FRAMEWORK FOR THE DESIGN AND IMPLEMENTATION OF SOFTWARE DEFINED RADIO BASED WIRELESS COMMUNICATION SYSTEM

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Pallavi Mannar Mannan

December, 2005
FRAMEWORK FOR THE DESIGN AND IMPLEMENTATION OF SOFTWARE DEFINED RADIO BASED WIRELESS COMMUNICATION SYSTEM

Pallavi Mannar Mannan

Thesis

Approved:                                                   Accepted:

_________________________                                _______________________
Advisor                       Department Chair               Dr. Okechukwu C. Ugweje    Dr. Alex De Abreu Garcia

_________________________                                _______________________
Committee Member               Dean of the College             Dr. Igor Tsukerman          Dr. George K. Haritos

_________________________                                _______________________
Committee Member               Dean of the Graduate School    Dr. George Giakos           Dr. George R. Newkome

Date

ii
ABSTRACT

The aim of this thesis is to design and implement a software defined radio based wireless communication system. 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 2E field programmable gate array. A comparison of the simulation results with the results obtained from implementing the system on Spartan 2E hardware is presented and discussed. It is shown that the simulation results and experimental results are similar.
ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Okechukwu Ugweje for giving me the freedom of thought and expression while performing my research. He has been a constant source of inspiration and has provided consistent succour and valuable suggestions throughout this project without which this work wouldn’t have been possible. I would like to thank Dr. Igor Tsukerman and Dr. George Giakos for serving in my committee. I would also like to thank the Department of Electrical and Computer Engineering at the University of Akron for giving me the opportunity to do a Master’s degree in this honorable institution. 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 cooperation.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF ACRONYMS</td>
<td>xi</td>
</tr>
</tbody>
</table>

## CHAPTER

### I. INTRODUCTION

1.1 Introduction ........................................... 1

1.2 Motivation ............................................. 2

1.3 Contribution of Thesis ............................... 5

1.4 Outline of Thesis ..................................... 6

### II. FUNDAMENTALS OF SOFTWARE DEFINED RADIO ....... 8

2.1 Introduction ........................................... 8

2.2 Definition of Software Defined Radio ................ 9

2.3 Software Defined Radio Concepts ..................... 13

2.3.1 Conventional Versus Software Defined Radio ....... 13

2.3.2 Characteristics of Software Defined Radio ....... 13

2.3.3 Advantages and Disadvantages of Software Defined Radio ...... 15

2.4 Software Defined Radio Implementation Platforms........ 16

2.5 Technical Challenges .................................. 19
2.6 Summary ................................................................. 21

III. FRAMEWORK FOR SDR BASED DESIGN AND IMPLEMENTATION ...... 23

3.1 Introduction .......................................................... 23

3.2 Wireless Communication System Model for Software Defined Radio ...... 23
  3.2.1 Radio Frequency Section ....................................... 23
  3.2.2 Intermediate Frequency Section ................................ 24
  3.2.3 Baseband Section ............................................... 25

3.3 Design Process for Reconfigurable Computing ............................. 25
  3.3.1 High Level Simulation ......................................... 26
  3.3.2 VHDL Description ............................................. 29
  3.3.3 HDL Co-Simulation ........................................... 30
  3.3.4 Hardware Co-Simulation ..................................... 30

3.4 Test-bed Implementation ............................................. 31
  3.4.1 Simulation with SIMULINK and System Generator .......... 32
  3.4.2 HDL Co-Simulation ........................................... 33
  3.4.3 Hardware Co-Simulation ..................................... 34

3.5 Summary .............................................................. 35

IV. EXPERIMENTAL SETUP, SIMULATION AND RESULTS ..................... 37

4.1 Introduction .......................................................... 37

4.2 System Model for Simulation ......................................... 37
  4.2.1 Transmitter Model ............................................. 38
  4.2.2 Receiver Model ................................................ 40

4.3 System Simulation .................................................. 42
4.4 Simulation Results ..................................................... 46
  4.4.1 Constellation Diagrams ................................. 46
  4.4.2 Eye Diagrams ..................................................... 48
  4.4.3 Output Waveform ............................................ 50
  4.5 Real time Implementation ........................................ 50
  4.6 Summary ........................................................... 56

V. SUMMARY AND CONCLUSION .................................. 57
  5.1 Summary .......................................................... 57
  5.2 Suggestions for Future Work ................................. 59

REFERENCES ............................................................. 60
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Difference between Conventional and Software Defined Radio</td>
<td>13</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Model of software defined radio processes</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Hardware implementation platforms for SDR</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Functional block diagram of a wireless communication system</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>Design process for reconfigurable computing</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Illustration of test-bed implementation process</td>
<td>31</td>
</tr>
<tr>
<td>3.4</td>
<td>System Generator dialog box</td>
<td>34</td>
</tr>
<tr>
<td>4.1</td>
<td>Software definable baseband communication system</td>
<td>38</td>
</tr>
<tr>
<td>4.2</td>
<td>Example of puncturing rate 1/2 encoder</td>
<td>40</td>
</tr>
<tr>
<td>4.3</td>
<td>Example of depuncturing rate 2/3 encoded data</td>
<td>41</td>
</tr>
<tr>
<td>4.4</td>
<td>Encoding and puncturing using System Generator blocks</td>
<td>43</td>
</tr>
<tr>
<td>4.5</td>
<td>Modulation and spreading of the encoded signal</td>
<td>44</td>
</tr>
<tr>
<td>4.6</td>
<td>Despreading and demodulation of the received signal</td>
<td>44</td>
</tr>
<tr>
<td>4.7</td>
<td>Depuncturing and decoding model using System Generator blocks</td>
<td>45</td>
</tr>
<tr>
<td>4.8</td>
<td>Constellation diagram for QPSK modulated signal with SNR = 15 dB</td>
<td>47</td>
</tr>
<tr>
<td>4.9</td>
<td>Constellation diagram for QPSK modulated signal with SNR = 20 dB</td>
<td>47</td>
</tr>
<tr>
<td>4.10</td>
<td>Constellation diagram for BPSK modulated signal with SNR = 20 dB</td>
<td>48</td>
</tr>
<tr>
<td>4.11</td>
<td>Eye diagram for QPSK modulated signal with SNR = 15 dB</td>
<td>49</td>
</tr>
<tr>
<td>4.12</td>
<td>Eye diagram for QPSK modulated signal with SNR = 20 dB</td>
<td>50</td>
</tr>
</tbody>
</table>
4.13 Input and output waveforms of the communication system ..................................51
4.14 Experimental setup for hardware co-simulation .................................................52
4.15 Spartan 2E FPGA hardware board used for hardware co-simulation ............ 53
4.16 Command window showing the progress of implementation tools ............... 54
4.17 Hardware co-simulation library ................................................................. 55
4.18 Constellation diagram for hardware co-simulation ........................................ 56
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analog-to Digital Converter</td>
</tr>
<tr>
<td>AM</td>
<td>Analog Modulation</td>
</tr>
<tr>
<td>AMPS</td>
<td>Advanced Mobile Phone Service</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Intergrated Circuit</td>
</tr>
<tr>
<td>AWGN</td>
<td>Additive White Guassian Noise</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>BPSK</td>
<td>Binary Phase Shift Keying</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital-to-Analog Converter</td>
</tr>
<tr>
<td>DCS</td>
<td>Digital Cellular System</td>
</tr>
<tr>
<td>DDC</td>
<td>Digital Down Converter</td>
</tr>
<tr>
<td>DDS</td>
<td>Direct Digital Synthesizer</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
</tr>
<tr>
<td>DUC</td>
<td>Digital Up Converter</td>
</tr>
<tr>
<td>FEC</td>
<td>Forward Error Correction</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fouier Transform</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>FM</td>
<td>Frequency Modulation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>GPP</td>
<td>General Purpose Processor</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HR</td>
<td>Hardware Radio</td>
</tr>
<tr>
<td>IF</td>
<td>Intermediate Frequency</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual Property</td>
</tr>
<tr>
<td>IS</td>
<td>Interim Standard</td>
</tr>
<tr>
<td>ISE</td>
<td>Integrated Software Environment</td>
</tr>
<tr>
<td>ISDR</td>
<td>Ideal Software Defined Radio</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>LCM</td>
<td>Least Common Multiple</td>
</tr>
<tr>
<td>LO</td>
<td>Local Oscillator</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PCS</td>
<td>Personal Communication Services</td>
</tr>
<tr>
<td>PN</td>
<td>Psuedorandom Noise</td>
</tr>
<tr>
<td>PSD</td>
<td>Power Spectral Density</td>
</tr>
<tr>
<td>QPSK</td>
<td>Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>SCR</td>
<td>Software Controlled Radio</td>
</tr>
<tr>
<td>SDR</td>
<td>Software Defined Radio</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
</tbody>
</table>
TACS  Total Access Communication System
USR  Ultimate Software Radio
VHSIC  Very High Speed Integrated Circuit
VHDL  VHSIC Hardware Description Language
WI-FI  Wireless-Fidelity
WLAN  Wireless Local Area Network
XST  Xilinx Synthesis Technology
CHAPTER I

INTRODUCTION

1.1 Introduction

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 [1]. 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. 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 [2]. 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 [3].

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 [3].

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 [4]. 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 [5]. 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 [3].

(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 [5]. 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 [6].

(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 [3].

(4) Cost: Every time a new technology evolves, it results in the migration of func-
tions 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 [7].

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 [6].

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 future of wireless generation services, especially as we migrate from third generation (3G) to fourth generation (4G) wireless standards.

1.3 Contribution of Thesis

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

(1) The development of a framework for the design and implementation of software defined radio based wireless communication systems.
(2) 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.

(3) 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).

(4) Initiation of the development of a testbed for the design and implementation of SDR based wireless communication system.

(5) Presentation of results based on the simulation and actual experimentation.

(6) Evaluation of the performance of the SDR system in terms of signal-to-noise ratio in an additive white Gaussian noise (AWGN) channel.

(7) Evaluation of the performance for the simulation of an SDR system and real time implementation on the Xilinx Spartan 2E field programmable gate array (FPGA) platform [8]. The results obtained during simulation and experiment are compared.

1.4 Outline of Thesis

In this thesis, the fundamentals of software defined radio are first presented in Chapter II, which includes the general background information and various definition for SDR. The presentation includes the difference between the SDR and conventional radio, char-
acteristics and advantages of SDR, possible design issues, and the platform choices for implementing SDR based wireless communication systems.

In Chapter III, 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 IV. Results of the simulation including constellation diagrams, eye diagrams, output waveforms, for different modulation techniques are presented and analyzed. The system is implemented on the Xilinx Spartan 2E FPGA platform [8]. The results of the implementation are compared with the simulation results.

Chapter V summarizes the content of the thesis. Also, possible extensions of the thesis are discussed.
CHAPTER II

FUNDAMENTALS OF SOFTWARE DEFINED RADIO

2.1 Introduction

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 [9]. 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 [10].

Efforts to define a unique worldwide standard to overcome the above problems often results in a new standard [11]. 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 envi-
ronment in which it is located [12]. 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 [13], [14], [15], [16]. 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 [2].

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

Because of the many features of SDR, there are many definitions available. The level of reconfigurability required to define a radio function in software is still not clear. A radio
that includes a microprocessor or digital signal processor does not necessarily qualify as a software radio. However, a radio that defines in software its functions such as modulation, error correction, encryption processes, exhibits some control over the RF hardware, and can be reprogrammed, qualifies as a software defined radio [3]. The degree of configurability is largely determined by a complex interaction between a number of common issues in radio design, including system engineering, antenna form factors, RF electronics, baseband processing, speed and reconfigurability of the hardware and power supply management [3].

One definition of SDR is provided by the SDR forum [17], 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, superheterodyne 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.

Please note that, software radios use digital techniques, but software controlled digital radios are generally not software radios [2]. 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 [2], [12], [18], [19]:

1. “Flexible transceiver architecture, controlled and programmable by software.”
2. “Signal processing able to replace, as much as possible, radio functionalities.”
3. “A system with air interface downloadability. That is, it is possible to dynamically reconfigure radio equipment by downloadable software, at every level of the protocol stack.”
4. “Software realization of terminals.”
5. “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 reconfigurability, 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 [20]. 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 [20].
2.3 Software Defined Radio Concepts

In this section, the difference between conventional radio and software defined radio is presented. Also, the characteristics of SDR, its advantages and disadvantages are discussed.

2.3.1 Conventional Versus Software Defined Radio

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

Table 2.1: Difference between conventional and software defined radios

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

2.3.2 Characteristics of Software Defined Radio

Consider a system model of a software defined radio shown in Figure 2.1. The receiver implemented with a smart antenna that provides a gain versus direction characteristic to minimize interference, multipath and noise. The smart antenna provides similar benefits for the transmitter. Digitization of the signal is carried out as close as possible to the antenna in
the receiver using the analog-to-digital converter. Similarly, the signal is converted to the
analog domain as late as possible in the transmitter using digital-to-analog converter. The
digitization of the signal is mostly done in the intermediate frequency (IF) range. Software
defined radio uses analog-to-digital converter to digitize the signal in the IF range, thus
overcoming the problems like carrier offset and imaging involved in digitizing the signal
using super-heterodyne method commonly used in conventional radios. Channelization and
sample rate conversion on the transmit path is used to interface the digital hardware to the
digital-to-analog converter and to interface ADC to the processing hardware on the receive
path. Baseband processing is performed in software using digital signal processors (DSPs),
field programmable gate arrays (FPGAs), application specific integrated circuits (ASICs),
or general purpose processors (GPPs). The algorithm used to modulate or demodulate
signal may use a variety of software methodologies, such as middleware, e.g., common
object request broker architecture (CORBA) [3].

![Diagram of software defined radio processes](image)

Figure 2.1: Model of software defined radio processes [3].
2.3.3 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 [4].

With SDR, the same piece of hardware will be configured to perform different functions. The reconfigurability of the platform will ensure hardware reusability. System reprogrammability 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 [3].

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 [6]. 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:

1. The difficulty of designing software for various target systems or standards.
2. The difficulty of designing air interfaces to digital signals and algorithms for different standards.
3. 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 2.2. All the four platforms shown in Figure 2.2 possess a level of reprogrammability or reconfigurability (i.e., the ability to modify the hardware or software) [21].

The DSP platform is essentially a microprocessor based system optimized for digital signal processing applications [3], [22]. DSPs can be programmed repeatedly with a high level language such as C, MATLAB [3]. 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.

![Software Defined Radio Hardware Implementation Platforms](image)

**Figure 2.2:** 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 (bitstream) into the static on-chip random access memory [7]. This is similar to the object code of a microprocessor, in which the bitstream is the product of compilation tools that translate the high level abstractions produced by a designer into equivalent but low level executable code [23].

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 [24].

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 [3]. 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 [25].

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 trade-off 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 [7].

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 [7].

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:

(1) 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 [12]. The primary goal of this transceiver was to extend the digital domain at the IF stage and keeping the RF stage analog [12].

(2) 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 reprogrammability 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 [12]. Hence, implementing communication functions in software presents a major challenge in practical systems.

3) 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 multirate digital signal processing that can be used to improve the flexibility of the digitization process [3], [26].

4) 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 [27], [28].

5) 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 ref-
ereference 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 [20].

(6) Receiver complexity is typically four or more times the transmitter complexity [3]. 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.

(7) 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 [3].

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 thesis. More information can be found in [20], [29].

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,
advantages and disadvantages were also presented. The choices of hardware available for real time implementation and technical challenges involved in implementation were discussed.
CHAPTER III

FRAMEWORK FOR SDR BASED DESIGN AND IMPLEMENTATION

3.1 Introduction

In this chapter, a conventional wireless communication system model is briefly reviewed. This is followed by the design process for reconfigurable computing for the SDR based wireless communication system. Finally, steps involved in implementing the design on a reconfigurable computing platform are presented. In this thesis, only the baseband section of a communication system is modeled and simulated.

3.2 Wireless Communication System Model for Software Defined Radio

The generic wireless communication system consists of a transmitter, channel and a receiver. The functional block diagram of the digital transceiver is shown in Figure 3.1 [3], [30].

3.2.1 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.

![Diagram of a wireless communication system](image)

Figure 3.1: Functional block diagram of a wireless communication system.

### 3.2.2 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.

3.2.3 Baseband Section

The baseband section performs operations, such as error correction, equalization, fre-
quency hopping, modulation, demodulation, spreading, despreading 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 com-
munication system, many modulation techniques such as MPSK, QPSK, DPSK, etc., are
used. In this thesis, the QPSK and BPSK modulation and demodulation techniques are
dealt. More details on the specific blocks that were implemented in this thesis are provided
in Chapter IV.

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 II.

3.3 Design Process for Reprogrammable 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 [31]. The only common approach involves block diagram system model [31].

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.

3.3.1 High Level Simulation

The algorithms and concepts used to define the system is 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 [8]. 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.

![Design process for reconfigurable computing.](image)

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 blocksets for DSP, communications, wavelets and many more. MATLAB SIMULINK is used in this thesis as the high level development tool in the design process.
Xilinx System Generator [8], 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 [32].

Xilinx’s System Generator for DSP tool was the first tool to bridge the gap between DSP and FPGA applications [31]. 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 [33].

System Generator automates the design process, debugs, 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 [31].

System Generator provides high-level abstractions that are automatically compiled into FPGA bitstream. It is delivered both with a predefined Xilinx blockset 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 [31].
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 [33].

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 [34]. These test vectors are later used to check the differences between the SIMULINK simulation, HDL simulation and hardware implementation of the model.

3.3.2 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 [34]. 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.
3.3.3 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 testbench 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 [35] 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.

3.3.4 Hardware Co-Simulation

Hardware co-simulation capability accelerates simulation and verification of design in hardware. System Generator’s hardware-in-the-loop co-simulation [33] interfaces makes a push-button flow and brings 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 [23], [36]. The VHDL modules can be transferred to the hardware using Xilinx synthesis technology (XST) synthesis tool, which comes with Xilinx’s ISE [37].
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 [36].

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.

3.4 Test-bed Implementation

![Figure 3.3: Illustration of test-bed implementation process [38].](image)

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 synthesiz-
able VHDL code including testbench [37]. 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.

3.4.1 Simulation with SIMULINK and System Generator

In this thesis, the algorithm is designed and simulated using Xilinx System Generator system level tool. For an exact representation of the FPGA implementation, the Xilinx blockset in MATLAB SIMULINK is used. The Xilinx blockset 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 [37]:

(1) Start the design by implementing the Xilinx blocks in the MATLAB SIMULINK model design.

(2) Select the Xilinx System Generator block and add it on the top of the design hierarchy.

(3) “Gateway In” and “Gateway Out” blocks are used to define the inputs and outputs to the Xilinx design. Xilinx gateway blocks automatically converts 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 thesis, we used all three representations in the simulation.

(4) 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.

3.4.2 HDL Co-Simulation

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 testbench enables design verification upon implementation. The following are the key steps in the HDL co-simulation process [37]:

(1) 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, testbench 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, Synplify Pro and Mentor Graphics. Also, HDL can be chosen either as VHDL or Verilog.

(2) 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 testbench 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.

![System Generator dialog box](image)

Figure 3.4: System Generator dialog box.

3.4.3 Hardware Co-Simulation

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 [37]:

1. 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.

2. After initiating the “Generate” button, the code generator is invoked and produces an FPGA configuration bitstream for the design that is suitable for hardware co-simulation. System Generator not only generates the HDL and netlist 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 Figure 4.16 in Chapter IV.

3. After the FPGA configuration bitstream 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 thesis, only hardware co-simulation is performed using Spartan 2E FPGA [8].

3.5 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 system model 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 next chapter.
CHAPTER IV

EXPERIMENTAL SETUP, SIMULATION AND RESULTS

4.1 Introduction

This chapter describes the experimental setup, simulation and results for the software defined radio based wireless communication system. The system is modeled and simulated using MATLAB SIMULINK and Xilinx’s System Generator. The simulated system is then targeted to the Xilinx’s Spartan 2E FPGA for real time implementation.

4.2 System Model for Simulation

In this thesis, software definable baseband section of the 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 III. The transmitter section of the baseband is implemented in Spartan 2E 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 is simulated. The receiver side consists of despreading, demodulation, depuncturing and Viterbi decoding. Figure
4.1 shows the software definable baseband section of the communication system model implemented in this thesis.

![Software definable baseband communication system](image1)

**Figure 4.1**: Software definable baseband communication system.

### 4.2.1 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. More information on convolutional encoder can be found in [39] - [46].

In this thesis, a rate 1/2 convolutional encoder with constraint length 7 and code array 127 and 117 is used. Please note that the code array used is the optimal code array of constraint length 7 [40]. 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 [34]. The puncture codes are a bit pattern that identifies the bits from the encoder to be transmitted and the codes used in this thesis 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 [41].

Figure 4.2 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 4.2, 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 4.2: 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 thesis.

4.2.2 Receiver Model

At the receiver, the signal is first despread and then demodulated. Then error correction is applied to the demodulated data. The signal is despread with PN sequence generator with a spreading gain of 16 and demodulated using QPSK technique.

The demodulated and despread 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 4.3 shows an example of depuncturing rate 2/3 punctured data.

![Diagram of depuncture and Viterbi decoder](image)

**Figure 4.3: Example of depuncturing rate 2/3 encoded data.**

The depuncture blocks have the same puncture 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 null 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 [39]. 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 [34]. 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 traceback length parameter [34]. More information on Viterbi decoder can be found at [34] - [46].

The constraint length of 7 and the code array 127 and 117 used for decoding are the same as in convolutional encoder. The traceback length parameter, that is, the number of trellis states processed before the decoder makes a decision on a bit, is set to 96. The decoder outputs the data bits which are later grouped accordingly.

4.3 System Simulation

The simulation model for punctured convolutional encoder is shown in Figure 4.4. In this thesis, punctured convolutional 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 convolutional 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 (vin) 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 (vout) output is set to high, which is sent to the Viterbi decoder valid input port.
Figure 4.4: Encoding and puncturing using System Generator blocks.

Figure 4.5 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 concatenator 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 despreaded using the same PN sequence generator and demodulated using QPSK demodulator. This is shown in Figure 4.6. 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 4.5: Modulation and spreading of the encoded signal.

Figure 4.6: 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
4.7 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.

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

Figure 4.7: Depuncturing and decoding model using System Generator blocks.
output waveform is compared. The other outputs from the Viterbi decoder (ber, ber_done, norm) are used to check for the errors while decoding.

4.4 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.

4.4.1 Constellation Diagrams

Figure 4.8 shows the constellation diagram of the modulated signal with signal-to-noise ratio (SNR) of 15 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 20 dB is shown in Figure 4.9.
Since the SNR is high, the constellation points form a more dense cloud around the central point thus reducing the transmission error.

Figure 4.8: Constellation diagram for QPSK modulated signal with SNR = 15 dB.

Figure 4.9: Constellation diagram for QPSK modulated signal with SNR = 20 dB.

Comparing Figures 4.8 and 4.9, it can be seen that the constellation points of Figure 4.9 are denser than in Figure 4.8. 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 4.9 compared to Figure 4.8.

Figure 4.10: Constellation diagram for BPSK modulated signal with SNR = 20 dB.

The constellation diagram for BPSK is shown in Figure 4.10. 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 [39]. 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 [47].

4.4.2 Eye Diagrams

The measure of distortion, timing jitters and noise margin can be found from the eye diagrams. Figure 4.11 shows the eye diagram of the modulated signal with SNR of 15 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.

![Eye Diagram](image)

Figure 4.11: Eye diagram for QPSK modulated signal with SNR = 15 dB.

Figure 4.12 shows the eye diagram for SNR of 20 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 4.11 and 4.12, it can be seen that the distortion in Figure 4.12 is 0.25 when compared with the distortion in Figure 4.11 which is 0.5. Similarly, the timing jitter is 0.1 for SNR = 20 dB and 0.2 for SNR = 15 dB. The other parameter noise margin is 0.65 and 0.5 for SNR = 30 dB and 15 dB respectively.
4.4.3 Output Waveform

Another indicator of performance is the observation of the output waveform compared to the input waveform. Figure 4.13 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 4.12: Eye diagram for QPSK modulated signal with SNR = 20 dB.

4.5 Real time Implementation

After simulation and analysis of the results, the system is implemented in real time via hardware co-simulation on a Spartan 2E FPGA [8]. The Digilent D2-SB hardware board,
which has a Spartan 2E FPGA chip on it is used to implement the system in real time. The experimental setup for hardware co-simulation is shown in Figure 4.14. The Spartan 2E is programmed through the joint test action group (JTAG) programming cable.

The hardware co-simulation is performed by powering on the Digilent D2-SB board and connecting the JTAG cable from the board to the printer port of the personal computer (PC) as shown in Figure 4.14.

![Input and output waveforms of the communication system.](image)

The Digilent D2-SB circuit board provides a complete circuit development platform centered on a Xilinx Spartan 2E FPGA [48] and is shown in Figure 4.15.

The Digilent D2-SB is made by Digilent Inc., and is used for hardware co-simulation. Some of the features of this board include [48]:

1. A Xilinx XC2S200E-200 FPGA with 200K gates and 350 MHz operation.
2. 143 user I/Os routed to six standard 40-pin expansion connectors.
3. A socket for a JTAG-programmable 18V00 configuration Flash ROM.
4. Dual on-board 1.5A power regulators (1.8V and 3.3V).
(5) An 50 MHz oscillator, and a socket for a second oscillator if needed.

(6) A JTAG programming port, a status LED and pushbutton for basic I/O.

The Spartan 2E FPGA and the 18V00 ROM on the Digilent D2-SB, and any programmable devices on peripheral boards attached to the Digilent D2-SB can be programmed via the JTAG port. In this thesis, only the Spartan 2E FPGA was used. The transmitter side of the model, which was simulated, is targeted to Spartan 2E FPGA through the JTAG programming chain.

Figure 4.14: Experimental setup for hardware co-simulation.

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 2E 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.

![Spartan 2E FPGA hardware board for hardware co-simulation.](image)

Figure 4.15: Spartan 2E FPGA hardware board for hardware co-simulation.

The code generator produces a FPGA configuration bitstream for the design, suitable for hardware co-simulation. System Generator not only generates the HDL and netlist 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 bitstream file as shown in Figure 4.16. This window shows the progress and output of each tool as it is runs. It can be seen from Figure 4.16 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 bitstream contains the hardware associated with the model.

![Command window showing the progress of implementation tools.](image)

Figure 4.16: Command window showing the progress of implementation tools.

Hardware co-simulation compilation targets automatically create bitstreams 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 bitstream. A SIMULINK library is also created in order to store the hardware co-simulation block information.

Figure 4.17 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 thesis, Figure 4.17 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 4.17: Hardware co-simulation library.

Figure 4.18 shows the constellation diagram output obtained by implementing the model in hardware. Comparing Figure 4.18 to Figure 4.10, 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.
4.6 Summary

In this Chapter, the experimental setup, simulation and corresponding results for the baseband section of a wireless communication system is presented. The parameters set to carry out the simulation were also presented. Also real time implementation of the system is presented. The results obtained from the simulation were discussed and the simulation results were compared with the results from real time implementation. It was demonstrated that the results obtained from simulation and hardware implementation are the same.
CHAPTER V

SUMMARY AND CONCLUSION

5.1 Summary

This thesis consists of two major tasks. First, a study of the software defined radio concept and, second testbed development and implementation of software radio transmitter and receiver.

The first task of this thesis was to investigate the current state of the art in software defined radio. This is a very large subject area with many promising applications. A global overview of the software radio is given, and some of the most important aspects of software radio such as the definition, characteristics, advantages, technological challenges and the hardware choices available for implementation are described and analyzed.

The second task is to build a software defined radio based wireless communication system. In this thesis, the baseband section of the communication system was simulated and then targeted to hardware implementation.

In Chapter III, 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 IV, the system that has been implemented, the simulation and the results are presented. 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 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.

This thesis 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, testbench 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.
5.2 Suggestions for Future Work

The software defined radio that has been implemented in this thesis is not fully complete. Only the baseband stage of the wireless communication system has been implemented. The other two major stages, radio frequency and intermediate frequency stages have not been implemented. Therefore, further research could focus on an end-to-end implementation of a wireless communication system in software defined radio.

The channel is considered to have only additive white Gaussian noise. Inter-symbol interference, Doppler shift, phase error, multipath fading, etc., can be added to the channel, in order to closely simulate real life systems.

The baseband stage that has been implemented is also not complete. The implemented software defined radio is simple with coding, modulation and spreading. This thesis could be extended by adding more functionalities such as equalization, carrier recovery, phase recovery, etc., needed to counteract the noise and interference problems of a communication system.

A complete communication system for any particular air standard (e.g., GSM, IS-95, or CDMA2000), can be defined in software. In this thesis, 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.
REFERENCES


