN sequence). Move the Initial State array inside the for loop. Delete the wire connecting the shift register and the Rotate 1D Array function (left-click it to mark and backspace). Remove broken wires (use CTRL-B). Right-click on the BD and choose Functions>>Comparison>>Select. Place the Select function inside the loop. Right-click on the BD and choose Functions>>Synchronization>>First Call?. Place it inside the loop and wire according to figure A-19.

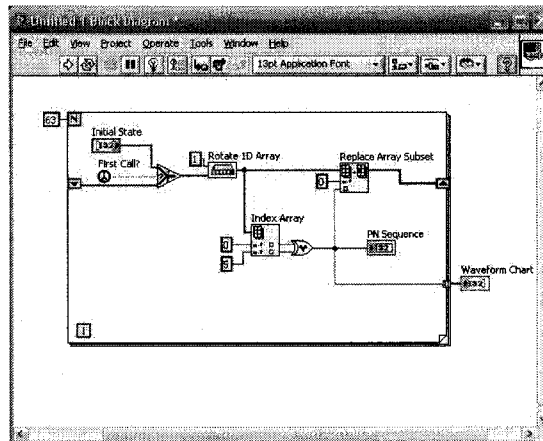


Figure A-19 – Generation of 63 PN samples

- 2.21. Run the VI. Observe that now the VI executes until it generates 63 PN sequence samples. The First Call? function has a true Boolean value on the first time the VI is called. After that, its value is set to false. The Select function passes the Initial State array values to the Rotate 1D Array

function only when the VI is called the first time. After that, the old values in the shift registers are passed from the previous execution of the VI.

- 2.22. It is time to prepare the VI to be used as a subVI. Move the PN Sequence indicator outside the for loop. Delete the Waveform Chart and the PN Sequence indicator. Remove broken wires (use CTRL-B). Move the mouse cursor over the loop tunnel and when the wiring appear right-click and choose Create>>Indicator. Double left-click on its label and change it to PN Sequence. This new indicator now is ready for displaying all the data for every loop iteration. Right-click on the Initial State array icon and choose Change to Constant. Your BD should look like the one in figure A-20. The BD is now finished.

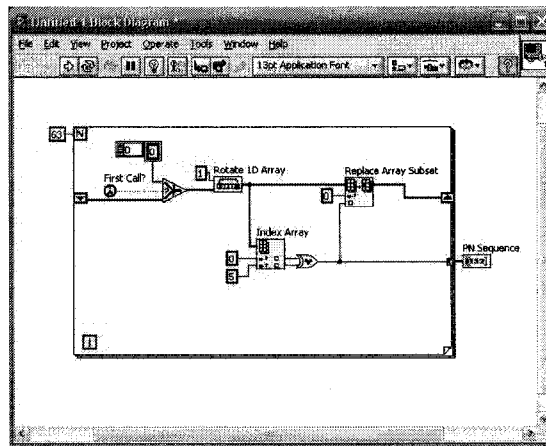


Figure A-20 – Final BD

- 2.23. Edit the VI icon by right-clicking on it in the FP and choosing Edit Icon... In the Icon Editor window left-click Edit and choose Clear. Edit the icon using the tools available in the tools palette of the Icon Editor window. Let it looking like the one in the figure A-21. Click OK.

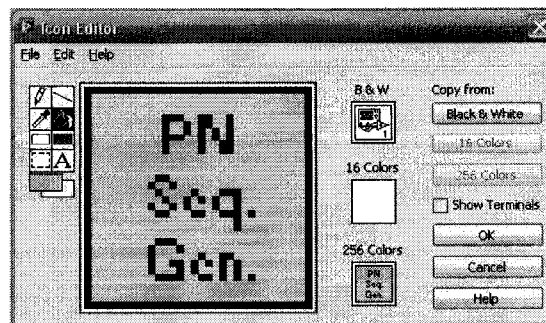


Figure A-21 – Icon editor window

Right-click on the VI icon in the FP and choose Show Connector. The only terminal in this VI is the PN sequence output; therefore, you can choose a simple connector pattern. Right-click on the connector pane, go to Patterns and choose the pattern shown by figure A-22. It contains only two terminals.

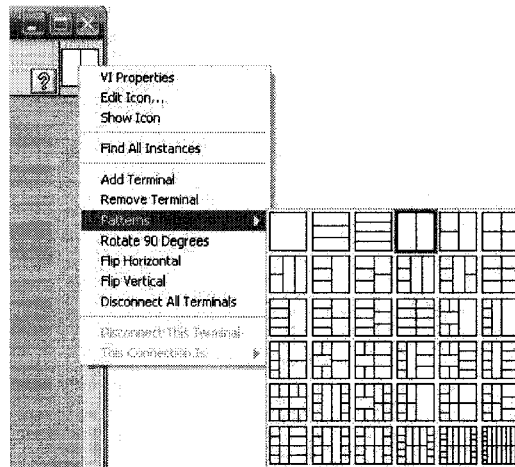


Figure A-22 –VI connector pane

- 2.24. Position the mouse cursor over the connector pane and observe that the wiring tool appears. Left-click on the right terminal. Position the mouse cursor over the PN Sequence indicator and left-click again to associate the icon right terminal to this indicator. Right-click on the VI icon again and choose Show Icon.
- 2.25. The last step is to document the VI. Left-click on the toolbar File menu and choose VI Properties. In the VI Properties window, under Category menu choose Documentation. Document the VI by typing in the VI description frame, as indicated by figure A-23.

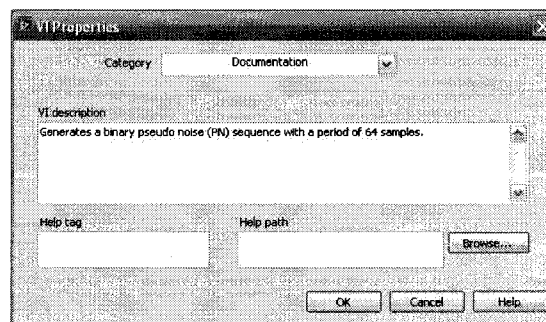


Figure A-23 – Documenting the VI

Now when you place the mouse cursor over the VI icon the Context Help window will show the description of the VI.

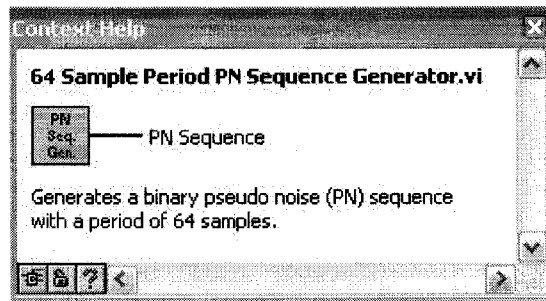


Figure A-24 – Context Help Window

2.26. Save the VI as “63 Sample Period PN Sequence Generator.vi”

LAB #2 - Introduction to the Software-Defined Radio System Using LabVIEW and the PC Sound Card

Abstract:

The digital communication system that you will be working with is implemented by software-defined radio using LabVIEW and the PC sound card. In this system, two computers communicate with each other through the use of a digital modulation scheme. At one side, the transmitter (implemented in software) generates a digital modulated waveform from a particular binary sequence (the message) and sends this signal to the PC sound card output. An audio cable connects the sound card output from the transmitting computer (TX) to the sound card input of the receiving PC (RX). The receiver (also implemented in software) processes the incoming signal and decodes the message. By using this system, you can change modulation parameters and sound card settings to observe the effects of these modifications at several nodes of the block diagrams using the friendly graphical user interface of LabVIEW.

References:

- [1] LabVIEW Graphical Programming Course, available online at <http://cnx.org/content/col10241/latest/>
- [2] Jeffrey Travis, Jim Kring, "LabVIEW for Everyone: Graphical Programming Made Easy and Fun," Prentice Hall PTR, New Jersey, 2006
- [3] C. Richard Johnson Jr., William A. Sethares, "Telecommunication Breakdown," Pearson Prentice Hall, Upper Saddle River, New Jersey, 2004

Required Equipment:

1. (2) Personal computer with soundcards and LabVIEW software installed.
2. (1) Audio cable.

1. Background Information

1.1. Digital Modulations

In digital communication systems, there are three basic types of digital modulation techniques. Each type uses a sinusoid to represent the information to be sent, where one of the sinusoid parameters is varied according to the modulation. The three basic modulation schemes are:

- Amplitude Shift Keying (ASK) – amplitude of carrier is varied
- Frequency Shift Keying (FSK) – frequency of carrier is varied
- Phase Shift Keying (PSK) – phase of carrier is varied

Figures A-25a, A-25b, and A-25c show an example for each of these modulation types. A 4-QPSK (quadrature phase shift keying) scheme is depicted by figure A-25d. Note that in the 4-QPSK modulation the carrier phase can assume 4 different values. PSK modulation can be regarded as a special case of quadrature amplitude modulation.

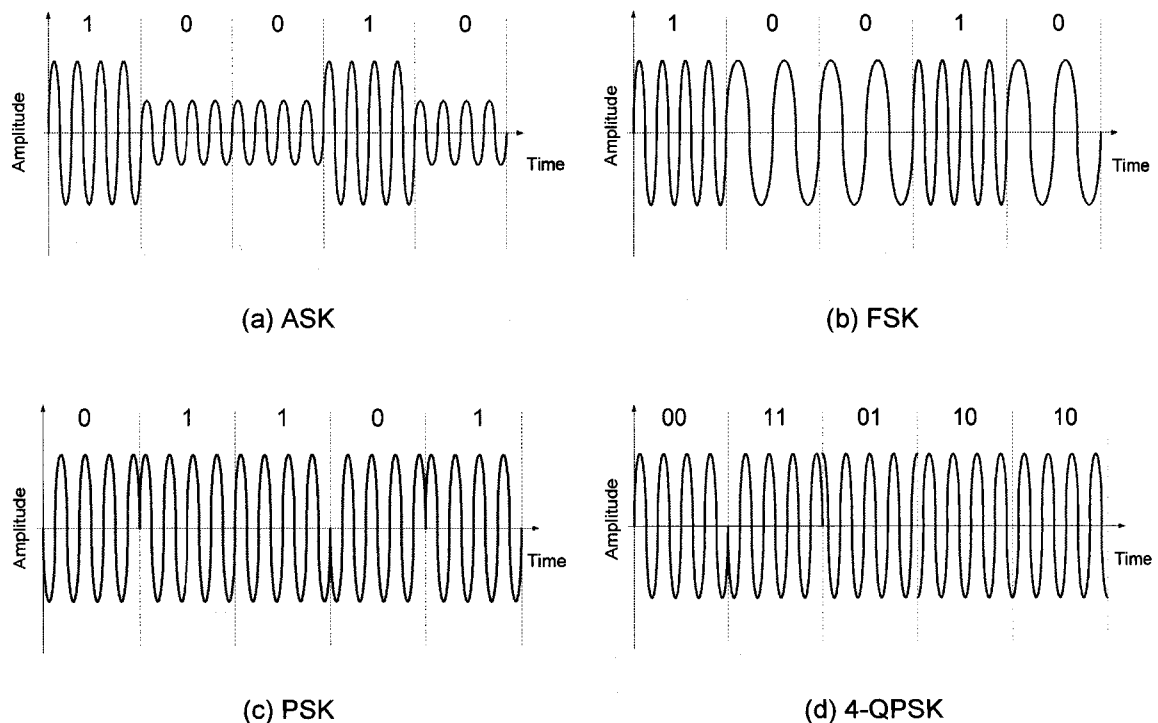


Figure A-25 – Digital modulation schemes

Quadrature amplitude modulation is a technique where two sinusoids out of phase with each other by 90° are used as carriers. The information to be sent

is divided into two channels: in-phase ($I(t)$) and quadrature-phase ($Q(t)$). The transmitted signal $s(t)$ is given by the form:

$$s(t) = I(t) \cos(2\pi f_c t + \theta) - Q(t) \sin(2\pi f_c t + \theta) \quad (\text{A-2})$$

where f_c is the carrier frequency, θ is its phase, and t is the time instant. At the receiver side, the in-phase $I'(t)$ and quadrature-phase $Q'(t)$ components are recovered by mixing the incoming signal with two locally generated sinusoids with frequency f_o and some phase ϕ and a low pass filtering scheme with cutoff frequency around f_o (or less, depending on the bandwidth of the modulating signal). The frequency f_o should match the carrier frequency f_c .

1.2. The Software-Defined Radio System

The 4 QAM scheme is the modulation type used in our software-defined radio system. You will use the LabVIEW program code 4 QAM SDR Transceiver.vi to transmit and receive digital modulated signals using the computer sound card. Each computer can be set as the transmitter (TX) or the receiver (RX) during the radio operation, depending on the user commands (click on TX or RX tab, respectively). The modulation and sound card parameters can be seen and modified in the front panel controls. To get back the default values for all the controls in the front panel click Edit in the menu bar and choose Reinitialize Values to Default. Signals at relevant nodes of the system can be observed by their corresponding charts in the control panel. These charts are selected by clicking on the proper tab name.

The structure of the transmitter and the receiver are given by figures A-26 and A-27, respectively. Each stage of the system is explained next with some detail.

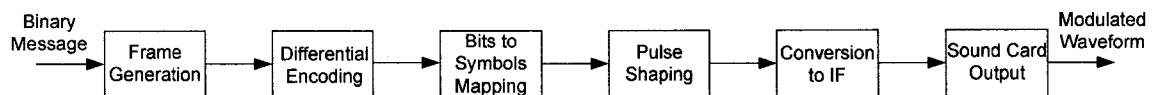


Figure A-26 – Transmitter structure

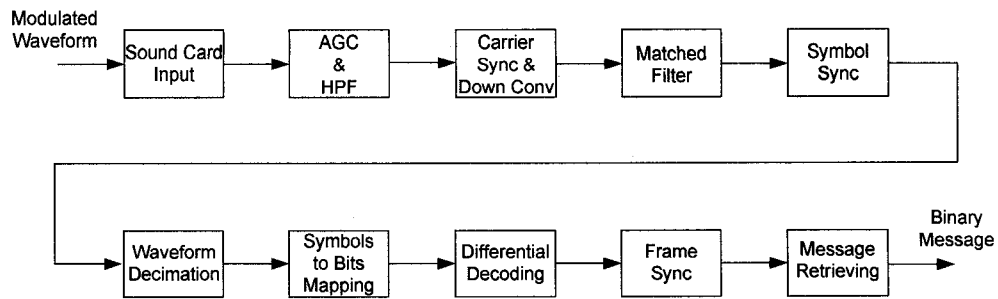


Figure A-27 – Receiver structure

1.2.1. The Transmitter

Frame Generation

The binary message is encapsulated into a frame pattern to be transmitted, as shown by figure A-28. The frame structure starts with 20 bits of a predefined pattern, called SOF (start of frame) bits. After the SOF bits are appended the message bits. The message bits used in this VI comprise a 64-bit sequence generated by the 64 Sample Period PN Sequence Generator.vi (it has a period of 64 samples). Another 20-bit sequence of a different predefined pattern, called EOF (end of frame) bits is appended after the message bits. In the receiver, the SOF and EOF bits are correlated with the received bit sequence in order to extract the correct frame boundaries. They are also used for resolving the phase ambiguity of an even multiple of $\pi/2$ radians at the carrier synchronization stage. These bit patterns can be seen and modified in the front panel.

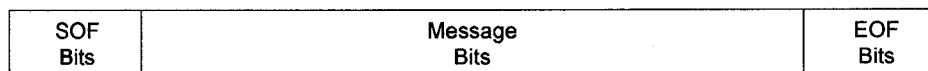


Figure A-28 – Frame structure

The message bits and the frame bits can be observed from the Message Bits and Frame Bits tabs in the front panel, respectively, when the VI is running as transmitter.

Differential Encoding

Differential encoding is applied to eliminate the phase ambiguity of a multiple of π radians at the carrier synchronization block. The frame bits are encoded using a Boolean XOR operation bitwise, according to equation A-3:

$$y_n = x_n \oplus y_{n-1} \quad (\text{A-3})$$

where:

y_n ... encoded sequence

x_n ... input sequence

n ... sequence index

The encoded frame bits can be observed from the Frame Bits tab in the front panel, choosing the proper radio button in the tab.

Bits to Symbols Mapping

The frame bits are mapped into the in-phase (I) and quadrature-phase (Q) components. In this 4 QAM scheme, each pair of bits is mapped into a point in the constellation diagram according to the figure A-29. These points can be observed from the Constellation Diagram chart.

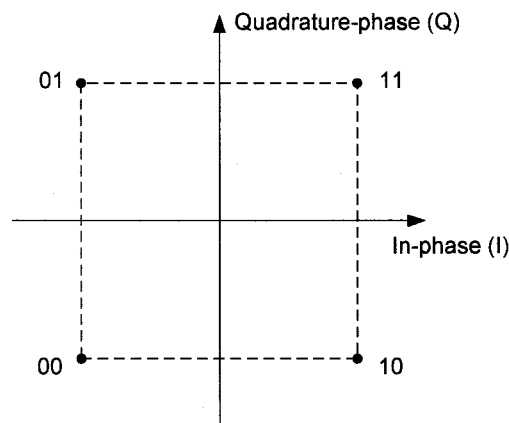


Figure A-29 – 4 QAM constellation diagram

Pulse Shaping

In this stage, the signal is up-sampled and passed through a pulse shaping filter. This is done to limit the bandwidth of the transmitted signal, minimize the intersymbol interference (ISI), and maximize the signal-to-noise ratio (SNR) at the receiver. Each symbol is up-sampled by the value specified in the Samples/Symbol control in the front panel. The default value is 16.

The type of pulse shaping filter used is defined in the Pulse Shaping Filter control. It is possible to use the raised cosine (RC), the root raised cosine (RRC), or the rectangular (none) pulse shaping filter. The default type is root raised cosine (RRC). The filter parameter alpha (roll-off factor) is defined in the Filter Parameter control. The default value is 0.5.

This block is also responsible for the desired transmitted symbol rate. It can be set in the Symbol Rate control. The default value is 2000 baud. The waveforms of the in-phase and quadrature-phase baseband signals can be observed by clicking on the Pulse Shaping tab and choosing the proper radio button in the tab.

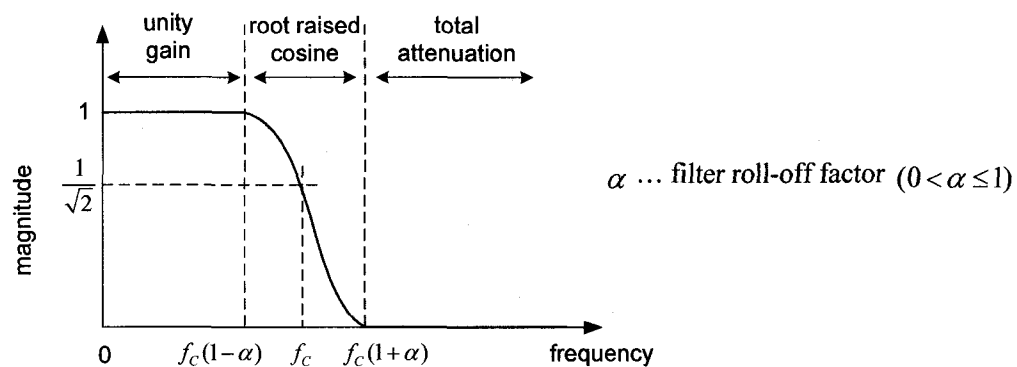


Figure A-30 – Ideal root raised cosine frequency response

Conversion to IF

The complex baseband signal modulates a quadrature pair of intermediate frequency (IF) carriers before it is sent to the sound card for transmission. The value of the carrier frequency can be adjusted in the control panel using the IF control. Its default value is 10 kHz. This block is also responsible for the actual sample rate generated by the code. The symbol rate and the number of samples per symbol determine the sample rate of the carrier.

$$F_s = (\text{Symbol Rate}) \cdot (\# \text{ of Samples per Symbol}) \quad (\text{A-4})$$

F_s ... Sample rate [Hz], Symbol Rate [symbols/sec]

The up-converted waveform can be observed in the time and frequency domains by clicking on the Output Signal (sound card) and FFT Output Signal (sound card) tabs, respectively, in the front panel.

The effective sample rate of the output signal is defined by the sound card parameters, as explained next.

Sound Card Output

Ultimately, the signal is output by the sound card at a sample rate defined in the Sample Rate control in the front panel. Its default value is 44100 Hz. This sample rate can be set in 1 Hz steps.

Buffers are used to transmit data between the SDR code and the Windows API (the core set of application programming interfaces). When the time between calls to the Windows API is in excess of the total buffer time per channel, buffer over-runs or under-runs occur. In order to avoid this problem, the number of buffers must be at least two, but a higher number is more reliable. The number of buffers can be set in the # of Buffers control in the control panel. Its default value is 10.

The Device ID control in the control panel specifies the output device used by the system. If the computer has more than one sound card, it is possible to choose which one to use. The default device ID is 0.

1.2.2. The Receiver

Sound Card Input

This stage is responsible for the acquisition of the transmitted signal. Here, the sound card sample and the number of buffers are defined in the control panel. The default value for the sample rate is also 44100 Hz. Ideally, the sound cards' sample rates should have the same value for the transmitter and the receiver computers. However, small discrepancies are likely to occur and they can cause synchronization problems in the demodulation process, depending on which algorithms are used.

The demodulation process involves heavy computations. Normally, high-performance digital signal processors and/or FPGAs are used to perform these operations. Since we are using personal computers running under Windows operating systems, there will be delays between the processing operations and signal acquisition. Therefore, we can expect buffer overflow at some point during a continued demodulation process. In order to have the receiver working properly during a certain time frame, we can adjust the number of buffers to a larger value. Typically, 3000 buffers will let the system work properly for about three minutes during the demodulation process. The number of buffers must be increased when more demodulation processing time is needed. The default value for the number of buffers is 3000.

Automatic Gain Control (AGC) and High Pass Filtering (HPF)

The amplitude of the incoming signal must be adjusted in order to make the demodulation process work properly. Amplitude variations in the received signal can cause erroneous behavior of subsequent processing stages. The automatic gain control (AGC) provides a constant average signal level applied to the demodulation stages. It works by measuring the RMS (root mean squared) value of the input signal from the sound card and adjusting the signal to a predefined RMS level.

A high pass filter (HPF) is also applied in this stage in order to clean the incoming signal of the low frequency noise added by the sound cards and other interference.

Carrier Synchronization and Down-Conversion

The incoming signal needs to be down-converted back into its baseband form in order to recover the transmitted message. The down-conversion to baseband is done by mixing the incoming signal with a sinusoid with the same phase and frequency as the carrier, and this mixing output is passed through a low pass filter with a cutoff frequency less than the intermediate frequency (IF) value. Therefore, the receiver needs to determine both the frequency and phase of the modulating sinusoid. Even though the transmitter and the receiver are set to work with the same carrier frequency, slightly frequency offsets are expected to happen.

This stage uses the Quadrature Costas Loop method to extract the carrier phase and frequency of the 4-QAM signal. It is a scheme where the recovered phase converges to an integer multiple of 90° shift from the true carrier phase. This phase ambiguity is resolved using differential encoding and correlation of the SOF and EOF bits. The Costas loop uses an adaptation constant, here called μ (μ), the value of which depends on how large the frequency offset is. Larger frequency offsets require larger values of μ to make the Costas loop acquire the right frequency and phase from the incoming signal. The outputs of the carrier synchronization and down-conversion stage block are the I (in-phase) and the Q (quadrature-phase) baseband signals, which can be observed from the Baseband Conversion tab in the control panel. The I and Q components can be selected by clicking on the corresponding radio button in that tab. The phase update output of the Costas loop algorithm can be observed by clicking the Phase Update tab in the front panel.

Matched Filter

After the incoming signal is returned to its baseband form, it needs to be passed through a filter that will reduce the intersymbol interference (ISI) and maximize the signal-to-noise (SNR) ratio. In order to do this, the receive filter should be matched to the pulse shape filter created in the transmitter (its impulse response must be a scaled time reversal of the pulse shape). It is called a matched filter for this reason. The outputs of the matched filter stage can be observed by clicking on the Matched Filter tab in the front panel. The I and Q components can be selected by clicking on the corresponding radio button in that tab.

Symbol Synchronization and Alignment

This block is responsible for finding the first ideal symbol time instant from the matched filter output. Its output is a symbol time aligned complex waveform, where the first sample corresponds to the optimal symbol instant. This is done to prepare the waveform for the decimation process, where one sample of each symbol must be extracted at an optimal time. The outputs of the Symbol Timing Recovery.vi are the complex array containing the quadrature symbol values corresponding to optimal sampling times and zero values for other sampling times, and the recovered symbol clock signal bundled with the in-phase component of the input.

The symbol timing recovery process can be observed in the 4 QAM SDR Transceiver.vi by clicking on the Symbol Timing tab in the front panel, which displays the in-phase signal and the recovered symbol clock.

Waveform Decimation

During the modulation process at the transmitter, each symbol was up-sampled by the pulse shape filter. The Waveform Decimation stage down-samples the waveform by the same factor used in the Samples/Symbol control of the transmitter. This value must be set in the Samples/Symbol control in the receiver's control panel. Its default value is 16.

The Eye Diagram chart in the control panel is a visualization tool that shows how well-formed the incoming pulse shaped signal is. A number of superimposed traces of the waveform (after the Symbol Synchronization and Alignment stage) starting at an integer multiple of the symbol time are added to the chart.

The decimated I and Q components can be observed in the front panel by clicking on the Decimation tab and choosing the appropriate radio button.

Symbol to Bits Mapping

The in-phase (I) and quadrature-phase (Q) waveform components are mapped back into bits according to the scheme used in the transmitter (see figure A-29). The received symbols can be observed from the Constellation Diagram chart in the control panel. This chart displays the received 4 QAM symbols in the complex I/Q plane after the decimation process.

Differential Decoding

Differential decoding is applied to recover the frame bits. The received sequence bits are decoded using a Boolean XOR operation bitwise, according to the following equation

$$x_n = y_n \oplus y_{n-1} \quad (\text{A-4})$$

where:

- y_n ... encoded received sequence
- x_n ... output sequence (frame bits)
- n ... sequence index

Frame Synchronization and Alignment

When the modulated signal arrives at the receiver, the corresponding start of frame (SOF) bits can be anywhere inside the receive buffer. It is necessary to identify the start of the frame (SOF) bits inside the received bit sequence and manipulate two consecutive bit sequences in order to obtain the entire frame. See figure A-31 for reference.

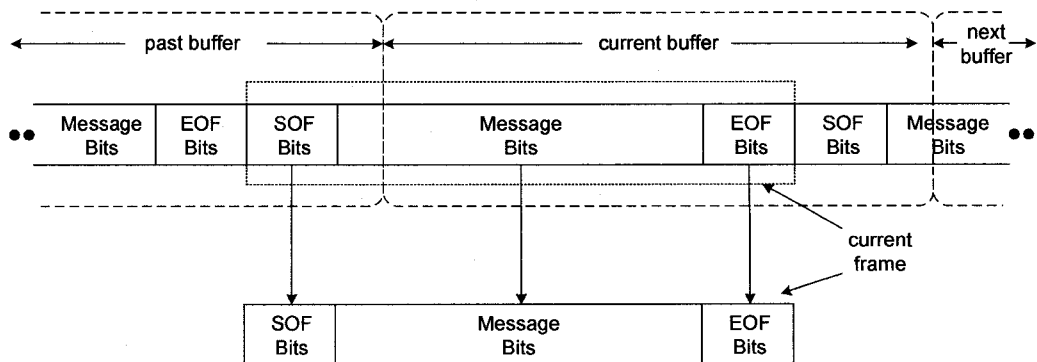


Figure A-31 – Frame synchronization

The SOF bits index indicates the position of the first SOF bit in the buffer. This index is determined by correlating the known SOF bit pattern with the received bit sequence.

The received frame can be observed from the Received Frame tab in the front panel of the 4 QAM SDR Transceiver.vi. The SOF bit pattern index can also be observed from the SOF Index tab. This can be useful to observe how the receiver sound card sample rate differs from the one at the transmitter, as explored by experiment procedure 2-14.

Message Retrieving

In the final stage of the system the message is separated from the SOF and EOF bits. The received message is then compared with a copy of the original message. A message error is flagged if there is an error in any received message bit. The Received Message tab in the receiver's front panel displays the received and original messages, according to the radio button selection. The message error is shown in the Message Error tab.

2. Procedure

- 2.1. Start LabVIEW on the transmitter computer and open the 4 QAM SDR Transceiver.vi.
- 2.2. Click on the TX tab and run the VI using the default values.
- 2.3. Observe the waveforms on the charts by clicking on the different tabs.
- 2.4. Increase the carrier frequency value (IF) to 13 kHz. How does the frequency spectrum of the output signal change? What should be the maximum value for the carrier frequency considering that the VI is working with 2000 symbols per second and 16 samples per symbol? Check your result experimentally.
- 2.5. Set the carrier frequency to its default value (10 kHz). Run the VI and observe the frequency spectrum of the output signal. Increase the symbol rate to 4000 samples per symbol. What difference can you observe in the frequency spectrum? Explain. Include the FFT chart of the two observations in your report.

Note: to include a copy of a chart in your report, click on the pause button (II) to pause the VI execution, right-click on the chart, and choose Export Simplified Image. In the Export Simplified Image window, choose

Enhanced Metafile (.emf) and Export to clipboard. Click in Export and paste the image in your word processor document.

- 2.6. Considering that the sound card sample rate is 44100 Hz, what is the maximum symbol rate supported by the transmitter when the samples per symbol is 16?
- 2.7. Connect the sound card output of the transmitter computer to the sound card input of the receiver computer with an audio cable. Adjust sound level on both computers to 30% of maximum.
- 2.8. Start LabVIEW on the receiver computer and open the 4 QAM SDR Transceiver.vi.
- 2.9. Select the TX tab on the control panel of the transmitter and the RX tab on the receiver.
- 2.10. Using the default values, run the VI on the receiver first. Observe that the receiver processes only the input signal graph while the signal is weak (no energy signal) by clicking on the chart tabs.
- 2.11. Run the VI on the transmitter using the default values. Observe how the receiver detects the incoming signal and starts the demodulation process.
- 2.12. Observe the waveforms at different stages of the receiver.
- 2.13. Verify the presence of aliasing in the frequency spectrum. Stop the transmitter and the receiver. Change the symbol rate to 1700 symbols per second for transmitter and receiver. Run the receiver first and then the transmitter. How does the presence of aliasing change in the frequency spectrum of the receiver? What should be the lowest symbol rate in order to avoid aliasing in the receiver considering the transmission parameters of 16 samples per symbol, 10 kHz carrier frequency, and 44.1 kHz sound card sample rate? Show the FFT chart for each case on your report.
- 2.14. The sound cards of the transmitter and the receiver computers should be set at the same 44100 kHz sample rate. However, you may detect a small offset between the two sample frequencies by observing the SOF (start of frame) index. Reinitialize to the default values and run both VIs (transmitter and receiver). Observe how the SOF index varies at the receiver. Calculate the sample frequency offset between the two computers based on the SOF index chart. Include the SOF Index chart on your report.

LAB #3 - Pulse Shaping

Abstract:

The need to operate over band-limited channels while reducing intersymbol interference (ISI) makes the use of pulse shaping a requirement in communication systems. In this laboratory session, we will investigate different types of pulse shaping filters in the time and frequency domains to understand how they affect the frequency spectrum of the modulated signal, its eye diagram, and the consequences in the demodulation process in the receiver.

References:

- [1] C. Richard Johnson Jr., William A. Sethares, "Telecommunication Breakdown," Pearson Prentice Hall, Upper Saddle River, New Jersey, 2004
- [2] National Instruments, "Pulse Shape Filtering in Communication Systems," available online at <http://zone.ni.com/devzone/cda/tut/p/id/3876>
- [3] National Instruments, "Pulse Shaping to Improve Spectral Efficiency," available online at <http://zone.ni.com/devzone/cda/ph/p/id/200>
- [4] LabVIEW Graphical Programming Course, available online at <http://cnx.org/content/col10241/latest/>
- [5] Jeffrey Travis, Jim Kring, "LabVIEW for Everyone: Graphical Programming Made Easy and Fun," Prentice Hall PTR, New Jersey, 2006

Required Equipment:

1. (2) Personal computer with sound cards and LabVIEW software installed.
2. (1) Audio cable.

1. Background Information

During the transmission process, the digital message needs to be converted into an analog signal. The pulse shaping filter converts each digital symbol into a corresponding analog pulse by up-sampling the original signal into a shape that helps to overcome the problem of intersymbol interference (ISI), thus limiting the signal bandwidth, and improving the signal to noise ratio (SNR) at the receiver. Different types of filters can be applied, where each type will determine specific characteristics of the transmitted signal in the time and frequency domains.

1.1. The Task of the Pulse Shaping Filter

Consider an analog signal $w_a(t)$ passing through a filter with impulse response $p(t)$. In the time domain, the output of the filter is given by the convolution

$$x(t) = w_a(t) * p(t) \quad (\text{A-5})$$

Considering $W_a(f)$ and $P(f)$ as the Fourier transform of $w_a(t)$ and $p(t)$, respectively, the Fourier transform of the output signal is given by the product

$$X(f) = W_a(f) P(f) \quad (\text{A-6})$$

By looking at this last equation, it can be observed that the filter response scales the output signal and limits its bandwidth ($X(f)$ is zero at frequencies where $P(f)$ is zero).

When choosing a particular pulse shaping, the tradeoff between the advantages and disadvantages in the time and frequency domains must be considered. For example, a rectangular pulse in the time domain is the most compact form of representing the signal. However, in the frequency domain, the corresponding frequency response is a sinc function, which has infinite bandwidth. On the other hand, the sinc pulse in the time domain has a corresponding rectangular form in the frequency spectrum, which limits the bandwidth of the incoming signal. Figure A-32 shows some pulse shape examples in time and frequency domains.

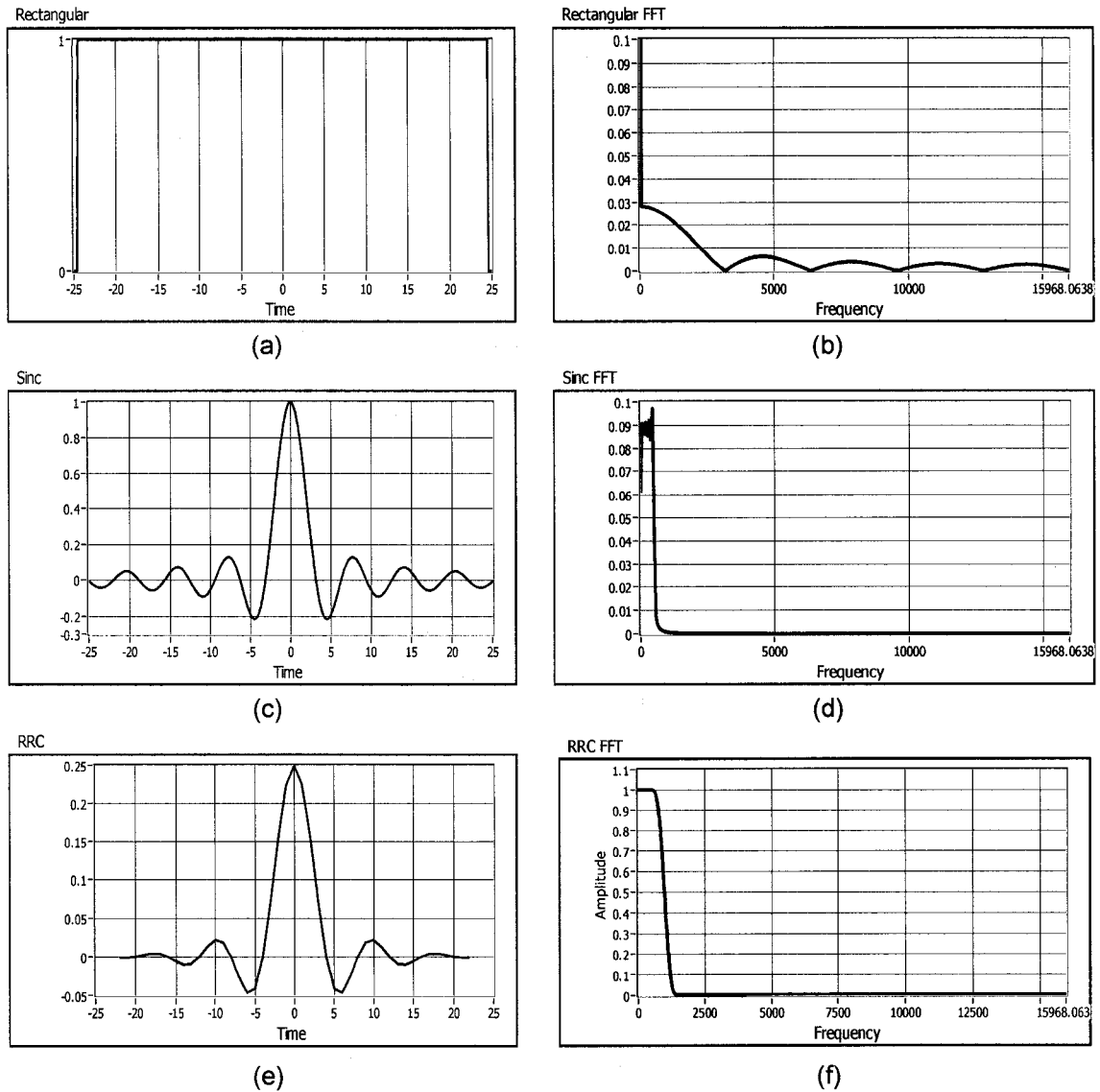


Figure A-32 – Some pulse shapes in the time and frequency domains

1.2. Intersymbol Interference

Intersymbol interference (ISI) is a phenomenon that occurs when adjacent symbols interfere with each other. It can occur when the pulse shape is wider than a symbol interval and when the channel causes distortions on neighboring pulses. Pulse shapes that are wider in time occupy narrower bandwidth. However, as the pulse shape becomes wider, more ISI will degrade the signal.

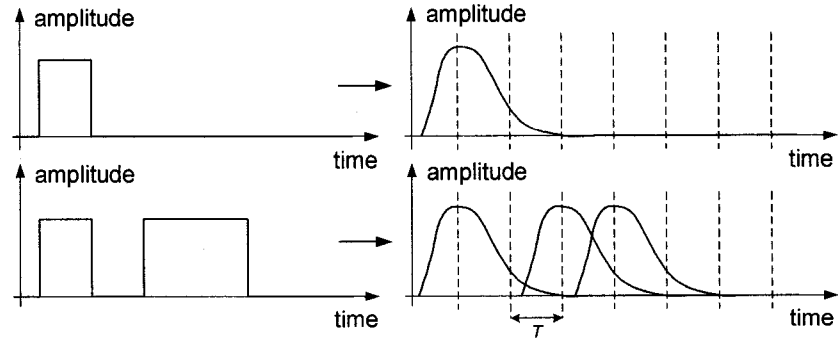


Figure A-33 – Intersymbol interference (ISI)

A pulse shape with a longer duration in time and satisfying the Nyquist condition, where it must be zero at all integer multiples of the symbol period T but one, provides a narrow bandwidth and avoids ISI. A pulse $h(t)$ is said to be a Nyquist pulse if it obeys the following equation:

$$h(kT + \tau) = \begin{cases} c & k = 0 \\ 0 & k \neq 0 \end{cases} \quad (\text{A-7})$$

for some τ , where k is an integer and c is a constant. Raised cosine (RC) and root raised cosine (RRC) pulse shapes are commonly used in communication systems because they have the properties of zero crossings at desired times, sloped band edges in the frequency domain, and a considerable fast decay in the time domain while preserving a narrow bandwidth.

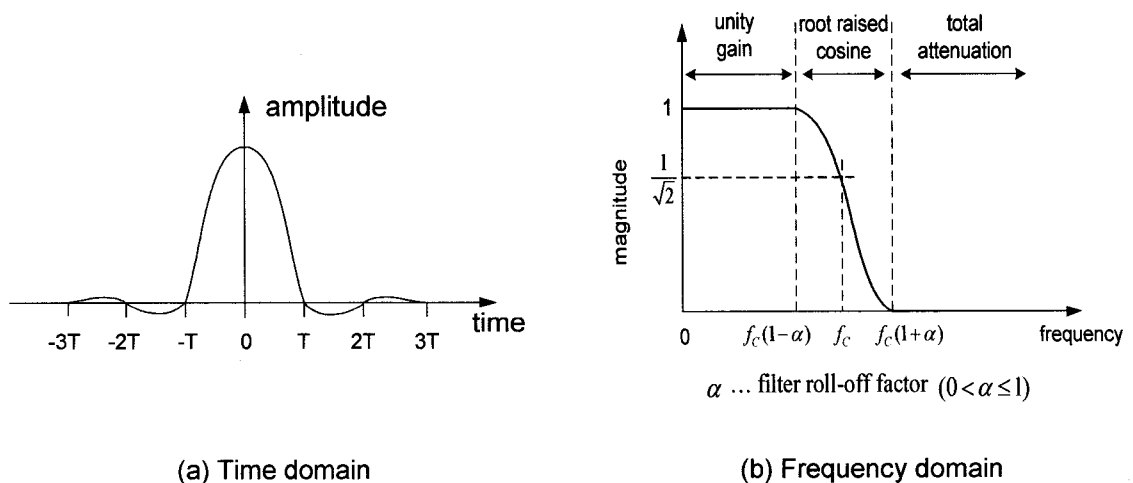


Figure A-34 – Ideal root raised cosine response

1.3. Pulse Shaping and the Eye Diagram

The pulse shaping operation involves the convolution of the incoming signal with the pulse shape. An example using the root raised cosine pulse shape is shown by figure A-35.

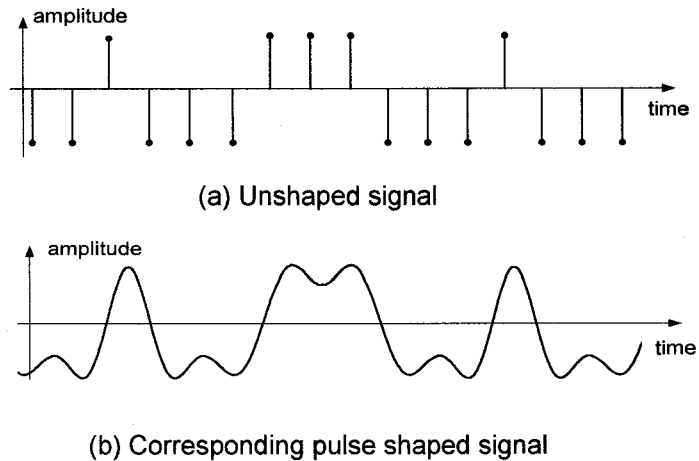


Figure A-35 – Pulse shaping example

The eye diagram is a visualization tool used for inspecting the pulse shaped signal. Several traces of the waveform are superimposed in the same picture, starting at integer multiples of the symbol time. Ideally, the original message should be recovered by sampling the pulse shaped signal exactly in the middle of each symbol. In this diagram, this point is said to be in the center of the eye. As the sample point deviates from the center of the eye, the recovery process becomes more susceptible to errors. Therefore, the eye diagram illustrates the limitation for sampling time errors. Also, if the signal distortion is extremely severe, the diagram will have the eye closed, showing that it is impossible to recover the original message. Refer to figure A-36 for the eye diagram details.

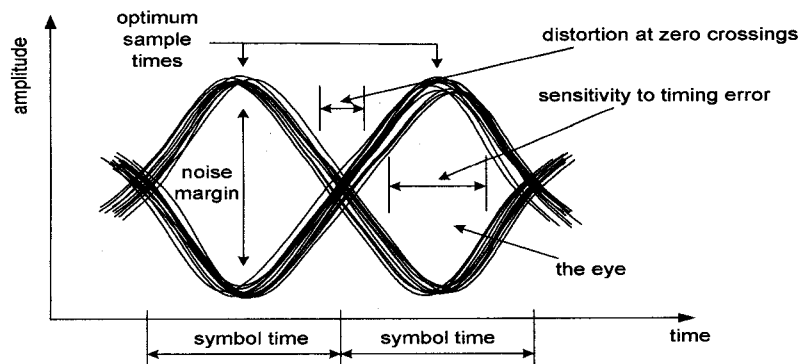


Figure A-36 – Eye diagram

1.4. Matched Filter

A receive filter is necessary in the demodulation process in order to decode the message contained in the transmitted signal. Since the transmitted signal is likely to arrive at the destination with some amount of noise, besides helping to avoid ISI, the receive filter must also help to improve the signal to noise ratio (SNR) for a better performance. In order to do this, the convolution of the pulse shape with the impulse response of the receive filter must be a Nyquist pulse. For this reason, the receive filter is called a matched filter and it has an impulse response that is a scaled, time-reversed form of the pulse shape used in the transmitter.

2. Procedure

- 2.1. Start LabVIEW and open the 4 QAM SDR Transceiver.vi on the transmitter computer.
- 2.2. Using the default values on the transmitter, select the TX tab and run the VI to observe the eye diagram for each pulse shaping scheme: Root Raised Cosine, Raised Cosine, and none (Rectangular). Print and comment on the shape for each pulse shaping filter. Check the symbol period.
- 2.3. Observe the frequency spectrum of the transmitted signal for each pulse shape. Print and comment on the shape for each case.
- 2.4. Stop the transmitter and connect the sound card output of the transmitter computer to the sound card input of the receiver computer with an audio cable. Start LabVIEW and open the 4 QAM SDR Transceiver.vi on the receiver computer. Adjust the sound level on both computers to 30% of maximum. Select the TX tab on the control panel of the transmitter and the RX tab on the receiver.
- 2.5. For each pulse shaping scheme, run the VIs on both transmitter and receiver computers using the default values and observe the eye diagram at the receiver. Print and comment on the shape for each case.
- 2.6. Add white noise to the transmitted signal by clicking on the Apply Noise switch and set the SNR (dB) value to 10 in the transmitter's control panel. Repeat step 2.5 using the addition of noise to the transmitted signal and verify the effects of each pulse shape filter on the eye diagram and

received message. Print and comment on the shape for each case. Which pulse shape is better? Explain.

- 2.7. Repeat step 2.5 with no noise added to the signal at the transmitter. Adjust the filter parameter (roll-off factor) to 0.1 on both computers. Run the VIs on transmitter and receiver computers and observe the eye diagram on the receiver. Observe again the eye diagram at the receiver for a roll-off factor of 0.9 (set this value on both computers). Print and comment on the shape.

3. Questions

- 3.1. Explain the reasons to apply pulse shaping in a communication system.
- 3.2. Why is it desired to have limited bandwidth signals in a communication system?
- 3.3. Explain the reasons to have a matched filter at the receiver.

LAB #4 - Carrier Recovery

Abstract:

Carrier recovery is a crucial stage in the demodulation process in a communication system. The signal that arrives at the receiver is a complicated waveform that must be down-converted into its baseband form. In order to do this, it is necessary to determine the frequency and phase of the modulating sinusoid. Errors in the estimation of these parameters can cause errors in the received demodulated message. In this laboratory, we will examine the Costas Loop for 4-QAM technique used in the carrier recovery process.

References:

- [1] C. Richard Johnson Jr., William A. Sethares, "Telecommunication Breakdown," Pearson Prentice Hall, Upper Saddle River, New Jersey, 2004
- [2] C. Richard Johnson Jr., "A Digital Quadrature Amplitude Modulation (QAM) Radio" available on CD-ROM accompanying Johnson and Sethares, "Telecommunication Breakdown"
- [3] LabVIEW Graphical Programming Course, available online at <http://cnx.org/content/col10241/latest/>
- [4] Jeffrey Travis, Jim Kring, "LabVIEW for Everyone: Graphical Programming Made Easy and Fun," Prentice Hall PTR, New Jersey, 2006

Required Equipment:

1. (2) Personal computer with sound cards and LabVIEW software installed.
2. (1) Audio cable.

1. Background Information

During the transmission process, the baseband signal is up-converted into a RF waveform by using a sinusoidal carrier. It is necessary to estimate the frequency and phase of the carrier when the waveform arrives at the receiver to get back the original signal in its baseband form.

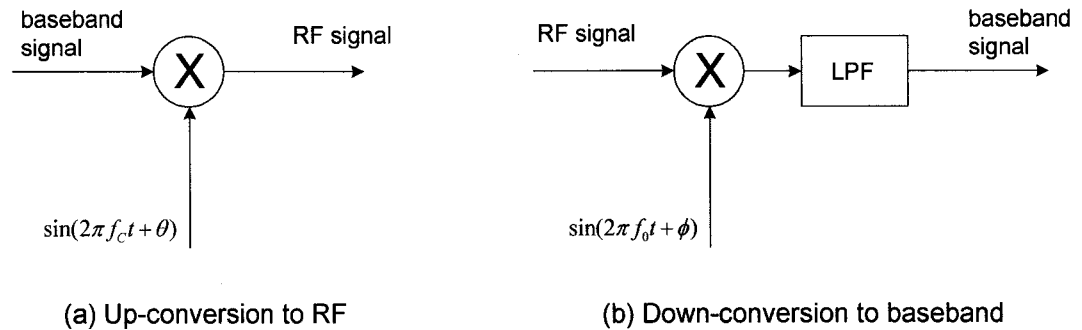


Figure A-37 – Up- and down-conversion

Ideally, the receiver should know the carrier frequency in advance and use this information to synchronize the carrier. However, small frequency offsets are likely to occur and this makes necessary the use of techniques for finding the carrier frequency and phase estimates. There are different techniques that can be used in the carrier recovery process, each one with different requirements and performance depending on the modulation scheme. In this laboratory the Costas Loop for 4 QAM scheme will be explored.

1.1. Up-Conversion to RF and Down Conversion to Baseband

In quadrature modulation, the transmitter converts the baseband signal into an RF signal by mixing a cosine with the desired carrier frequency f_c and some phase θ with the in-phase baseband channel and a sine with the same frequency f_c and phase θ with the quadrature-phase baseband channel. See figure A-38 for reference.

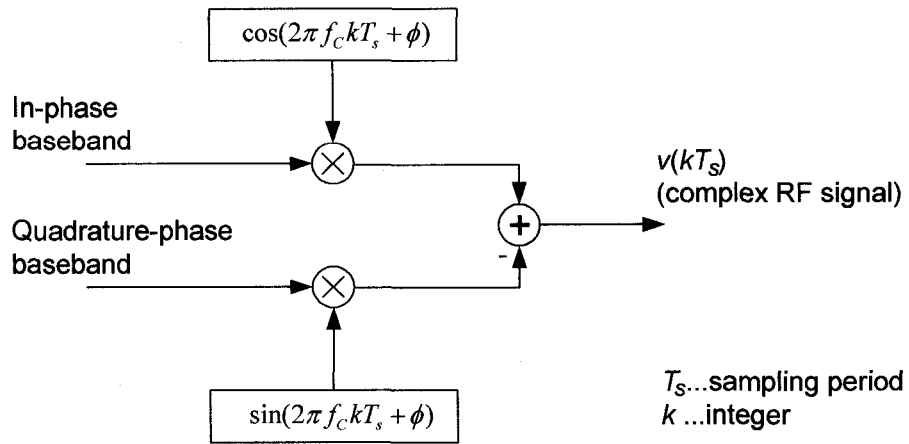


Figure A-38 – Quadrature up-conversion to RF

After recovering the carrier at the receiver, the incoming RF signal is down-converted into its baseband form by mixing it with a locally generated cosine with a frequency f_0 and phase ϕ and is passed through a low pass filter with cutoff frequency approximately equal to f_0 (f_0 is the estimated frequency of the carrier by the receiver) to provide the in-phase baseband signal. Similarly, the quadrature-phase baseband signal is obtained by mixing the incoming RF signal with a locally generated sine of same frequency f_0 and phase ϕ , and the mixer output is passed through a low pass filter with the same cutoff frequency f_0 and then multiplying by -1. The block diagram of the carrier recovery and down-conversion stage is depicted by figure A-39.

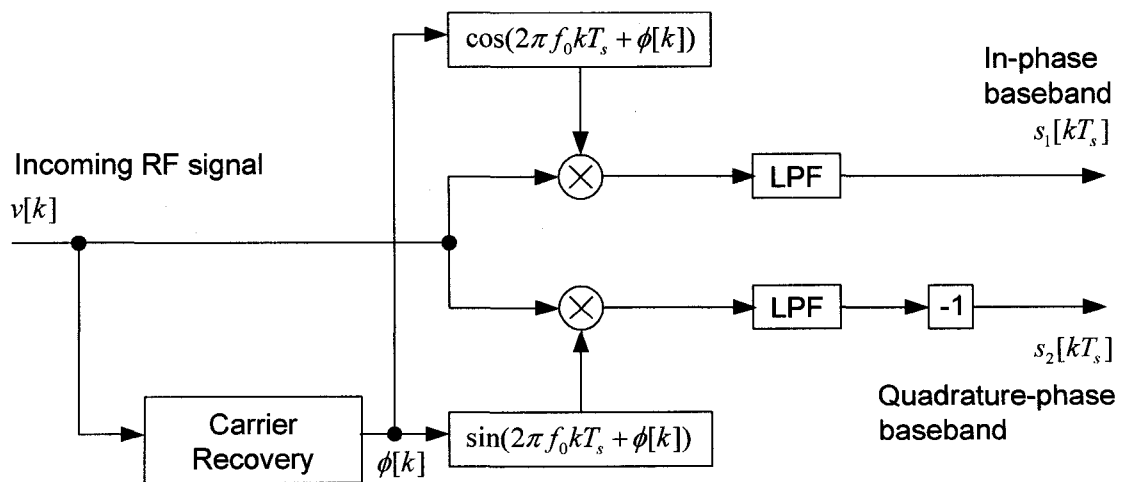


Figure A-39 – Carrier recovery and down-conversion

1.2. The Costas Loop for 4 QAM

The Costas Loop is an adaptation algorithm widely used in communication systems. It operates directly on the received signal and uses the “hill climbing” adaptive method to maximize the cost function J_C , which for our 4 QAM scheme is

$$J_C = \cos^2(2(\theta - \phi)) \quad (\text{A-8})$$

where θ is the actual phase of the carrier and ϕ is the estimated one. The phase estimate algorithm searches for values of ϕ that approach θ (for maximizing J_C). It starts with some initial value for ϕ and follows a path defined by the gradient of the cost function J_C evaluated at the current point using an adaptation constant μ (μ has a positive value). The adaptation constant μ determines the convergence of the algorithm described by equation A-9:

$$\phi[k+1] = \phi[k] + \mu \left. \frac{\partial J_C}{\partial \phi} \right|_{\phi=\phi[k]} \quad (\text{A-9})$$

In order to implement this algorithm, we need to determine the derivative $\frac{\partial J_C}{\partial \phi}$ evaluated at $\phi = \phi[k]$. With some algebra and trigonometry manipulations it can be shown (see [4]) that this derivative can be implemented for the 4 QAM scheme by defining the four signals:

$$x_1(t) = LPF\{v(t) \cos(2\pi f_0 t + \phi)\} \quad (\text{A-10})$$

$$x_2(t) = LPF\{v(t) \cos(2\pi f_0 t + \phi + \pi/4)\} \quad (\text{A-11})$$

$$x_3(t) = LPF\{v(t) \cos(2\pi f_0 t + \phi + \pi/2)\} \quad (\text{A-12})$$

$$x_4(t) = LPF\{v(t) \cos(2\pi f_0 t + \phi + 3\pi/4)\} \quad (\text{A-13})$$

and taking the average of their product ($v(t)$ is the received RF signal and f_0 is the carrier frequency estimate). The 4-QAM Costas Loop algorithm becomes:

$$\phi[k+1] = \phi[k] + \mu x_1(t) x_2(t) x_3(t) x_4(t) \Big|_{t=kT_s, \phi=\phi[k]} \quad (\text{A-14})$$

where T_s is the sampling period of the signal. The block diagram for the 4 QAM Costas Loop is shown in figure A-40.

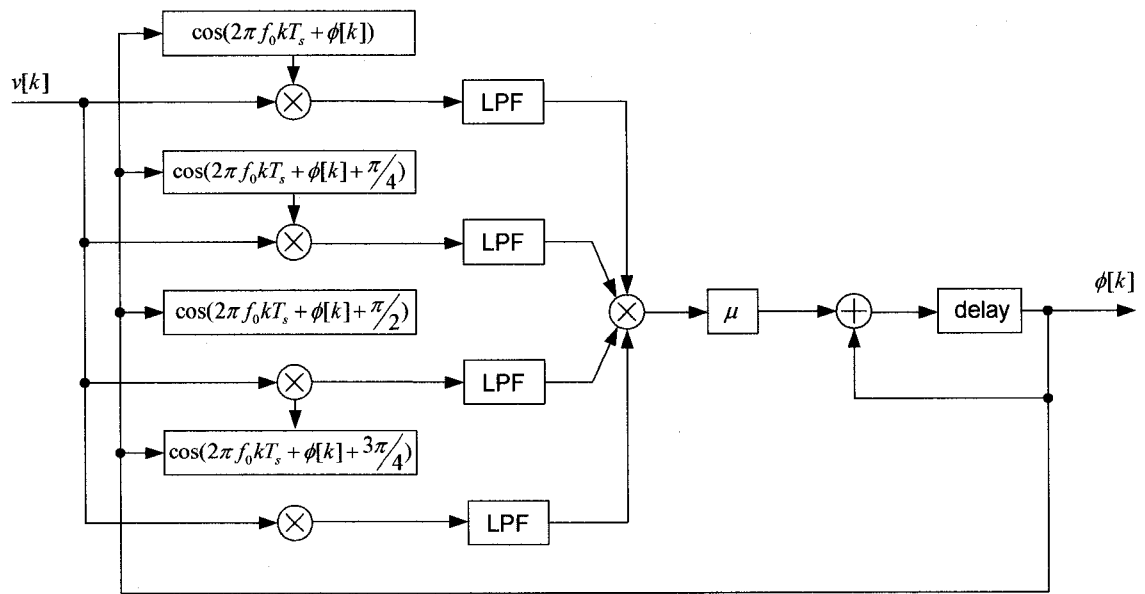


Figure A-40 – 4 QAM Costas Loop

This scheme will provide a phase estimate ϕ that will converge to the phase of the carrier plus an integer multiple of $\pi/2$ radians. In the presence of a frequency offset between the carrier frequency and the local oscillator at the receiver, the phase update will not converge to a constant value. Instead, it will converge to a line with slope m which represents the frequency offset, as shown in figure A-41.

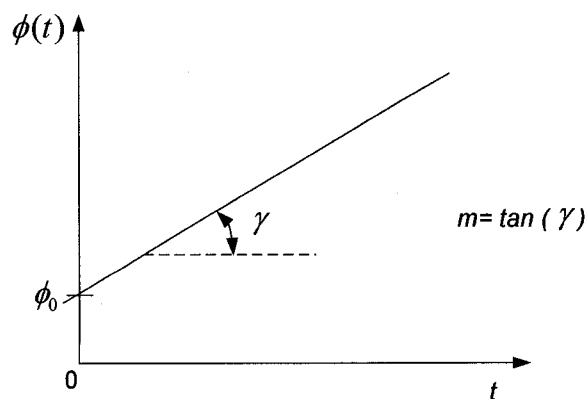


Figure A-41 – Phase update in the presence of frequency offset

The phase update is given by the following equation:

$$\phi(t) = mt + \phi_0 \quad (\text{A-15})$$

Comparing equation A-15 with equation A-16:

$$\phi(t) = 2\pi f_{\text{offset}} t + \phi_0 \quad (\text{A-16})$$

gives us:

$$f_{\text{offset}} = \frac{\tan(\gamma)}{2\pi} \quad (\text{A-17})$$

The phase ambiguity issue is discussed next.

1.3. Phase Ambiguity Resolution

The phase ambiguity can be resolved by:

- a. correlating the down-converter output with a known training sequence, or
- b. using a differential encoding technique, or
- c. using a trained equalizer.

The 4 QAM SDR Transceiver.vi uses differential encoding and bit pattern correlation to eliminate the phase ambiguity problem. The in-phase and quadrature-phase channels are encoded separately in the transmitter and decoded separately in the receiver. This takes care of phase ambiguities of π radians. Phase ambiguities of $\pi/2$ and $3\pi/2$ radians are resolved by checking the start and end of frame patterns. If an error is detected, the phase is changed by a $\pi/2$ rotation.

2. Procedure

- 2.1. Connect the sound card output of the transmitter computer to the sound card input of the receiver computer with an audio cable. Start LabVIEW and open the 4 QAM SDR Transceiver.vi on both computers. Adjust the sound level on both computers to 30% of maximum. Select the *TX* tab on the control panel of the transmitter and the *RX* tab on the receiver.

- 2.2. Run the transmitter using the default values. Run the receiver using the default values and verify that there is no error in the received message (click on the Message Error tab).
- 2.3. Observe the Phase Update graph. Does it have a constant value over time? Print the graph and explain the values on it.
- 2.4. Stop the receiver and change the IF value to 9,995 Hz. Run the receiver and verify the Message Error tab. Click on the Phase Update graph tab and observe how its values change over time. Print the two graphs (Message Error and Phase Update) and comment on the phase update values and on the message error.
- 2.5. Repeat step 2.4 while changing the IF value to 9,990 Hz. Are there any errors in the received sequence? Why? Print the two graphs and explain.
- 2.6. Repeat step 2.5 (IF value set to 9,990 Hz) and change the Costas Loop adaptation constant μ to 20. Check if there are errors in the received sequence. Print the two graphs and explain.
- 2.7. Repeat step 2.6 (IF set to 9,990 Hz and μ set to 20) while adding noise to the transmitted signal by switching the Apply Noise switch to ON and setting the SNR (dB) control to 10 in the transmitter's control panel. Run the VI and check for occurrence of errors in the received sequence.
- 2.8. Repeat step 2.7 with μ set to 30 in the receiver. Can you observe a different result for the message errors? Explain.
- 2.9. Determine experimentally the maximum frequency offset at which the system can still work properly (no error in received message) by varying the IF value and the adaptation constant μ in the receiver.

3. Questions

- 3.1. Explain the role of the adaptation constant μ in the Costas Loop. What does happen if it is too large? What does happen if it is too small?
- 3.2. Will the phase always converge to an "optimal value" in the Costas Loop algorithm? Explain.
- 3.3. How does noise interfere with the Costas Loop algorithm?
- 3.4. Why does the constellation diagram at the receiver rotate as the frequency offset increases?

- 3.5. Find the frequency offset value experimentally for a given setup using figure A-41 and equation A-17. Print and comment on the phase update graph.

LAB #5 - Symbol Timing Recovery

Abstract:

After the carrier recovery and down-conversion to baseband stage, the received signal passes through a matched filter in order to get the best estimate of the transmitted symbols. The original bits which were up-sampled in the transmission process now need to be down-sampled in order to recover the original bit sequence. In order to do this, the receiver has to sample the incoming signal after the matched filter at "optimal times". This is the role of the symbol timing recovery stage in the receiver. There are different techniques which can be used in this process. In this laboratory experiment, we will focus on the timing recovery process using the delay-locked loop (DLL) and the max-eye algorithms.

References:

- [1] C. Richard Johnson Jr., William A. Sethares, "Telecommunication Breakdown," Pearson Prentice Hall, Upper Saddle River, New Jersey, 2004
- [2] LabVIEW Graphical Programming Course, available online at <http://cnx.org/content/col10241/latest/>
- [3] Jeffrey Travis, Jim Kring, "LabVIEW for Everyone: Graphical Programming Made Easy and Fun," Prentice Hall PTR, New Jersey, 2006

Required Equipment:

1. (2) Personal computer with sound cards and LabVIEW software installed.
2. (1) Audio cable.

1. Background Information

After down-converting the quadrature modulated signal into its baseband form and passing through a matched filter, the signal contains the up-sampled complex symbols. It is necessary to recover the symbol timing clock to take one sample from each symbol interval at optimal times. Since the transmitted signal may suffer from noise interference, the techniques to recover the symbol timing must take this problem into account.

1.1. The Delay-Locked Loop (DLL) Algorithm

The delay-locked loop (DLL), also called early-late sampling, is a simple technique used to recover the symbol timing. In quadrature modulation, the DLL algorithm works on one of the two components of the quadrature signal to recover the symbol timing. For instance, it takes the in-phase component and tries to sample it at the peaks to ensure better performance when noise is present. The sample obtained at a peak of the incoming signal is said to be an on-time sample. If the sample is not at the peak, then it is said to be a too early or too late sample. Figure A-42 depicts the three possible cases in this scheme while figure A-43 shows the block diagram of the DLL algorithm.

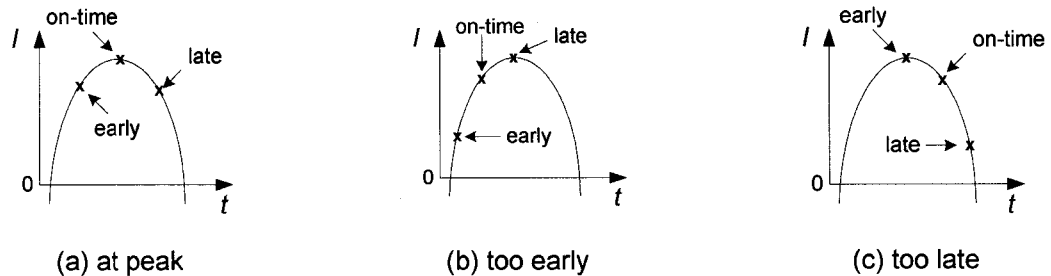


Figure A-42 – DLL sampling

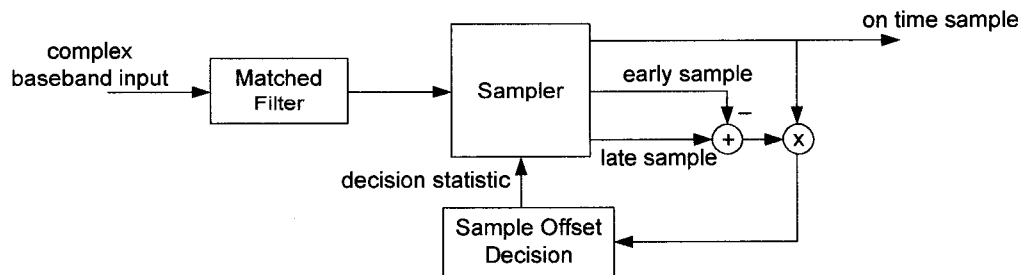


Figure A-43 – Delay-Locked Loop block diagram

This algorithm takes three sequential samples and compares their values to verify which one has the larger value in magnitude by using the decision statistics input at the sample offset decision block. Since the number of samples per symbol N is known by the decision block, it adjusts the next on-time sample index according to the decision statistics input:

- a. If the on-time sample is at the peak, the next on-time sample will be at the next N sample index.
- b. If the on-time sample is too early, the next on-time sample will be at the next $N+1$ sample index.
- c. If the on-time sample is too late, the next on-time sample will be at the next $N-1$ sample index.

In the presence of noise, instead of using only the decision statistics as the parameter for updating the on-time sample, a more reliable approach is taken. For each DLL iteration, the offset value from the decision statistics input is added to the previous one. The next on-time sample is taken every other N sample index until the sum exceeds a threshold value. After that, the next on-time sample takes into account the value of the sum: if it is negative, the next on-time sample is at the next $N-1$ sample index; otherwise (if the sum is positive), the next on-time sample is taken at the next $N+1$ sample index.

1.2. The Max Eye Algorithm

The Max Eye algorithm is a technique that calculates the best sampling times by choosing the offset that maximizes the opening in the eye diagram. A weighted average method is applied to find an average timing that maximizes the eye opening for each symbol. In the presence of high noise levels this technique can have issues when the opening of the eye diagram diminishes.

1.3. Implementation of the Delay-Locked Loop and Max-Eye Algorithms

In our experiments, we will be using the *4 QAM SDR Transceiver.vi* and the *4 QAM SDR Transceiver_maxeye.vi*. The first VI uses the DLL algorithm while the second uses the Max-Eye algorithm to recover the symbol timing.

2. Procedure

- 2.1. Connect the sound card output of the transmitter computer to the sound card input of the receiver computer with an audio cable. Start LabVIEW and open the *4 QAM SDR Transceiver.vi* on both computers. Adjust the

sound level on both computers to 30% of maximum. Select the *TX* tab on the control panel of the transmitter and the *RX* tab on the receiver.

- 2.2. Run the transmitter and the receiver using the default values. Click on the Symbol Timing tab at the receiver and observe the symbol clock and the I channel component (pause the execution at the receiver to take a better look at the chart). Observe the decimated I and Q components in the Decimation tab. Is the DLL algorithm doing its job properly? Print the graphs and explain. Verify the *Message Error* tab. Are there any errors? Stop the receiver.
- 2.3. Repeat step 2.2 applying noise to the transmitted signal by setting the SNR (dB) control in the transmitter to 6 dB. Observe the symbol timing and decimation graphs. What difference can you observe from the last step (no noise) on the graphs? Print and comment on the graphs. Observe if there are any message errors.
- 2.4. Using the default values as a starting point, find out experimentally the minimum value for the SNR (dB) control at the transmitter where the receiver can still work without message errors (wait for approximately 2000 iterations).
- 2.5. Using the default values and the value for SNR (dB) you found in step 2.4, decrease the value of the DLL threshold E_0 to 0.01. Run both VIs and verify the Message Error graph. What do you observe? Print the graph and explain. What is the role of the constant E_0 ?
- 2.6. Run the VIs on the transmitter and receiver computers and observe the waveforms at different nodes of the system by clicking on the tabs. Click on the Apply Time Delay? Switch on the BD to insert a two second delay for each buffer processing. Observe the waveforms at different nodes of the system.
- 2.7. Close the 4 QAM SDR Transceiver.vi and open the 4 QAM SDR Transceiver_maxeye.vi in the receiver computer. Repeat step 2.4 to determine the minimum SNR value at the transmitter for which the receiver can still work without any errors (wait for approximately 2000 iterations).
- 2.8. Compare the results from step 2.4 and step 2.7. Are they different?
- 2.9. Repeat step 2.6 using the 4 QAM SDR Transceiver_maxeye.vi.
- 2.10. Verify which of the two receivers have better performance using the default values (use a SNR value lower than the one you found in step 2.4 at the transmitter for both receivers and compare the BER values after

1000 iterations). Print the BER graphs for both receivers. Comment on the results.

3. Questions

- 3.1. Describe the importance of the matched filter in the symbol timing recovery.
- 3.2. Explain why the input of the Sample Offset Decision block is taken from the multiplication of the sum (late-sample – early-sample) by the on-time sample in the DLL algorithm.

APPENDIX B

4 QAM SDR TRANSCEIVER.VI CODE

The main code implementation of the 4 QAM SDR Transceiver.vi is presented in this appendix.

This VI's front panel is divided into transmitter and receiver structures, as well as the block diagram. The block diagram, however, has many more details which are hidden in the case structures. These details are also shown in separated figures.

All of the subVIs code used to implement important stages of the 4 QAM SDR Transceiver.vi were presented in Chapter 5 and are not replicated here. Only their icons and descriptions (as shown by LabVIEW Context Help window) are included.

Codes corresponding to the Apply Noise.vi and Input Scaling and Energy Detection.vi, which are used in the main code of the 4 QAM SDR Transceiver.vi, are also included in this appendix.

The code implementation of the 4 QAM SDR Transceiver_maxeye.vi, included in the experiments of Appendix A, is presented here. Its front panel and other case structures that have the same code of the 4 QAM SDR Transceiver.vi are not shown.

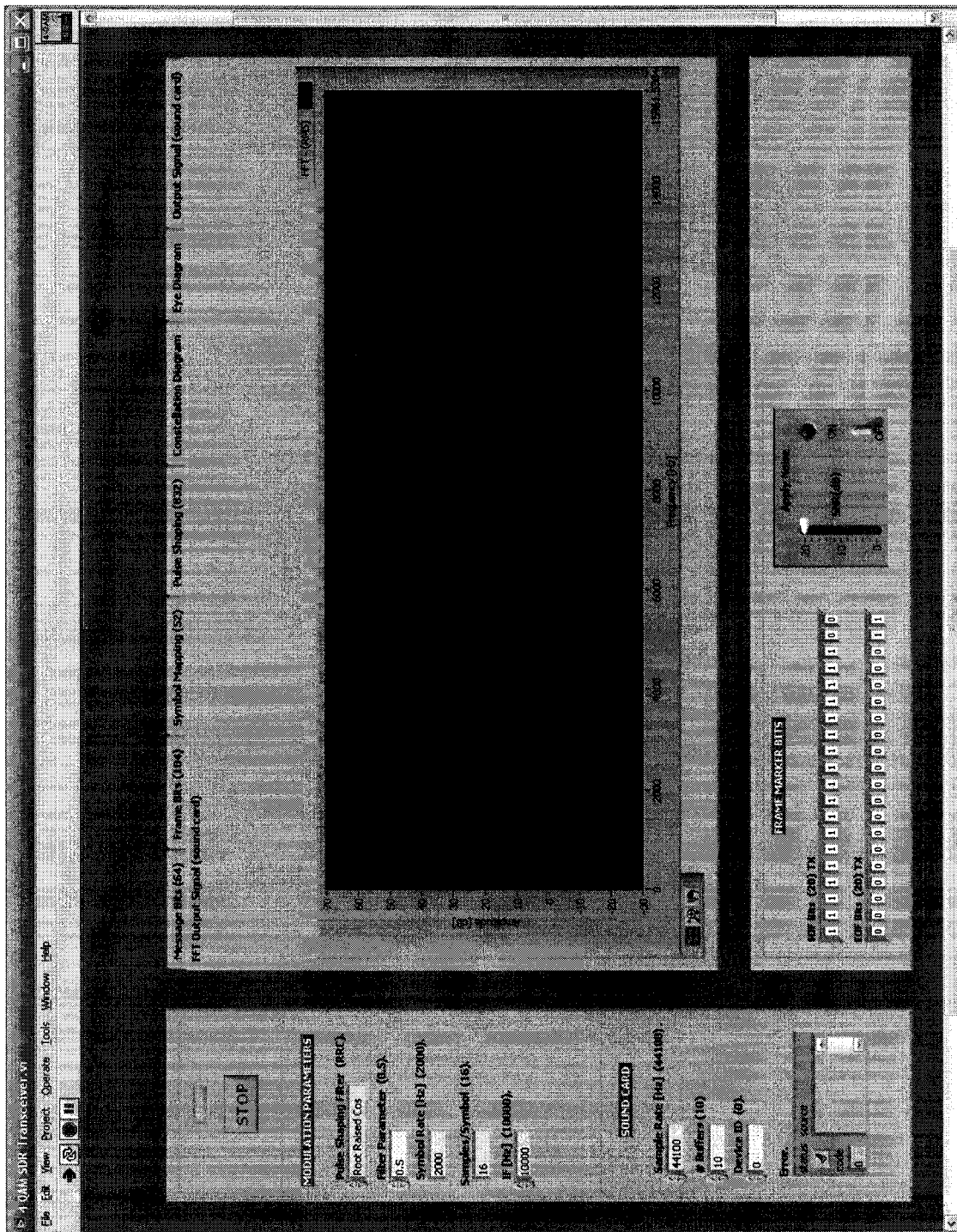


Figure B-1 – 4 QAM SDR Transceiver front panel: transmitter mode

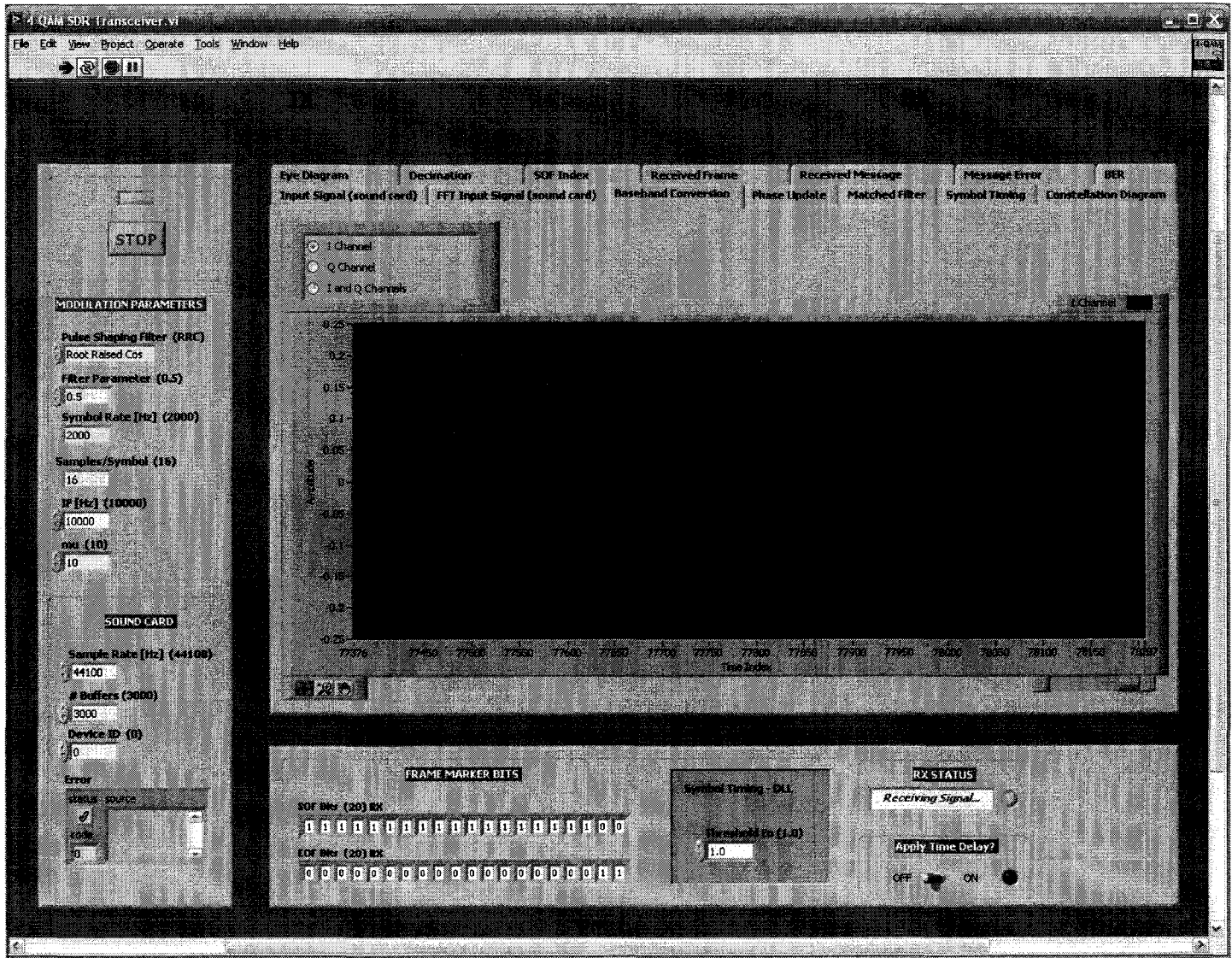


Figure B-2 – 4 QAM SDR Transceiver front panel: receiver mode

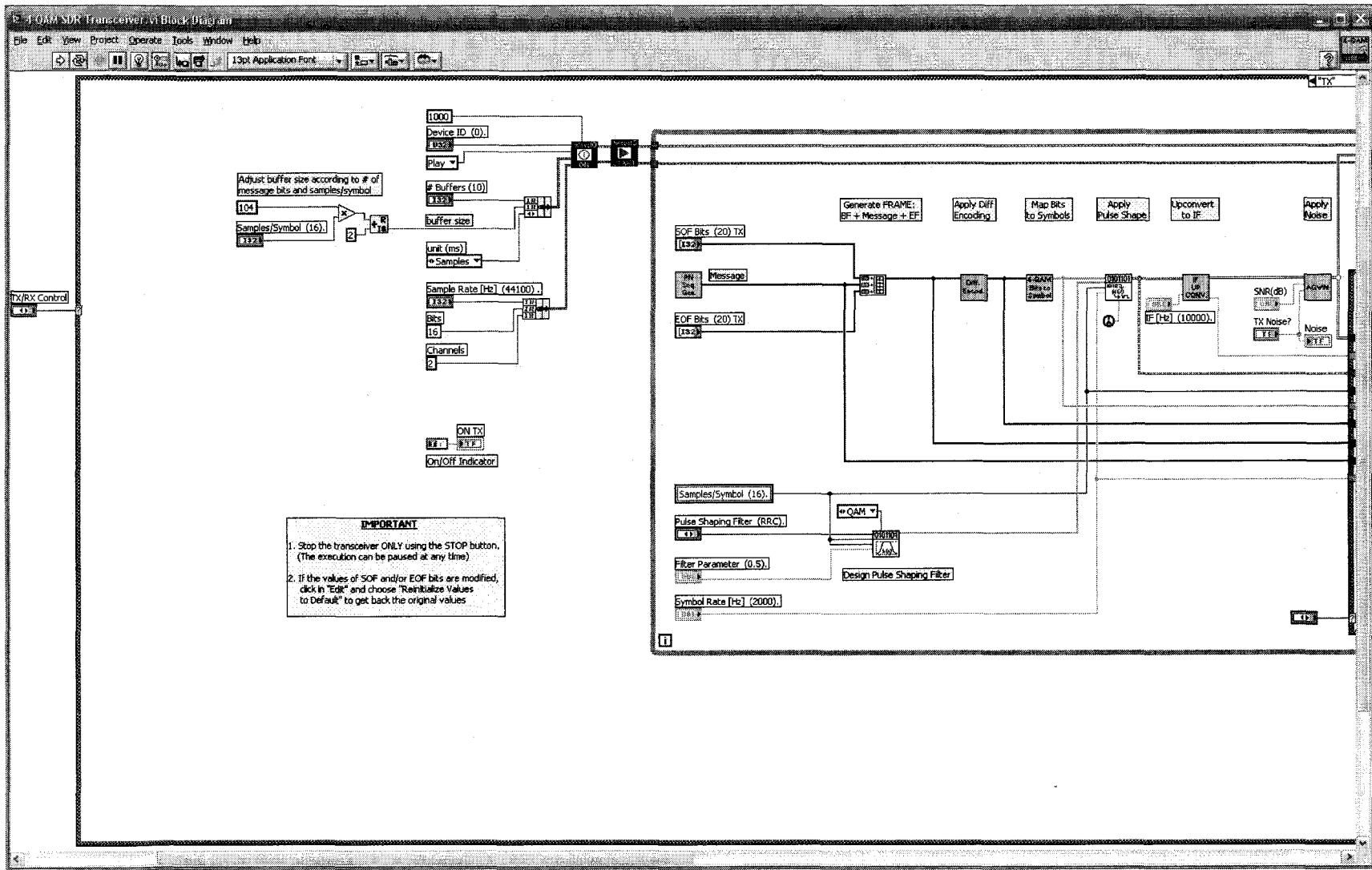


Figure B-3 – 4 QAM SDR Transceiver block diagram, transmitter mode: part 1

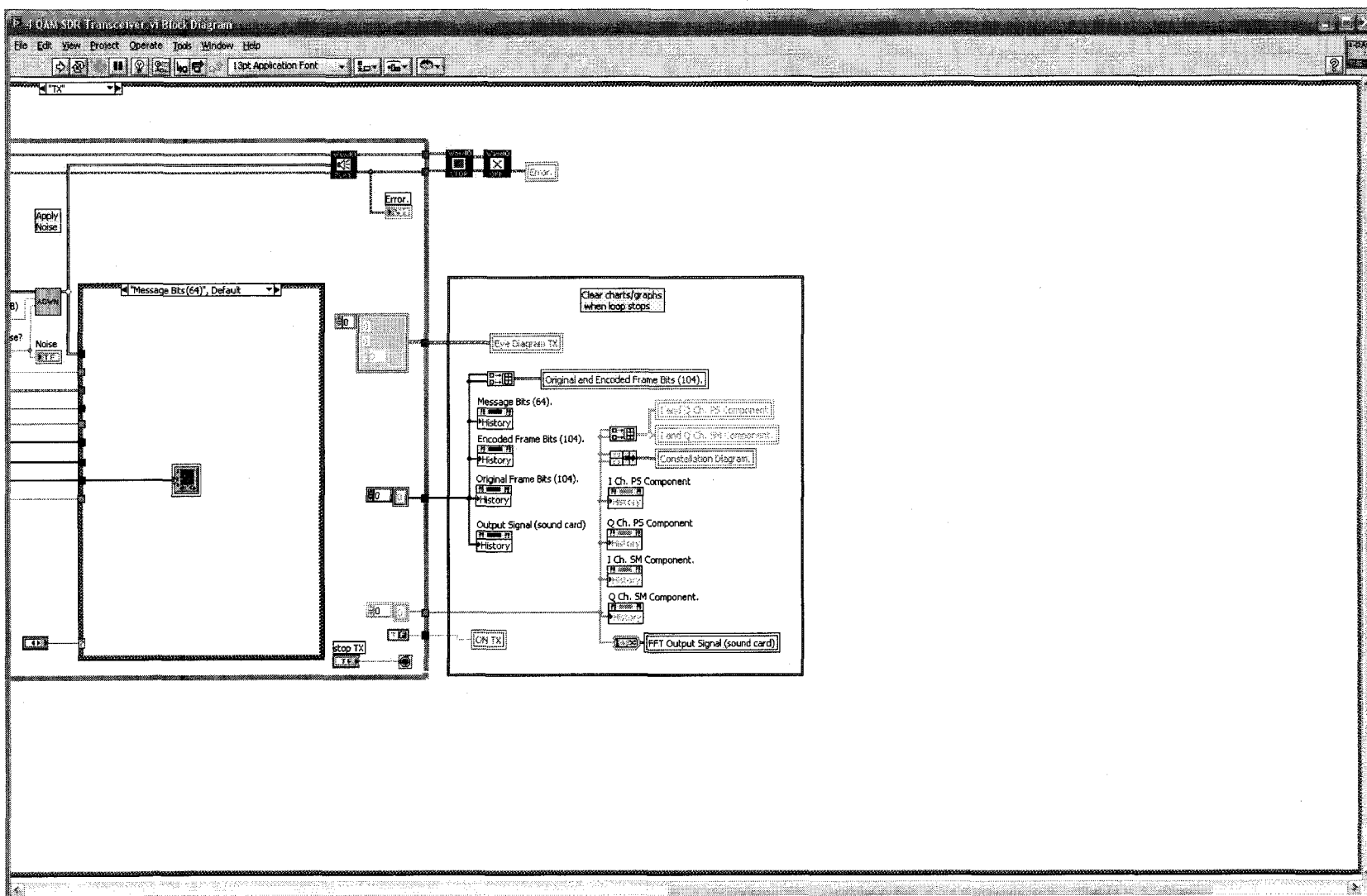
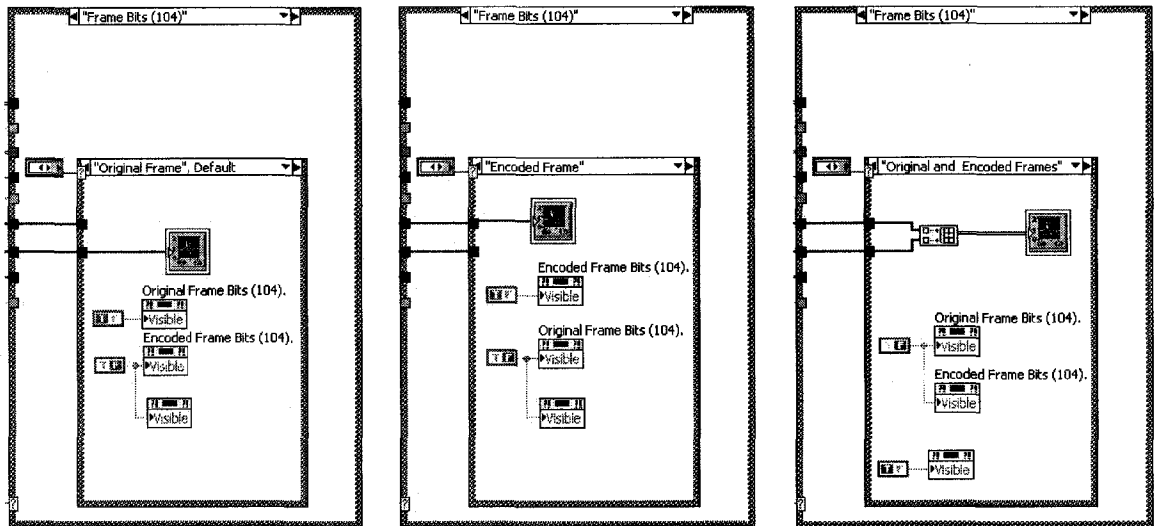


Figure B-4 – 4 QAM SDR Transceiver block diagram, transmitter mode: part 2

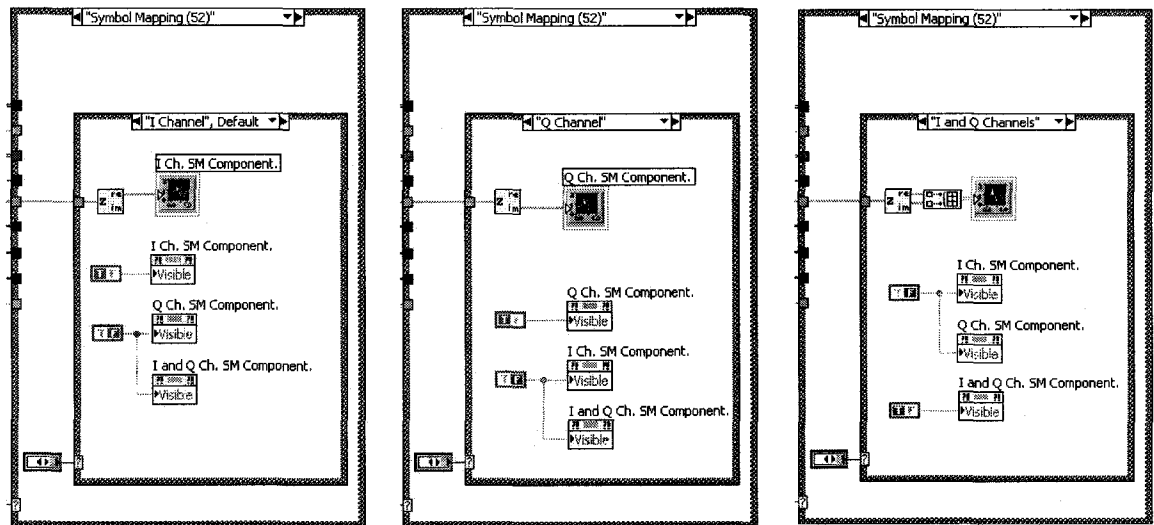


(a) Original Frame

(b) Encode Frame

(c) Original and Encoded Frame

Figure B-5 – 4 QAM SDR Transceiver.vi BD, transmitter mode: Frame Bits case structures

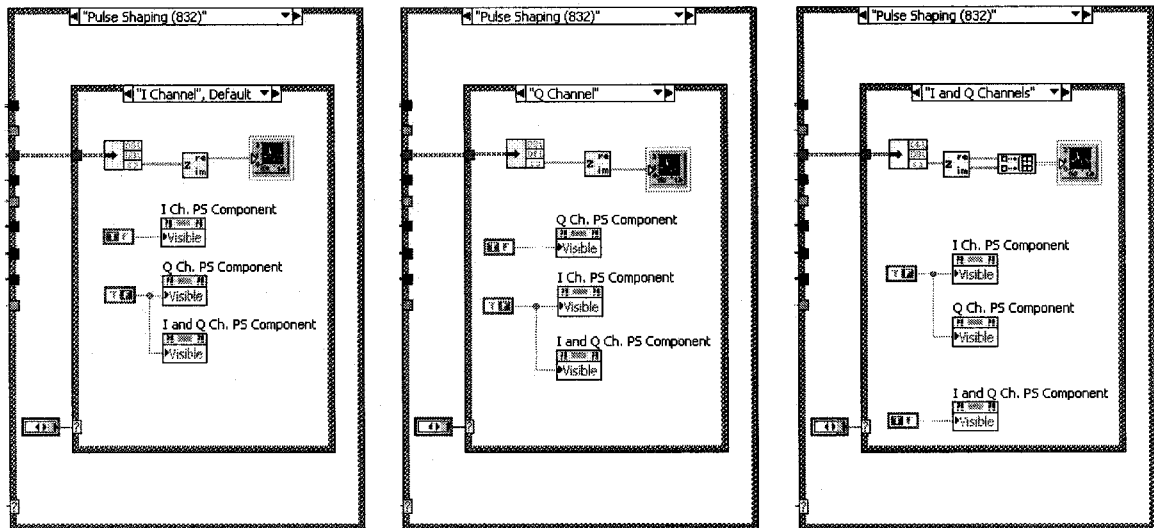


(a) I Channel

(b) Q Channel

(c) I and Q Channels

Figure B-6 – 4 QAM SDR Transceiver.vi BD, transmitter mode: Symbol Mapping case structures

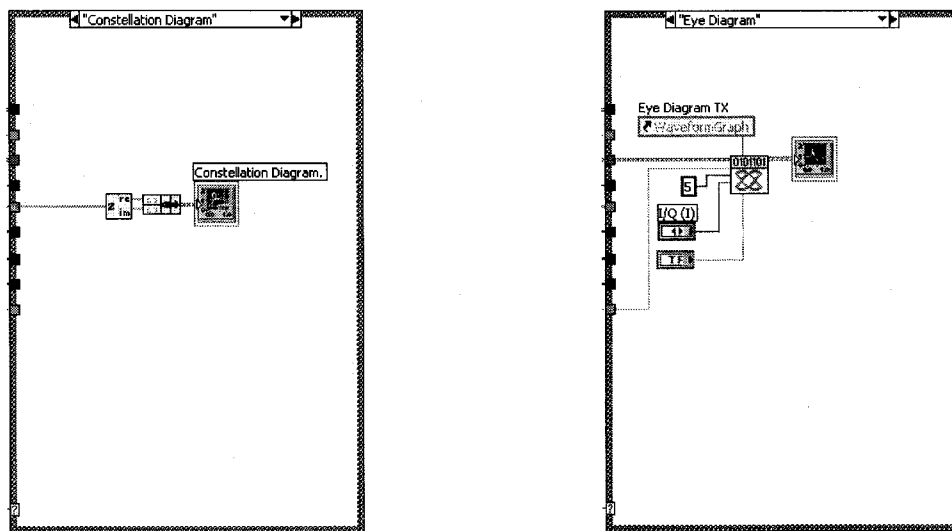


(a) I Channel

(b) Q Channel

(c) I and Q Channels

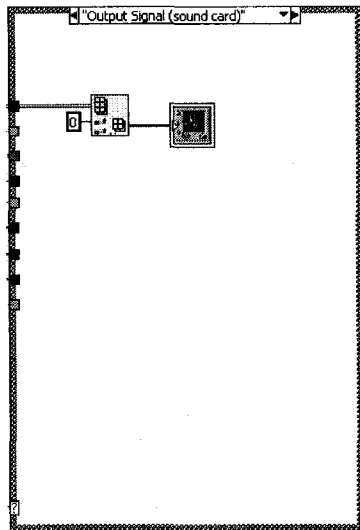
Figure B-7 – 4 QAM SDR Transceiver.vi BD, transmitter mode: Pulse Shaping case structures



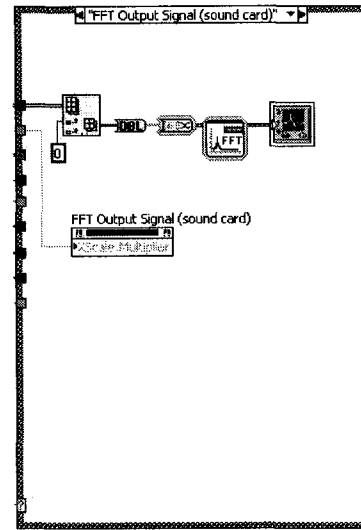
(a) Constellation Diagram

(b) Eye Diagram

Figure B-8 – 4 QAM SDR Transceiver.vi BD, transmitter mode: Constellation and Eye Diagrams case structures



(a) Output Signal



(b) FFT Output Signal

Figure B-9 – 4 QAM SDR Transceiver.vi BD, transmitter mode: Output Signal and FFT Output Signal case structures

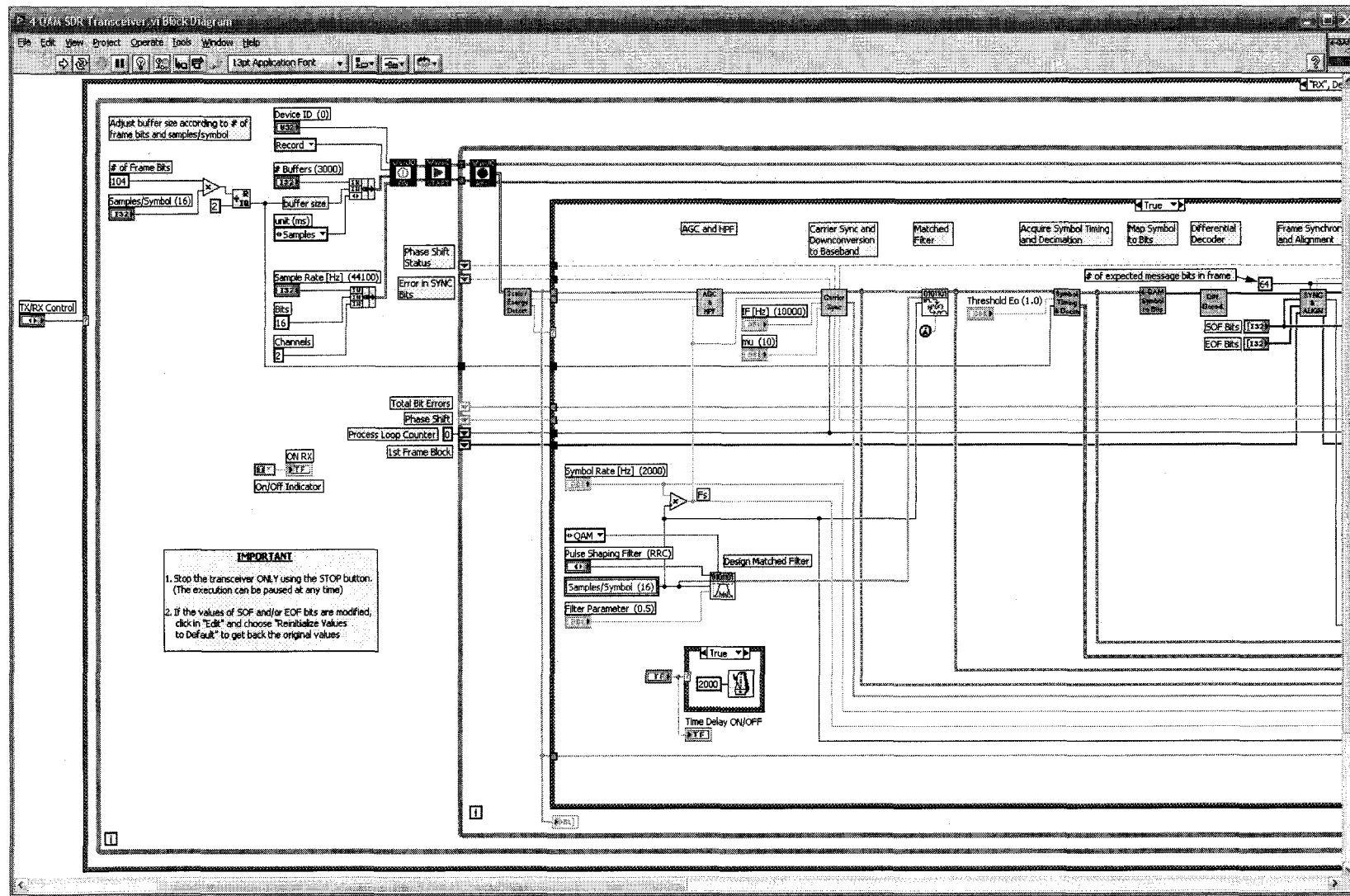


Figure B-10 – 4 QAM SDR Transceiver block diagram, receiver mode: part 1

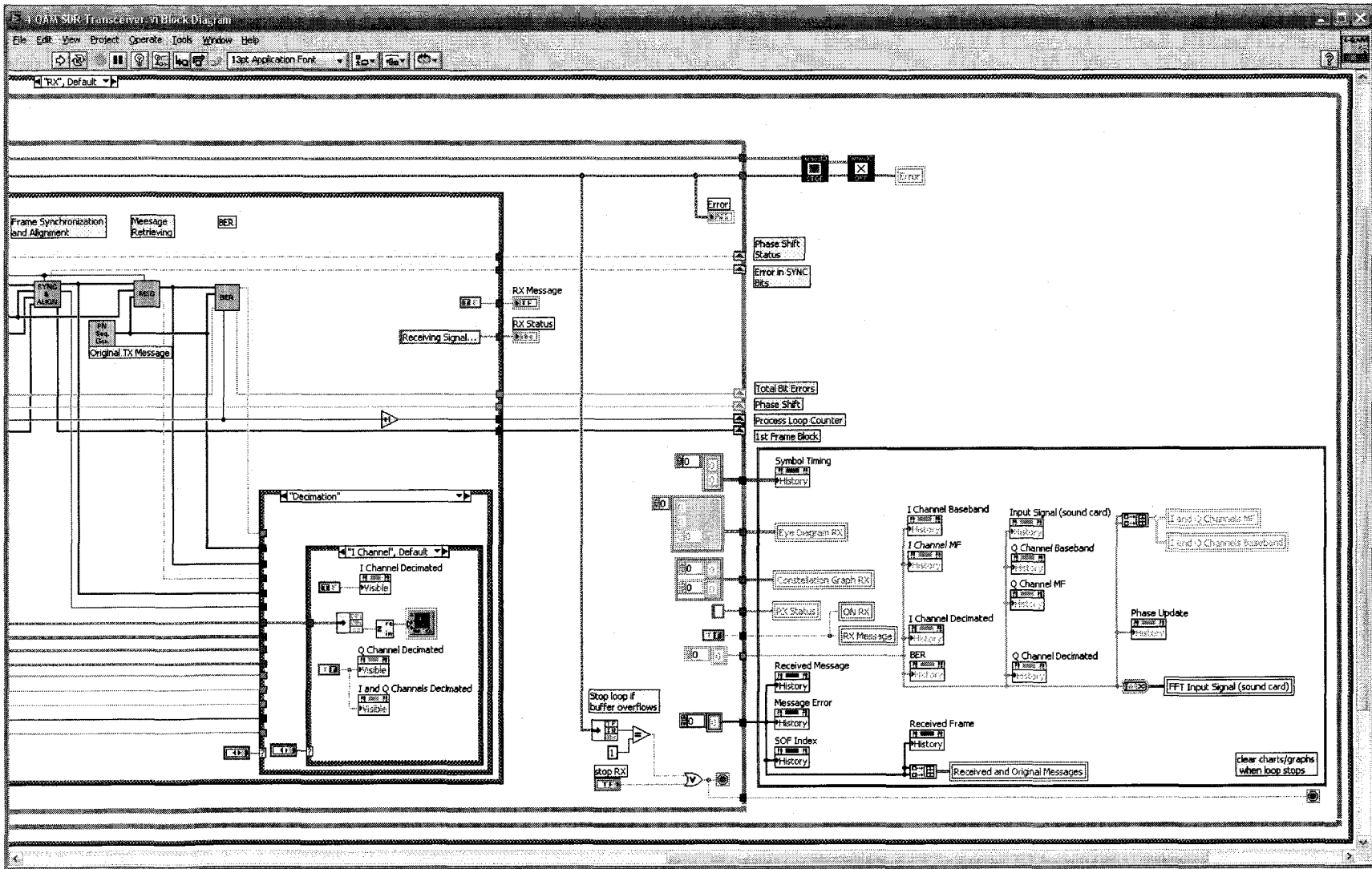


Figure B-11 – 4 QAM SDR Transceiver block diagram, receiver mode: part 2

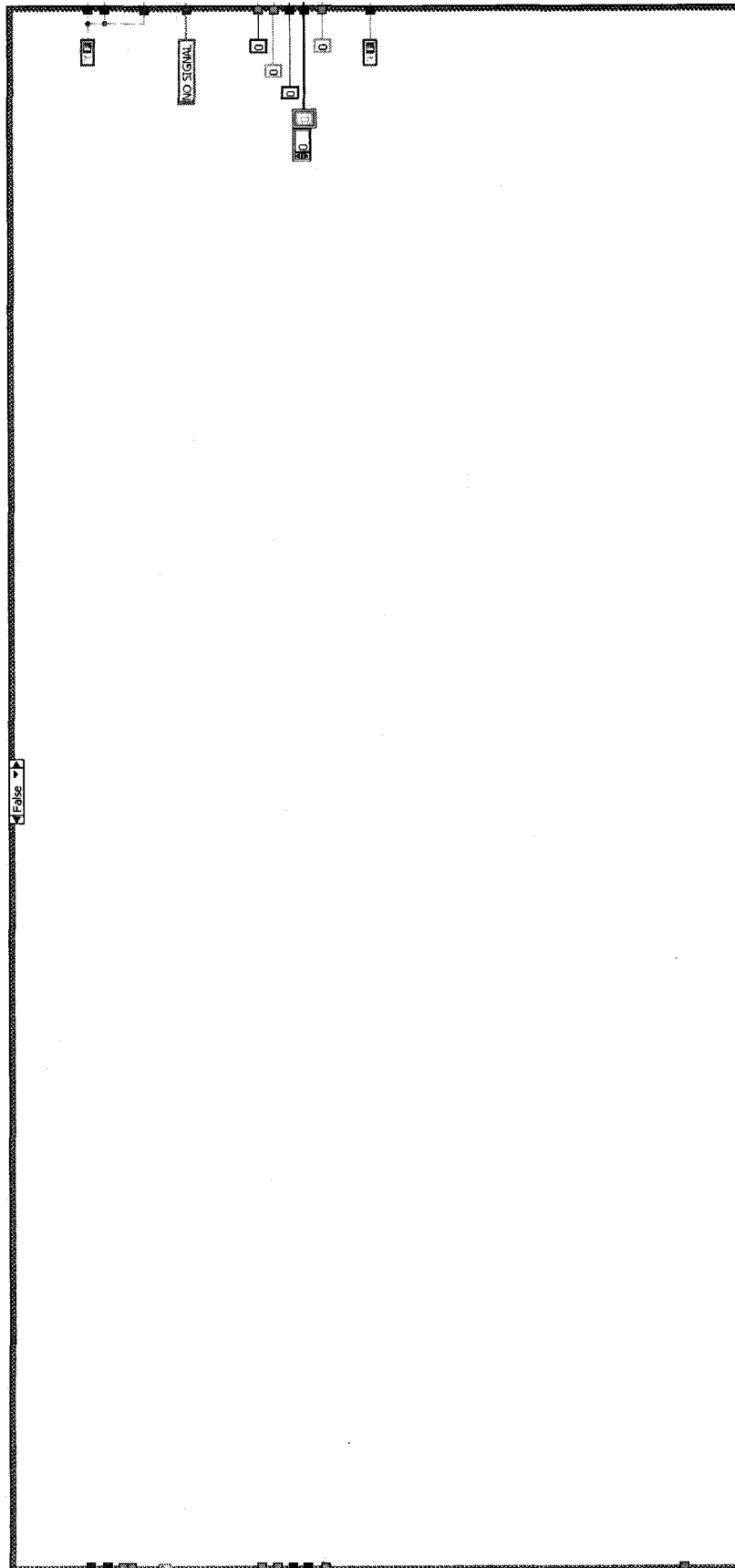


Figure B-12 – 4 QAM SDR Transceiver.vi block diagram, receiver mode: false case structure

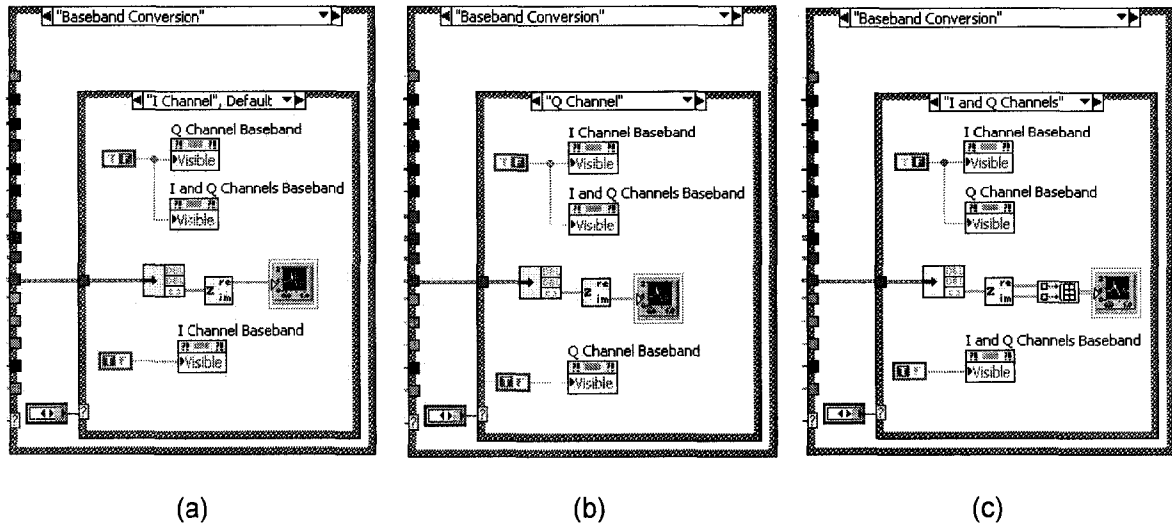


Figure B-13 – 4 QAM SDR Transceiver.vi BD, receiver mode: Baseband Conversion case structures

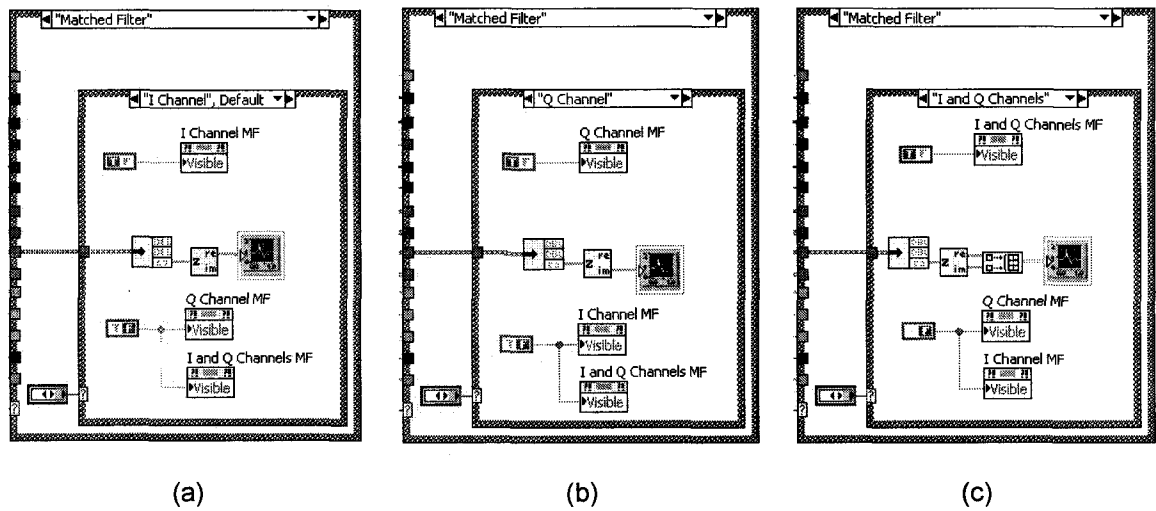
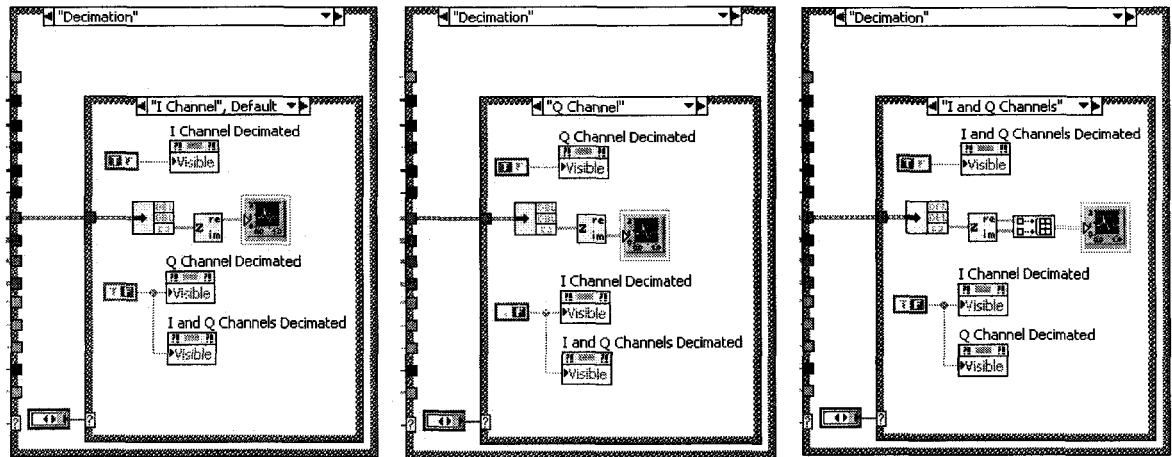


Figure B-14 – 4 QAM SDR Transceiver.vi BD, receiver mode: Matched Filter case structures

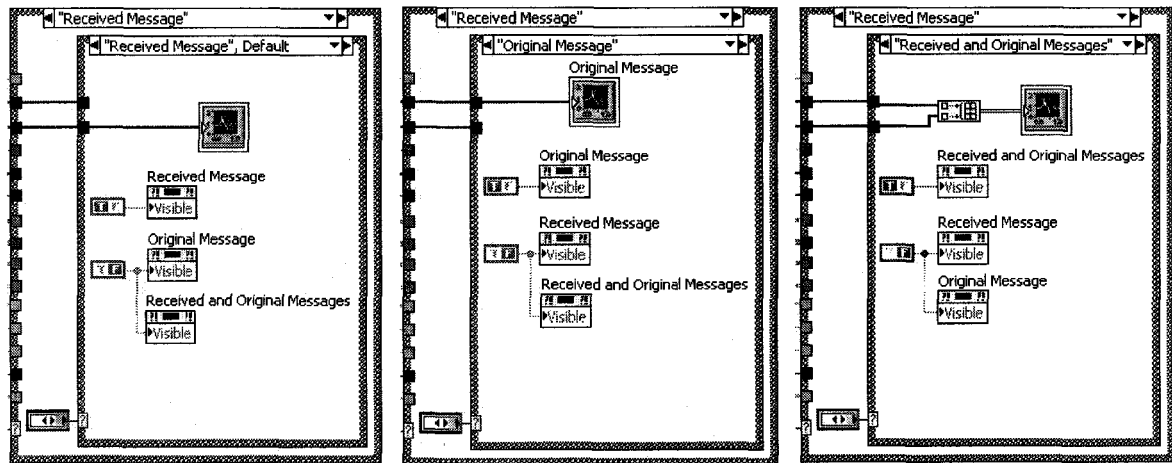


(a) I Channel

(b) Q Channel

(c) I and Q Channels

Figure B-15 – 4 QAM SDR Transceiver.vi BD, receiver mode: Decimation case structures



(a) Received Message

(b) Original Message

(c) Received and Original Messages

Figure B-16 – 4 QAM SDR Transceiver.vi BD, receiver mode: Received Message case structures

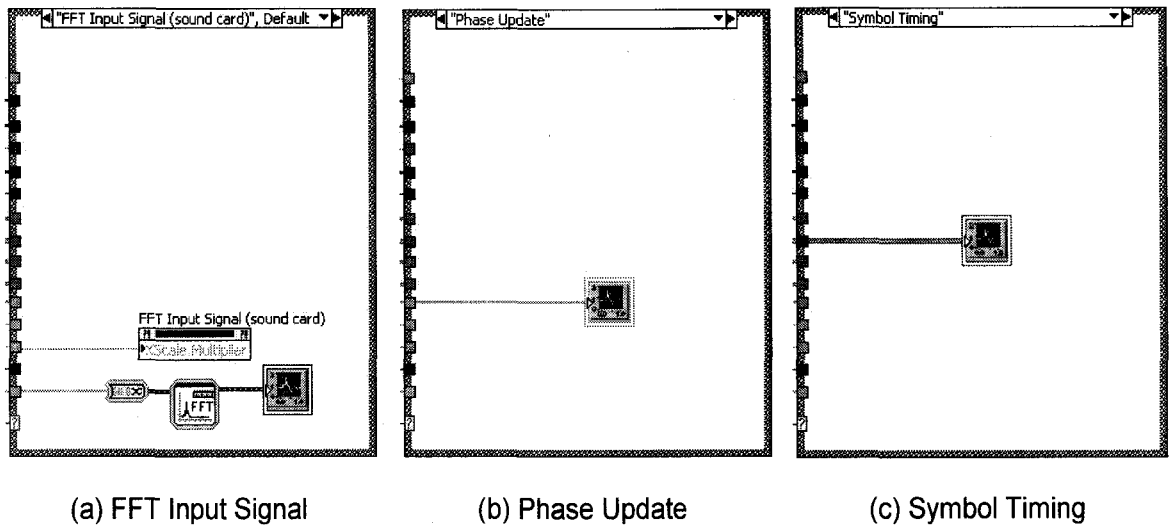


Figure B-17 – 4 QAM SDR Transceiver.vi BD, receiver mode: FFT Input Signal, Phase Update, and Symbol Timing case structures

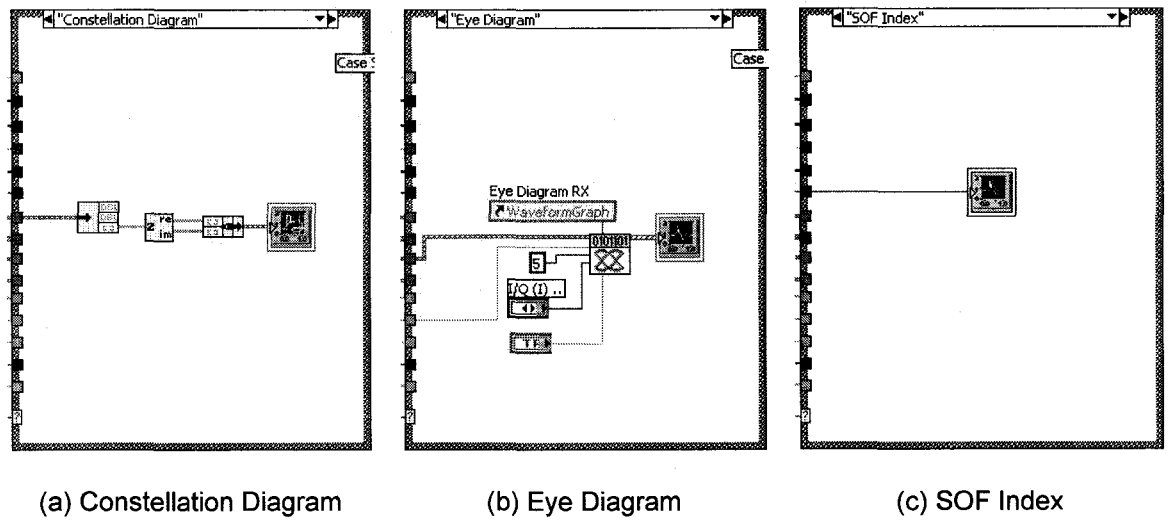


Figure B-18 – 4 QAM SDR Transceiver.vi BD, receiver mode: Constellation Diagram, Eye Diagram, and SOF Index case structures

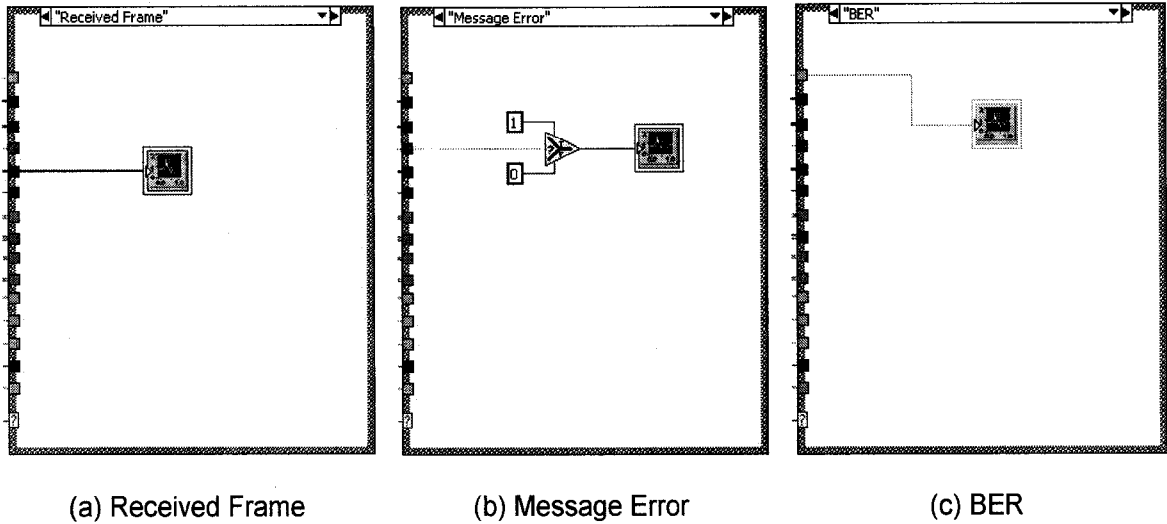
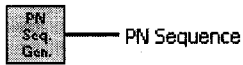


Figure B-19 – 4 QAM SDR Transceiver.vi BD, receiver mode: Received Frame , Message Error, and BER case structures

64 Sample Period PN Sequence Generator.vi



Generates a binary pseudo noise (PN) sequence with a period of 64 samples.

Figure B-20 – 64 Sample Period PN Sequence Generator.vi icon and description

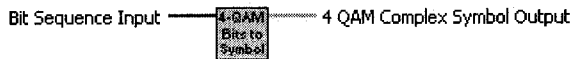
Differential Encoder.vi



Applies differential encoding to the binary input sequence to solve for carrier phase ambiguity at receiver. The Differential Decoder.vi must be used in the receiver.

Figure B-21 – Differential Encoder.vi icon and description

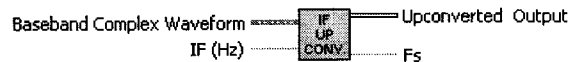
4 QAM Bits to Symbol Mapping.vi



Converts pair of binary sequence into a 4-QAM symbol (complex value). The 4 QAM Symbol to Bits Mapping.vi must be used in the receiver to recover the transmitted binary sequence.

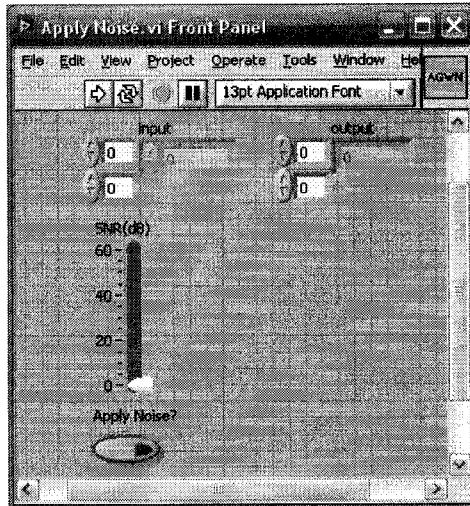
Figure B-22 – 4 QAM Bits to Symbol Mapping.vi icon and description

Upconvert to IF.vi

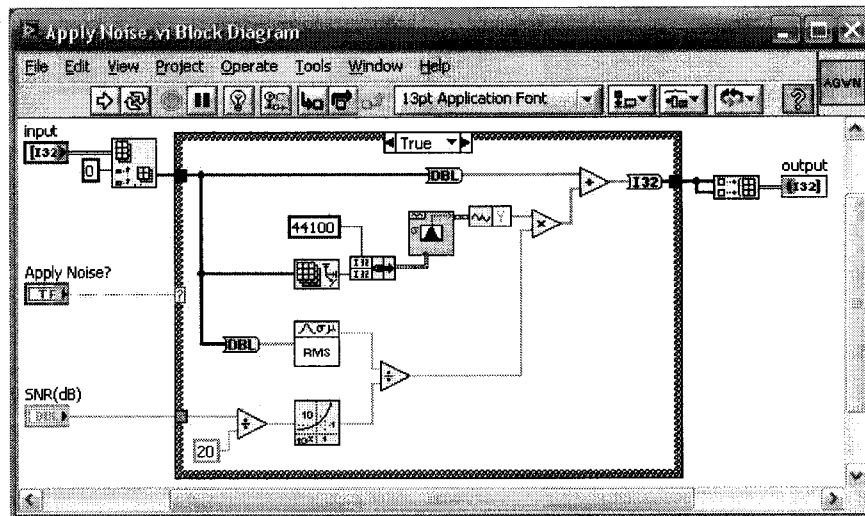


Converts the baseband signal into an intermediate frequency (IF) modulated signal. The carrier frequency (IF) is set by the value of the IF input in Hz. Fs is the sample frequency in Hz.

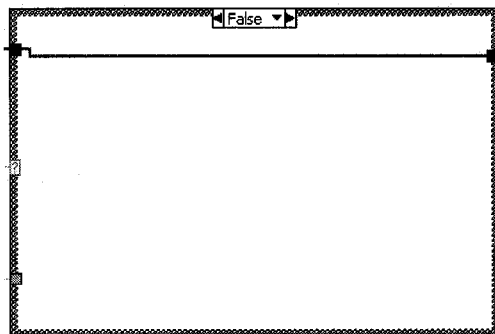
Figure B-23 – Upconvert to IF.vi icon and description



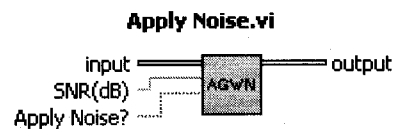
(a) Front panel



(b) Block diagram



(c) False case structure



Adds white Gaussian noise to the transmitted signal to simulate channel impairments when the Boolean Apply Noise? control is set to TRUE.

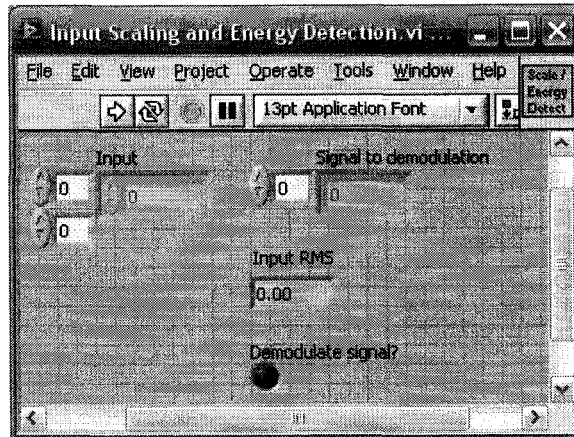
Input: signal to be transmitted.

Apply Noise?: when set to TRUE adds white Gaussian noise to the transmitted signal.

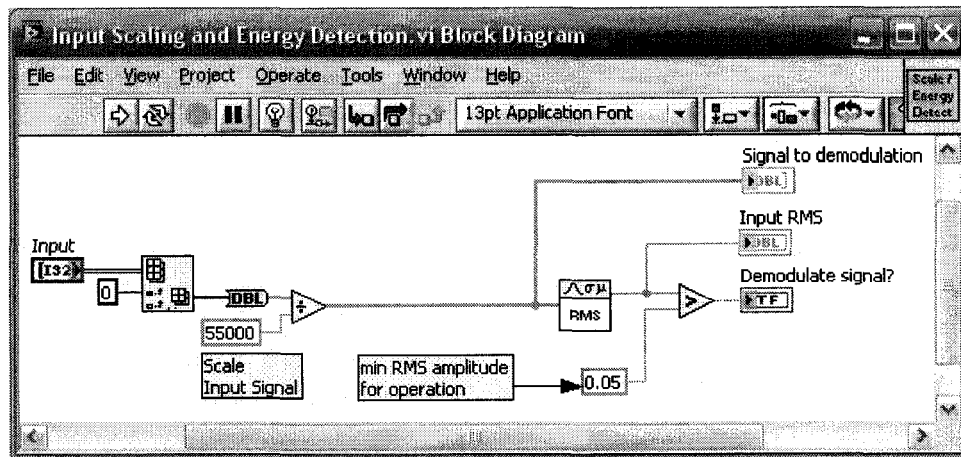
SNR (dB): adjusts the signal to noise ratio in dB.

(d) Icon and description

Figure B-24 – Apply Noise.vi

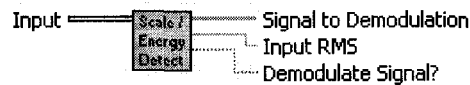


(a) Front panel



(b) Block diagram

Input Scaling and Energy Detection.vi



This VI scales the amplitude of the incoming signal from the sound card back to the proper range used in the code. It also detects if there is enough energy in the signal for proper demodulation. If the RMS amplitude value of the incoming signal is lower than the minimum RMS amplitude value for operation, the Demodulate Signal? output is set to FALSE. Otherwise, this output is set to TRUE in order to pass the signal to the demodulation process.

Input: received signal from sound card buffer.

Signal to Demodulation: Scaled signal.

Input RMS: root mean square (RMS) value of the incoming signal.

Demodulate Signal?: set to TRUE when there is enough energy in the signal to demodulation. Otherwise, it is set to FALSE.

(c) Icon and description

Figure B-25 – Input Scaling and Energy Detection.vi

AGC and HPF.vi



This VI applies automatic gain control (AGC) and high pass filtering (HPF) on the input signal. The AGC stage brings the signal to an optimum amplitude range to be used in the demodulation process. The HPF stage removes low frequency noise added to the signal by the sound card and/or other interferences.

Input Signal: received signal from sound card after scaling.

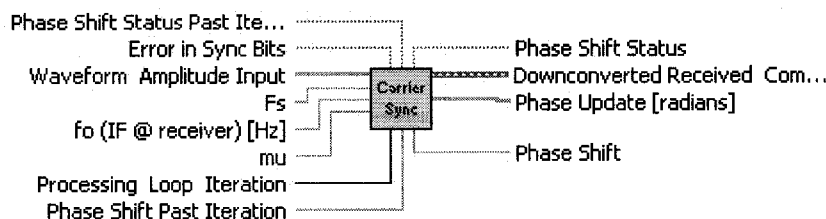
Input RMS: RMS value of the incoming signal.

Sampling Freq (Fs): input signal sampling frequency.

Output: signal output after applying AGC and HPF.

Figure B-26 – AGC and HPF.vi icon and description

Carrier Synchronization and Downconversion.vi



This VI uses the Quadriphase Costas Loop method for recovering carrier phase and frequency from the 4-QAM signal. Later, it down converts the signal to baseband.

Phase Shift Status: set to TRUE if a phase shift was applied, set to FALSE otherwise.

Error in Sync Bits: set to TRUE if there was an error in synchronization bits (SOF and/or EOF) during past loop iteration, set to FALSE otherwise.

Fs: sampling rate, in Hertz [Hz].

fo (IF @ receiver): intermediate frequency (IF) of carrier used in receiver, in Hertz [Hz]. This frequency should match the intermediate frequency used on the transmitter.

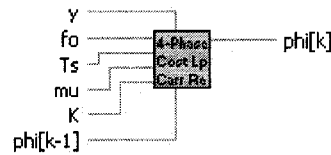
mu: adaptation constant of Costas Loop. It has to be adjusted according to the frequency offset value between transmitter and receiver. Higher frequency offsets demand higher values for mu. Default value = 10.

Processing Loop Iteration Number: iteration number value during signal processing time. When the signal starts being processed, this number starts at value 0 and increases by 1 each loop iteration.

Phase Shift: value of phase shift, in radians [rad], applied to the Carrier Frequency and Phase Recover block to resolve phase ambiguity problem.

Figure B-27 – Carrier Synchronization and Downconversion.vi icon and description

Quadrphase Costas Loop Carrier Recovery.vi

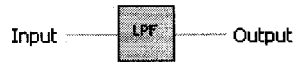


Recovers phase from the incoming IF signal using the Costas loop for 4 QAM modulation. When the value set for the carrier frequency at the receiver (f_0) does not match the carrier frequency of the received signal, this VI updates the phase $\phi[k]$ to compensate for the frequency offset. Higher frequency offsets require higher values for the Costas loop adaptation constant (μ).

- y: amplitude value of the complex input waveform.
- fo: carrier frequency set at receiver in Hz. Default value = 10 kHz.
- Ts: sampling period of the incoming waveform in seconds.
- mu: Costas loop adaptation constant. Default value = 10.
- phi: phase update in radians.
- k: Costas loop iteration number.

Figure B-28 – Quadrphase Costas Loop Carrier Recovery.vi icon and description

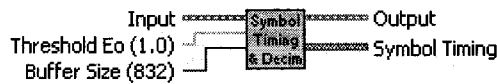
1st Order Butterworth LPF 7k.vi



1st order Butterworth low pass filter with cutoff frequency of 7 kHz.

Figure B-29 – 1st Order Butterworth LPF 7k.vi icon and description

Symbol Timing and Decimation.vi

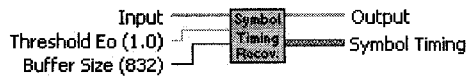


Recovers the symbol timing using the DLL algorithm and down samples the incoming complex waveform.

- Input: complex waveform from matched filter.
- Threshold Eo: successive decision statistics sum value needed to correct timing in the DLL algorithm. If this sum is larger or equal the threshold E_0 , the algorithm corrects the symbol clock. Default value = 1.0
- Buffer Size: size of sound card buffer. Default value = 832
- Output: complex decimated quadrature symbols.
- Symbol Timing: outputs the recovered symbol clock bundled with the in-phase component.

Figure B-30 – Symbol Timing and Decimation.vi icon and description

Symbol Timing Recovery.vi



Uses the delay-locked loop (DLL) algorithm to recover the symbol clock.

The output array has the same size of the input array.

If the sample is at an optimal time, the corresponding complex value is passed to the output array element.

Otherwise, zero values are passed to the output array element.

The final output array will have the corresponding complex symbol values for the optimal times and zero values for all the rest of times.

Input: complex quadrature signal.

Threshold Eo: successive decision statistics sum value needed to correct timing in the DLL algorithm. If this sum is larger or equal the threshold Eo, the algorithm corrects the symbol clock. Default value = 1.0

Buffer Size: size of sound card buffer. Default value = 832

Output: complex array containing quadrature symbol values corresponding to optimal sampling times and zero values for other sampling times.

Symbol Timing: outputs the recovered symbol clock bundled with the in-phase component.

Figure B-31 – Symbol Timing Recovery.vi icon and description

Down Sampler.vi



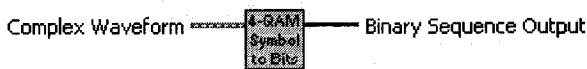
Down samples the incoming array by eliminating samples with zero values.

Input: complex output from Symbol Timing Recovery.vi

Output: complex decimated quadrature signal.

Figure B-32 – Down Sampler.vi icon and description

4 QAM Symbol to Bits Mapping.vi



Converts 4-QAM symbols into pair of bits, according to mapping defined in 4-QAM Bits to Symbol VI.

Figure B-33 – 4 QAM Symbol to Bits Mapping.vi icon and description

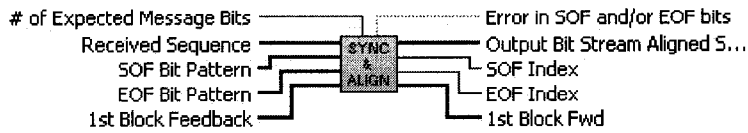
Differential Decoder.vi



Applies differential decoding to the incoming bit sequence to solve for carrier phase ambiguity.

Figure B-34 – Differential Decoder.vi icon and description

Frame Sync and Alignment.vi



This VI detects the beginning of the frame in the current buffer and aligns the remaining part of this frame on next buffer in order to recover the entire transmitted frame.

of Expected Message Bits: number of expected message bits in a frame.

Received Sequence: received frame bits.

SOF Bit Pattern: bit pattern used as start of frame.

EOF Bit Pattern: bit pattern used as end of frame.

1st Block Feedback: block saved at the past iteration that must be appended to the current block to recover the current frame.

Error in SOF and/or EOF Bits: If an error occurs in any of the bits inside the SOF and/or the EOF bits, then this signal is set to TRUE. Otherwise, it is set to FALSE.

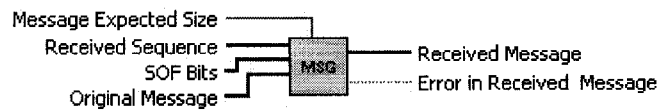
Output Bit Stream Aligned SOF+MESSAGE+EOF: aligned frame output.

SOF Index: first bit index of the SOF bit pattern inside the received sequence.

EOF Index: first bit index of the EOF bit pattern inside the received sequence.

Figure B-35 – Frame Synchronization and Alignment.vi icon and description

Message Extraction.vi



Extracts the message from the received frame and detects if there is error in the received message when a copy of the original message is available at the Original Message input. If there is an error, the output Error in Received Message is set to TRUE.

Message Expected Size: number of bits expected in the message.

Received Sequence: received aligned frame bits.

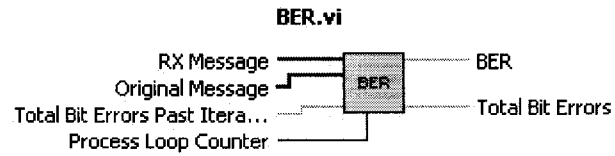
SOF Bits: start of frame (SOF) bit pattern.

Original Message: binary message sent by the transmitter used to determine occurrence of errors in the received message.

Received Message: received message bits.

Error in Received Message: if there is error in one or more bits of the received message, this output is set to TRUE. Otherwise, it is set to FALSE.

Figure B-36 – Message Extraction.vi icon and description



Calculates the bit error rate (BER) of a transmitted digital message.

RX Message is the received binary message.

Original Message: bit sequence transmitted (message).

Total Bit Errors Past Iteration: number of message bit errors in past iteration.

Process Loop Counter: number of loop iterations (or number of received frames) used to calculate the total received bits.

BER: bit error rate.

Total Bit Errors: total of bit errors accumulated from all iterations. It must be wired to a shift register on the right side of the main loop, which will be used as the input for Total Bit Error Past Iteration.

Figure B-37 – BER.vi icon and description

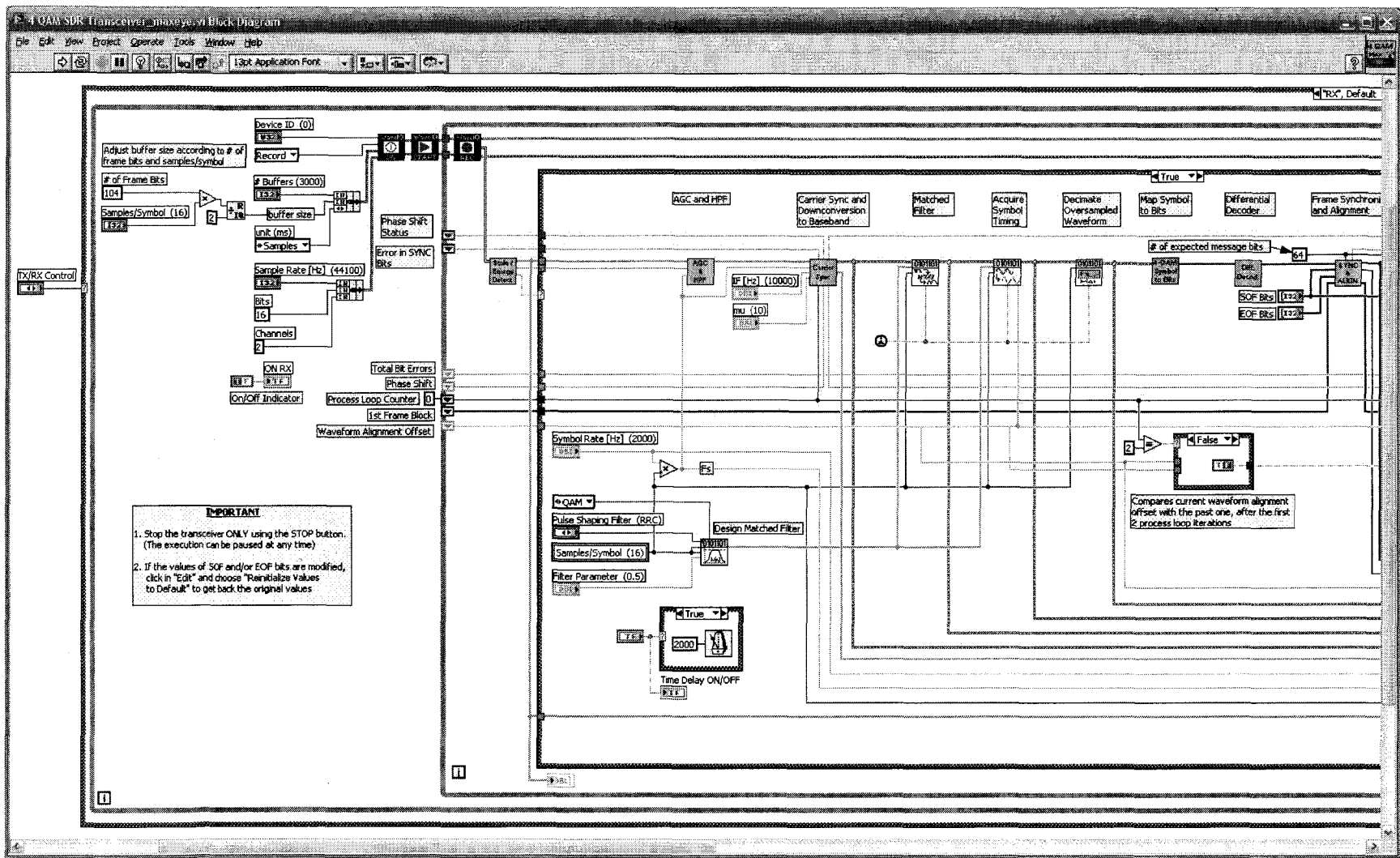


Figure B-38 – 4 QAM SDR Transceiver_maxeye.vi block diagram, receiver mode: part 1

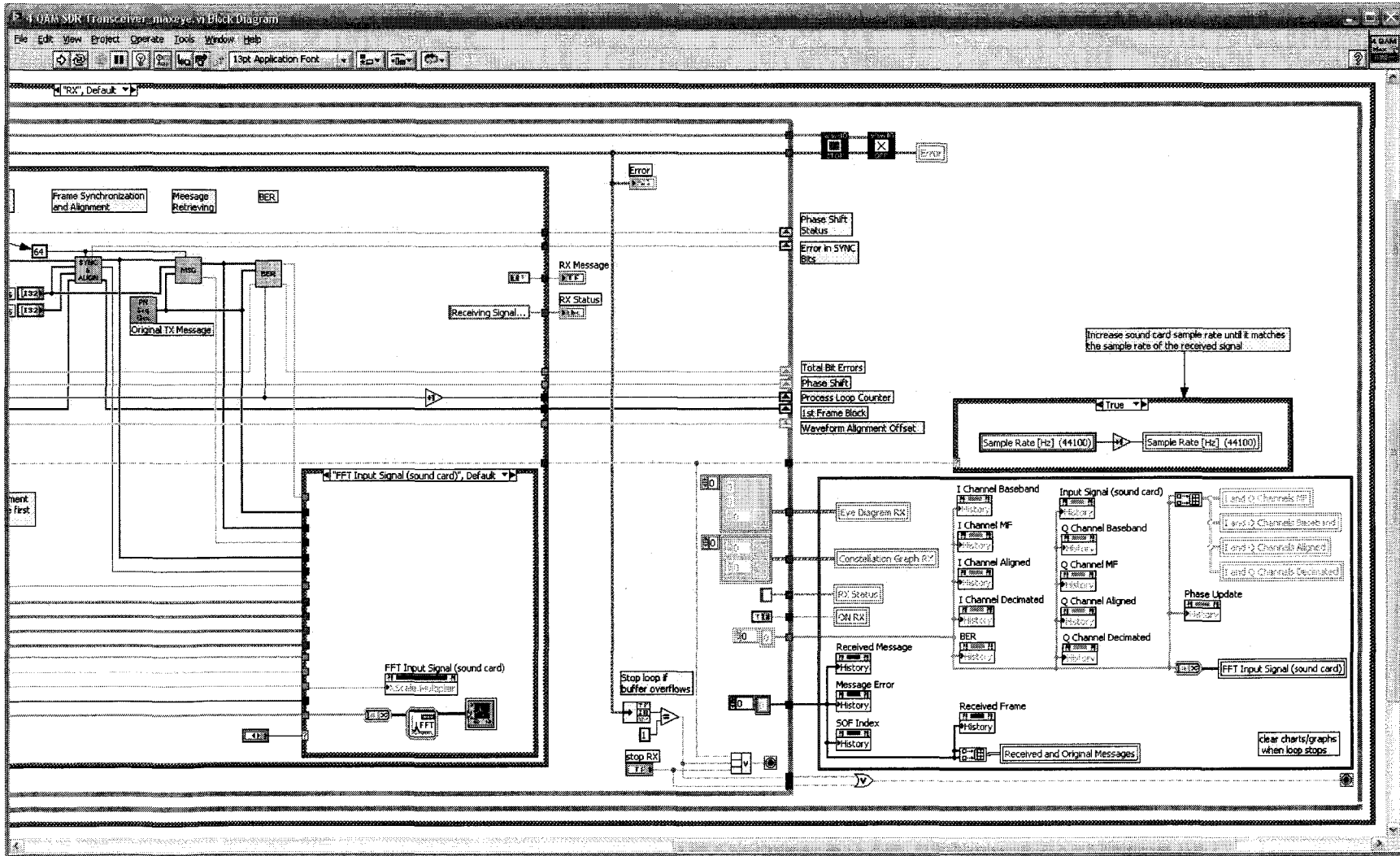


Figure B-39 – 4 QAM SDR Transceiver_maxeye.vi block diagram, receiver mode: part 2

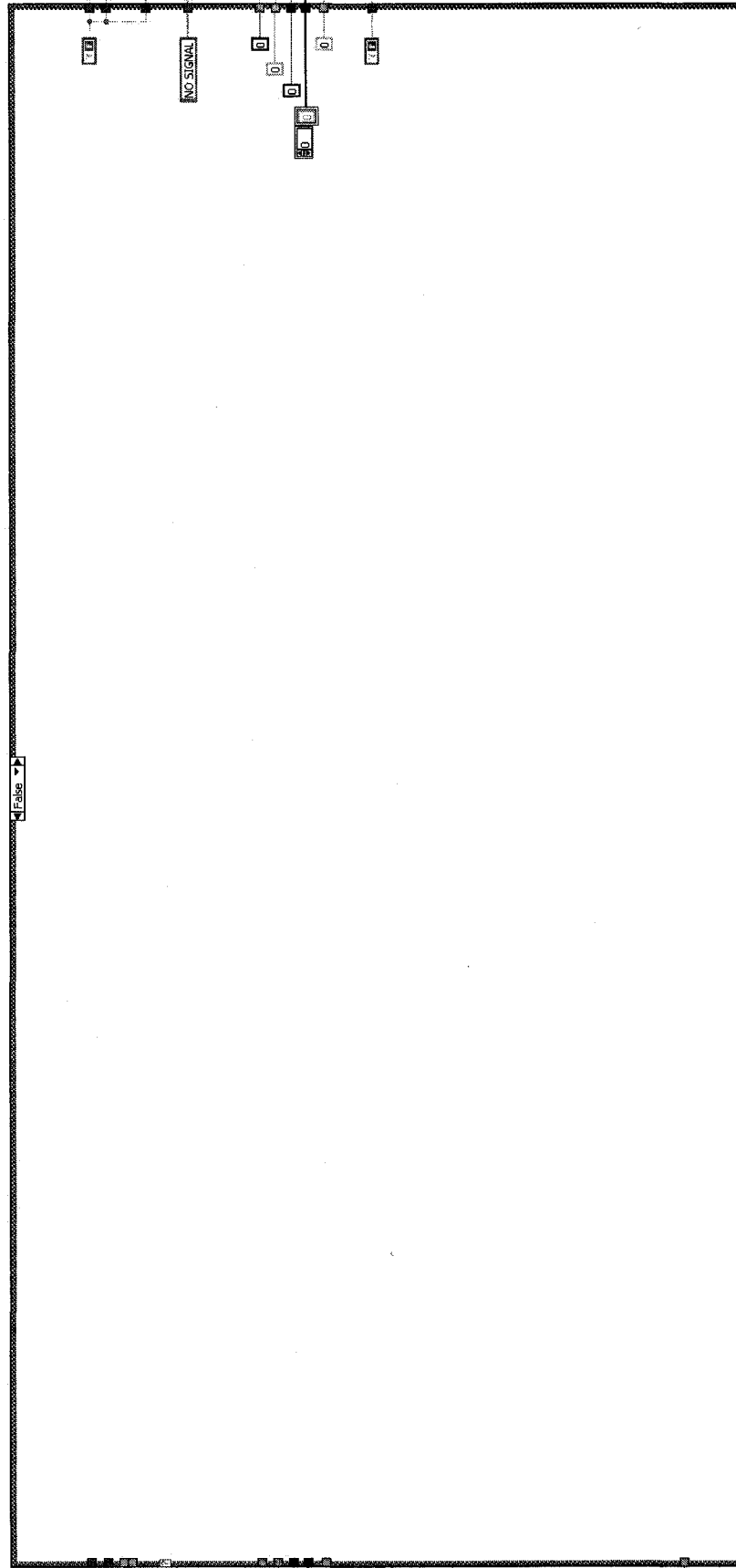


Figure B-40 – 4 QAM SDR Transceiver_maxeye.vi block diagram, receiver mode: false case structure

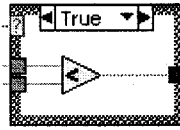


Figure B-41 – 4 QAM SDR Transceiver_maxeye block diagram, receiver mode: sample rate update case structure 1 detail

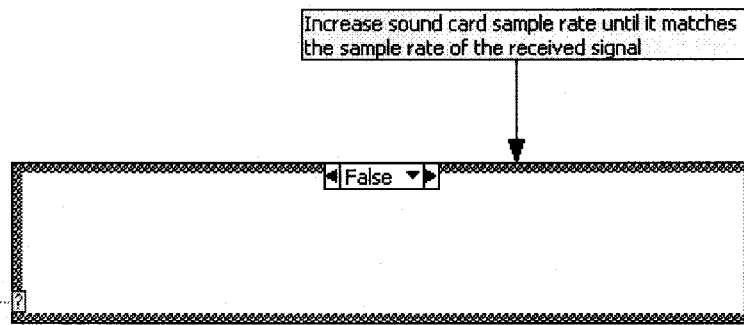


Figure B-42 – 4 QAM SDR Transceiver_maxeye block diagram, receiver mode: sample rate update case structure 2 detail