

CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

INTEGRATING FPGA WITH MULTICORE SDR DEVELOPMENT PLATFORM TO
DESIGN WIRELESS COMMUNICATION SYSTEM

A graduate project submitted in partial fulfillment of the requirements
for the degree of Masters of Science
in Electrical Engineering.

By

SANKET PRAKASH JOSHI

MAY 2012

The graduate project of Sanket Joshi is approved:

Dr. Nagwa Bekir

Date

Dr. Ramin Roosta

Date

Dr. Ichiro Hashimoto, Chair

Date

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Ichiro Hashimoto for giving me the freedom of thought and expression while performing my project work. He has been a constant source of inspiration and has provided consistent succor and valuable suggestions throughout this project without which this work wouldn't have been possible.

I would also like to thank Dr. Nagwa Bekir and Dr. Ramin Roosta for serving in my committee. In addition to that I would also like to thank the Department chair Dr. Ali Amini and the Department of Electrical and Computer Engineering at the California State University Northridge for giving me this opportunity. I would like to express my heart-felt gratitude to my family for their encouragement and support without which I wouldn't have come so far. Finally I would like to thank all my friends for their invaluable support and co operation.

TABLE OF CONTENTS

SIGNATURE PAGE.....	ii
ACKNOWLEDGEMENT.....	iii
LIST OF TABLES.....	vii
LIST OF FIGURES.....	viii
ABSTRACT.....	x
CHAPTER 1 Introduction.....	1
1.1 Introduction.....	1
1.2 Motivation.....	2
1.3 Contribution.....	5
1.4 Outline of project report.....	6
CHAPTER 2 Fundamentals of Software Defined Radio.....	7
2.1 Introduction.....	7
2.2 Definition of Software Defined Radio.....	8
2.3 Software Defined Radio Concept.....	12
2.3.1 Comparison of conventional and SDR.....	12
2.3.2 Architecture of SDR.....	14
2.3.3 Rate of adoption and value chain of SDR.....	17
2.3.4 SDR related technologies.....	19
2.3.5 Advantages and disadvantages of SDR.....	22

2.4 SDR implementation platform.....	23
2.5 Technical challenges.....	26
2.6 Summary.....	28
CHAPTER 3 Framework for system design.....	29
3.1 Wireless communication system model for SDR.....	29
3.1.1 Radio frequency section.....	29
3.1.2 Intermediate frequency section.....	30
3.1.3 Baseband section.....	30
3.2 Design process for reprogrammable computing.....	31
3.2.1 High level simulation.....	32
3.2.2 VHDL description.....	34
3.3.3 HDL co-simulation.....	35
3.3.4 Hardware co-simulation.....	35
3.3 Test bed implementation.....	36
3.3.1 Simulation with simulink and system generator.....	37
3.3.2 HDL co-simulation procedure.....	37
3.3.3 Hardware co-simulation procedure.....	39
3.3.4 Summary.....	40
CHAPTER 4 System design and implementation.....	41
4.1 Environment required.....	41
4.2 System model for simulation.....	41

4.3 The system generator design.....	42
4.4 Convolutional encoder and Virterbi decoder.....	44
4.5 Transmitter model.....	46
4.6 Receiver model.....	49
CHAPTER 5 Simulations.....	51
5.1 System simulation.....	51
5.2 Simulation results.....	56
5.2.1 Constellation diagrams.....	56
5.2.2 Eye diagrams.....	59
5.2.3 Output waveform.....	61
CHAPTER 6 Real time implementation and simulation.....	62
6.1 Real time implementation and simulation.....	62
CHAPTER 7 Summary and Conclusion.....	67
7.1 Summary.....	67
7.2 Conclusion.....	68
7.3 Future work.....	68
BIBILOGRAPHY.....	69
APPENDIX-A.....	74

LIST OF TABLES

Table 2.1-Comparison between conventional and Software Defined Radio.....	13
Table 4.1-The hardware and software requirements.....	41
Table 4.2-The convolutional code with puncturing configuration.....	45

LIST OF FIGURES

Figure 2.1-Generalized Functional Architecture.....	10
Figure 2.2-Block diagram of a generic digital transceiver.....	14
Figure 2.3-Transceiver of an Ideal SDR.....	16
Figure 2.4-Architecture of software components of an SDR.....	17
Figure 2.5-SDR Value Chain: Product and Service Based Providers and Supporting Organizations.....	19
Figure 2.6-Venn diagram of advance wireless technology.....	20
Figure 2.7-Cognitive Radio Concept Architecture.....	21
Figure 2.8-Hardware implementation platforms for SDR.....	24
Figure 3.1-functional block diagrams of the digital transceiver.....	30
Figure 3.2-Design process for reconfigurable computing.....	32
Figure 3.3-Illustration of test-bed implementation process.....	36
Figure 3.4-System Generator dialog box.....	38
Figure 4.1-Software definable baseband communication system.....	42
Figure 4.2-The System Generator design flow.....	43
Figure 4.3-Basic architecture of design with Xilinx block set.....	44
Figure 4.4-Three essential Xilinx blocks.....	44
Figure 4.5-Convolution Encoder setting.....	45
Figure 4.6-Viterbi Decoder setting.....	46
Figure 4.7-Example of puncturing rate 1/2 encoder.....	48

Figure 4.8-Transmitter design.....	48
Figure 4.9 Example of depuncturing rate 2/3 encoded data.....	49
Figure 4.10-Receiver design.....	50
Figure 5.1-Encoding and puncturing using System Generator blocks.....	52
Figure 5.2-Modulation and spreading of the encoded signal.....	53
Figure 5.3-Despreading and demodulation of the received signal.....	53
Figure 5.4-Depuncturing and decoding model using System Generator blocks.....	54
Figure 5.5-Communication system model using system generator blocks.....	55
Figure 5.6-Constellation diagram for QPSK modulated signal with SNR = 20 dB.....	57
Figure 5.7-Constellation diagram for QPSK modulated signal with SNR = 25 dB.....	57
Figure 5.8-Constellation diagram for BPSK modulated signal with SNR = 25 dB.....	58
Figure 5.9-Eye diagram for QPSK modulated signal with SNR = 20 dB.....	59
Figure 5.10-Eye diagram for QPSK modulated signal with SNR = 25 dB.....	60
Figure 5.11 Input and output waveforms of the communication system.....	61
Figure 6.1-Spartan 3E FPGA hardware board for hardware co-simulation.....	63
Figure 6.2-Command window showing the progress of implementation tools.....	65
Figure 6.3-Hardware co-simulation library.....	66
Figure 6.4-Constellation diagram for hardware co-simulation.....	66

ABSTRACT

INTEGRATING FPGA WITH MULTICORE SDR DEVELOPMENT PLATFORM TO DESIGN WIRELESS COMMUNICATION SYSTEM

By

SANKET PRAKSH JOSHI

Master of Science in Electrical Engineering

This project uses integrating FPGA with multicore Software Defined Radio development platform to design wireless communication system. It constructs the system on the Matlab/Simulink environment in the way of Model-based Design, and realizes transmitter section with FPGA. Finally it uses Hardware-in-the-loop co-simulation to observe the bits error rate of system under the white Gaussian noise channel disturbance of different SNR values. Software defined radio is a feasible solution for reconfigurable radios, which can perform different functions at different times on the same hardware.

The baseband section of a wireless communication system is first simulated and then implemented in hardware. The performance of the baseband transmitter is analyzed using constellation and eye diagrams for different modulation techniques and different signal to noise ratios, while considering an additive white Gaussian noise channel. The performance of the receiver is analyzed by comparing the input and output waveforms. The performance of the system in real time is also analyzed by implementing the system in hardware using Xilinx Spartan 3E FPGA (XC3S500E-4FG320C). A comparison of the simulation results with the results obtained from implementing the system on Xilinx Spartan 3E FPGA (XC3S500E-4FG320C) hardware is presented and discussed. It is shown that the simulation results and experimental results are similar.

CHAPTER 1

INTRODUCTION

1.1 Introduction:

Modern society is increasingly becoming dependent on digital communication systems in order to function properly, with a growing number of applications relying on these devices, e.g., personal health/body networks, defense/homeland security, navigation/localization, social networking, vehicular transportation. There is a multitude of telecommunications standards in use today, ranging from mobile communication standards such as GSM, PHS, and UMTS, to wireless LAN standards such as WiMAX and IEEE802.11x as well as future standards based on UWB technology. Wireless communication networks have become more popular in the past two decades since the advent of cellular communications. The rapid growth in cellular communications has proved that wireless communication is viable for voice and data services. Traditional wireless devices are designed to deliver a single communication service using a particular standard. With the steady increase of new wireless services and standards, single purpose devices with dedicated hardware resources can no longer meet the user's needs. It is also expensive to upgrade and maintain a wireless system each time a new standard comes into existence.

A feasible solution to make communication systems more flexible and user friendly can be achieved through the software defined radio (SDR) concept, because as traditional transceiver technology requires users to have separate equipment for each standard however SDR offers the possibility of using one terminal to receive many standards through the use of wideband reconfigurable transceivers and software signal processing. The main challenge is to optimize the tradeoffs between performance, power consumption and cost. This is achieved through the use of readily available low cost mobile communications components and a standard laptop PC for data processing and configuration. Software defined radio refers to the class of reprogrammable or reconfigurable radios in which the same piece of hardware can perform different functions at different times. Software defined radio is an emerging technology, for multi-

service, multi-standard, multi-band, reconfigurable radio systems, which are reprogrammable by software. A working definition of a software defined radio is a radio that is considerably defined in software and whose physical layer behavior can be significantly altered through changes to its software. Thus, the same piece of hardware can be used to realize different applications by modifying the software. Software defined radio has generated tremendous interest in the wireless communication industry because of the wide-ranging economic and deployment benefits it offers.

Programmable hardware modules are increasingly being used in communication systems design at different functional levels. Software defined radio (SDR) technology can be used to take advantage of programmable hardware modules to build open system architecture based on software. In this case, a variety of transceiver functions such as automatic gain control, frequency translation, filtering, modulation and demodulation can be integrated on a single hardware platform. This could result in maximizing the number of radio functions for a particular application. Software defined radio offers the flexibility and upgradeability necessary to satisfy these requirements.

1.2 Motivation

Consider a typical communication system scenario where the user would like to have access to information through different wireless networks (e.g., wireless local area network (WLAN), Bluetooth, etc.), or a mobile phone user may be traveling between two regions around the globe, where the wireless technologies or standards are different. To utilize the services offered by the broad range of technology alternatives around the world, the user has to carry different devices due to incompatibility of systems and standards. The practical solution to overcome this problem is to use a single device that can adapt to different technologies. This could be possible using software defined radio, since it represents a radio that uses a reprogrammable hardware to create a generic hardware base. On top of the generic hardware platform, flexible software architecture is embedded.

The software allows for multiple protocols, fast upgrades, and complete reconfigurations of radio features and functions. Some of the attractive features of SDR are as follows:

(1) Performance:

The functionality of conventional radio architectures is usually determined primarily by hardware with minimal configurability through software. The hardware consists of the amplifiers, filters, mixers and oscillators dedicated to a particular mode of transmission. The software is confined to functions such as controlling the interface with the network, and error correction. Since the hardware dominates the design, upgrading a conventional radio design essentially means completely abandoning the old design and starting over again, resulting in a waste of time and resources. Software defined radio solves this problem by implementing radio functionalities as software modules running on generic hardware platforms. Since the radio functionalities are defined in software, when a new technology is introduced, it can be easily implemented by dynamic selection of parameters for its functional modules, i.e., reprogramming the software. Software defined radio provides a greater advantage to normal radio systems, since such radio systems can provide only fixed parameters with limited performance.

(2) Flexibility:

The inflexibility of conventional radio systems limits the ability to get the right information to the right users at the right time. Conventional radio systems do not provide the waveform agility necessary to achieve this objective. With software defined radio, modulation waveforms and multiple air interface standards are possible. Thus, SDR platforms can serve a range of applications including analog cellular, digital cellular, personal communications services (PCS), wideband systems, spread spectrum, navigation waveforms (e.g., global positioning system), emergency radio, public safety, and other radio systems. Depending on the waveform, architecture, and implementation, a single software radio platform has the flexibility or potential to support a broad range of communication services.

(3) Compatibility:

The concept of seamless global coverage requires that the radio support two distinct features. (a) Global roaming or seamless coverage across geographical regions; (b) interfacing with different systems and standards to provide seamless services at a fixed location. Existing technologies for voice, video, and data use different packet structures, data types, and signal processing techniques. Integrated services can be obtained with either a single device capable of delivering various services or with a radio that can communicate with devices providing complementary services. The supporting technologies and networks that the radio might have to use can vary with the physical location of the user. To successfully communicate with different systems, the radio has to communicate and decode the signals from devices using different air interfaces. Furthermore, to manage changes in networking protocols, services, and environments, mobile devices supporting reconfigurable hardware also need to seamlessly support multiple protocols. Such radios can be implemented efficiently using software radio architectures in which the radio reconfigures itself based on the system it will be interfacing with and the functionalities that will be supported.

(4) Cost:

Every time a new technology evolves, it results in the migration of functions from an older design to the new design. Implementing a new design involves manufacturing and testing. The cost of this process increases since upgrading to a newer design is not always possible in conventional systems. Software based radio can reduce the cost of manufacturing and testing, while providing a quick and easy way to upgrade the product to take advantage of newer signal processing techniques and new service applications. It is evident from the above list of benefits that SDR technology is very beneficial and has wide applications. Fundamentally, SDR technology can be used in any device that uses radio frequency (RF) for communication, which encompasses a wide range of products including cellular base stations, military communications systems and public safety radios. Because of the above reasons, we are motivated to investigate the different system design and implementation processes for SDR based wireless communication system. We

believe that this could be a major thrust in the of wireless generation services, especially as we migrate from third generation (3G) to fourth generation (4G) wireless standards.

1.3 Contribution

The analysis presented in this project report has many features and several contributions to the current state of knowledge. The general and specific contributions include the following:

- The development of a framework for the design and implementation of Integrated FPGA with multicore SDR development platform based wireless communication systems.
- The analysis of a sample implementation of software defined radio based wireless communication system with coding (i.e. convolutional encoding), viterbi decoding, puncturing and depuncturing, modulation and demodulation, spreading and de-spreading.
- The comparison of the performance of different modulation and demodulation techniques in a SDR implementation environment. The modulation techniques considered are Quadrature Phase Shift Keying (QPSK) and Binary Phase Shift Keying (BPSK).
- Initiation of the development of a test bed for the design and implementation of multicore SDR platform based wireless communication system.
- Presentation of results based on the simulation and actual experimentation.
- Evaluation of the performance of the SDR system in terms of (SNR) signal-to-noise ratio in an Additive White Gaussian Noise (AWGN) channel.
- Evaluation of the performance for the simulation of an SDR system and real time implementation on the Xilinx Spartan-3E FPGA (XC3S500E-4FG320C) field programmable gate array (FPGA) platform part number HW-SPAR3E-SK-UNI-G. The results obtained during simulation and experiments are compared.

1.4 Outline of the Project report

In this report, the fundamentals of software defined radio are first presented in Chapter 2, which includes the general background information and various definitions for SDR. The presentation includes the difference between the SDR and conventional radio, architecture and advantages of SDR, adoption and chain value of SDR, SDR related technologies, possible design issues, and the platform choices for implementing integrated FPGA with multicore SDR based wireless communication systems.

In Chapter 3, the framework for the implementation of a wireless communication system in SDR is presented. This includes a brief introduction of wireless communication systems, with a block diagram of the end-to-end communication system architecture, and the methodology of implementation.

An illustrative baseband communication system implementation, simulation and results are presented and discussed in Chapter 4. Results of the simulation including constellation diagrams, eye diagrams, and output waveforms, for different modulation techniques are presented and analyzed in chapter 5. The system is implemented on the Xilinx Spartan 3E FPGA (XC3S500E-4FG320C) platform. The results of the implementation are compared with the simulation results in chapter 6.

Chapter 7 summarizes the content of the project report. Also, possible extensions of the report are discussed.

CHAPTER 2

FUNDAMENTALS OF SOFTWARE DEFINED RADIO

2.1 Introduction

With the exponential growth in the ways and means by which people need to communicate - data communications, voice communications, video communications, broadcast messaging, command and control communications, emergency response communications, etc. - modifying radio devices easily and cost-effectively has become business critical. Software defined radio (SDR) technology brings the flexibility, cost efficiency and power to drive communications forward, with wide-reaching benefits realized by service providers and product developers through to end users.

Cellular communication systems have undergone tremendous growth since the early 1980's. As a result, mobile communication has become a major worldwide business. Because of this rapid growth, many analog and digital communication standards such as total access communication system (TACS), global system for mobile (GSM), digital cellular system-1800 (DCS-1800), interim standard-95 (IS-95), code division multiple access 2000(CDMA2000), have been developed. In fact, many competing standards have been introduced. The proliferation of standards is not only difficult for manufacturers but also for consumers. Manufacturers have to develop a new device for each technology or standard. This results in extra development costs and divided markets. It is also bad for consumers because users cannot use their mobile communication systems everywhere.

Efforts to define a unique worldwide standard to overcome the above problems often results in a new standard. A unique common worldwide standard has its own advantages, but the industrial competition between different manufacturers introduces many difficulties. Therefore, software defined radio (SDR) concept is considered by many as an emerging technology that offers potential pragmatic solutions. For example, a software implementation of the user terminal will be able to dynamically adapt to the radio environment in which it is located. Software defined radio concepts can be viewed

as a means to make users, service providers, and equipment manufacturers more independent of standards.

Software radio also describes radio functionalities defined by software. The possibility to define the typical functionalities of a radio interface by software will be an excellent opportunity to improve system performance. Currently the radio functionalities in communication systems are usually implemented by dedicated hardware. The presence of software defining the radio interface implies the use of digital signal processors (DSPs) replacing dedicated hardware to execute in real time, the necessary radio functions. To completely realize a digital programmable transceiver, it is necessary for the digital signal processors and the programmable logic such as field programmable gate arrays (FPGAs) to have a high processing power. Although advances have been made in digital signal processing since the 1980's, the processing power of DSPs and FPGAs is still not enough to realize fully functional software defined radios. The required processing power is expected to become available in the near future.

In this chapter, the definitions and meanings of the software defined radio are presented. The difference between SDR and conventional radio is highlighted, as well as the characteristics, advantages and disadvantages. Then, different hardware platforms available to implement SDR are discussed. The design issues in implementing SDR are highlighted.

2.2 Definition of Software Defined Radio

A number of definitions can be found to describe Software Defined Radio, also known as Software Radio or SDR. The Wireless Innovation Forum, working in collaboration with the Institute of Electrical and Electronic Engineers (IEEE) P1900.1 group, has worked to establish a definition of SDR that provides consistency and a clear overview of the technology and its associated benefits.

Simply put Software Defined Radio is defined as:
"Radio in which some or all of the physical layer functions are software defined"

A radio is any kind of device that wirelessly transmits or receives signals in the radio frequency (RF) part of the electromagnetic spectrum to facilitate the transfer of information. In today's world, radios exist in a multitude of items such as cell phones, computers, car door openers, vehicles, and televisions.

Traditional hardware based radio devices limit cross-functionality and can only be modified through physical intervention. This results in higher production costs and minimal flexibility in supporting multiple waveform standards. By contrast, software defined radio technology provides an efficient and comparatively inexpensive solution to this problem, allowing multi-mode, multi-band and/or multi-functional wireless devices that can be enhanced using software upgrades.

SDR defines a collection of hardware and software technologies where some or all of the radio's operating functions (also referred to as physical layer processing) are implemented through modifiable software or firmware operating on programmable processing technologies. These devices include field programmable gate arrays (FPGA), digital signal processors (DSP), general purpose processors (GPP), programmable System on Chip (SoC) or other application specific programmable processors. The use of these technologies allows new wireless features and capabilities to be added to existing radio systems without requiring new hardware.

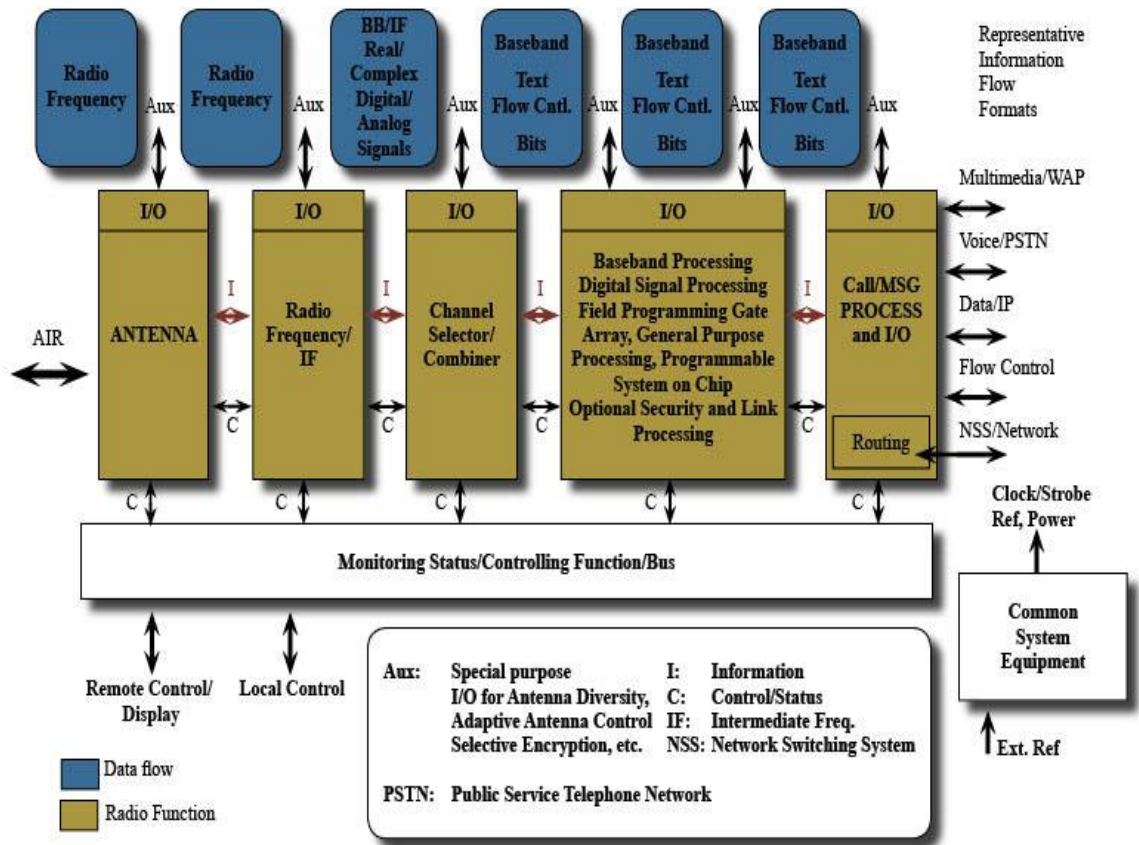


Figure 2.1-Generalized Functional Architecture

One definition of SDR is provided by the SDR forum, is that SDR is the radio that accepts fully programmable traffic and control information and supports a broad range of frequencies, air interfaces, and application software. The SDR forum discriminates between different levels of flexibility in a radio. These are:

- (1) **Hardware Radio (HR):** In a HR, system attributes cannot be changed since the functionality of the hardware radio is fixed. However, this radio can use internal software as long as it cannot be changed externally.
- (2) **Software Controlled Radio (SCR):** This is the radio in which only the control functions are implemented in software. For example, the transmitted power level of a radio can be controlled by software, while all other functions are fixed in hardware. Current radio designs often fall under this category.
- (3) **Software Defined Radio (SDR):** These are radios that provide software control of almost every radio function, including modulation, multiplexing, amplification,

super heterodyne mixers, multiple access and other transmitter and receiver processes. The software should have the capability to add new air interfaces without reloading the entire set of software.

(4) Ideal Software Defined Radio (ISDR): This radio has the same functionality as the SDR, but it does not have an analog front-end (amplification, mixers, etc.), thereby unable to eliminate analog noise and distortions. The analog front-end contains an antenna, analog-to-digital converters (ADCs) and digital-to-analog converters (DACs), directly attached to it.

(5) Ultimate Software Radio (USR): The USR is an ideal, flexible, small, lightweight, low-power radio which is fully programmable.

Software radios use digital techniques, but software controlled digital radios are generally not software radios. The difference between software controlled digital radios and software radios is the total programmability of software defined radio. This programmability includes programmable radio frequency bands, channel access modes, and modulation.

It is obvious that unique definition for the software radio concept may not be possible.

The most common definitions are summed up below and quoted from:

- “Flexible transceiver architecture, controlled and programmable by software.”
- “Signal processing able to replace, as much as possible, radio functionalities.”
- “A system with air interface downloads ability. That is, it is possible to dynamically reconfigure radio equipment by downloadable software, at every level of the protocol stack.”
- “Software realization of terminals.”
- “A transceiver with frequency band and radio channel bandwidth, modulation and coding scheme, radio resource and mobility management protocols, and user applications.”

It appears that in SDR, the parameters of interest can be adapted and changed by the network operator, service provider, and end users. A software defined radio system can operate in multi-service environments. This means that the system is able to offer

services of any already standardized systems or future ones, on any radio frequency band. The system is not constrained to a particular standard. For that reason the software radio system is very flexible. The compatibility of a software radio system with any defined mobile radio standard is guaranteed by its reconfigure ability, which is achieved by DSP processors. These processors implement in real time radio interface and upper layer protocols.

A software defined radio not only transmits and receives signals but it does more in an advanced application. Before transmission, SDR can distinguish the available transmission channel, select suitable channel modulation, direct the transmit beam in the direction of interest, check for proper power level and then transmit the signal. Similarly, on the receive path, apart from just receiving the signal, SDR can characterize the energy distribution in the desired channel and adjacent channels, provide adaptive equalization, null interference, approximate the dynamic properties of the desired signal, decode the channel modulation using appropriate schemes, correct errors through forward error correction (FEC), and hence help in obtaining the desired signal with less bit error rate (BER).

Finally, software radio supports incremental service enhancements through a wide range of software tools. These tools assist in analyzing the radio environment, defining the required enhancements, prototyping incremental enhancements via software, testing the enhancements in the radio environment, and finally delivering the service enhancements via software and/or hardware.

2.3 Software Defined Radio Concepts

In this section, the difference between conventional radio and software defined radio is presented. Also, the architecture of SDR, Rate of adoption, SDR related technologies, its advantages and disadvantages are discussed.

2.3.1 Comparison of Conventional and Software Defined Radio

To compare the functionalities of the conventional radio and SDR, we provide a tabulation of their functions in Table 2.1, and also provide benefits of SDR.

Table 2.1-Comparison between conventional and software defined radios

Conventional Radios	Software Defined Radios
Radio functionalities are primarily defined in hardware with minimum configurability in software.	Radio functionalities are defined in software.
Since the design is dominated by hardware, upgrading the design is not possible.	Software based architecture allows for easy upgrade of the design without abandoning the older design.
The user has to use different mobile devices due to incompatibility of standards.	Global mobility can be achieved by downloading the appropriate air interface thus overcoming the incompatibility of standards.
Multi-function radios design including separate silicon for each system decreases the efficiency and becomes bulky.	Reprogrammability makes SDR to be efficient and compact.
Results in waste of silicon area since each system has to be implemented separately.	Silicon area is conserved by using the same chip to perform a function and changing the configurations during runtime to perform another function.

Benefits of SDR

- A family of radio “products” to be implemented using a common platform architecture, allowing new products to be more quickly introduced into the market.
- Software to be reused across radio "products", reducing development costs dramatically.
- Over-the-air or other remote reprogramming, allowing "bug fixes" to occur while a radio is in service, thus reducing the time and costs associated with operation and maintenance.
- New features and capabilities to be added to existing infrastructure without requiring major new capital expenditures, allowing service providers to quasi-future proof their networks.
- The use of a common radio platform for multiple markets, significantly reducing logistical support and operating expenditures.

- Remote software downloads, through which capacity can be increased, capability upgrades can be activated and new revenue generating features can be inserted.
- Reduce costs in providing end-users with access to ubiquitous wireless communications – enabling them to communicate with whomever they need, whenever they need to and in whatever manner is appropriate.

2.3.2 Architecture of Software Defined Radio

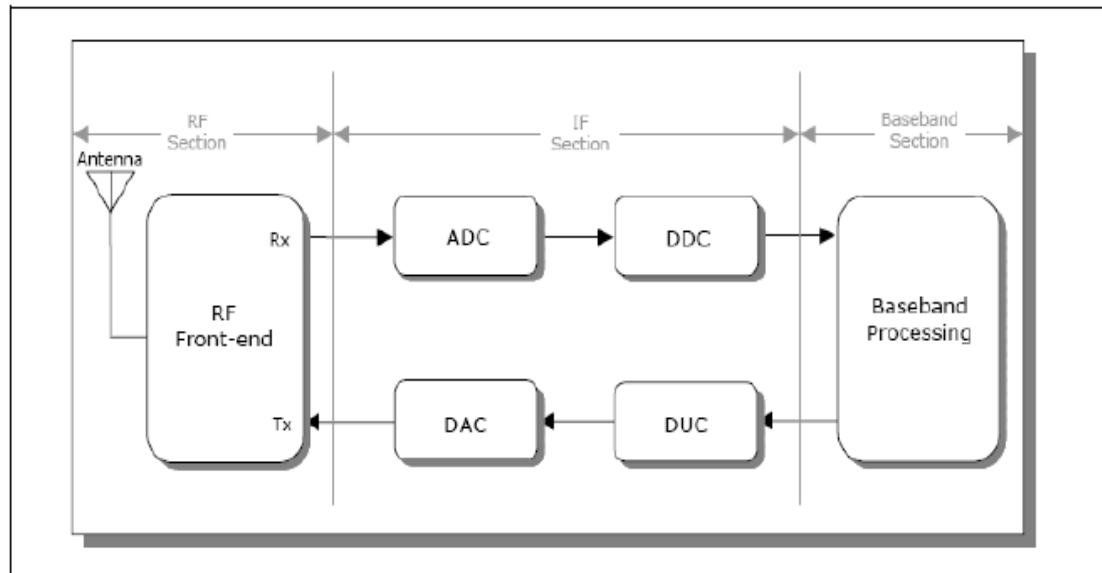


Figure 2.2-Block diagram of a generic digital transceiver

The digital radio system consists of three main functional blocks:

RF section, IF section and baseband section.

The RF section deals with up conversion from IF to RF and down conversion from RF to IF. The ADC/DAC blocks interface between the analog and digital sections of the radio system. The ADC/DAC blocks perform analog-to-digital conversion and digital-to-analog conversion, respectively. DDC/DUC blocks perform digital-down-conversion and digital-up-conversion, respectively.

Additionally it performs modulation and demodulation of the signal. The baseband section performs baseband operations (connection setup, equalization,

frequency hopping, timing recovery, correlation) and also implements the link layer protocol.

The DDC/DUC and baseband processor are implemented digitally in a SDR and they require large computing power. If the baseband section is implemented using ASICs, the function of the radio remains fixed reducing the flexibility of the radio. If DSPs are used for baseband processing, a programmable digital radio (PDR) system can be realized. In other words, in a PDR system baseband operations and link layer protocols are implemented in software. The limitation of this system is that any change made to the RF section of the system will impact the DDC/DUC operations and will require non-trivial changes to be made in DDC/DUC ASICs.

In comparison to that the software-defined radio (SDR) system is one in which the baseband processing as well as DDC/DUC modules are programmable. This is possible because of the availability of smart antennas, wideband RF front-end, wideband ADC/DAC technologies and ever increasing processing capacity of DSPs and general-purpose microprocessors.

In SDR system, the link-layer protocols and modulation/demodulation operations are implemented in software. If the programmability is further extended to the RF section (i.e., performing analog-to-digital conversion and vice-versa right at the antenna) an ideal software radio systems can be implemented. However, the current state-of-the-art ADC/DAC devices cannot support the digital bandwidth, dynamic range and sampling rate required to implement this in a commercially viable manner. Figure2.3 shows a parameter-controlled (Pac) SDR where the control bus supplies parameters for the desired operation.

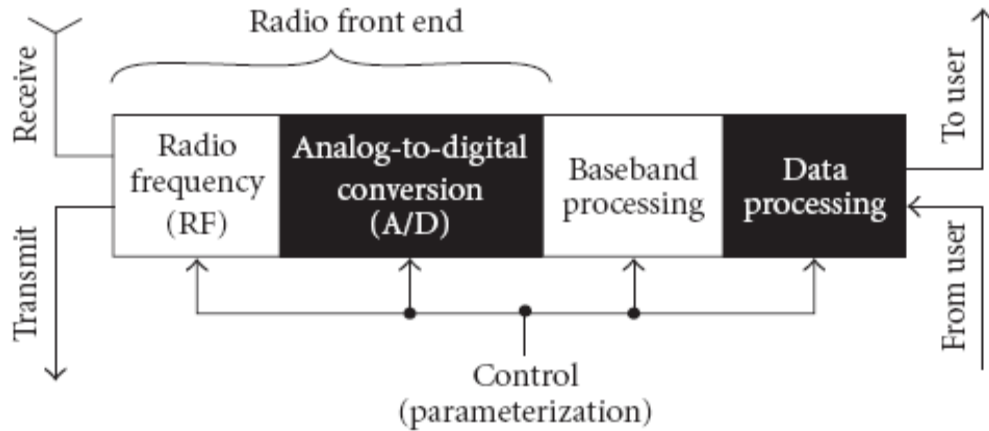


Figure 2.3-Transceiver of an Ideal SDR

Figure 2.4 illustrates the architecture of software components in a typical SDR system. The system uses a generic hardware platform with programmable modules (DSPs, FPGAs, microprocessors) and analog RF modules. The operating environment performs hardware resource management activities like allocation of hardware resources to different applications, memory management, and interrupts servicing and providing a consistent interface to hardware modules for use by applications. In SDR system, the software modules that implement link layer protocols and modulation/demodulation operations are called radio applications and these applications provide link-layer services to higher layer communication protocols such as WAP (Wireless Application Protocol) and TCP/IP.

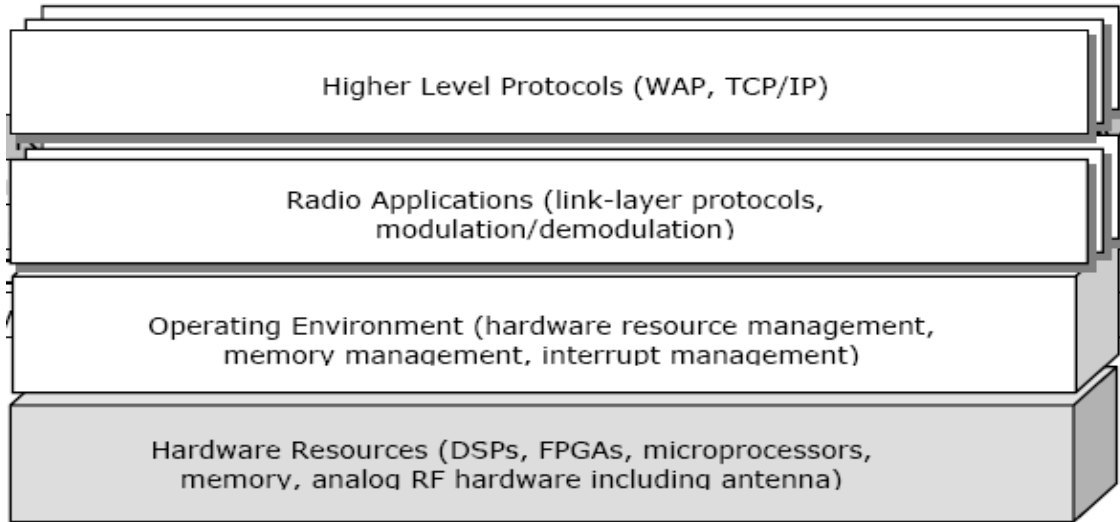


Figure 2.4-Architecture of software components of an SDR

2.3.3 Rate of Adoption and value chain of SDR

The Wireless Innovation Forum commissioned a number of research reports in 2006 to evaluate the adoption of SDR technologies in various markets. The results of these studies demonstrated that, in certain markets, SDR is moving beyond the innovators and early adopters as defined by Geoffrey Moore in “Crossing the Chasm” into the early majority phase defining the mainstream market*. In this phase, adopters select a technology not because it is innovative or visionary but because it has been shown to successfully solve a problem within their specific market.

Examples of SDR adoption illustrating the transition to the mainstream are abundant:

- Thousands of software defined radios have been successfully deployed in defense applications
- Cellular infrastructure systems are increasingly using programmable processing devices to create “common platform” or “multiband-multiprotocol” base stations supporting multiple cellular infrastructure standards

- Cellular handsets are increasingly utilizing System on Chip (SoC) devices that incorporate programmable “DSP Cores” to support the baseband signal/modem processing
- Satellite “modems” in the commercial and defense markets make pervasive use of programmable processing devices for intermediate frequency and baseband signal processing

While these types of systems are often not marketed as “SDR’s”, they utilize and benefit from SDR technologies to solve market specific problems such as; cost of development, cost of production, cost of upgrades and maintenance, time to market in supporting new and evolving air interface standards, or problems associated with network interoperability.

The benefits and anticipated opportunities for SDR technology are having a significant impact on the wireless industry’s value chain. This chain consists of product-based and service-based providers, with value added at each stage, ultimately resulting in SDR end products and services that meet the needs of the end users and subscribers.

Throughout the chain, the providers may be supported by external organizations such as educational institutions, research laboratories, industry standards bodies, investors, tests & verification and government. These supporting organizations provide critical input as development progresses through the chain, ultimately reaching the end user. The detail of the chain and the relationship within the context of the Wireless Innovation Forum membership is outlined below.

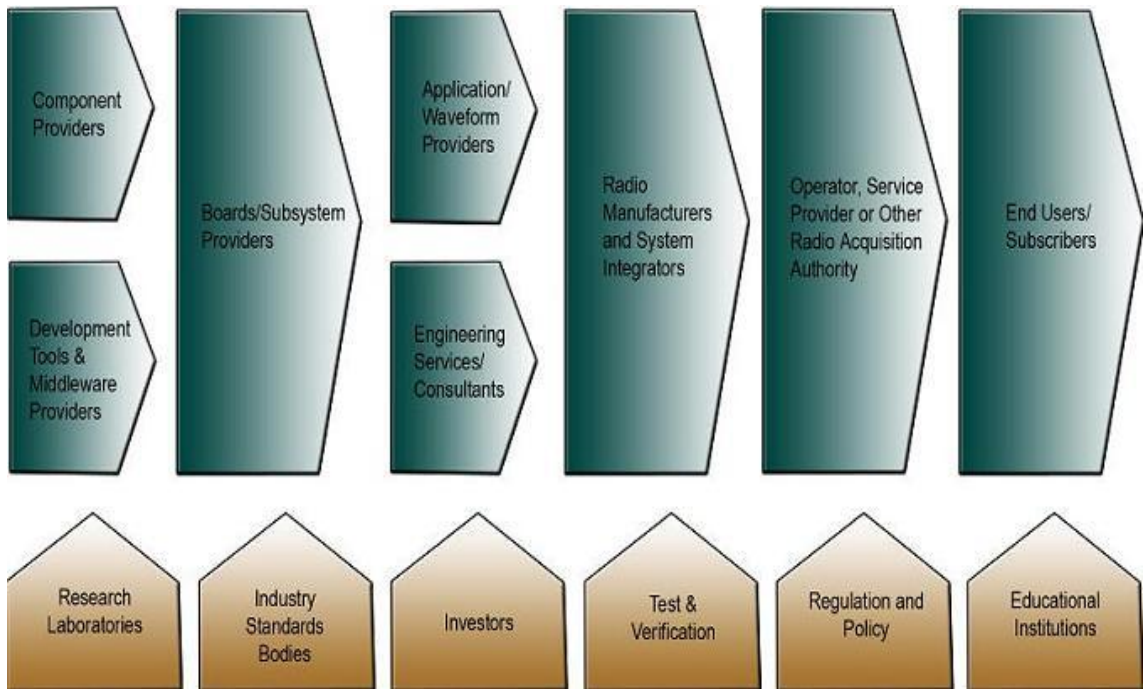


Figure 2.5-SDR Value Chain: Product and Service Based Providers and Supporting Organizations

SDR has far reaching implications within the chain impacting a variety of organizations and industry sectors through the radio frequency (RF) chain (front end components, software developers, chips makers, etc) and throughout business modes (service providers, OEMs, IP holders, etc.). In order to provide viable products and services to meet the future development potential of SDR technology, organizations must look to structure SDR into all levels of the value chain. With successful applications seen in a number of markets, the opportunity to fully engage SDR at all levels of the chain is now.

2.3.4 SDR related Technologies

SDR can act as a key enabling technology for a variety of other reconfigurable radio equipments commonly discussed in the advanced wireless market. While SDR is not required to implement any of these radio types, SDR technologies can provide these types of radio with the flexibility necessary for them to achieve their full potential, the benefits of which can help to reduce cost and increase system efficiencies:

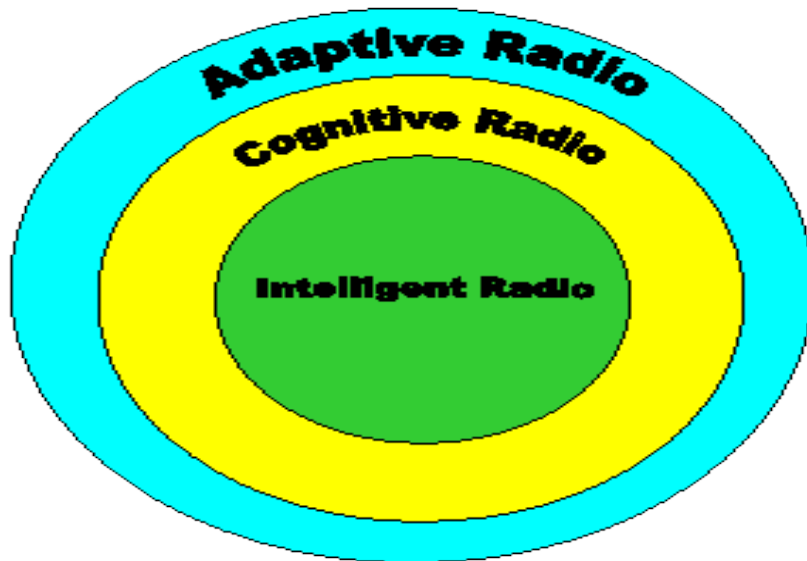


Figure 2.6-Venn diagram of advance wireless technology

Adaptive Radio

Adaptive radio is radio in which communications systems have a means of monitoring their own performance and modifying their operating parameters to improve this performance. The use of SDR technologies in an adaptive radio system enables greater degrees of freedom in adaptation, and thus higher levels of performance and better quality of service in a communications link.

Cognitive Radio

Cognitive radio is radio in which communication systems are aware of their internal state and environment, such as location and utilization on RF frequency spectrum at that location. They can make decisions about their radio operating behaviour by mapping that information against predefined objectives.

Cognitive radio is further defined by many to utilize Software Defined Radio, Adaptive Radio, and other technologies to automatically adjust its behavior or operations to achieve desired objectives. The utilization of these elements is critical in allowing end-users to make optimal use of available frequency spectrum and wireless networks with a common set of radio hardware. This will reduce cost to the end-user while allowing him

or her to communicate with whomever they need whenever they need to and in whatever manner is appropriate.

There are two major subsystems in a cognitive radio; a cognitive unit that makes decisions based on various inputs and a flexible SDR unit whose operating software provides a range of possible operating modes. A separate spectrum sensing subsystem is also often included in the architectural a cognitive radio to measure the signal environment to determine the presence of other services or users. It is important to note that these subsystems do not necessarily define a single piece of equipment, but may instead incorporate components that are spread across an entire network. As a result, cognitive radio is often referred to as a cognitive radio system or a cognitive network.

The cognitive unit is further separated into two parts as shown in the block diagram below. The first labeled the “cognitive engine” tries to find a solution or optimize a performance goal based on inputs received defining the radio’s current internal state and operating environment. The second engine is the “policy engine” and is used to ensure that the solution provided by the “cognitive engine” is in compliance with regulatory rules and other policies external to the radio.

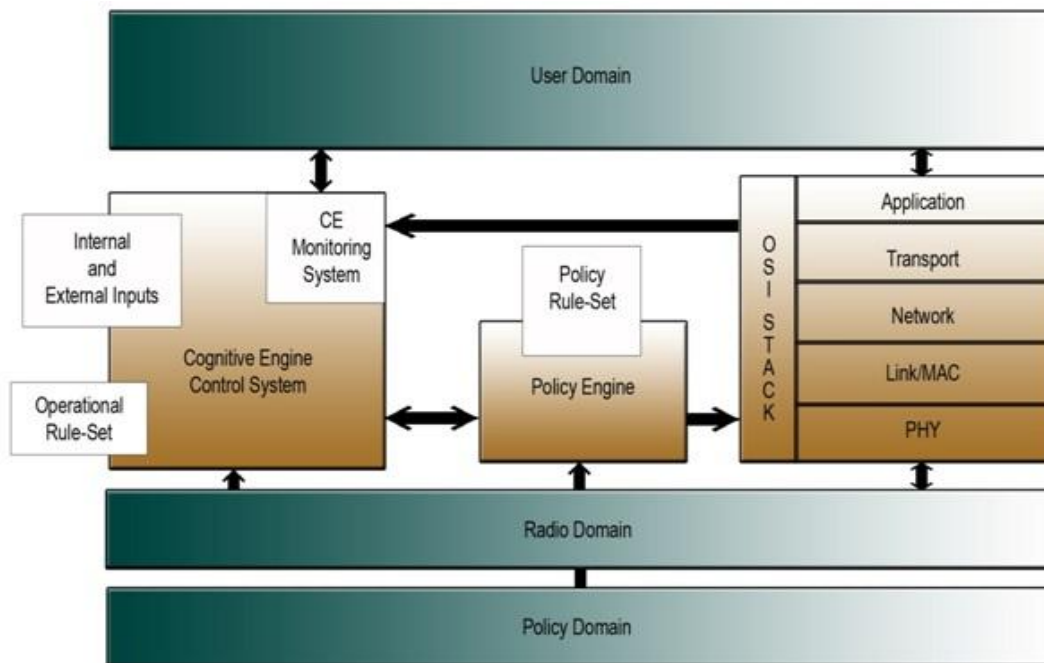


Figure 2.7-Cognitive Radio Concept Architecture

Intelligent Radio

Intelligent radio is cognitive radio that is capable of machine learning. This allows the cognitive radio to improve the ways in which it adapts to changes in performance and environment to better serve the needs of the end user.

In addition to utilizing SDR technologies, adaptive radio, intelligent radio and cognitive radio systems may all support dynamic spectrum access (DSA), allowing the systems to select the frequency spectrum in which they will operate at a given location and over a given period of time to optimize the use of available spectrum and avoid interference with other radios or other systems.

2.3.5 Advantages and Disadvantages of Software Defined Radio

It is believed by many that the successful deployment of SDR will revolutionize the field of communication. One of the advantages of SDR is that it can be changed quickly to support multiple standards. The ability to configure devices, which may be used by many communication systems (e.g., cellular phones, wireless-fidelity (WI-FI) transceivers, frequency modulation (FM) and analog modulation (AM) radios, terminals of satellite communications), will be remarkable.

With SDR, the same piece of hardware will be configured to perform different functions. The reconfigure ability of the platform will ensure hardware reusability. System reprogram ability allows hardware reuse until a new generation of hardware platforms is available. This will provide cost and time savings. Manufacturers will not be limited to reduced hardware platform set. As a consequence, mass production will allow lowered costs.

Another advantage of SDR would be the possibility to improve the software in successive steps, and the correction of software errors and bugs discovered during the operation. In addition, SDR can enhance the interoperability of different systems in many applications such as the military, law enforcement, or search and rescue teams. Incompatibility of radio systems that has always hindered the seamless operation of the military, the law enforcement agencies and many rescue teams, will be eliminated. With

the increase of channel data rates through multiplexing and spectrum spreading, SDR could be used in cellular networks, GSM based PCS network, and future generation systems network. A new approach to wireless base station design using SDR has the potential of offering significant benefits such as reduced size, complexity, and power consumption. More importantly, SDR can support a variety of air interface standards, modulation schemes and protocols, simultaneously. Some commercial telephone service providers have begun expressing interest in the SDR economic benefits in long term. More highlights on the benefits of SDR are given in Section 1.2.

While SDRs offer benefits as outlined above, there are drawbacks in the design and implementation of SDR. Those include:

- The difficulty of designing software for various target systems or standards.
- The difficulty of designing air interfaces to digital signals and algorithms for different standards.
- The problem of poor dynamic range in some communication systems design.

2.4 Software Defined Radio Implementation Platforms

As indicated above, the global trend in the communication industry is to replace hardware by software, because of software flexibility. Real time software defined radio design can be implemented using a variety of digital hardware namely (a) field programmable gate arrays, (b) digital signal processors, (c) application specific integrated circuits and (d) general purpose processors. The different implementation platforms are shown in Figure 2.8. All the four platforms shown in Figure 2.8 possess a level of reprogram ability or reconfigure ability (i.e., the ability to modify the hardware or software).

The DSP platform is essentially a microprocessor based system optimized for digital signal processing applications. DSPs can be programmed repeatedly with a high level language such as C, MATLAB. Modifications and upgrades to the design are made through these high level languages, thus reducing the design times for each iteration. The flexibility offered by the digital signal processor comes at the cost of efficiency. When there are several computations to be performed, parallel executions of

these computations will slow down the rate at which data is processed and this leads to the use of more than one DSP. This solution is limited since synchronizing several DSPs is difficult.

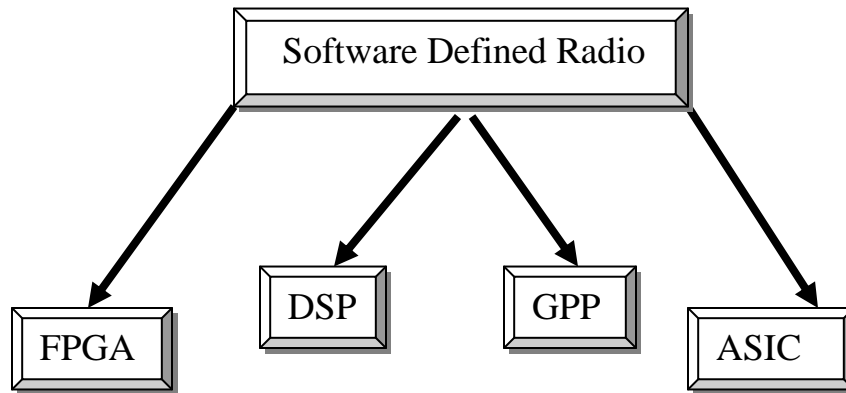


Figure 2.8-Hardware implementation platforms for SDR

A field programmable gate array is a general purpose integrated circuit that is programmed by the designer rather than the device manufacturer. A unique feature of FPGA is that it can be reprogrammed, even after it has been deployed into a system. Field programmable gate array is programmed by downloading a configuration program (bit stream) into the static on-chip random access memory. This is similar to the object code of a microprocessor, in which the bit stream is the product of compilation tools that translate the high level abstractions produced by a designer into equivalent but low level executable code.

Field programmable gate arrays were designed for multilevel circuits to handle complex circuits on a single chip. Since they are reprogrammable, their configurations can be easily changed to upgrade systems or correct system bugs, making it ideal for prototyping.

Field programmable gate arrays are now used in various configurations, as in multimode systems, and are very useful in meeting the needs of a software defined radio implementation.

Application specific integrated circuits (ASICs) implement the system circuitry in fixed hardware, resulting in the most optimized implementation of the circuit in terms of speed and power consumption. However, ASIC design requires sophisticated circuit design and layout software tools. Also, as the name implies, their use are for specific application and not subject to modification at a later date.

A general purpose processor is similar to DSP as a hardware platform in the design of software defined radio. Like DSP, it offers flexibility and ease of design. Radio functionalities can be implemented in high level languages such as C and C++. Designers can use the familiar approaches of object oriented programming and debugging to develop real time software radio systems. This increases productivity significantly and thus reduces system development time.

Digital signal processor is the most generalized type of hardware that can be programmed to perform various functions, while ASIC is the most specialized and can be used only in specific application. Field programmable gate arrays offers a compromise in flexibility between ASIC and DSP platforms.

In general, these hardware components constitute design spaces that trade flexibility, processing speed, and power consumption among other things. There should be a tradeoff between the maximum flexibility and high power consumption of DSP platforms to minimum flexibility and less power consumption of ASICs compared to FPGAs, which have good hardware optimization. Recently, FPGAs have become increasingly popular due to their ability to reduce design and development cycle time. Furthermore, latest FPGAs come with intellectual property (IP) cores, which are used for specific applications.

There are other advantages of using FPGAs instead of DSPs for signal processing in commercial telecommunication systems. The power consumption is lower; the size is smaller, quicker to use and the costs are much lower in comparison to DSPs. Since the chip can be reused after fixing the bugs or upgrading a system, they are ideal for prototyping and testing the circuit design. Since FPGAs are reprogrammable, one chip

can be configured to perform more than one function and the configurations can be changed during run time.

2.5 Technical Challenges

This section discusses the technical issues, which have to be solved before software radio can be commercially available. The important technical issues involved in the development of a software radio system are as follows:

- In transceivers, the border between analog and digital domain should be moved closer, as much as possible, towards the RF. This requires ADC and DAC wideband converters placed as near as possible to the antenna. Increasing the border between the analog and digital domain is not exclusively for software defined radio. Much research has been carried out in the wideband transceiver realization. The primary goal of this transceiver was to extend the digital domain at the IF stage and keeping the RF stage analog.
- The process of replacement of dedicated hardware in communication systems with DSPs or FPGAs should be further developed. In other words, we need to define the radio functionalities as much as possible in software. This opens the way to two possible horizons: software implementation of baseband functions, such as coding, modulation, equalization and pulse shaping; and reprogram ability of the system to guarantee multi-standard operation. Though DSP technology has been used in implementing the baseband processing in base stations, it is not possible to categorize it as SDR since not all baseband functionalities are implemented in DSPs. Also, the software is limited and pre-loaded; therefore the system is constrained to a specific radio interface and cannot be reconfigured. Hence, implementing communication functions in software presents a major challenge in practical systems.
- Analog-to-digital and digital-to-analog conversions for the ideal software defined radio are difficult to achieve. In practice, the selection requires trading power consumption, dynamic range and bandwidth. Current conversion technology is limited and is often the weak link in the overall system design. There are post

digitization techniques based on multi rate digital signal processing that can be used to improve the flexibility of the digitization process.

- Power management is also a major challenge. For example, sleep modes of DSPs or other hardware save power but introduce a probability that the radio will be asleep during a paging message. A possible solution is a structured timing of paging messages, which reduces the probability of a miss, and further conserves battery life.
- The clock generation and distribution is another challenge in SDR design. Every standard such as GSM or IS-95 has its own clock rate. Using one reference oscillator per standard may increase parts count, increase complexity, and therefore cost. A single master clock may use the least common multiple (LCM) of the required clocks, but this leads to a high clock rate, which is power inefficient. A possible solution is to use normalize standards to avoid clock rates with large LCMs.
- Receiver complexity is typically four or more times the transmitter complexity. Thus, the receiver architecture has a first order impact on handset cost. The challenge is to develop a simple receiver. With the current technology, the support of many standards leads to complex and power inefficient solutions. Application specific integrated circuits are power efficient but inflexible. Field programmable gate arrays could be a possible solution. Hybrids of platform implementation could be utilized.
- The ideal radio frequency stage for SDR should incorporate flexibility in selection of power gain, bandwidth, dynamic range, etc. Achieving strict flexibility is impractical and trade-offs must be made. These are the major challenges that must be addressed before full realization of SDR. Besides these important issues there are other challenges, which have to be solved like software architecture selection, hardware architecture selection etc., which are not discussed in this report.

2.6 Summary

The fundamentals of SDR are presented in this chapter. It also dealt with definition and concepts of SDR. The difference between the conventional radio and SDR, characteristics, architecture were also presented. The choices of hardware available for real time implementation and technical challenges involved in implementation were discussed.

CHAPTER 3

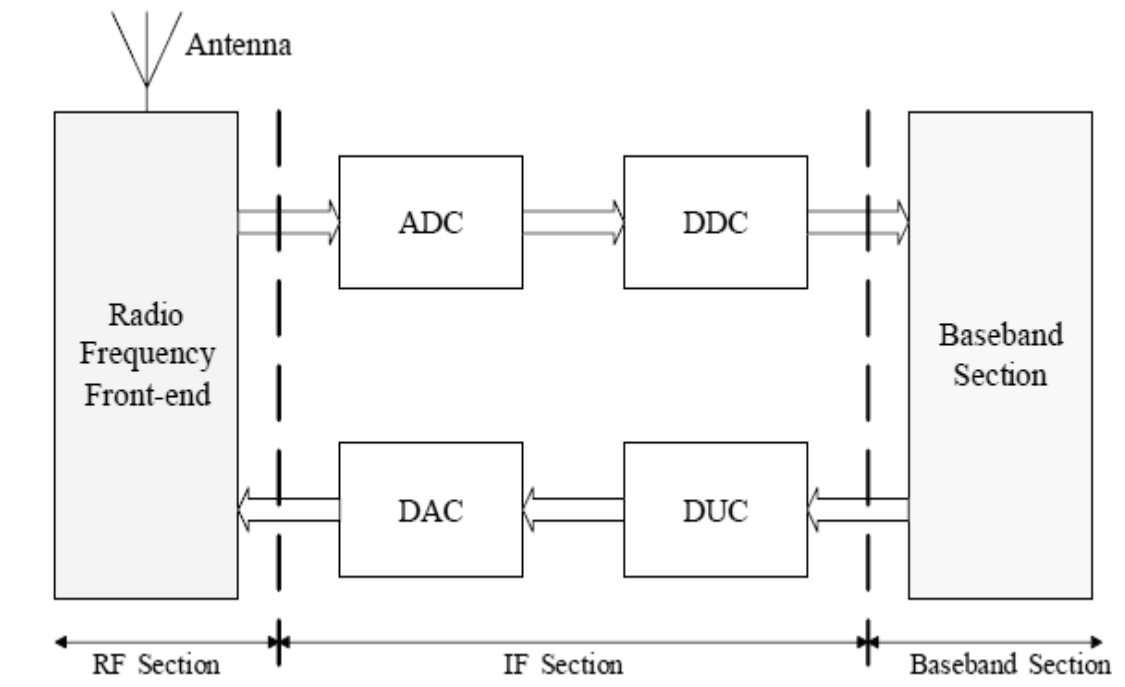
FRAMEWORK FOR SYSTEM DESIGN

3.1 Wireless Communication System Model for Software Defined Radio

The generic wireless communication system consists of a transmitter, channel and a receiver. The functional block diagrams of the digital transceiver are shown in Figure 3.1.

3.1.1 Radio Frequency Section

The radio frequency (RF) section is responsible for transmitting and receiving the RF signal and converting the RF signal into an intermediate frequency (IF) signal. The RF section consists of antennas and analog hardware modules. The RF front-end is designed in such a way to reduce interference, multipath and noise. The RF front-end on the receive side performs RF amplification and down conversion from RF to IF. On the transmit side, the RF section performs analog up conversion and RF power amplification.



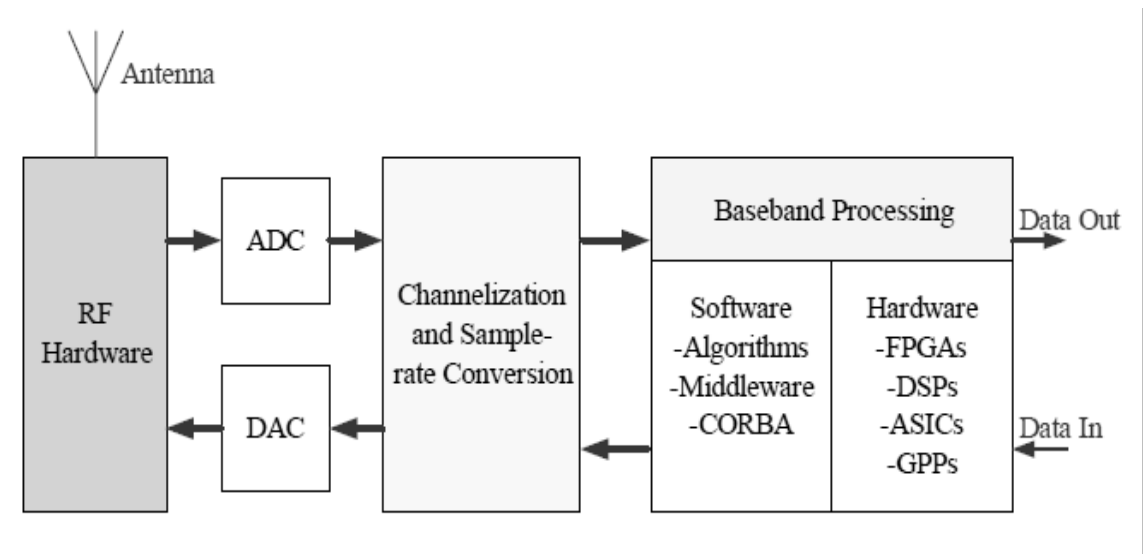


Figure 3.1-functional block diagrams of the digital transceiver

3.1.2 Intermediate Frequency Section

The ADC/DAC performs analog-to-digital conversion on the receive path, and digital to analog conversion on the transmit path. These blocks interface between the analog and digital sections of the radio system. Usually, the above conversion takes place in the IF stage. Digitizing the signal with an ADC eliminates the last stage in the conventional model, where problems such as carrier offset and imaging are encountered.

As the names imply, the digital down converter (DDC) and digital up converter (DUC) perform digital down conversion on the receive path and digital up conversion on the transmit path, respectively. Digital filtering and sample rate conversion are often needed to interface the output of the ADC to the processing hardware at the receiver. The same happens in the reverse direction in the transmitter, where digital filtering and sample rate conversion are necessary to interface the digital hardware to the DAC that converts the modulated waveform to an analog waveform.

3.1.3 Baseband Section

The baseband section performs operations, such as error correction, equalization, frequency hopping, modulation, demodulation, and spreading, dispreading and timing

recovery. Forward error correction is a method of obtaining error control in data transmission in which the transmitter sends redundant data and the receiver recognizes only the portion of the data that contains no apparent errors. Equalization is done to counteract the inter symbol interference in the channel. Frequency hopping and spreading is used to minimize unauthorized interception or jamming of the communication system by repeated switching of frequencies during radio transmission using a specified algorithm. In a wireless communication system, many modulation techniques such as MPSK, QPSK, DPSK, etc., are used. In this report, the QPSK and BPSK modulation and demodulation techniques are dealt. More details on the specific blocks that were implemented in this report are provided in Chapter 4.

The DDC, DUC and the baseband processing requires large computing power and these modules are generally implemented using DSPs, FPGAs, and ASICs. The advantages of using the above platforms in SDR were discussed in Chapter 2.

3.2 Design Process for Reconfigurable Computing

This section discusses the design process for reconfigurable computing, which helps in implementing a wireless communication system model in SDR. Field programmable gate arrays are currently being used in DSP applications, though there is no simpler way to create fast DSP applications using FPGAs. Combining FPGA and DSP technologies is difficult because DSP designers primarily use MATLAB or C/C++ to specify systems, whereas FPGA designers use very high speed integrated circuit (VHSIC) hardware description language (VHDL) or Verilog. The only common approach involves block diagram system model.

Though the DSP algorithms and FPGA architecture are based on two different implementations backgrounds, they must work together to make an effective reconfigurable system. In FPGA design, the DSP algorithms may be modified to obtain the best possible FPGA implementation, and FPGA implementation must be verified to match the original specification given by the DSP algorithm. This requires a constant exchange of information about simulation results, design performance, DSP algorithm changes, and implementation results throughout the design process. Deciding on a single

tool and language that meets the requirements of SDR design specification can be difficult.

A general overview of the design process for reconfigurable computing is given in Figure 3.2.

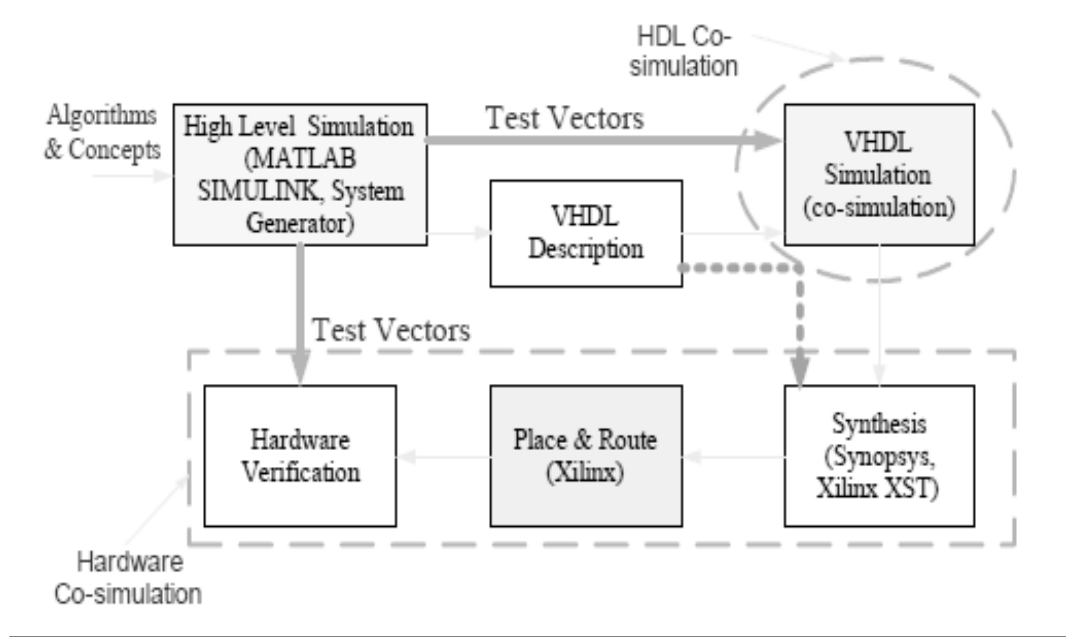


Figure 3.2-Design process for reconfigurable computing

3.2.1 High Level Simulation

The algorithms and concepts used to define the system are modeled using high level software languages like MATLAB, SIMULINK and C. The Xilinx’s System Generator for DSP is a new tool, which comes with a predefined block set along with MATLAB SIMULINK software packet and can be used to implement the algorithms. These high level languages can also be used to verify the accuracy of the algorithms. However they do not directly aid in the hardware implementation.

MATLAB is widely used by many DSP algorithm developers. It is considered the best environment for algorithm development and debugging because of its built-in functions and toolbox extensions for communications, signal processing and wavelet

processing. In addition to the intellectual property functions provided in MATLAB, the software packet is uniquely adept with vector- and array-based waveform data at the core of algorithms, which is suitable for applications such as wireless communications and image processing. MATLAB SIMULINK is fully integrated with the MATLAB engine for visual data flow environment for modeling and simulation of dynamic systems. In addition to the graphical block editor, event-driven simulator, and extensive library of parameterizable functions, it has block sets for DSP, communications, wavelets and many more. MATLAB SIMULINK is used in this report as the high level development tool in the design process.

Xilinx System Generator is a system-level modeling tool that aids in FPGA hardware design. It extends SIMULINK capabilities in many ways to provide a modeling environment well suited for hardware design. System Generator for DSP is a tool for developing and debugging high performance DSP systems based on advanced Xilinx FPGAs. System Generator for DSP and MATLAB SIMULINK tool, provide the graphical design environment commonly used by DSP architects and FPGA designers.

Xilinx's System Generator for DSP tool was the first tool to bridge the gap between DSP and FPGA applications. System Generator along with SIMULINK is a powerful visual data flow environment ideally suited for modeling and simulating DSP algorithms, and allows the developer to automatically generate bit- and cycle-accurate hardware implementation from the system model.

System Generator automates the design process, debugs, and implements and verifies the Xilinx-based FPGAs. It comes with DSP core libraries for high-level modeling and automatic validation code generation, and also provides a high-speed hardware description language (HDL) co-simulation interface, system-level resource estimation, and high-speed hardware co-simulation interfaces for design verification using FPGA hardware platforms.

System Generator provides high-level abstractions that are automatically compiled into FPGA bit stream. It is delivered both with a predefined Xilinx block set library and other languages such as VHDL which are commonly used in FPGA

platforms. Finally, it facilitates the design at the system level, and allows simulation, implementation, and verification within the same environment, usually without writing a single line of HDL code or even looking at the Xilinx integrated software environment (ISE) tools.

In spite of these advantages, System Generator fails to satisfy certain needs of the DSP algorithm developing functions like handling matrix operations and vector based processing. Examples of such algorithms include linear algebra, which involves matrix inverse and factorization operations, and complex number operations such as calculating magnitude and angle, and normalizing complex numbers.

Test vectors can also be created using the System Generator. To construct test vectors, System Generator simulates the design in SIMULINK, and saves the values of the outputs. These test vectors are later used to check the differences between the SIMULINK simulation, HDL simulation and hardware implementation of the model.

3.2.2 VHDL Description

The system that is modeled using System Generator can be compiled into low level representations. That is, the algorithms used to model the system can be broken into processes and coded in VHDL, which gives the description of the hardware. More precisely, it describes the architecture of the system i.e., its components and interconnections. The VHDL description results in a collection of HDL files that implement the design and are later used for HDL and hardware co-simulation. If required, test benches can also be created with other descriptions of the model.

One of the most important applications of VHDL is to capture the performance specification for a circuit, in the form of a “test bench”. Test benches are VHDL descriptions of circuit inputs and corresponding expected outputs that verify the behavior of a circuit over time. Test benches are an integral part of any VHDL project and should be created in tandem with other descriptions of the circuit.

3.2.3 HDL Co-Simulation

Simulation may be defined as the process of verifying the functional characteristics of models at any level of abstraction. VHDL simulation verifies the functionality of the system i.e., given the expected inputs and test whether the outputs are as expected or not. A VHDL test bench and data vectors, which has been created by System Generator for DSP represents the inputs and expected outputs seen in the MATLAB SIMULINK simulation, and allow the designer to easily see any discrepancies between the SIMULINK and VHDL simulation results. ModelSim is required, when HDL co-simulation is done. ModelSim provides a complete HDL simulation environment that enables to verify the functional and timing models of the design, and the VHDL source code.

3.2.4 Hardware Co-Simulation

Hardware co-simulation capability accelerates simulation and verification of design in hardware. System Generator's hardware-in-the-loop co-simulation. Interfaces make a push-button flow and bring the full power of MATLAB and SIMULINK analysis functions to hardware verification.

Once VHDL has been generated by System Generator and before the design is implemented in FPGA, it is necessary that it is synthesized for optimal FPGA implementations. In synthesis, the conceptual HDL design definition is used to generate the logical or physical representation for the targeted silicon device. That is, synthesis maps the HDL to the gate level representation. The VHDL modules can be transferred to the hardware using Xilinx synthesis technology (XST) synthesis tool, which comes with Xilinx's ISE the next step, is to place and route the design in order to verify it on the FPGA. This is achieved using the Xilinx's ISE implementation tools. The place and route function in FPGA design places the synthesized subsystems into FPGA locations and makes connections between these subsystems, enabling their operation as an integrated system.

Placing and routing is followed by hardware verification. The design is implemented on the desired hardware. Hardware verification checks if the module created in high level simulation works well on the desired FPGA device. Test vectors are used to check any discrepancies between the simulation and the hardware implementation.

3.3 Test-bed Implementation

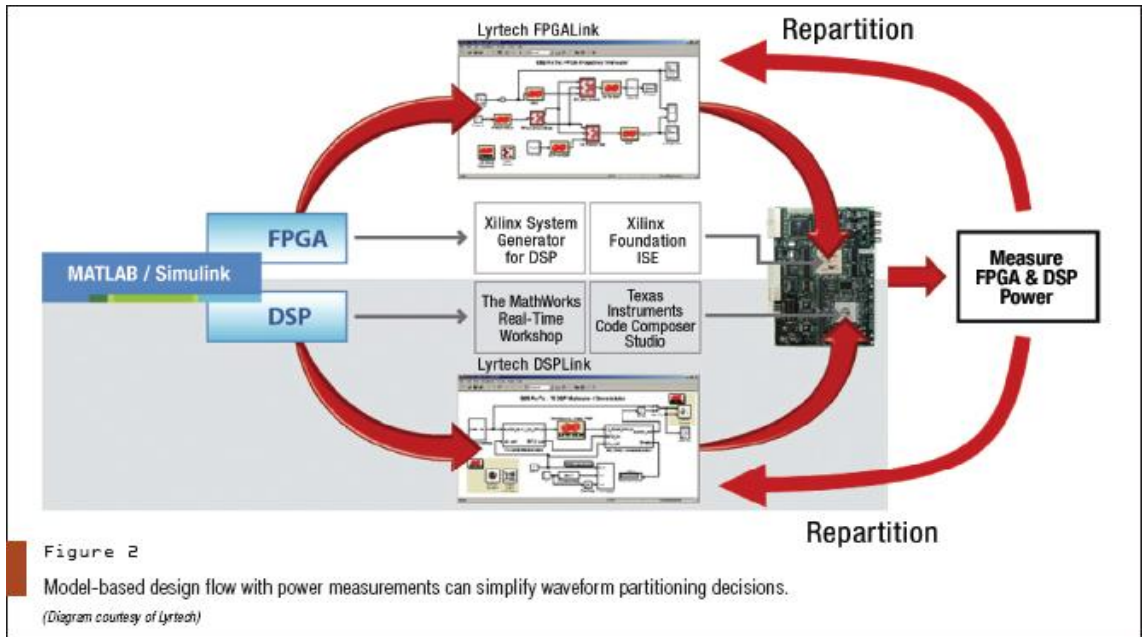


Figure 3.3-Illustration of test-bed implementation process

Using MATLAB SIMULINK along with Xilinx System Generator and the Xilinx ISE synthesis and implementation tool, it is possible to implement DSP designs in FPGA. As a plug-in to the MATLAB SIMULINK modeling software, the Xilinx System Generator provides a bit accurate model of FPGA circuits and automatically generates a synthesizable VHDL code including test bench. This synthesized VHDL design can be used for implementation in the Xilinx’s FPGAs platform. Figure 3.3 illustrates the test-bed implementation of the design process for reconfigurable computing. The design implementation is described below.

3.3.1 Simulation with SIMULINK and System Generator

In this report, the algorithm is designed and simulated using Xilinx System Generator system level tool. For an exact representation of the FPGA implementation, the Xilinx block set in MATLAB SIMULINK is used. The Xilinx block set enables bit-true and cycle-true modeling and includes common parameterizable blocks such as finite impulse response (FIR) filter, fast Fourier transform (FFT), direct digital synthesizer (DDS), multipliers, and much more. The following are the key steps in the design simulation process using MATLAB SIMULINK and System Generator.

- Start the design by implementing the Xilinx blocks in the MATLAB SIMULINK model design.
- Select the Xilinx System Generator block and add it on the top of the design hierarchy.
- “Gateway In” and “Gateway Out” blocks are used to define the inputs and outputs to the Xilinx design. Xilinx gateway blocks automatically convert the double precision floating point numbers from the MATLAB SIMULINK environment into the fixed point numbers for the Xilinx environment. All the system components inside the gateway blocks should be Xilinx blocks only. However, any other MATLAB SIMULINK blocks such as scope, scatter plots and eye diagrams can be used to interface with Xilinx design and represented in system level. In this report, we used all three representations in the simulation.
- Then the design can be simulated and the outputs can be verified using visual output blocks like scopes or by writing the output to the MATLAB workspace.

3.3.2 HDL Co-Simulation procedure

A system level design can be converted to the gate level representation using System Generator, which will automatically generate the VHDL code for all Xilinx blocks contained in the hierarchy. Additionally, automatic generation of test bench enables design verification upon implementation. The following are the key steps in the HDL co-simulation process.

- Double click on the System Generator block and bring up the graphical user interface (GUI). This is illustrated in Figure 3.4, which shows the System Generator dialog box. Set different options, such as targeted FPGA family, test bench generation and IP core generation. Notice that, the compilation type can be selected for HDL simulation or hardware co-simulation. The tool used for synthesis can be chosen from a choice of synthesis tools Xilinx's XST, Simplify Pro and Mentor Graphics. Also, HDL can be chosen either as VHDL or Verilog.

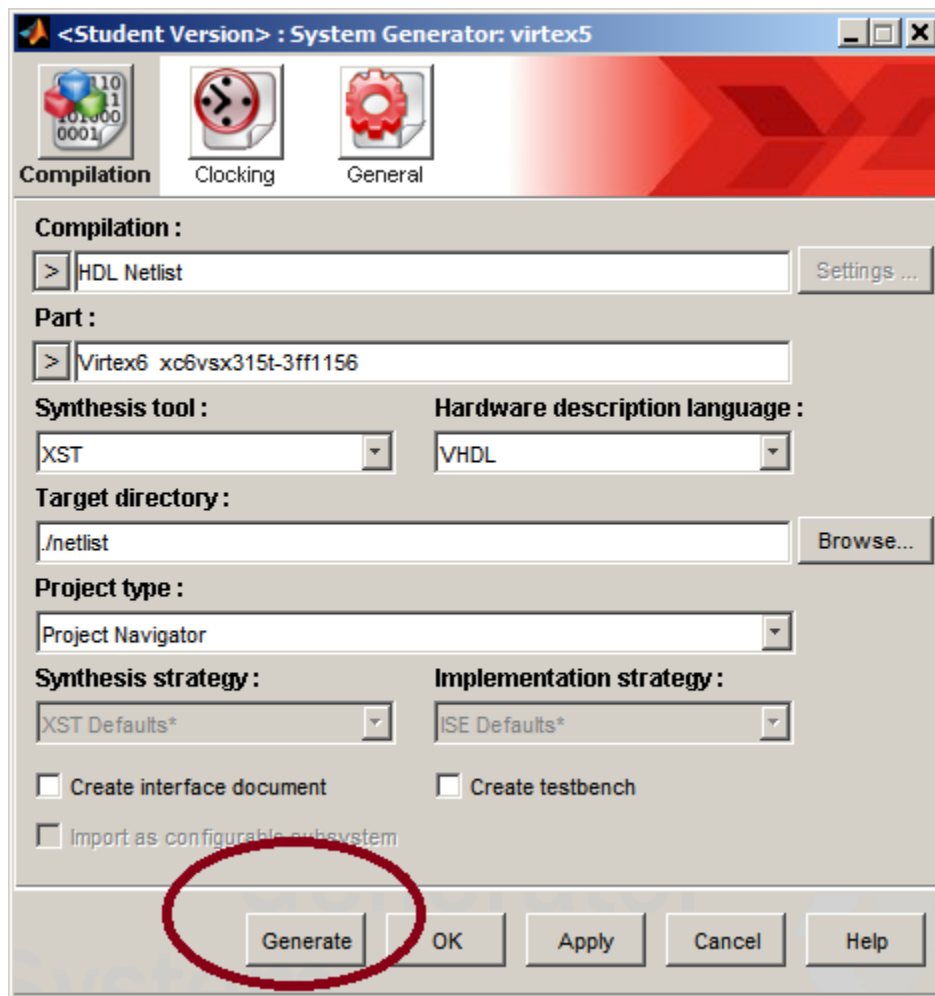


Figure 3.4-System Generator dialog box

- The next step is to run the System Generator by initiating the “Generate” button. This creates a top-level VHDL file, automatically generating IP cores using Xilinx Core Generator System or generates synthesizable VHDL. A VHDL test bench and data vectors are created if selected. These vectors represent the inputs and expected outputs stated in the MATLAB SIMULINK simulation step, and allow additional verification using a behavioral simulator. The design can be synthesized or a VHDL functional simulation can be run.

3.3.3 Hardware Co-Simulation procedure

The key steps involved in hardware co-simulation are similar to that in HDL co simulation. The System Generator will automatically synthesize, and place and route the design on the target FPGA platform upon selecting the appropriate options, such as compilation type, target FPGA, synthesis tool, and so on. The key steps in the hardware co-simulation process can be summarized as follows.

- The hardware co-simulation platform can be chosen from the System Generator dialog box. When the compilation target is selected, the fields on the System Generator dialog box are automatically configured with settings appropriate for the selected compilation target.
- After initiating the “Generate” button, the code generator is invoked and produces an FPGA configuration bit stream for the design that is suitable for hardware co-simulation. System Generator not only generates the HDL and net list files for the model during the compilation process, but it also runs the downstream tools necessary to produce an FPGA configuration file. The compilation process using code generator is shown in next chapter 6.
- After the FPGA configuration bit stream is created, a new hardware co-simulation block is created by the System Generator and stored in the MATLAB SIMULINK library. Hardware co-simulation blocks can be used in the design with other MATLAB SIMULINK blocks. When the hardware co-simulation block is simulated, it interacts with the underlying FPGA platform and facilitates the design implementation and verification of the desired FPGA.

In this report, only hardware co-simulation is performed using Xilinx Spartan 3E FPGA (XC3S500E-4FG320C) platform.

3.4 Summary

This chapter is a description of the general design and simulation processes for implementing a wireless communication system on a SDR. In this chapter, the framework for system design used for SDR implementation was discussed. The design process involved in implementing the system on a reconfigurable platform has been presented. The steps involved in simulating and implementing the system in real time has been discussed. The actual implementation is presented in the chapter 6.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 Environment Required

Before we start doing wireless communication system design for integrating FPGA with multicore SDR development platform, our computer must meet the requirements outlined below.

Table 4.1-The hardware and software requirement

Operating system	Hardware
Windows 7	<ul style="list-style-type: none">• IBM-compatible desktop or laptop• Processor: Pentium i3 or better• RAM: 3 GB• Hard disk drive: 40 GB of free space or more• Display: 800 × 600 pixels or more
Software	
ARM/DSP development software <ul style="list-style-type: none">• Texas Instruments, Code Composer Studio	
FPGA development software <ul style="list-style-type: none">• Xilinx, ISE Foundation• Xilinx Synthesis Technology (XST)	
Model-based design software <ul style="list-style-type: none">• The Math Works, MATLAB 7.11 (R2010b)• Simulink• Real-Time Workshop• Signal Processing block set and toolbox• Xilinx, System Generator for DSP	

4.2 System Model for Simulation

In this report, software definable wireless communication system is designed, simulated and implemented. MATLAB SIMULINK and Xilinx's System Generator are used as high level modeling tools in the design process. Simulation of the system with these tools forms the first step of the design process for reconfigurable computing as discussed in Chapter 3. The transmitter section of the baseband is implemented in Xilinx

Spartan 3E FPGA (XC3S500E-4FG320C) hardware board using the hardware co-simulation process, which is used to analyze the system in real time.

In the baseband section of the communication system, the transmitter consisting of the convolutional encoding, puncturing, modulation and spreading are simulated. The receiver side consists of despreading, demodulation, depuncturing and Viterbi decoding. Figure 4.1 shows the software definable section of the communication system model implemented in this report.

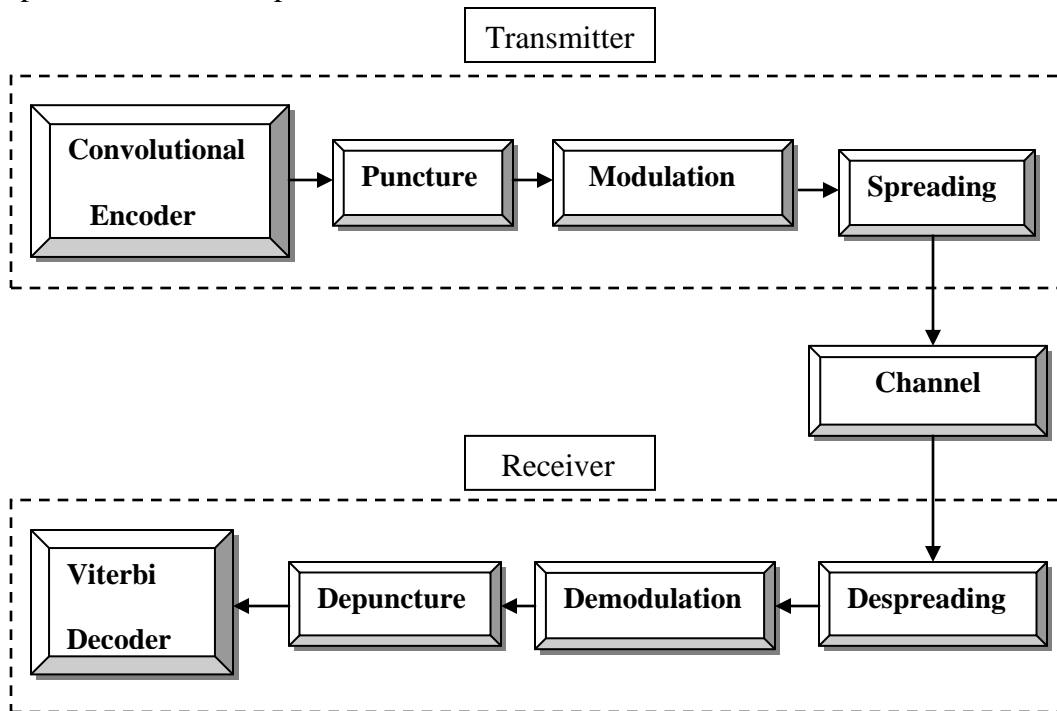


Figure 4.1-Software definable baseband communication system

4.3 The System Generator Design

System Generator works within the Simulink model-based design methodology. Often an executable spec is created using the standard Simulink block sets. This spec can be designed using floating-point numerical precision and without hardware detail. Once the functionality and basic dataflow issues have been defined, System Generator can be used to specify the hardware implementation details for the Xilinx devices. System Generator uses the Xilinx DSP block set for Simulink and will automatically invoke Xilinx Core Generator to generate highly optimized net lists for the DSP building blocks.

System Generator can execute all the downstream implementation tools to product a bit stream for programming the FPGA. An optional test bench can be created using test vectors extracted from the Simulink environment for use with ModelSim or the Xilinx ISE Simulator.

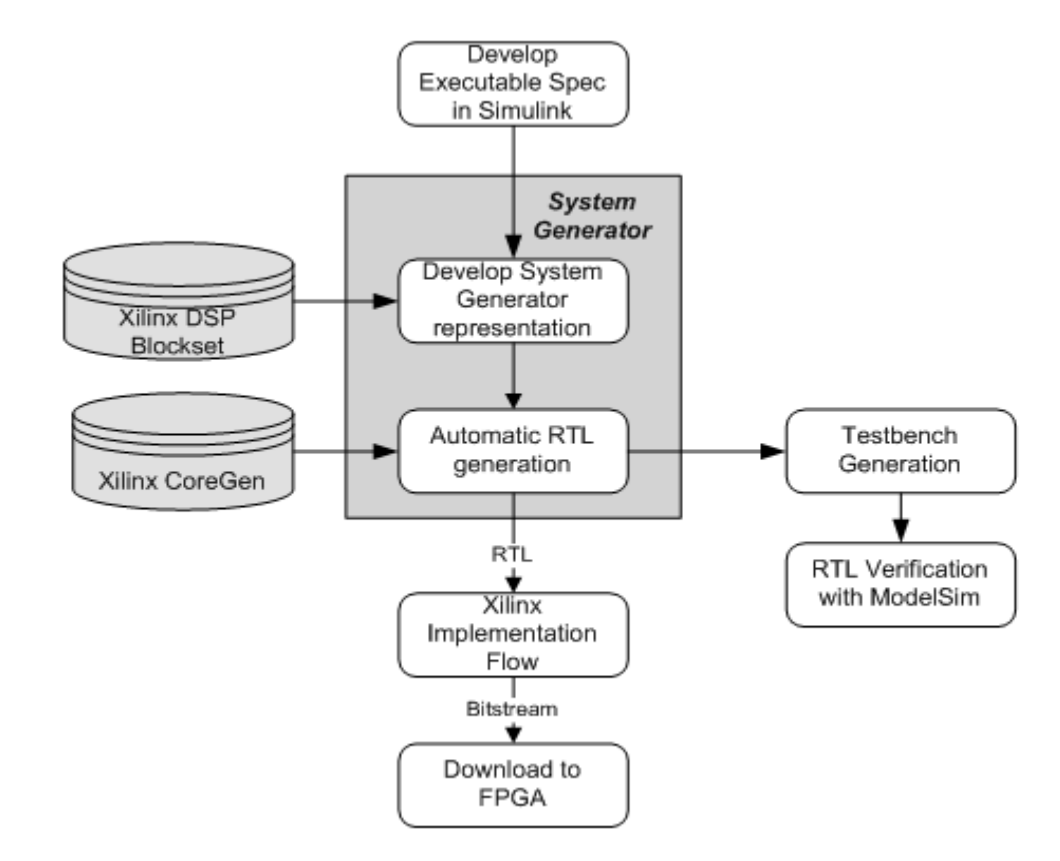


Figure 4.2-The System Generator design flow

When designing a system including FPGA part, we have to add System Generator block when using Xilinx block set to construct a model. This block can let user set the frequency of FPGA and file saving dictionary ...etc. The Resource Estimator block provides fast estimates of FPGA resources required to implement a System Generator subsystem or model.

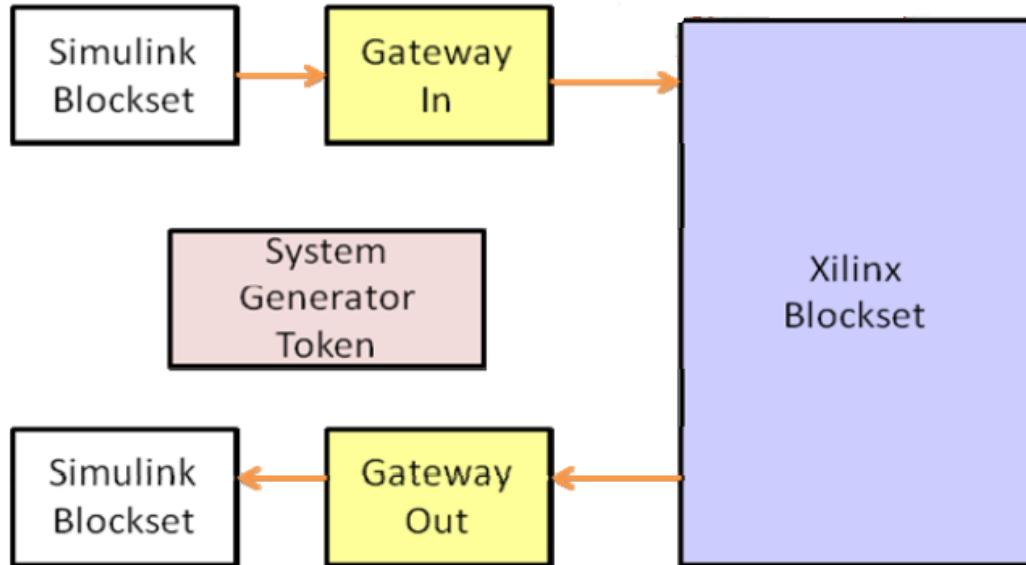


Figure 4.3-Basic architecture of design with Xilinx block set

Two blocks called Gateway In and Gateway Out defines the boundary of the FPGA from the Simulink simulation model. The Gateway In block converts the floating point input to a fixed-point number, and the Gateway Out converts the FPGA outputs back to double precision.

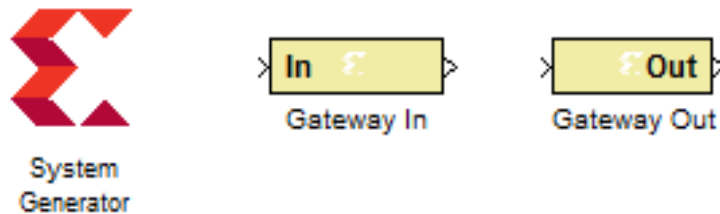


Figure 4.4-Three essential Xilinx blocks

4.4 Convolutional Encoder & Virterbi Decoder

The convolutional encoder of IEEE 802.16 Wireless OFDM PHY standard, which shall have native rate of 1/2, a constraint length equal to 7, and shall use the generator polynomials codes shown in Equation to derive its two code bits:

$$\begin{cases} G_1 = 171_{OCT} & \text{for } X \\ G_2 = 133_{OCT} & \text{for } Y \end{cases}$$

For the 7th mode of channel coding, the Convolutional Code (CC) code rate is $\frac{1}{2}$ is used in this project. The puncture of code rate $\frac{1}{2}$ is listed in the Table 4.2

Table 4.2-The convolutional code with puncturing configuration

	Code rates			
Rate	1/2	2/3	3/4	5/6
d_{free}	10	6	5	4
X	1	10	101	10101
Y	1	11	110	11010
XY	X_1Y_1	$X_1Y_1Y_2$	$X_1Y_1Y_2X_3$	$X_1Y_1Y_2X_3Y_4X_5$

Figure 4.5 and Figure 4.6 show the setting of convolutional encoder and viterbi decoder.

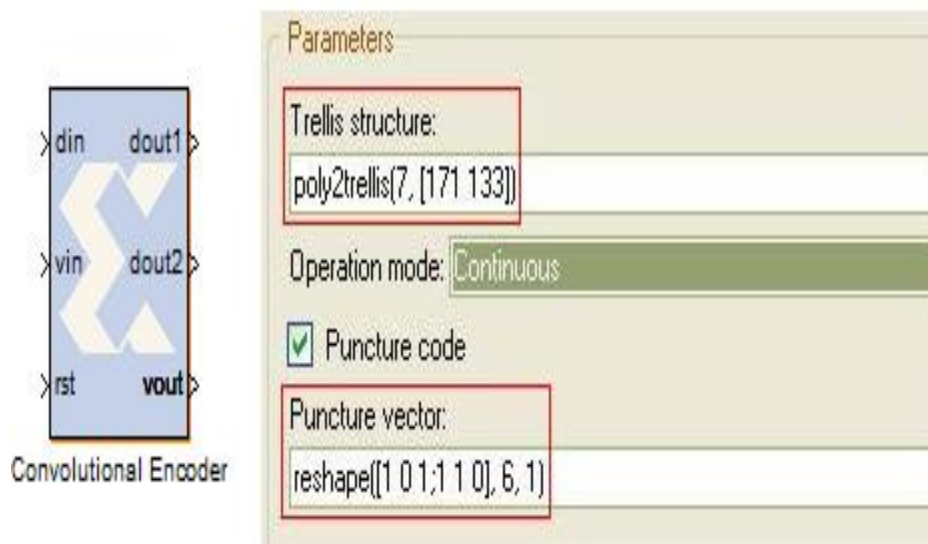


Figure 4.5-Convolution Encoder setting

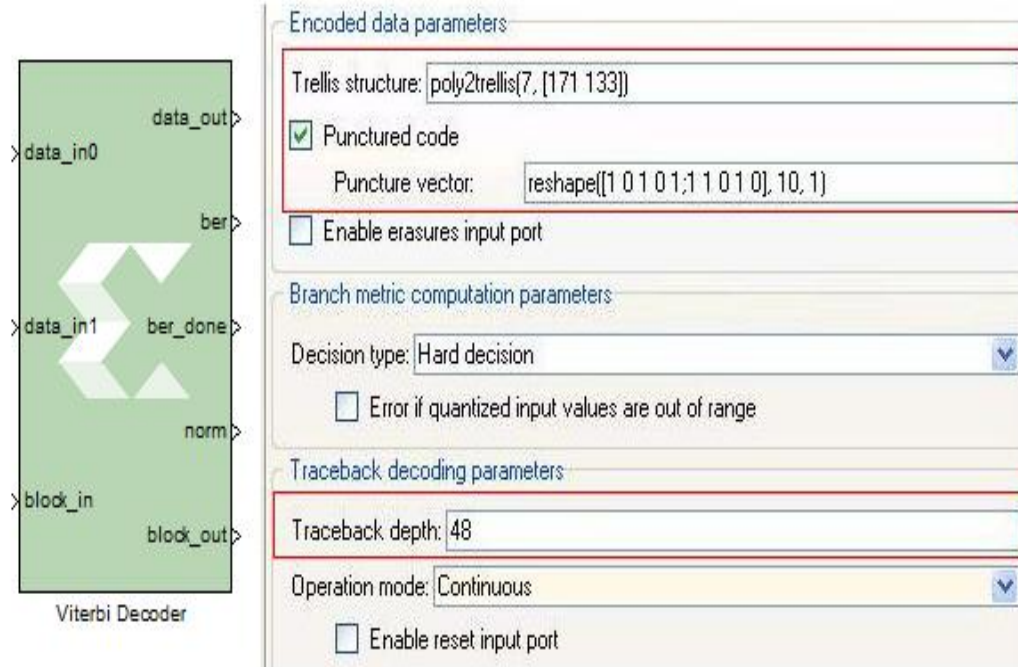


Figure 4.6-Viterbi Decoder setting

4.5 Transmitter Model

At the transmitter, the data from the source is input to the forward error correction block, which comprises of the convolutional encoder and puncturing system. The convolutional encoder is used for error correction in data transmission and it encodes the data sequence by inserting redundant bits. In the convolutional encoder, values are encoded by a linear feed forward shift register, which computes modulo-2 sums over a sliding window of input data.

In this report, a rate 1/2 convolutional encoder with constraint length 7 and code array 171 and 133 is used. Please note that the code array used is the optimal code array of constraint length 7. The constraint length denotes the number of shift registers over, the constraint length denotes the number of shift registers over, which the modulo-2 sum of the input data is performed. The rate 1/2 signifies that for every 1 bit input, the encoder will output 2 encoded bits.

Puncturing is a method of constructing new codes by removing the user-defined bits from the encoded data. The use of puncturing significantly reduces the number of bits to be transmitted over the channel. The puncture codes are a bit pattern that identifies the bits from the encoder to be transmitted and the codes used in this report are 10 and 11. Based on the puncture code parameter, the binary vector decides the bits that are to be removed. In a puncturing sequence, 0 and 1 means that the corresponding code symbol is not transmitted or is transmitted, respectively.

Figure 4.7 shows how the rate $1/2$ convolutionally encoded output can be punctured with puncture code 10 and 11 to give a rate $2/3$ punctured output. In Figure 4.7, the puncture block 0 and 1 uses 10 and 11 as puncture codes respectively. Consider bits A, B, C and D as input to the convolutional encoder. Bit A input to the encoder is encoded as A0 and A1. Similarly bits B, C and D are encoded as B0, B1, C0, C1, D0 and D1. The encoded bits are now input to the puncture blocks. Since the puncture codes are 10 and 11, two bits in parallel should be input to the puncture blocks. Thus, when A0 and B0 are input to puncture block 0, bit B0 is not transmitted because the puncture code 10 will delete every second bit input to the puncture block 0. However, all the bits input to the puncture block 1 will be transmitted because the puncture code 11 will not delete any bits.

Similarly, bit D0 will be deleted when C0 and D0 are input to the puncture block 0 and bits C1 and D1 will be transmitted from puncture block 1. Thus, the output rate of the forward error correction block is $2/3$, i.e., for every 2 input bits, 3 out of 4 encoded bits are transmitted.

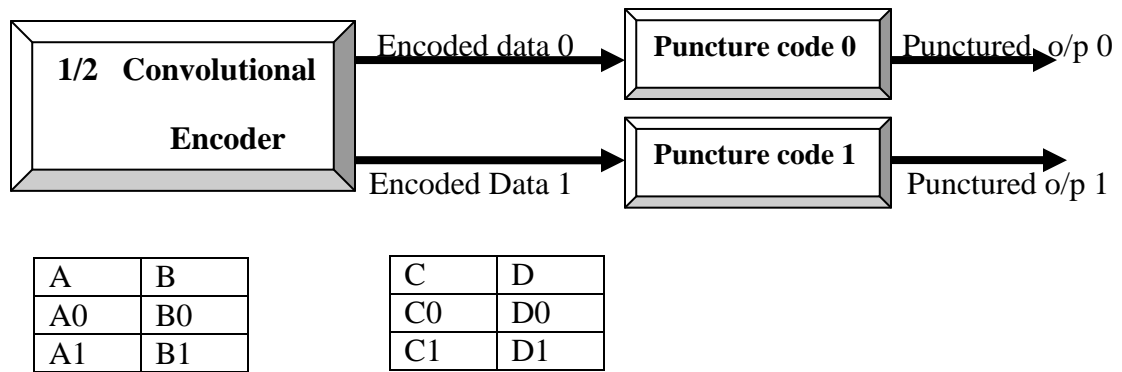


Figure 4.7-Example of puncturing rate 1/2 encoder

The punctured data is modulated using QPSK modulation technique. The QPSK modulated data is then spread by pseudorandom noise (PN) sequence with a spreading gain of 16 before transmission. Please note that the spreading gain used in this thesis, is very small when compared with practical processing gains and is used only for illustrative purposes. The channel is modeled as an AWGN channel. Hence, only the AWGN model is considered in this report. Figure 4.8 shows the transmitter design for this project.

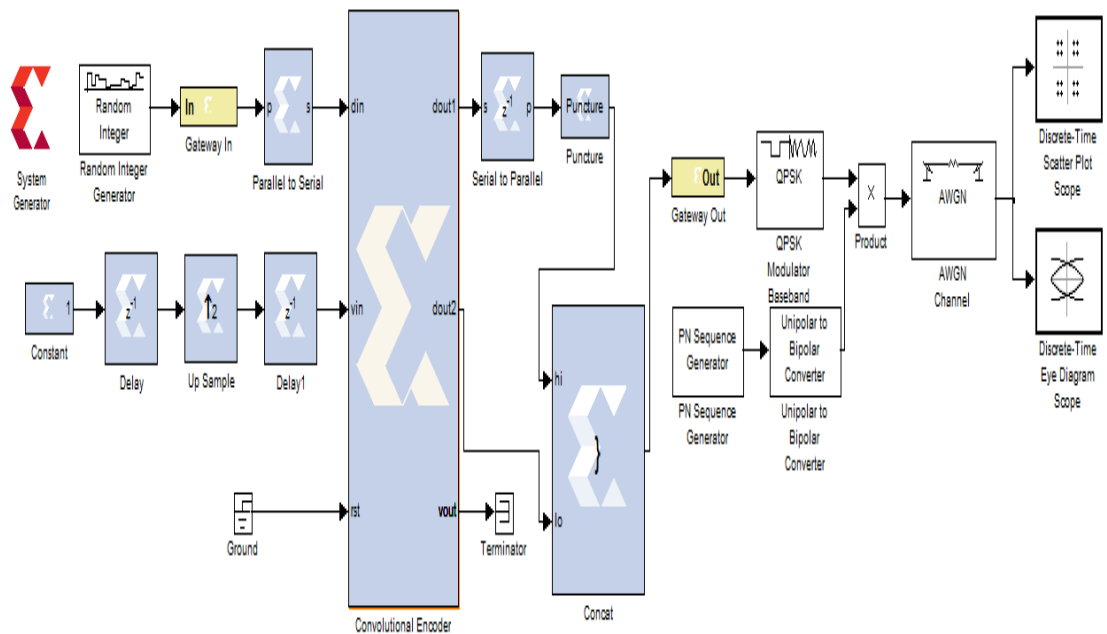


Figure 4.8-Transmitter design

4.6 Receiver Model

At the receiver, the signal is first dispread and then demodulated. Then error correction is applied to the demodulated data. The signal is dispread with PN sequence generator with a spreading gain of 16 and demodulated using QPSK technique. The demodulated and dispread data is depunctured prior to decoding, by inserting null-symbols in the punctured locations. The null-symbols are inserted according to the puncture code patterns. Figure 4.9 shows an example of depuncturing rate 2/3 punctured data.

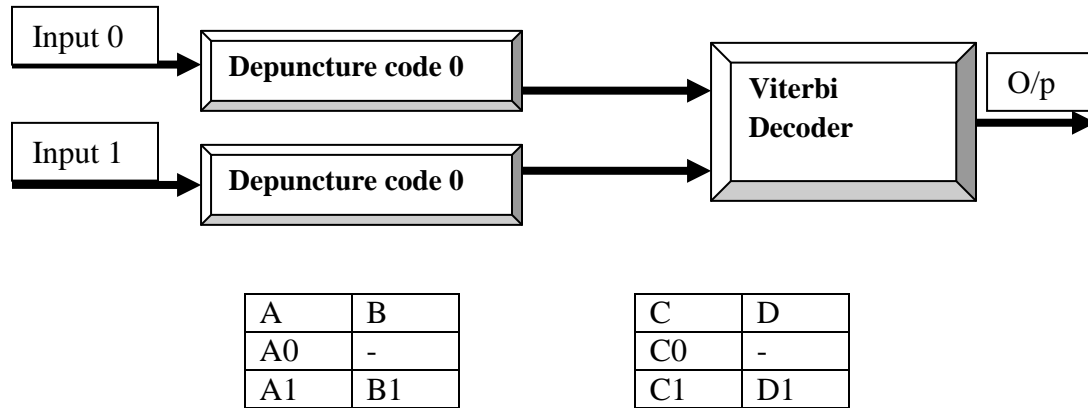


Figure 4.9 Example of depuncturing rate 2/3 encoded data

The depuncture blocks have the same depuncture codes as in the puncture blocks. Hence depuncture block 0 has a code of 10 and block 1 has a code of 11. Therefore, the null symbol is inserted after every other bit coming out of depuncture block 0. No symbols are inserted for block 1 as no bits were punctured. Since the bit B0 and D0 were punctured before transmission, null symbol is inserted in those locations and input to the Viterbi decoder along with the other bits.

Viterbi decoder is used to decode the convolutionally encoded signal by finding an optimal path through all the possible states of the encoder. There are two steps to the decoding process. The first step is to weigh the cost of incoming data against all possible data input combinations. Either a Hamming or Euclidean metric may be used to determine the cost. The second step is to trace back through the trellis and determine the optimal path. The length of the trace through the trellis can be controlled by the trace back length parameter.

The constraint length of 7 and the code array 171 and 133 used for decoding are the same as in convolution encoder. The trace back length parameter, that is, the number of trellis states processed before the decoder makes a decision on a bit, is set to 48. The decoder outputs the data bits which are later grouped accordingly. Figure 4.10 shows the design for receiver for this project work.

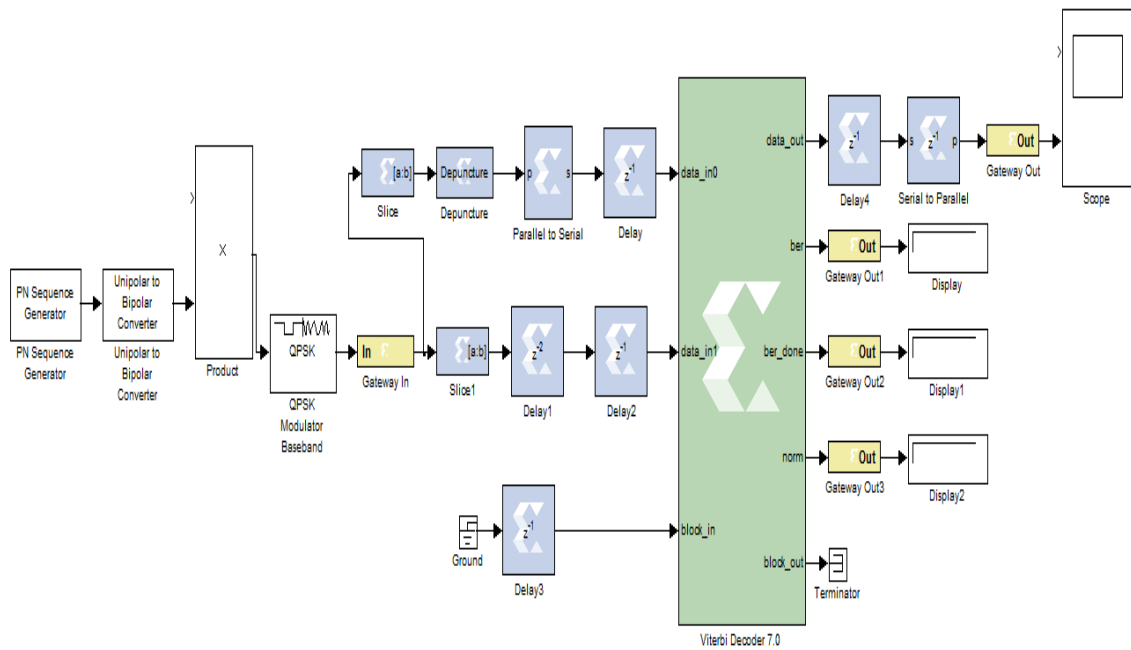


Figure 4.10-Receiver design

CHAPTER 5

SIMULATION

5.1 System Simulation

The simulation model for punctured convolution encoder is shown in Figure 5.1. In this thesis, punctured convolution encoder of rate $2/3$ with constraint length 7 is used. The data source random integer generator, output is input to the “Gateway In” block. This block converts the data in double precision to the Xilinx fixed point representation. From the “Gateway In” block, the data is parallel-to-serial converted and given to the data input port of the convolution encoder. The output from the data output port 1 of the encoder is serial-to-parallel converted and given to the puncture block with puncture code 10. Every second, encoded output bit from data output port 1 is deleted by the puncture block after serial-to-parallel conversion. However, there is no puncture block on data output port 2 since the puncture code is 11 and all the encoded bits from output port 2 are transmitted.

A constant value of 1 is used as input to the input port (V_{in}) to specify to the encoder that the data on its input port is valid and is ready to be encoded. When there is valid output on the output ports of the encoder, the valid output port (V_{out}) output is set to high, which is sent to the Viterbi decoder valid input port.

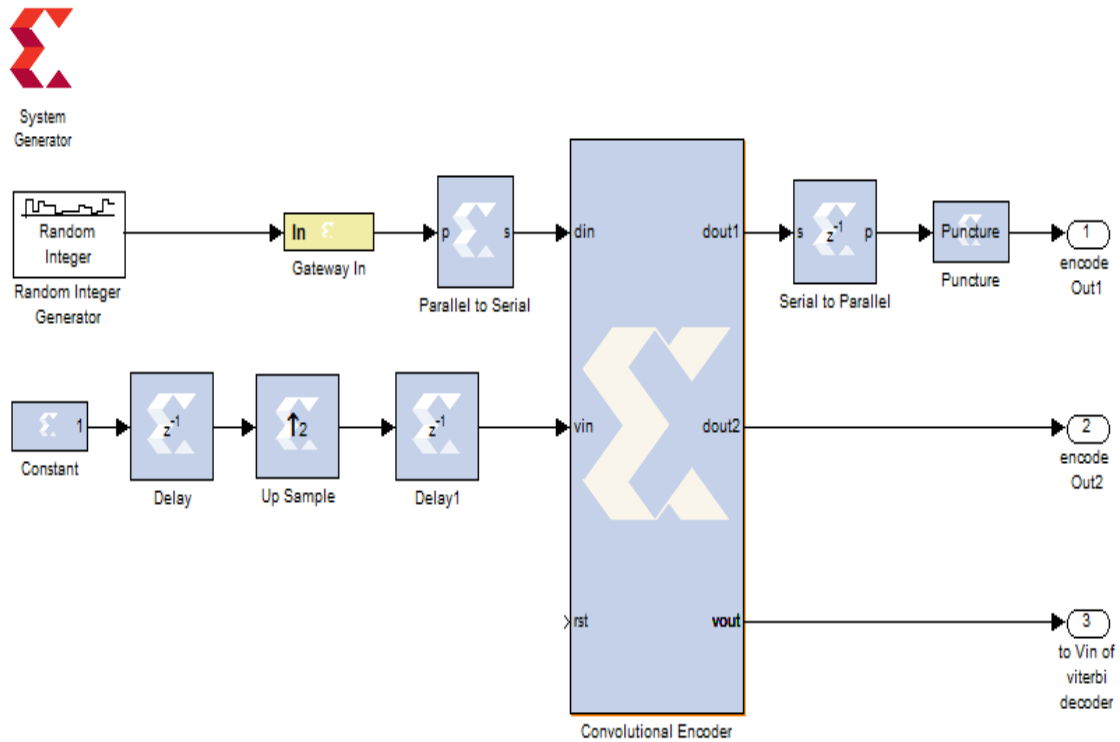


Figure 5.1-Encoding and puncturing using System Generator blocks

Figure 5.2 shows the modulation and spreading design process. The punctured data from the two puncture blocks are concatenated and then modulated using QPSK modulator. The data from the concatenate passes through a "Gateway out" block so that the Xilinx's fixed point representation is converted to SIMULINK double precision since QPSK modulation and spreading is done with SIMULINK blocks. After modulation, the modulated data is then spread using PN sequence generator and passed through the AWGN channel.

At the receiver, the signal is despreading using the same PN sequence generator and demodulated using QPSK demodulator. This is shown in Figure 5.3. After demodulation, the input data to the two depuncture blocks is first passed through the "Gateway In" block converting it to Xilinx's fixed point representation. Resulting data is then passed through the slicer to separate the encoded bits.

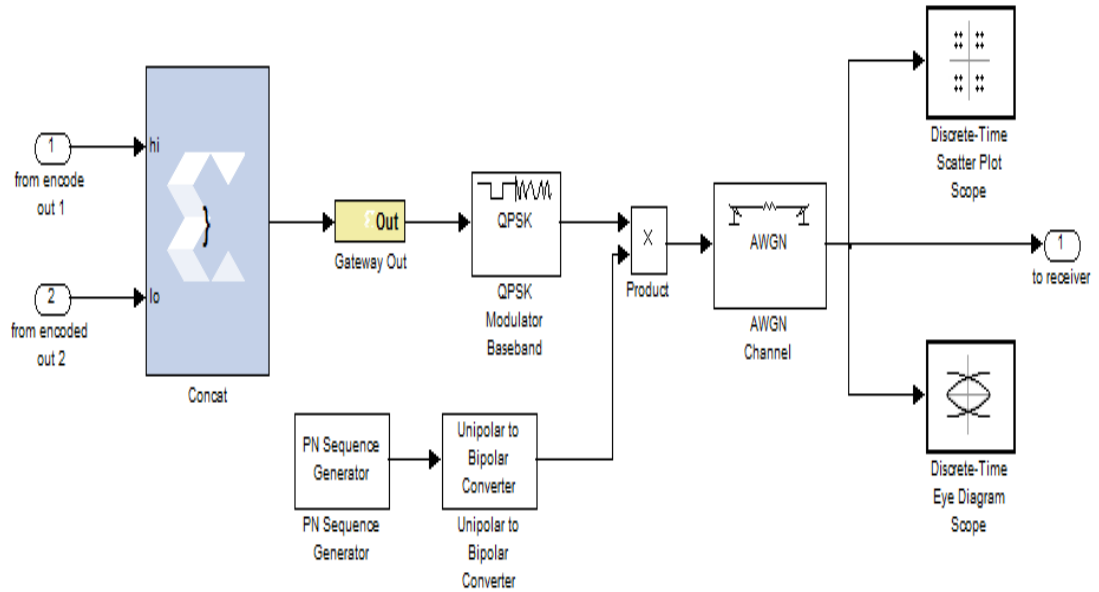


Figure 5.2-Modulation and spreading of the encoded signal

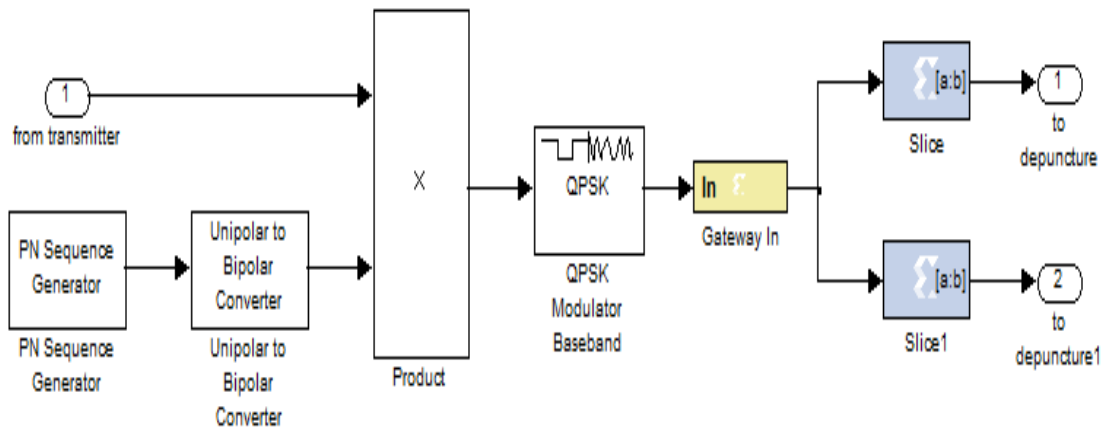


Figure 5.3-Despreading and demodulation of the received signal

Recall that the purpose of depuncturing and decoding is for forward error correction. The same puncture code 10 and 11 used for puncturing is used for depuncturing. Figure 5.4 shows the depuncturing and decoding blocks used in the simulation. The decoder depunctures the received data, prior to Viterbi decoding, by inserting null-symbols in the punctured locations.

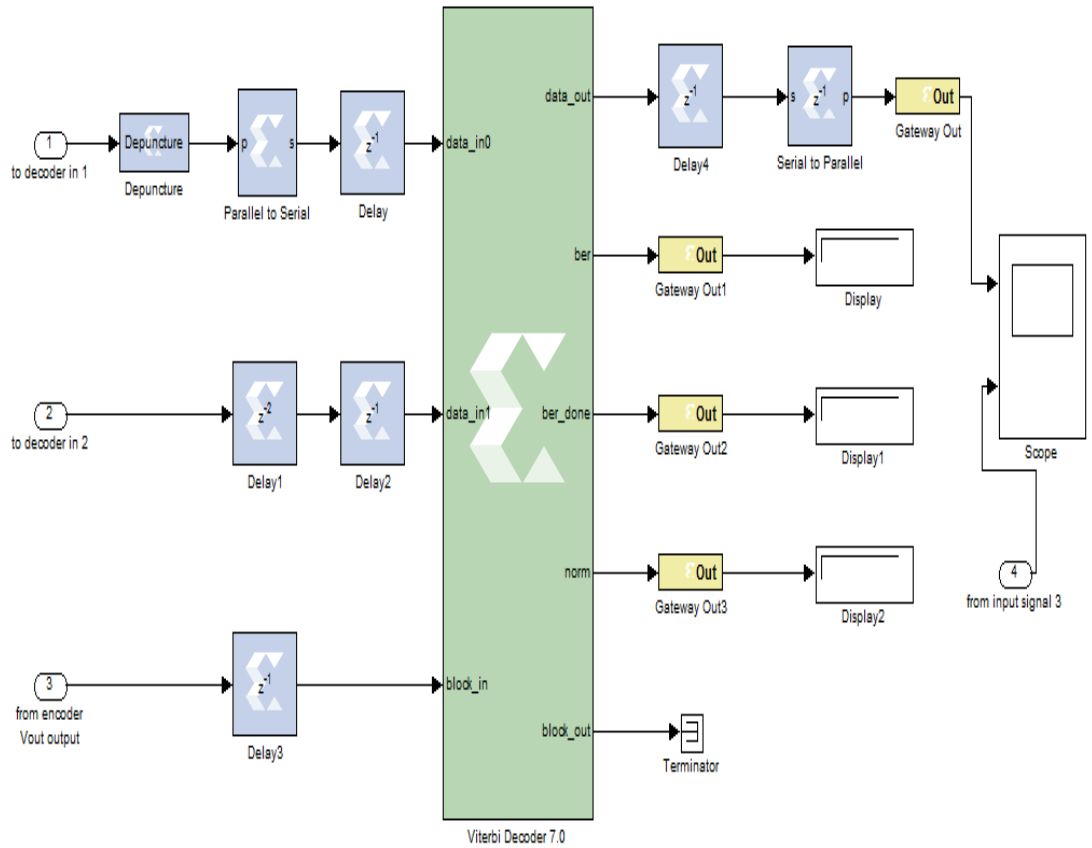


Figure 5.4-Depuncturing and decoding model using System Generator blocks

The null-symbol is inserted after every input to the data input port 1 of the Viterbi decoder through the depuncture block and no null-symbol is inserted at the data input port 2 of the decoder as no bits were punctured during transmission. After depuncturing from block 1, the data is parallel-to-serial converted and input to the Viterbi decoder.

The input to the second port of the Viterbi decoder and the input to the valid input port are given after few delays in order to synchronize with the output from depuncture block 1. At the decoder, the depunctured data is then decoded using a rate 1/2 Viterbi decoder. The decoder output is serial-to-parallel converted accordingly and the input and output waveform is compared. The other outputs from the Viterbi decoder (ber, ber_done, and norm) are used to check for the errors while decoding.

Figure 5.5 shows the circuit diagram of the full wireless communication system using system generator blocks from which we can simulate the circuit and get results and shows communication on integrating FPGA with multicore SDR development platform.

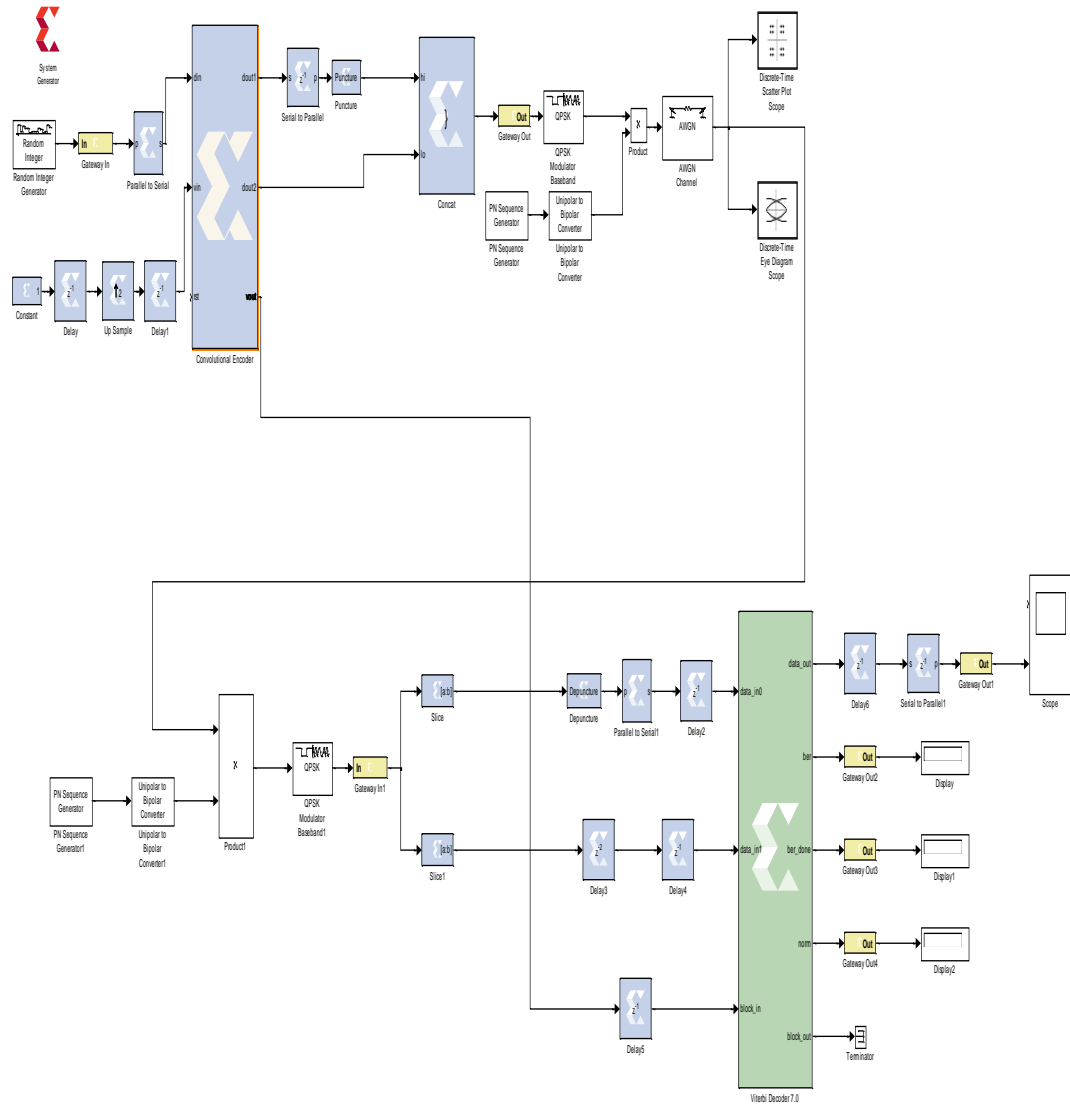


Figure 5.5-Communication system model using system generator blocks

5.2 Simulation Results

The results obtained from the simulation is presented and discussed in this section. The constellation diagram and eye diagram of the modulated signal from which conclusions about the modulated signal can be drawn is observed at the output of the channel. These diagrams reveal the modulation characteristics of the signal and help to depict the impact of impairments, such as pulse shaping or channel distortions. They are commonly used to evaluate the overall performance of the digital communication systems. Since the channel used in this thesis is AWGN, the extent to which the noise has affected the modulated signal can be seen from constellation and eye diagrams.

5.2.1 Constellation Diagrams

Figure 5.6 shows the constellation diagram of the modulated signal with signal-to-noise ratio (SNR) of 20 dB. The figure shows that each constellation point becomes a cloud around the central point. When the noise is more in the channel, the constellation points spreads around the central point. While demodulating in the receiver, the chances of misinterpretation of one point as other is more and this leads to incorrect demodulation and error.

Increasing the SNR of the AWGN channel will increase the performance of the system. The constellation diagram of modulated signal with SNR of 25 dB is shown in Figure 5.7. Since the SNR is high, the constellation points form a more dense cloud around the central point thus reducing the transmission error.

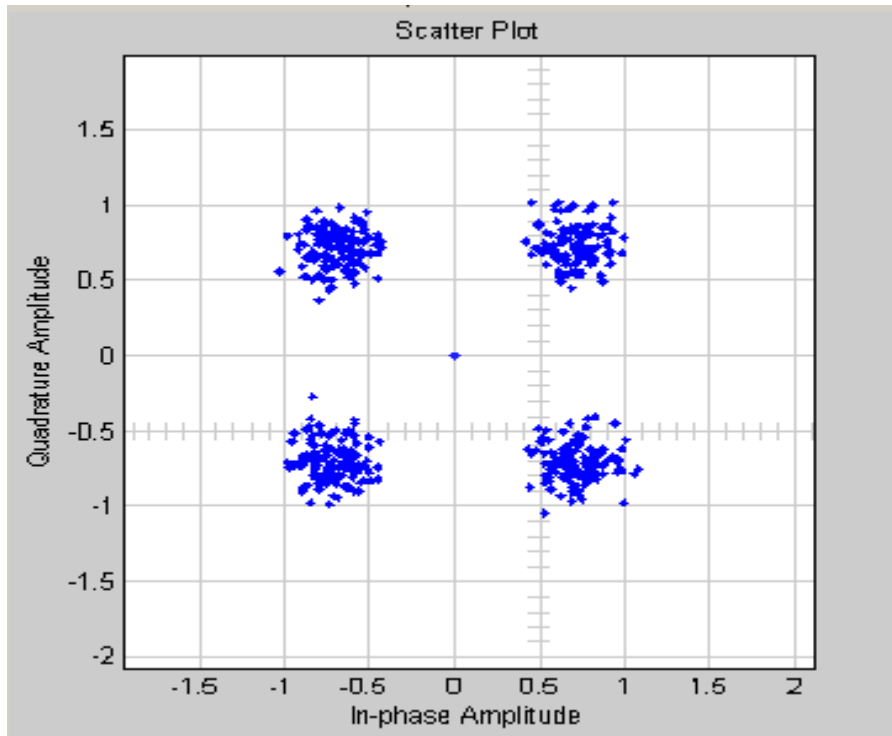


Figure 5.6-Constellation diagram for QPSK modulated signal with SNR = 20 dB

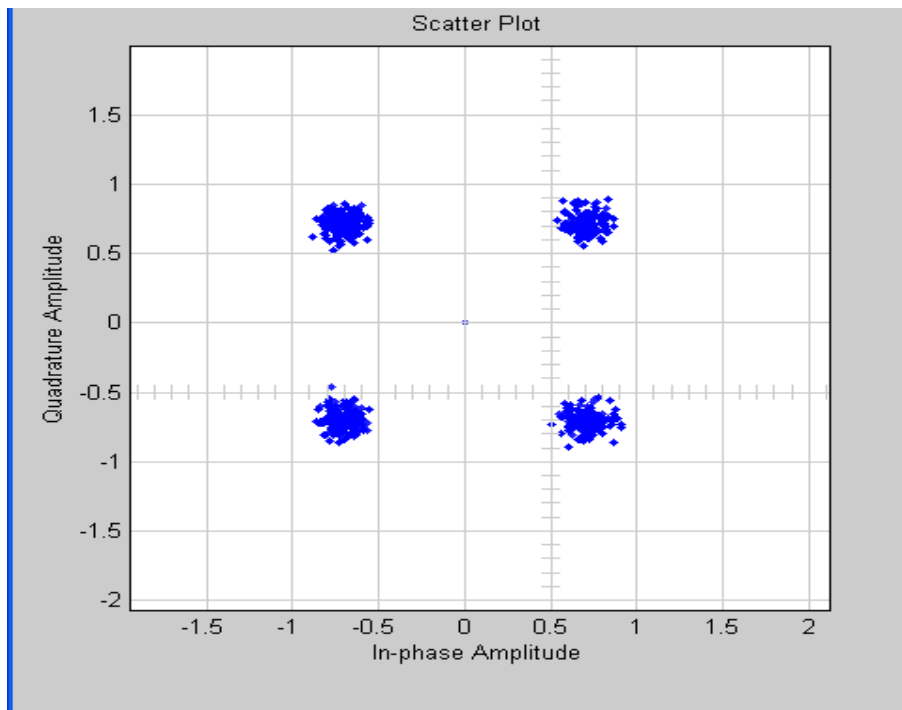


Figure 5.7-Constellation diagram for QPSK modulated signal with SNR = 25 dB

Comparing Figures 5.6 and 5.7, it can be seen that the constellation points of Figure 5.7 are denser than in Figure 5.6. If the noise in the channel is smaller, then the constellation points will be denser, and thus the transmission error is less and the receiver output is more accurate. This is clearly reflected on Figure 5.6 compared to Figure 5.7

The constellation diagram for BPSK is shown in Figure 5.8. Recall that the power spectral density (PSD) of a QPSK signal has a null-to-null bandwidth that is equal to the bit rate, which is half that of a BPSK signal. Therefore, QPSK has twice the bandwidth efficiency of BPSK, since 2 bits are transmitted in a single modulation symbol instead of 1 bit for BPSK. Further, the bit error probability of QPSK is identical to BPSK, while twice as much data can be sent in the same bandwidth. Thus, when compared to BPSK, QPSK provides twice the spectral efficiency with exactly the same energy efficiency.

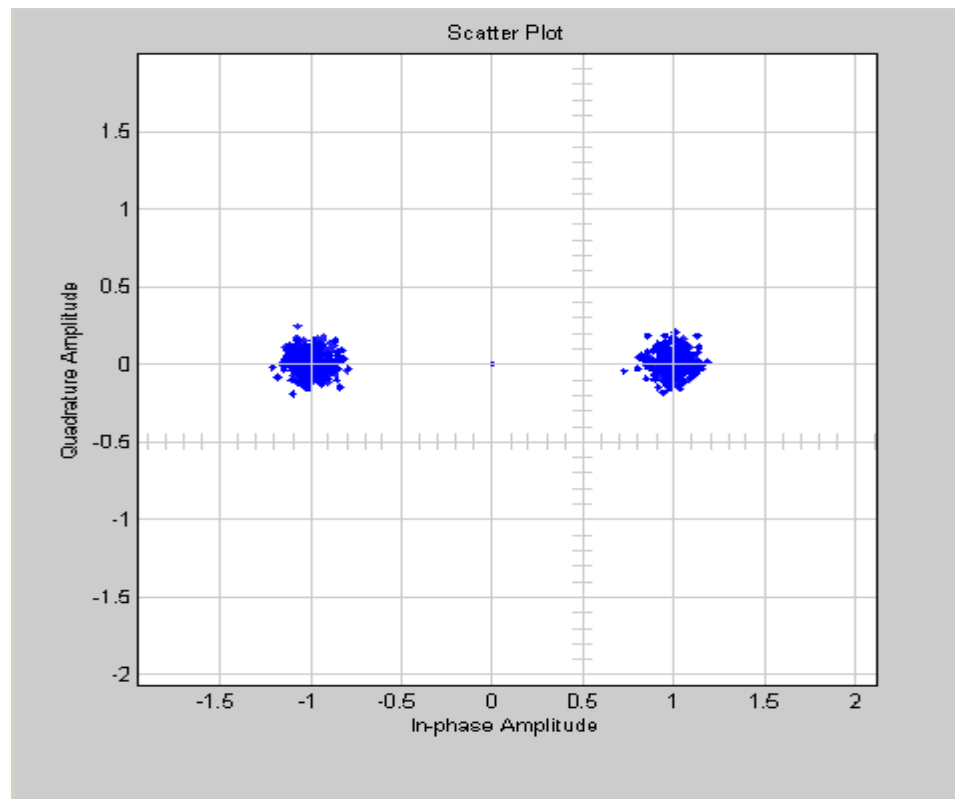


Figure 5.8-Constellation diagram for BPSK modulated signal with SNR = 25 dB

5.2.2 Eye Diagrams

The measure of distortion, timing jitters and noise margin can be found from the eye diagrams. Figure 5.9 shows the eye diagram of the modulated signal with SNR of 20 dB. From the figure, A shows the distortion which is equal to 0.5; B and C show the timing jitter and the noise margin which are equal to 0.2 and 0.5. Due to the noise in the channel, when the noise margin in the eye diagram decreases, the eye starts to close in, thus making the errors to increase. Since the SNR is less, it can be seen from the figure that the eye has more distortions and is not properly open due to the presence of noise in the channel.

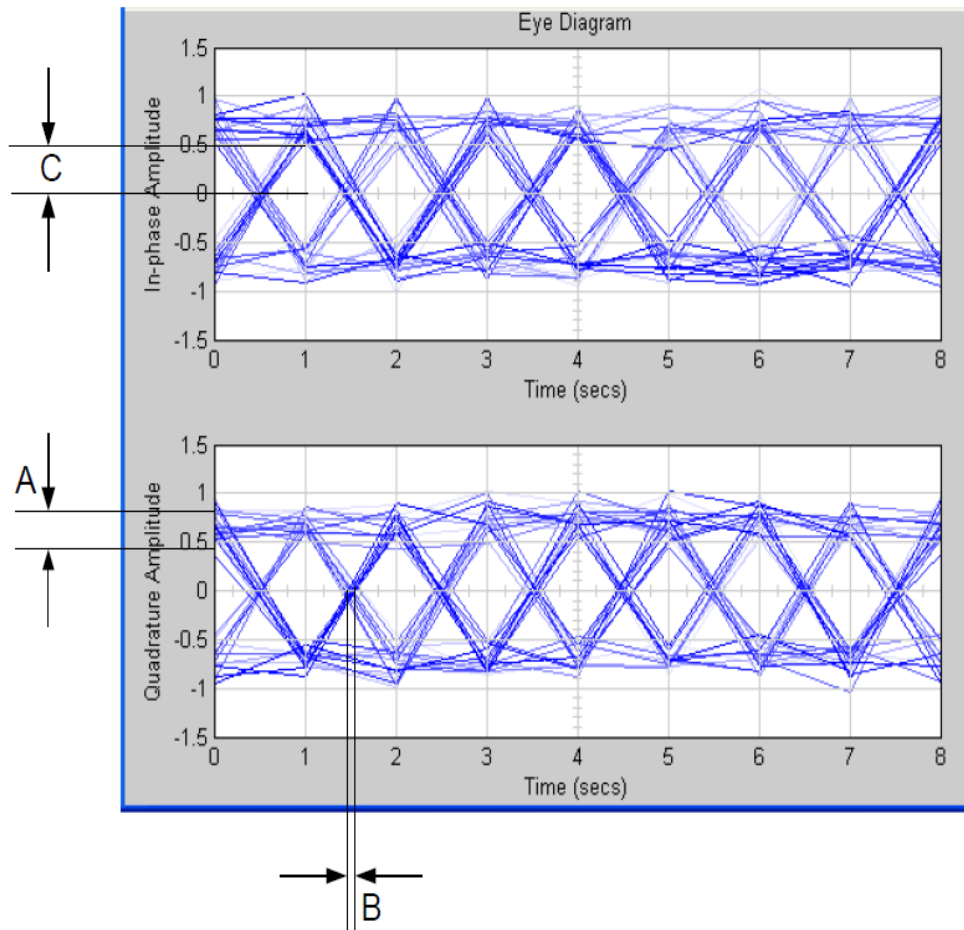


Figure 5.9-Eye diagram for QPSK modulated signal with SNR = 20 dB

Figure 5.10 shows the eye diagram for SNR of 25 dB. The eye diagram has less distortion given that the eye opening is more defined. The proper eye opening defines

less bit errors and hence less transmission error. Comparing Figures 5.9 and 5.10, it can be seen that the distortion in Figure 5.10 is 0.25 when compared with the distortion in Figure 5.10 which is 0.5. Similarly, the timing jitter is 0.1 for SNR = 25 dB and 0.2 for SNR = 20 dB. The other parameter noise margin is 0.65 and 0.5 for SNR = 30.

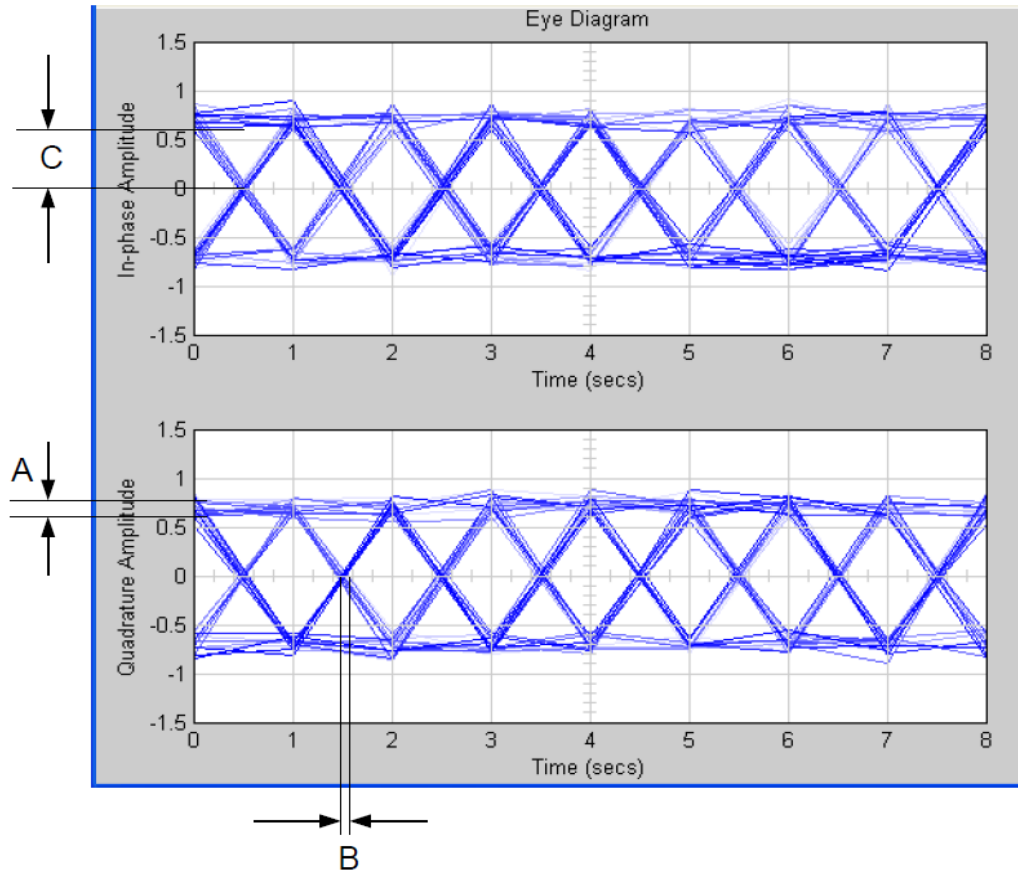


Figure 5.10-Eye diagram for QPSK modulated signal with SNR = 25 dB

5.2.3 Output Waveform

Another indicator of performance is the observation of the output waveform compared to the input waveform. Figure 5.11 shows the input signal to the transmitter and the output signal of the receiver when both are synchronized. Observe that the input and output signals are similar for the system specifications discussed above, showing that the received signal is demodulated and decoded without much of error.

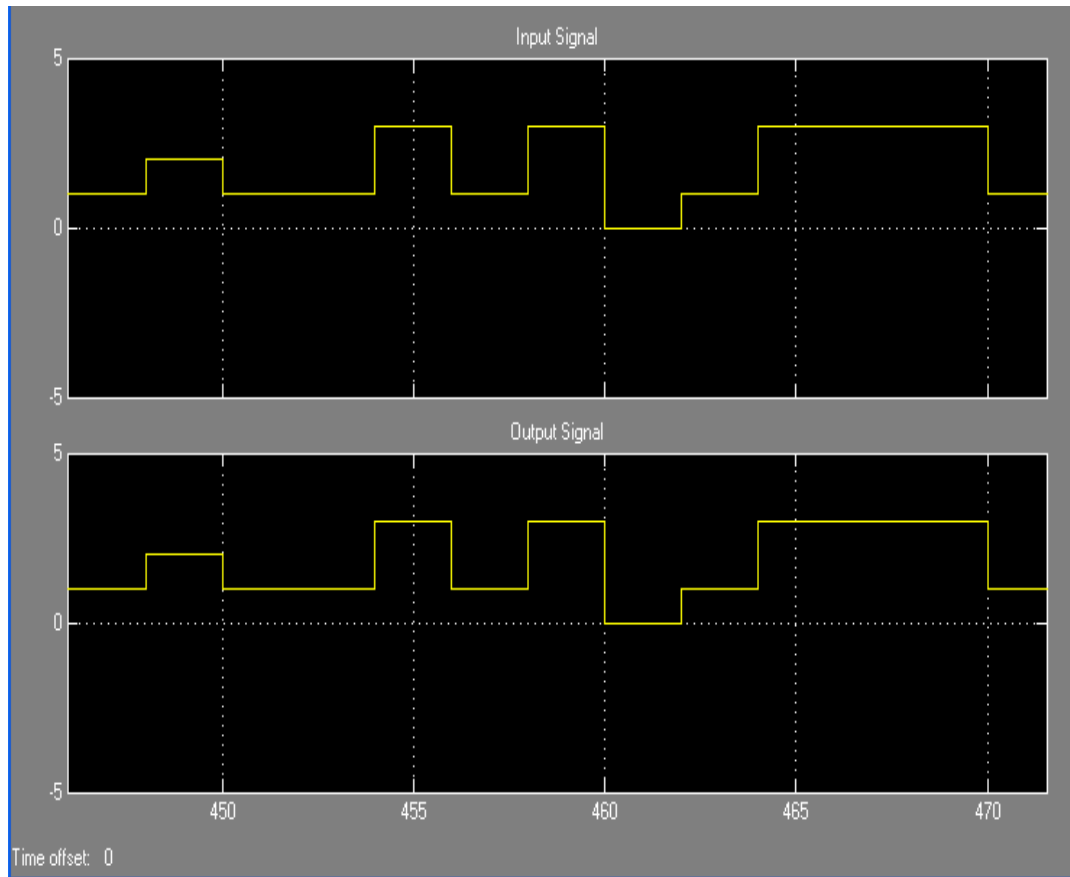


Figure 5.11 Input and output waveforms of the communication system

CHAPTER 6

REAL TIME IMPLIMENTATION AND SIMULATION

6.1 Real time Implementation and Simulation

After simulation and analysis of the results, the system is implemented in real time via hardware co-simulation on Xilinx Spartan 3E FPGA (XC3S500E-4FG320C) platform. The Diligent D2-SB hardware board, which has a Spartan 3E FPGA (XC3S500E-4FG320C) chip on it, is used to implement the system in real time. The Spartan 3E (XC3S500E-4FG320C) is programmed through the joint test action group (JTAG) programming cable.

The hardware co-simulation is performed by powering on the Diligent D2-SB board and connecting the JTAG cable from the board to the printer port of the personal computer (PC).

The Diligent D2-SB circuit board provides a complete circuit development platform centered on a Xilinx Spartan 3E FPGA (XC3S500E-4FG320C) and is shown in Figure 6.1. The Diligent D2-SB is made by Diligent Inc., and is used for hardware co-simulation. Some of the features of this board include,

Key Features

- **Xilinx Devices:**
- Spartan-3E FPGA (XC3S500E-4FG320C)
- CoolRunner™-II CPLD (XC2C64A-5VQ44C)
- Platform Flash (XCF04S-VO20C)
- **Clocks:** 50 MHz crystal clock oscillator
- **Memory:**
- 128 Mbit Parallel Flash

- 16 Mbit SPI Flash
- 64 MByte DDR SDRAM
- **Connectors and Interfaces:**
- Ethernet 10/100 Phy
- JTAG USB download
- Two 9-pin RS-232 serial port
- PS/2- style mouse/keyboard port, rotary encoder with push button
- Four slide switches
- Eight individual LED outputs
- Four momentary-contact push buttons
- 100-Pin expansion connection ports
- Three 6-pin expansion connectors
- **Display:** 16 character - 2 Line LCD

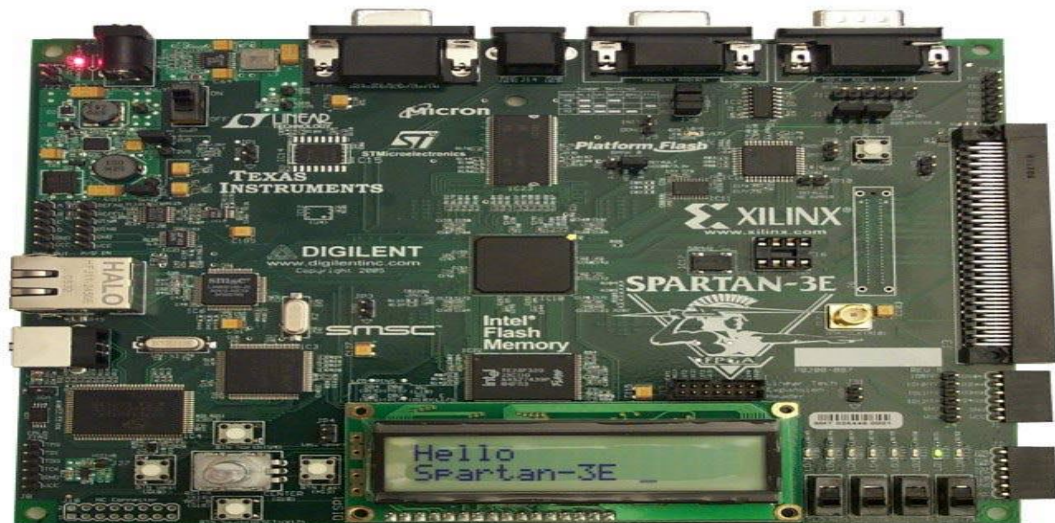


Figure 6.1-Spartan 3E FPGA hardware board for hardware co-simulation

The Spartan 3E FPGA and the 18V00 ROM on the Diligent D2-SB, and any programmable devices on peripheral boards attached to the Diligent D2-SB can be programmed via the JTAG port. In this thesis, only the Spartan 3E FPGA was used. The transmitter side of the model, which was simulated, is targeted to Spartan 3E FPGA through the JTAG programming chain.

Hardware co-simulation makes it possible to incorporate SIMULINK simulation into FPGA hardware directly. Hardware co-simulation is invoked by activating the System Generator, which should be present in all models containing System Generator blocks. Hardware co-simulation targets are organized under the hardware co-simulation sub-menu in the compilation dialog box field. When the compilation target, Spartan 3E already installed, is selected, the fields on the System Generator block dialog box are automatically configured with settings appropriate for the selected compilation target. System Generator remembers the dialog box settings for each compilation target. Once the compilation target is selected, the System Generator code generator is invoked to compile the model for hardware co-simulation.

The code generator produces a FPGA configuration bit stream for the design, suitable for hardware co-simulation. System Generator not only generates the HDL and net list files for the model during the compilation process, but it also runs the downstream tools necessary to produce an FPGA configuration file. A new command window is opened when the implementation tools are running to produce the configuration bit stream file as shown in Figure 6.2. This window shows the progress and output of each tool as it is runs. It can be seen from Figure 6.2 that the information about the completion of placing and routing of the system in hardware is also displayed. The figure shows that the placement routing and timing were successfully completed. The configuration bit stream contains the hardware associated with the model.

```
C:\MATLAB701\sys\perl\win32\bin\perl.exe
Generating Pad Report.
All signals are completely routed.
Total REAL time to PAR completion: 12 secs
Total CPU time to PAR completion: 9 secs
Peak Memory Usage: 59 MB
Placement: Completed - No errors found.
Routing: Completed - No errors found.
Timing: Completed - No errors found.
Writing design to file jtagcosim_top.ncd.
PAR done.
#-----#
# Starting program bitgen
# bitgen -w -intstyle xflow -g StartUpClk:JtagClk jtagcosim_top.ncd
#-----#
```

Figure 6.2-Command window showing the progress of implementation tools

Hardware co-simulation compilation targets automatically create bit streams and associate them with implementation blocks. When the design is simulated in SIMULINK, results for the compiled portion are calculated in hardware. This allows the compiled portion to be tested in actual hardware, and can speed up simulation dramatically. System Generator creates a new hardware co-simulation block for the design once it has finished creating the FPGA configuration bit stream. A SIMULINK library is also created in order to store the hardware co-simulation block information. Figure 6.3 shows the hardware co-simulation block generated for the transmitter stage of the model. At this point, the hardware co-simulation block can be copied out of the library and used in the SIMULINK simulations instead of the Xilinx blocks. In this report Figure 6.3 is used in the design and is simulated with other SIMULINK blocks. When simulation is complete, the hardware co-simulation block interacts with the underlying hardware and produces the output. A hardware co-simulation block consumes and produces the same types of signals as that of other System Generator blocks. When a value is written to one of the hardware co-simulation block's input ports, the block sends the corresponding data to the appropriate location in hardware. Similarly, the block retrieves data from hardware when there is an event on an output port.

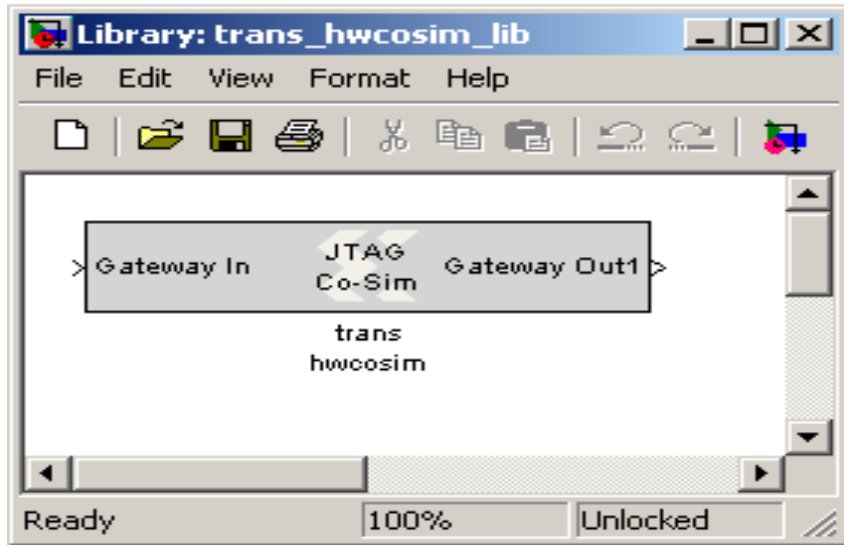


Figure 6.3-Hardware co-simulation library

Figure 6.4 shows the constellation diagram output obtained by implementing the model in hardware. Comparing Figure 6.4 to the previous results, the output from simulation, it is evident that the two results are similar, with little or no difference. Thus we have shown that the hardware implementation result is the same as the simulated results.

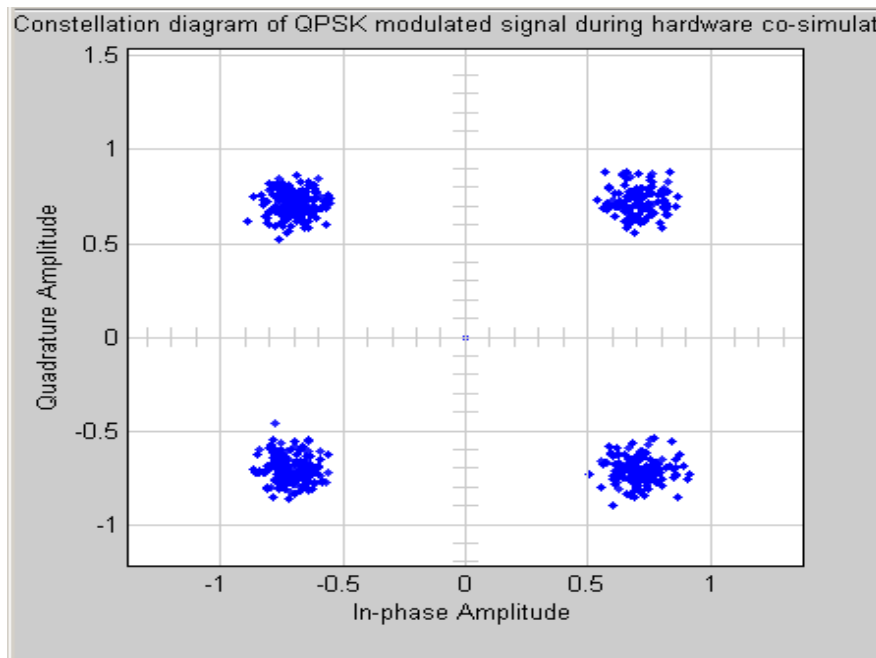


Figure 6.4-Constellation diagram for hardware co-simulation

CHAPTER 7

SUMMARY AND CONCLUSION

7.1 Summary

This project report consists of a study of the software defined radio concept and, test bed development and implementation of integrating FPGA with multicore software defined radio development platform to design wireless communication system.

In Chapter 3, a general description of wireless communication system is presented. The design methodology used for simulation and hardware implementation of the system was discussed.

In Chapter 4, framework for the system that has been implemented, the simulation and the results are presented in Chapter 5. Results such as the constellation diagrams and eye diagrams at the output of the transmitter are used to quantify the performance of the system. Furthermore, the system is tested by comparing the input waveform to the transmitter and the output waveform from the receiver. The results obtained in Chapter 5 and Chapter 6 indicate that the system produces similar results from simulation and hardware implementation. Several modulation techniques are used with similar results observed during simulation and hardware implementation. In this report I used the System Generator for DSP, a tool developed by Xilinx for implementing a model of a digital signal processing algorithm in a Xilinx FPGA platform. The Xilinx System Generator for DSP is compatible with MATLAB SIMULINK making it possible to provide design parameters, data path definition, bit- and cycle-true simulations, test bench generation, hardware co-simulation and VHDL code generation. System Generator is very helpful for system engineers without background in VHDL coding and FPGAs, since the generator automates the hardware implementation process. Though the automation seems to be helpful, there are few constraints encountered in this thesis while using System Generator. Not all the communication functions are available in the predefined library. This restricts the use of this tool to certain extent. But the functionalities can be written as code in MATLAB, C or VHDL and can be imported into the system using the black box features of the System Generator. The next constraint is in

the hardware implementation. There are chances that when the design is targeted in hardware it would not be placed and routed optimally. This could be a major problem if the system size is a requirement. In spite of these constraints, the automation facility of the System Generator proves to be useful for system implementation.

7.2 Conclusion

Designed and successfully implemented wireless communication system on multicore SDR development platform and also seen that during simulation if signal to noise ratio (SNR) for AWGN channel increased the system gave us more good results with less distortion.

Although some modes' BER are not good enough, here designed a model of wireless communication system on FPGA implemented SDR development platform. It's particular that we integrate FPGA into our system. It is more difficult that design a system by using Xilinx block set and Simulink block set at the same time. Because there are several difference between these two kinds of block set such as the difference of data format, block architecture etc. This report could be helpful for model-based design of next generation, and using Model-based FPGA design in Simulink can shorten the time to market.

7.3 Future work

A complete communication system for any particular air standard (e.g., GSM, IS-95, or CDMA2000), can be defined in software. In this report, the receiver is not implemented in hardware due to the licensing issues of the Viterbi decoder block. A workaround for this problem would be to write a VHDL or C code for the Viterbi decoder and import it into the system and implement it in hardware. These are some of the ways this thesis could be extended in the future

BIBLIOGRAPHY

- [1] IEEE std. 802.11a-1999, 30 Dec. 1999.
- [2] K. Wongthavarawat and A. Ganz, "IEEE 802.16 Based Last Mile Broadband Wireless Military Networks with Quality of Service Support," IEEE Military Commun. , vol. 2, pp. 779-784, Oct. 2003.
- [3] Ulrich Ramacher, "Software-Defined Radio Prospects for Multistandard Mobile Phones," Computer, Volume 40, Issue 10, p.p. 62-69, October 2007.
- [4] Luo Zhigang, Li Wei, Zhang Yan, Guan Wei, "A Multi-standard SDR Base Band Platform," Research & Innovation Center, Alcatel Shanghai Bell Co., Ltd., Shanghai, 2003.
- [5] Water Tuttlebee, "Software Defined Radio: Enabling Technologies," JOHN WILEY & SONS, LTD, NJ, 2002.
- [6] J. H. Reed, "Software Radio A Modern Approach to Radio Engineering," Prentice Hall, Upper Saddle River, NJ, 2002.
- [7] Paul Burns, "Software Defined Radio for 3G," Artech House Publishers, Boston, 2003.
- [8] Nikhil S. Bhatia, "A Physical Layer Implementation of Reconfigurable Radio," Bradley Department of Electrical and Computer Engineering Blacksburg, Virginia, December 3rd , 2004.
- [9] Lyrtech Inc., "Small Form Factor SDR Evaluation Module/Development Platform user's guide," May, 2007.
- [10] Lyrtech Inc., "SFF SDR Development Platform Model-based design guide," April 2007.
- [11] T. Rappaport, Wireless Communications - Principles & Practice, 2nd edition. Prentice-Hall, Upper Saddle River, NJ, 1996.

- [12] J. Mitola, III, "The software radio architecture," IEEE Commun. Mag., vol. 33, no. 5, May 1995, pp. 26-38.
- [13] J. Reed, Software Radio: A Modern Approach to Radio Engineering, Prentice Hall Communications Engineering and Emerging Technologies Series, 2002.
- [14] J. Noll and E. Buracchini, "Software radio - A key technology for adaptive access," Wireless Commun. and Mobile Computing, vol. 2, issue 8, Dec. 2002, pp. 789-798.
- [15] Vanu Software Radio. Retrieved on June 14, 2005 from <http://vanu.com/technology/softwareradio.html>
- [16] O. Ugweje and M. Souaré, Software Defined Radio in Communication Systems, Research Brief, Department of Electrical and Computer Engineering, The University of Akron, Akron, OH.
- [17] S. Srikanteswara, Design and Implementation of Soft Radio Architecture for Reconfigurable Platforms, Ph.D Dissertation, Virginia Polytechnic Institute and State University, July 2001.
- [18] System Generator product guide. Retrieved on July 23, 2005 from http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=dr_dt_system_generator.
- [19] R. Schiporst, Demonstration of Software Radio Concept, Master's Thesis, University of Twente, Netherkands, June 2000.
- [20] M. Laddomada, "Reconfiguration issues of future mobile software radio platforms," Wireless Commun. and Mobile Computing, vol. 2, issue 8, Dec. 2002, pp. 815-826.
- [21] M. Cummings and S. Haruyama, "FPGA in the software radio," IEEE Commun. Mag., vol. 37, no. 2, Feb. 1999, pp. 108-112.
- [22] E. Buracchini, "The software radio concept," IEEE Commun. Mag., vol. 38, no. 9, Sep. 2000, pp. 138-143.

- [23] A. Wiesler and F. Jondral, "A software radio for second- and third-generation mobile systems," *IEEE Trans. on Vehicular Technology*, vol. 51, no. 4, July 2002, pp. 738-748.
- [24] S. Weiss, A. Shligersky, S. Abendroth, J. Reeve, L. Moreau, T. E. Dodgson and D. Babb, "A software defined radio testbed implementation," In *Proceedings of IEE Colloquium on DSP Enable Radio*, Livingston, Scotland, 2003, pp. 268-274.
- [25] N. W. Anderson, H. R. Karimi, P. Mangold and M. Wezelenburg, "Software definable implementation of a dual mode TD-CDMA/DCS1800 transceiver." Retrieved on June 13, 2005 from <http://www.csem.ch/slats/files/ACTSUM98.PDF>.
- [26] N. W. Anderson, H. R. Karimi and P. Mangold, "Software-definable implementation of a TDMA/CDMA transceiver." Retrieved on June 13, 2005 from <http://www.csem.ch/slats/files/ICSPAT2.PDF>.
- [27] Software Defined Radio Forum FAQs. Retrieved on June 14, 2005 from <http://www.sdrforum.org/faq.html>.
- [28] R. Baines, "The DSP bottleneck," *IEEE Commun. Mag.*, vol. 33, issue 5, May 1995, pp. 46-54.
- [29] Z. Kostic and S. Seetharaman, "DSPs in cellular radio communications," *IEEE Commun. Mag.*, vol. 35, issue 12, Dec. 1997, pp. 22-35.
- [30] J. Mitola III, "Technical challenges in the globalization of software radio," *IEEE Commun. Mag.*, vol. 37, no. 2, Feb. 1999, pp. 84-89.
- [31] B. Mohebbi, E. Filho, R. Maestre, M. Davies and F. Kurdahi, "A case study of mapping a software defined radio application on a reconfigurable DSP core," *International Symposium on Systems Synthesis*, Newport Beach, CA, 2003, pp. 103-108.
- [32] G. Jo, M. Sheen, S. Lee and K. Cho, "A DSP-based reconfigurable SDR platform for 3G systems," *IEICE Trans. on Commun.*, Feb. 2005, pp. 678-683.
- [33] Design Flow using System Generator for DSP. Retrieved on June 13, 2005 from

http://www.xilinx.com/products/software/sysgen/design_flow.htm. 61

- [34] X. Zhanga and K. Nga, "A review of high-level synthesis for dynamically reconfigurable FPGAs," *Microprocessors and Microsystems*, vol. 34, issue 4, Aug. 2000, pp. 199-211.
- [35] V. Bose, *Design and Implementation of Software Radios Using General Purpose Processor*, Ph.D Dissertation, Massachusetts Institute of Technology, June 1999.
- [36] A. Haghghat, "A review on essentials and technical challenges of software defined radio," *Proceedings of MILCOM*, vol. 1, Oct 2002, pp. 377-382.
- [37] J. Gunn, K. Baron and W. Ruczczyk, "A low power DSP core-based software radio architecture," *IEEE Journal on Selected Areas in Commun.*, vol. 17, no. 4, Apr. 1999, pp. 574-590.
- [38] A. Truter, E. Wolmarans, "A software defined radio architecture with power control for 3GW-CDMA systems," *Alcatel Altech Telecoms*, Boksburg, South Africa.
- [39] K. Moessner, D. Bourse, D. Greifendorf and J. Stammen, "Software radio and reconfiguration management," *Computer Commun.*, vol. 26, issue 1, Jan., 2003, pp. 26-35.
- [40] N. Apostolou, *Signal Synthesis with Dynamically-Charged Power Spectral Density in a Software Defined Radio Transmitter*, Master's Thesis, Naval Postgraduate School, Monterey, California, Sept. 2003.
- [41] S. Lam, "Implementing DSP algorithms in FPGAs," *XCell Journal Online*, Winter 2004. Retrieved on June 13, 2005 from http://www.xilinx.com/publications/xcellonline/xcell_51/xcell_dspalgorithms51.htm
- [42] P. Fanden, *Evaluation of Xilinx System Generator*, Master's Thesis, Linkoping University, Sweden, Dec. 2001.

- [43] N. Lall and E. Cigan, "Plug and play methodologies for FPGA based signal processing". Retrieved on June 13, 2005 from http://www.fgpajournal.com/articles_2005/20050308_xilinx.htm.
- [44] Xilinx User Guide from http://www.xilinx.com/products/software/sysgen/app_docs/user_guide.htm.
- [45] ModelSim Reference. from http://www.xilinx.com/ise/optional_prod/mxe.htm.
- [46] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," ACM Computing Surveys, vol. 34, issue 2, June 2002, pp. 171-210.
- [47] D. Chin and S. Lam. "Implementing DSP designs with Xilinx System Generator and implementation tools." Retrieved on June 13, 2005 from <http://www.synplicity.com/literature/syndicated/pdf/DSP.pdf>.
- [48] Push-button performance using System Generator for DSP. from http://www.xilinx.com/products/software/sysgen/sysgen_sellsheet.pdf.
- [49] B. Sklar, Digital communications: Fundamentals and Applications, 2nd edition. Prentice-Hall, Upper Saddle River, NJ, 2001.
- [50] Blockset reference, Xilinx user guide. Retrieved on Sep 18, 2005 from http://www.xilinx.com/products/software/sysgen/app_docs/user_guide_Chapter_10_Section_3_Subsection_61.htm.
- [51] C. Wang, D. Sklar and D. Johnson, "Forward error correction coding," Crosslink-Space Commun., vol. 3, no.1, Winter 2001-2002. Retrieved on Nov 16, 2005 from <http://www.aero.org/publications/crosslink/winter2002/04.html>
- [52] O. Ugweje and S. Martin, "Channel coding performance of multitone CDMA system," Proceedings of SPIE, vol. 4529, Nov 2001, pp. 1-12.

APPENDIX – A

ADC - Analog-to Digital Converter

AM - Analog Modulation

AMPS - Advanced Mobile Phone Service

ASIC- Application Specific Integrated Circuit

AWGN - Additive White Gaussian Noise

BER - Bit Error Rate

BPSK - Binary Phase Shift Keying

CDMA - Code Division Multiple Access

CORBA - Common Object Request Broker Architecture

DAC - Digital-to-Analog Converter

DCS - Digital Cellular System

DDC - Digital Down Converter

DDS - Direct Digital Synthesizer

DSP - Digital Signal Processor

DUC - Digital Up Converter

FEC - Forward Error Correction

FFT - Fast Fourier Transform

FIR - Finite Impulse Response

FM - Frequency Modulation

FPGA - Field Programmable Gate Array

GPP - General Purpose Processor

GSM - Global System for Mobile

GUI - Graphical User Interface

HR - Hardware Radio

ISE - Integrated Software Environment

ISDR - Ideal Software Defined Radio

JTAG - Joint Test Action Group

LCM - Least Common Multiple

PN - Pseudorandom Noise

PSD - Power Spectral Density

QPSK - Quadrature Phase Shift Keying

RF - Radio Frequency

SCR - Software Controlled Radio

SDR - Software Defined Radio

SNR - Signal-to-Noise Ratio

TACS - Total Access Communication System

USR - Ultimate Software Radio

VHSIC - Very High Speed Integrated Circuit

VHDL - Very Hardware Description Language

WI-FI - Wireless-Fidelity

WLAN - Wireless Local Area Network