INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

NOTE TO USERS

This reproduction is the best copy available.

UMI®

University of Alberta

DESIGN AND IMPLEMENTATION OF TIER 3 SOFTWARE DEFINED RADIO RECEIVERS

by

Jung Ko 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2005

To my parents and my wife Peipei

Abstract

Software Defined Radio (SDR) is one of the key technologies in the future of wireless communication industry. A *Tier-3* SDR, or *Ideal Software Radio* receiver is a device that performs all demodulation (RF, IF and baseband) entirely in the digital domain. In this work we present the design and implementation of AM and Short Wave receivers that belong to this type of ideal software radios. These receivers are implemented in Field Programmable Gate Array (FPGA) devices and we show how to integrate both receivers into the same device along with their performance measurements. The AM and Short Wave receivers have an equivalent gate count of 690,000 and 780,000 gates respectively. The device utilization of 4-input LUTs is 6% for the AM receiver and 25% for the Short Wave receiver using a Xilinx Virtex-E 2000 FPGA. The output SNR are 50 dB for the AM receiver and 55 dB for the Short Wave receiver using 8 and 9 bits of quantization respectively.

Acknowledgements

The author gratefully acknowledges the financial support from the following agencies:

- Natural Science and Engineering Research Council (NSERC)
- Alberta Informatics Circle of Research Excellence (iCORE)
- Canadian Space Agency (CSA)
- University of Alberta Walter H. Johns Graduate Fellowship
- University of Alberta Department of Electrical and Computer Engineering

He also would like to thank the following people:

- Dr. Gaudet for his supervision and assistance in this project.
- Robert Hang for so many hours of insightful technical help and advice.
- Paul Greidanus for his friendly technical support regarding the use of software tools.
- Dave Nguyen for his assistance in using the milling machine.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ADC	Analog to digital converter
ALU	Arithmetic logic unit
AM	Amplitude modulation
ASIC	Application specific IC
AWG	American wire gauge
BPF	Bandpass filter
BRAM	Block RAM
CCM	Configurable computing machines
DAC	Digital to analog converter
DDC	Digital direct downconverter
DDS	Direct digital synthesizer
DSB	Double side-band
DSP	Digital signal processing
EDK	Embedded development kit
EMC	External memory controller
ESG	Enhanced signal-generator
ETC	Electric toll collection
FF	Flip-flop
FIR	Finite impulse response
FPGA	Field programmable gate array
FSL	Fast simplex link
FTP	File transfer protocol
GMSK	Gauss mean shift keying
GPIO	General purpose input/output
GPRS	General packet radio service
GSM	Global system for mobile communications
IF	Intermediate frequency
IIR	Infinite impulse response
IOB	Input output block
JTAG	Joint test action group
LED	Light emitting diode

LMB	Local memory bus
LNA	Low noise amplifier
LPF	Low pass filter
LSB	Least significant bit
LUT	Lookup table
MDM	Microblaze debug module
MEMS	Micro-electromechanical systems
MIPS	Million instructions per second
MMIC	Monolithic microwave integrated circuit
MSB	Most significant bit
MW	Medium wave
OPB	On-chip peripheral bus
OSR	Oversampling ratio
RAM	Random access memory
PHS	Personal handyphone system
RF	Radio frequency
SDR	Software defined radio
SIMD	Single instruction multiple data
SNR	Signal to noise ratio
SW	Short wave
UART	Universal asynchronous receiver-transmitter
UML	Universal modelling language
UMTS	Universal mobile telecommunications system
WAV	Waveform sound format
WCDMA	Wideband code division multiple access

List of Symbols

BW	Filter passband bandwidth
F_c	Carrier frequency
F_{local}	Local oscillator frequency
$F_{Nyquist}$	Nyquist frequency
F_s	Sampling frequency
IF	AM intermediate frequency
$IF1$	Short Wave first intermediate frequency
$IF2$	Short Wave second intermediate frequency
OPS	Operations per second
OSR	Oversampling ratio

Chapter 1

Introduction

> You see, wire telegraph is a kind of a very, very long cat. You pull his tail in New York and his head is meowing in Los Angeles. Do you understand this? And radio operates exactly the same way: you send signals here, they receive them there. The only difference is that there is no cat.
>
> Albert Einstein, when asked to describe radio
> US (German-born) physicist (1879 - 1955)

1.1 Motivation

With recent advances in digital communications, a vast number of devices can communicate over wireless channels very fast and efficiently. However, because of the large number of protocols and standards that define wireless communication, a given device can only communicate with other devices that support the same protocol. If one wishes to support multiple protocols at the same time, extra parallel hardware needs to be added to the device. This approach increases the cost and complexity of each device. Moreover, due to the rapidly changing nature of some wireless protocols, constant hardware changes are costly but inevitable. One novel approach to solve these problems was introduced in [1] and is known as *Software Defined Radio* (SDR). In a SDR scenario, the radio interface is implemented in a "programmable device" instead of application specific hardware. This method allows for software reconfigurability in the radio interface. With such a scheme, a device could support different standards by simply loading a different software "on-the-fly", thus avoiding costly hardware updates.

1.2 Definition

According to [2], the term *software radio* refers to "*reconfigurability of the radio interface by software, possibly using over-the-air download, with an often implicit assumption of analog-to-digital converter at the antenna.*"

The above definition is often referred as the *idealistic* definition. It usually implies that the transceiver has the *Analog-to-Digital Converter* (A/D) as close to the antenna as possible. This means that all three stages in a traditional superheterodyne transceiver, namely *Radio Frequency* (RF), *Intermediate Frequency* (IF) and *baseband* processing, are implemented in the digital domain. Figure 1.1 shows an example of such a system.

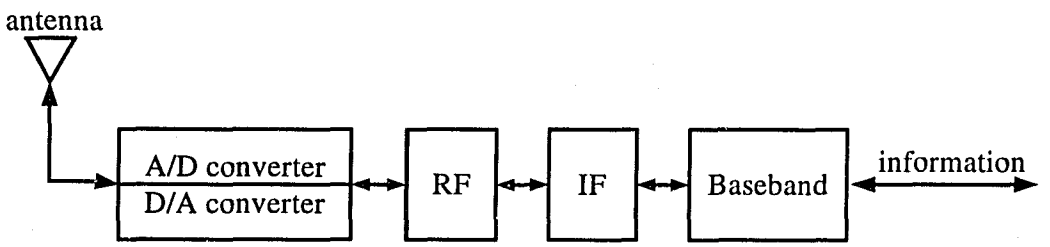

Figure 1.1: An "Ideal" Software Defined Radio architecture.

Clearly, the idealist definition is very hard to achieve. This is because we are required to digitally process the incoming samples in the RF domain, which is normally in the order of hundreds to even thousands of megahertz in frequency. This puts a very big burden on the A/D converter and the DSP blocks that are needed for such a system.

A more practical definition to software defined radio is therefore introduced in Figure 1.2. In this definition, we allow reconfiguration at *any* level in the protocol stack as opposed to *all* levels in the former idealistic definition. This takes away the burden of doing RF or even IF operations in the digital domain.

From Figure 1.2, we can see that as we go down in the protocol stack, the operating frequency becomes higher and therefore, the DSP complexity grows proportionally. Currently, practical systems such as state-of-the-art mobile phones do

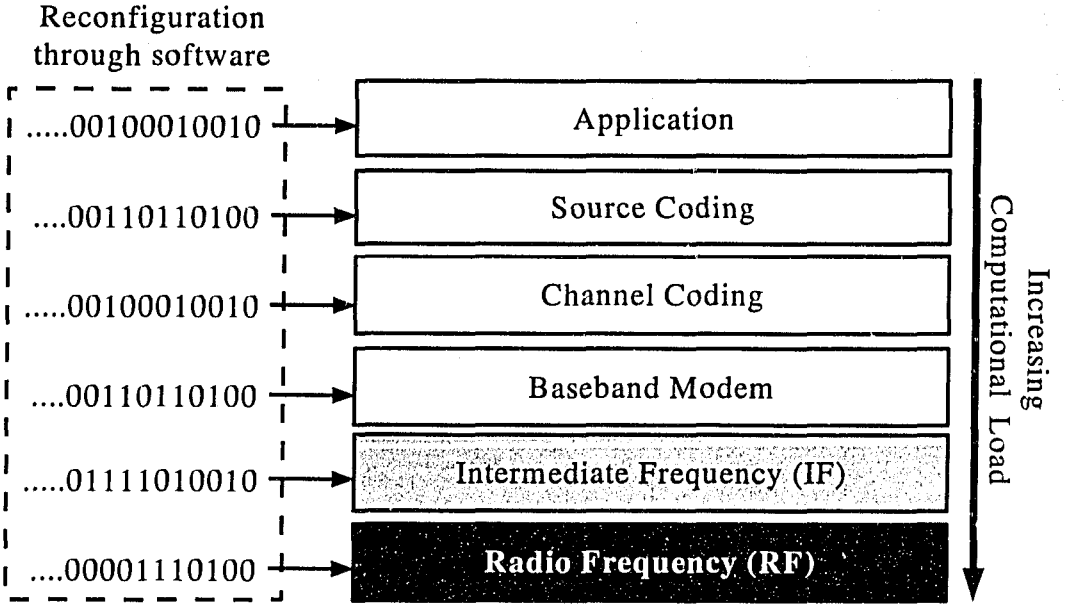

Figure 1.2: Pragmatic definition of Software Defined Radio architecture.

provide certain software reconfigurability at the application layer. However, software reconfigurability at the RF level is still in the research stage.

1.3 Classification

While the ideal definition of SDR is quite restrictive and impractical, the pragmatic definition is a little bit too vague. Indeed, a formal way to classify the different types of software radios is much needed. Fortunately, *SDR Forum* [3] provided a formal way to classify SDR into five different *tiers*.

Specifically, SDR can be classified as:

- *Tier 0. Hardware Radio:* This type of radio transceiver is entirely based on hardware. Any changes in the parameter of the device must be done through mechanical devices such as relays, switches or even opening the cover of the device.
- *Tier 1. Software Controlled Radio:* Some control functionality of this type of radio is done through software. However, the essential radio interfaces

such as RF and IF are still implemented in the analog domain.

- *Tier 2. Software Defined Radio:* In this type of wireless transceiver, the baseband processing is implemented in software. Higher frequency operations such as RF are still implemented in the analog domain using fixed hardware.
- *Tier 3. Ideal Software Radio:* In Tier 3 devices, we have all the benefits of Tier 2 devices; however, we replace all the analog parts, aside from the frontend *Low Noise Amplifier* (LNA), with their digital counterparts. This means that we are sampling the spectrum directly at the RF range. This type of radio closely resembles to the *ideal definition* described in Section 1.2.
- *Tier 4. Ultimate Software Radio:* This ultimate system is intended for comparison purposes as opposed to implementation. This type of device essentially would have very large processing capabilities that can adapt to any protocol in just few milliseconds. Tier 4 SDR devices can operate in any practical RF frequency ranges and it can output the desired information in any form that the user wants.

1.4 Benefits of SDR

Software Defined Radio provides several benefits to different sectors in the telecommunication industry, namely, the manufacturers, operators and end users [4].

From the manufacturer point of view, SDR provides the possibility to develop a generic hardware platform *while* the software is being developed at the same time. Incremental improvement in software is much less expensive compared to incremental improvements in hardware. In addition, mass production of generic hardware should lower the overall cost.

Another advantage of using a software driven radio interface is the ability to manage spectrum efficiently. This is an attractive feature for network operators that face a shortage of available spectrum. The transmitter and receiver could change "on-the-fly" the transmission frequency in order to avoid collision and interference

with other devices in the vicinity. This type of dynamic allocation of spectrum is indeed very urgently needed in metropolitan areas such as those in Japan [5].

Network operators can also introduce new and personalized services to the end user by simply uploading new software to the mobile receiver. This feature gives more competitive advantages to the network operators.

Finally, from the end user point of view, software defined radio should extend the lifetime of the equipment. As long as there is enough processing power, the user can use the same equipment and adapt it to newer communication protocols. In addition, because the radio interface is implemented in software, the roaming user can move from one geographical region to another without worrying about incompatible telecommunication protocols.

1.5 Research Work

The goal of this research work is to design and implement a working prototype of a Tier-3 software defined radio. Given the state of the art A/D converters, it should be possible to implement *ideal software radio* receivers for frequencies in the order of few megahertz. The main modulation scheme used in this frequency range is *Amplitude Modulation* (AM). Therefore, we set to build a software receiver for AM schemes in the Medium Wave (540 to 1600 kHz) and lower Short Wave (3 to 5 MHz) ranges.

The receiver should have the A/D converter as close to the antenna as possible. Moreover, we sample at the system clock frequency because we want to sample the entire available RF signal directly and do all the necessary processing in the digital domain. Sampling at such a high frequency enables this same architecture to work with different wireless standards, as long as the frequency of interest falls below the cutoff frequency of the anti-aliasing filter in front of the ADC. For instance, the same frontend configuration can be used for both the AM and Short Wave radio receivers described in this work.

Another requirement is to have a receiver that can demodulate the incoming signal in *real-time*. Given the number of operations per second required in the

receiver, the target platform will be *Field Programmable Gate Arrays* (FPGA).

As shown in Figure 1.3, the receiver architecture consists of an A/D converter, followed by the FPGA chip and one or two D/A converters. The AM demodulation algorithm for both MW and SW will be programmed into the FPGA and the resulting baseband signal is directed to the output speakers through the D/A converters.

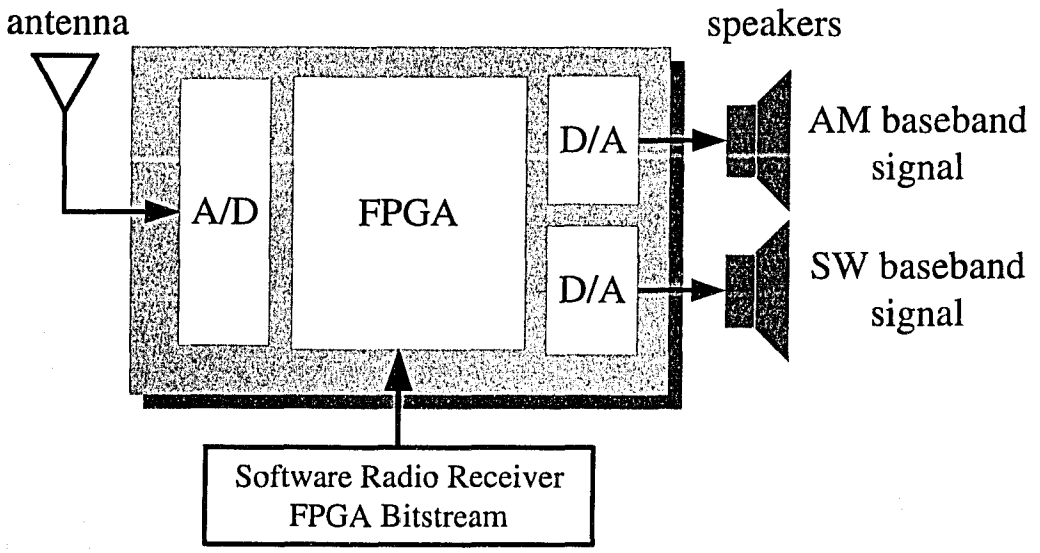

Figure 1.3: Proposed receiver architecture.

1.6 Thesis Outline

This thesis is organized as follows. In Chapter 2, we present a survey of the most recent work done in the area of software defined radio. This survey includes an overview of some receiver architectures proposed in recent literature. We also introduce the enabling technologies for a software defined radio platform followed by a brief description of some SDR simulators and prototypes. Based on this introduction, we show the advantages of a Tier 3 SDR and what are the challenges of building an ideal SDR.

In Chapter 3 we present the AM[1] receiver. The receiver is first simulated using high level languages such as MATLAB. The model is then translated into hardware

[1]The acronym AM is used to describe both the Amplitude Modulation scheme and the frequency

description language and synthesized for the target FPGA platform. Finally, we present the receiver testing and performance measurements for the AM receiver.

A similar approach is followed in Chapter 4 for the Short Wave receiver. In this chapter, we present the main differences between the implementation of the two receivers and show the testing and performance measurements of the Short Wave receiver.

The main advantage of SDR is the ability to change the modulation scheme without changing the hardware. Therefore, in Chapter 5 we explore this advantage by having both AM and Short Wave receiver in the same FPGA. We also describe some of the peripherals that are needed for the proper operation of the two receivers and how the user can control the two receivers through a command-line driven interface.

Finally, we conclude this work in Chapter 6 along with the proposed future works in the SDR area.

---

range where AM broadcast signals are transmitted. Depending on the context, the meaning of the word should not cause confusion. However, sometimes the acronym *Medium Wave* (MW) is used instead of AM to differentiate between the two possible uses of the word.

NO TEXT

Chapter 2

Background

Since Joseph Mitola [1] coined the term *software radio* in 1991, much work has been done in this area. In this section we give a brief survey of some of the work that has been presented in the recent literature.

2.1 Overview Papers

There are many introductory papers on the topic of SDR. Tuttlebee [2] gives a very through introduction of the concept of SDR. He introduces the benefits of such a receiver system and identifies the enabling technologies needed for SDR. In [4], Buracchini gives several definitions of SDR, its benefits and also what are the technologies needed in order to implement a software radio.

2.2 Receiver Architecture

Perhaps the subject that has received the most wide attention in SDR is the receiver architecture. In [1], Mitola gives a very detailed discussion on the receiver architecture of a software radio. He identified the different functional blocks in the receiver platform and presented the requirements for each block. He also presented a discussion on how to estimate the processing complexity for a given algorithm or modulation scheme. This information can then be used to determine which operations should be done in hardware and which ones in software.

Yoshida *et al.* [6] presented another possible architecture for SDR. It consists of

a broadband RF frontend and digital baseband processing units. A *direct conversion principle* is suggested as opposed to *heterodyning*. In this architecture, the received signal is downconverted directly to baseband, using a local oscillator. This architecture would fall into the *Tier 2* category because the downconversion is performed in the analog domain.

On the other hand, MacLeod *et al.* in [7] compare the following architectures: 1) *direct digitization with undersampling*, 2) *zero IF* and 3) *superheterodyne*. In a direct digitization with undersampling architecture, the RF signal is first filtered with a very sharp bandpass filter in the analog domain. The signal is then sampled at a frequency that is below the Nyquist frequency (hence the name *undersampling*) and further processed in the digital domain. In the zero IF architecture, the RF signal is downconverted directly to the baseband in the analog domain. The digitization takes place at the baseband frequency range. Finally, in the proposed superheterodyne architecture, the RF signal is first downconverted to a fixed IF frequency in the analog domain and then sampled and processed in the digital domain. The advantages and disadvantages of each architecture are presented in their work and they conclude that the superheterodyne architecture is the most practical architecture compared to the other two alternatives.

Pearson in his paper [8], treats SDR as "*System* Defined Radio". He argues that since the radio interface is implemented in software, the problem can be treated as a complex software system. SDR effectively converts an electromagnetics and electronics problem into a computer engineering problem; thus, one can use the conventional object-oriented approach and use tools such as *Universal Modelling Language* (UML) to implement a software radio.

In [9], Greifendorf *et al.* propose to optimize a SDR receiver as a whole rather than optimizing each of components separately. They conclude that in order to have a fully reconfigurable architecture, the number of analog components in a given receiver must be reduced as much as possible. This is because digital components can be programmable and tend to scale well with technology while analog parts do not scale as well. Several examples using this design paradigm are presented in

their paper.

Zhigang *et al.* propose a multiple standard SDR baseband platform in [10]. In this system, there are multiple antennas and RF frontends to cover the desired frequency range. The downconverted signal is sampled at the IF stage and handled by different software modules. These modules are implemented in an objective-oriented language so they can be inserted or removed from the baseband platform easily. The processing modules themselves are built from a set of common system libraries and common algorithm libraries so most of the code can be re-used.

In [11], Srikantesware *et al.* propose to use *Configurable Computing Machines* (CCM) as the processing engine for software radio handsets. In a CCM device, the processor itself can be reconfigured "on-the-fly" to support different modulation schemes. Several academic and commercial CCM devices are presented in their work, including [12–16].

2.3 Enabling Technologies

Another area that received much attention in the past few years regarding SDR is the set of enabling technologies that make SDR possible. An overview of the technical challenges of SDR is presented in [17]. This paper listed the following components as key elements in SDR: *intelligent antenna, programmable RF modules, high performance A/D/A converters, DSP techniques* and *interconnect technology*. The main challenges with all these enabling technologies are the need to be reconfigurable for different operating frequencies while at the same time operate at a very high frequency or sampling rate.

An interesting idea is the use of *Micro-Electromechanical Systems* (MEMS) for building smart antennas. In this setup, the antenna size is varied through MEMS switches. A frequency change of 3 to 8 GHz has been reported in [18,19]. The same MEMS technology can be applied to programmable RF modules. In [20,21], high performance and flexible RF components have been reported using MEMS-based variable reactive elements and switches.

Perhaps the main challenge in the realization of Tier 3 software radios lies in the

analog-to-digital converter. This is because the sampling rate of *commercially-off-the-shelf* (COTS) converters is still one order of magnitude lower than the carrier frequency range used in commercial receivers. However, development in higher performance converters is still ongoing. For instance, COTS A/D converters can reach sampling rates in the order of 500 MSPS at 12 bits [22] or even 2 GSPS at 10 bits resolution [23]. ADCs based on GaAs achieving 6 bits resolution at 6 GSPS and 8 bit resolution at 3 GSPS have been reported in [24]. Moreover, an actual working superconducting ADC prototype operating at 19.6 GSPS has been reported in [25].

Digital signal processing is the core component of an SDR receiver. Traditionally DSP processing are handled by DSP-processors, ASIC and FPGAs. General purpose processors used in desktop computers are sometimes also used in SDR platforms due to their availability, flexibility and inexpensive nature. According to [26], the following five criteria should be considered in the selection of a DSP engine:

- programmability
- level of integration
- development cycle
- performance
- power

Depending on the goal of the SDR receiver, the above criteria should be balanced in order to arrive at the ideal choice of DSP engine. An in-depth discussion of the trade-offs associated with the choice of DSP techniques can be found in [27].

Finally, *software download mechanism* is also one of the enabling technologies for SDR. Lachlan *et al.* in [28] provides an entire framework for software download for SDR devices that achieves the following:

- control and verification of the integrity of the downloaded software
- prevention of unauthorized software from executing on the SDR terminal

- transmission secrecy to ensure that the intellectual property is not stolen during the download process

The above goals are achieved through tamper-resistant hardware and several cryptographic techniques. Both the manufacturers of hardware and software companies need to cooperate with the government regulating bodies in order to implement the proposed security framework.

2.4 Simulators

Since software defined radio is concerned with the software implementation of radio interfaces, a major task in the design of such a system involves software simulations. Several simulators for SDR have been proposed in the literature. Among these, an object-oriented approach to model each hardware components that are used in a SDR environment is presented in [29]. In [30], another SDR simulator called *SDRsim* is presented. This simulator uses inexpensive PCs as the simulation platform. As a teaching aid, a communication design lab based on PC sound cards and MATLAB is presented in [31]. The sound card is used as the input and output device and MATLAB is used to process the stream of data. Finally, an open source implementation of software defined radio libraries called GNURadio can be obtained from [32]. It contains several examples for the implementation of software radio for platforms that support the GNU development tools such as Linux.

2.5 Prototypes

Due to the limited resolution and speed of commercial analog-to-digital converters (ADC), actual SDR prototypes are often limited to a Tier 2 software radio. A brief summary of some available prototypes can be found in Table 2.1.

Table 2.1: SDR Prototypes presented in recent literature.

Name/ Organization	Architecture	Frequency Range	Modulation Schemes
ARIB [33, 34]	Uses array of antennas and banks of analog processing elements. Signal is digitized at the baseband. Provides features such as radio monitoring, direction of arrival estimation, radio wave characteristic measurement, etc.	27 MHz, 900 MHz, 2 GHz	BPSK, QPSK, π/4-QPSK, GMSK, FM, AM
CRL [35]	Basic building blocks are implemented in DSP and FPGA devices. The specification of these blocks can be changed by external parameters in order to reduce the amount of information that needs to be downloaded into the device.	1.5 GHz, 1.9 GHz, 5.8 GHz	PHS, GPS, ETC, GMSK, π/4-QPSK, BPSK, QPSK
SOPRANO [36]	Composed of reconfigurable digital circuits, ADC, DACs and a low-power wideband analog component called *Monolithic Microwave Integrated Circuit* (MMIC). It uses direct conversion techniques to shift the signal to baseband for sampling and further processing in the digital domain. It is capable of handling wireless standards such as wireless LANs and cellular phone systems.	500 MHz to 9 GHz	BPSK, QPSK, 8-PSK, 16-QAM, 64-QAM
NTT [37]	SDR prototype for cellular base stations. It has three antennas and the incoming signal is band pass filtered and undersampled at the IF stage. It has four DSP engines capable of processing 1600 MIPS each. Three DSP engines are used for receiving and one for transmitting.	2.45 GHz	BPSK, QPSK, π/4-QPSK, GMSK, 16-QAM
Toshiba [38]	SDR designed for handheld applications. It uses the direct conversion and low IF principle to downconvert the RF signal for digital processing.	1.9 GHz	π/4-QPSK, GMSK

Continued on next page

Table 2.1 – continued from previous page

Name/ Organization	Architecture	Frequency Range	Modulation Schemes
Toyo [6]	SDR prototype for radio base stations. It has four simultaneous TX/RX channels. The TX/RX channels have a bandwidth of 1.25MHz and 650 kHz respectively, with a sampling resolution of 12 bits at 40 MSPS.	370-380 MHz	$\pi/4$-QPSK, FM, BPSK, FSK
Sandbridge Technologies [39]	An analog RF frontend is used to downconvert the signal down to IF. The IF signal is sampled and processed by a multi-threaded general purpose processor. A custom made compiler optimizes the C program in order to use SIMD instructions.	850 MHz, 900 MHz, 1.8 GHz, 1.9 GHz, 2.4 GHz	WCDMA, GSM/GPRS, 802.11b, Bluetooth

As we can see from the table, there are many SDR prototypes presented in the literature. All of them try to sample the incoming signal at either the IF or baseband stages using different receiver architectures. While this approach reduces the requirement on the A/D converters and DSP processing engines, the RF stage itself is not truly reconfigurable. For example, a change in the modulation scheme or IF value would require modifications in the analog parts of the receiver.

2.6 Summary

From the above discussion, it is clear that the main challenges in the implementation of a Tier 3 SDR are imposed by the availability of fast A/D converter, followed by the requirement of powerful DSP processing engines. This is why so far all the SDR prototypes available in the literature implement a Tier 2 SDR. As we see more advances in ADC technologies, the next logical step is to attempt to implement a Tier 3 ideal SDR. This is precisely what we are doing in the present work. In the following chapters we introduce Tier 3 SDR prototypes for AM and Short Wave radio receivers.

NO TEXT

Chapter 3

AM Receiver

A Tier 3 SDR implementing an AM receiver should perform all the receiver functions in the digital domain. In this chapter we design and implement this AM receiver. We first describe the target hardware platform in Section 3.1. In Section 3.2 we briefly introduce the building blocks of a heterodyne receiver. In Section 3.3 we present how the receiver is modelled in MATLAB through high level simulation. The hardware implementation of this receiver is described in Section 3.4. Finally, the test and performance measurement of the receiver are provided in Sections 3.6 and 3.7 respectively.

3.1 Target Hardware Platform

The target hardware platform is the *GVA 290* FPGA board from GVAssociates [40]. It contains four A/D converters, two Xilinx Virtex-E 2000 FPGAs and four D/A converters. The whole system is capable of operating at a maximum frequency of 100 MHz and the converters have a resolution of 12 bits. Although the maximum clock frequency is 100 MHz, the default clock is set to 50 MHz. Since we are only interested in a "proof of concept" prototype, it is decided that the default system clock speed will be used as our target frequency.

3.2 Heterodyne Receiver

The goal of the AM receiver is to demodulate incoming RF signals down to baseband. Standard AM broadcast signals in North America have a bandwidth of 10 kHz (for a baseband of 5 kHz) and a carrier frequency that ranges from 540 kHz to 1600 kHz [41]. This is shown in Figure 3.1. Notice that the carrier for each station is present in the spectrum and the two sideband components are symmetric.

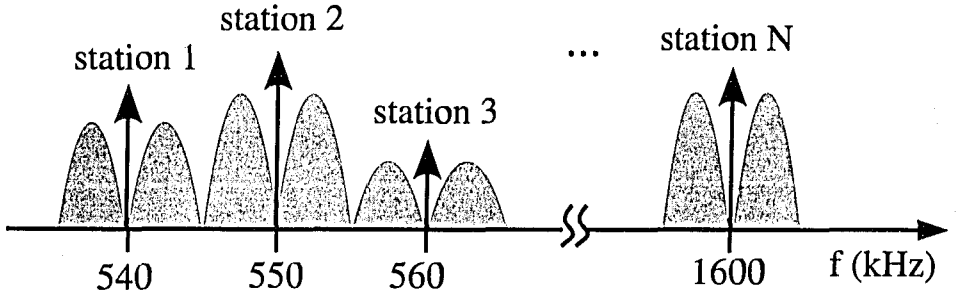

Figure 3.1: AM broadcast band showing different radio stations.

The most common receiver architecture by far is the *heterodyne* receiver. In this type of receiver, the carrier frequency is first *downconverted* to a fixed *intermediate frequency* (IF) for further processing. A generic heterodyne receiver is depicted in Figure 3.2.

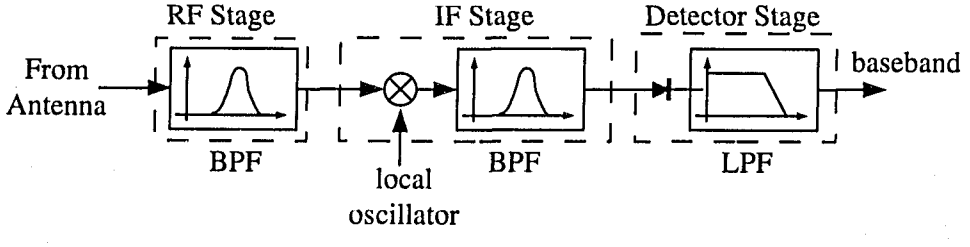

Figure 3.2: Generic heterodyne receiver architecture.

There are three main blocks in any heterodyne receiver, namely, *RF stage*, *IF stage* and *detector stage*.

3.2.1 RF Stage

The RF stage consists of a bandpass filter (BPF) centered at the carrier frequency and it is in charge of rejecting the image stations prior to the downconversion process. The bandwidth of this BPF, BW, is limited by the following inequality:

$$BW < 2 \times IF \tag{3.1}$$

where IF is the intermediate frequency. This inequality must be satisfied in order to prevent image channels appearing in the downconverted signal [41].

3.2.2 IF Stage

The IF stage is composed of a *mixer* and a sharp bandpass filter centered at IF frequency that rejects any adjacent radio stations.

The value of the local oscillator is determined by the following equation [42]:

$$F_{local} = F_c + IF \tag{3.2}$$

where F_{local} is the local oscillator frequency, F_c is the carrier frequency and IF is the intermediate frequency.

The bandwidth of the IF BPF is fixed to 10 kHz since this is the bandwidth of radio station that we wish to demodulate.

3.2.3 Detector Stage

Finally, the detector extracts the envelope in the input IF signal by first taking the absolute value (accomplished by the diode) and then followed by a low pass filter (LPF). The stop band of this LPF should be set to 5 kHz because this is the baseband bandwidth of the AM signals.

3.2.4 Digital Implementation

The proposed heterodyne receiver shown in Figure 3.2 is implemented in the digital domain. In order to do this, we should introduce some extra blocks that change

the sample rate of the incoming data. This is because the length of a given filter is proportional to the sample rate. Figure 3.3 shows the proposed receiver architecture.

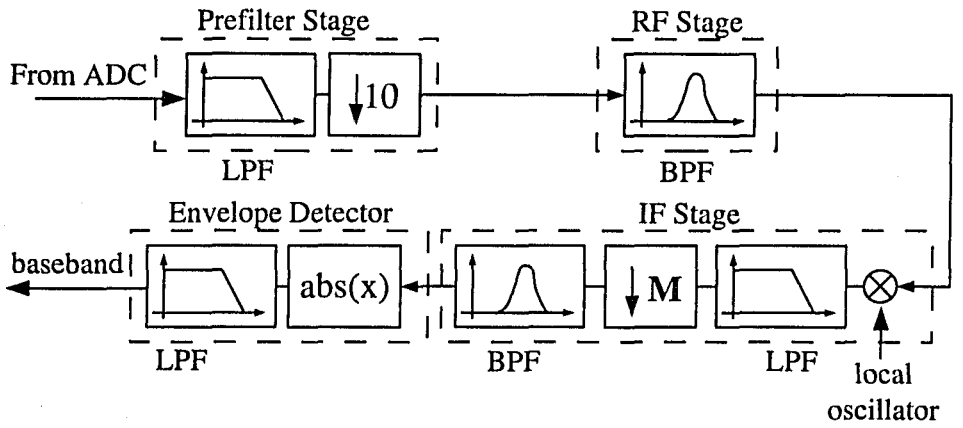

Figure 3.3: Modified heterodyne receiver suited for digital implementation.

The extra blocks are the *prefilter stage* and the decimators. In the prefilter stage, we low pass filter the incoming signal and then decimate by 10. This stage effectively reduces the sampling rate from 50 MSPS down to 5 MSPS which is still high enough for AM broadcast signals. A similar operation is performed in the IF stage to further reduce the sample rate. Notice that the diode operation shown in Figure 3.2 is achieved by taking the absolute value of the data sample.

3.3 Receiver Simulation

The AM receiver is first modelled in MATLAB in order to determine the ideal value of the intermediate frequency, the decimation factor at the IF stage and the following parameters for each filter:

- coefficient width,
- data width,
- filter length,
- filter coefficients

3.3.1 Selection of IF

The selection of the intermediate frequency affects the overall filter length. To see why this is true, consider the following relationships:

- The RF BPF length is mainly determined by the sampling rate and the transition bandwidth. While the sampling rate is fixed in our case (5 MSPS), the transition bandwidth depends on the intermediate frequency because Eq. (3.1) must be satisfied. This means that *increasing* the IF frequency will *decrease* the RF bandpass filter length.
- The length of the BPF at the IF stage is mainly determined by the incoming data rate. If we can lower the incoming data rate through decimation, we can use a shorter filter and accomplish the same results. However, the decimation factor is limited by the intermediate frequency. *Increasing* the IF frequency will decrease the allowed value for the decimation factor and hence *increase* the IF bandpass filter length.

It is clear from the above relationships that we have a tradeoff between the length of the IF BPF and the length of the RF BPF for different values of IF. In order to find the optimal value of IF such that the overall number of taps in the whole receiver is reduced, a simulation with different value of intermediate frequency is performed. The result is shown in Figure 3.4.

As we can observe from Figure 3.4, while RF BPF length decreases with higher IF values, the IF BPF increases with higher IF values. The total number of filter taps is drawn with thick lines in Figure 3.4 and it shows that the overall hardware resources are minimized for an IF value of *approximately* 50 kHz.

The decimation factor at the IF stage also changes with different values of IF. This effect is shown in Figure 3.5. For an IF value of 50 kHz, this corresponds to a decimation factor of 22.

It is worth mentioning that the chosen value of $IF = 50$ kHz minimizes the total number of taps required for the FIR filters in the receiver *regardless* of the filter implementation method. However, since the input sample rate of each filter

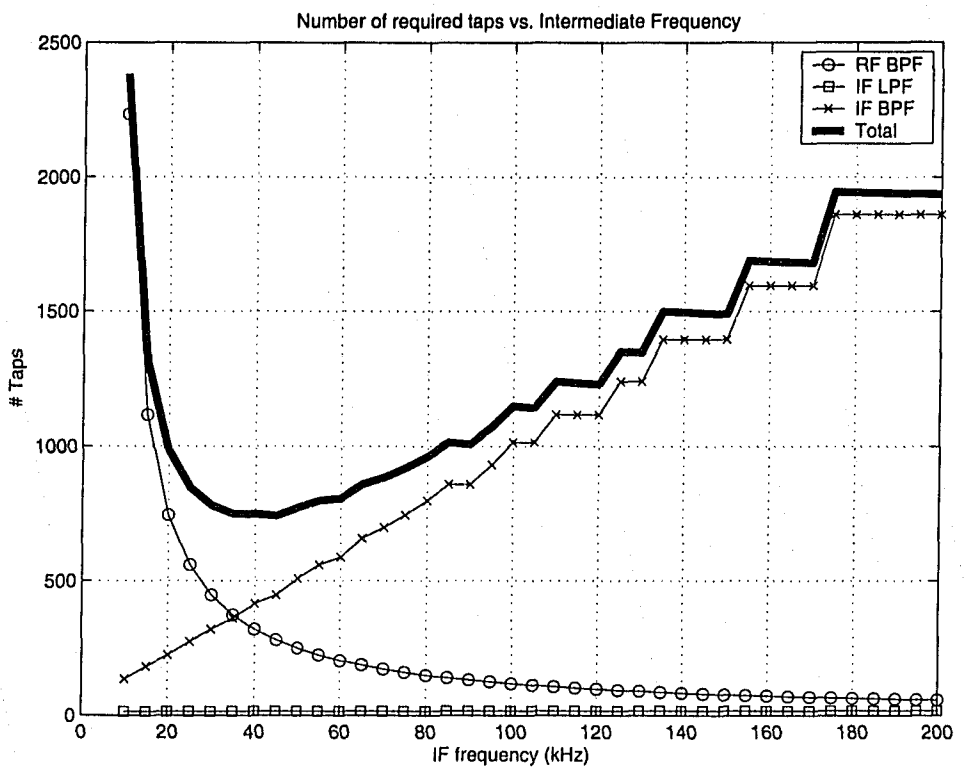

Figure 3.4: Filters length vs. intermediate frequency value.

may vary drastically due to decimation, a short filter with a high sampling rate may be doing the same amount of computations per second as a longer filter with a low sampling rate. This means that, for the chosen value of IF, the resulting total number of *operations per second* is not necessarily minimum. To illustrate this property, let's define the metric *operations per second*, *OPS* as:

$$OPS = \text{filter length} \times \text{filter sampling rate} \tag{3.3}$$

If we compute *OPS* for each filter and plot it against different values of IF we obtain the graph shown in Figure 3.6.

The thick line on Figure 3.6 shows the total number of operations per second for different values of IF. As we can see, *OPS* is minimized for an IF value of around 90 kHz.

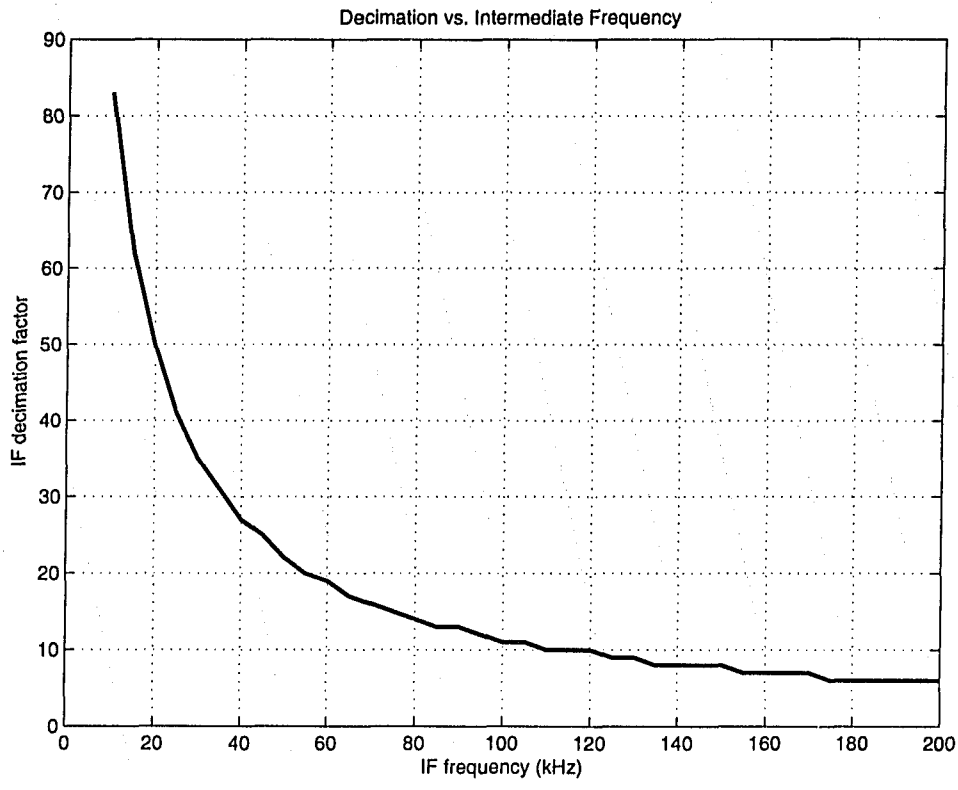

Figure 3.5: IF decimation factor vs. intermediate frequency value.

So which value of IF is optimal? The answer to the question depends on the metric that we wish to minimize. In our case, we choose to minimize the hardware resource utilization and hence, we choose an IF value of 50 kHz. On the other hand, if we were to implement this receiver on a general purpose microprocessor where there is only one time-shared ALU performing all the arithmetic operations, then an IF value of 90 kHz would be more desirable.

3.3.2 Filter Parameters

In order to determine the parameters for each filter, we use MATLAB to design and model the receiver. The digital filters are designed with the `kaiserord` and `firl` commands in MATLAB. The `kaiserord` command estimates the order of the filter that we wish to implement while the `firl` command computes the filter coefficients. The following example shows how to create a low pass filter in

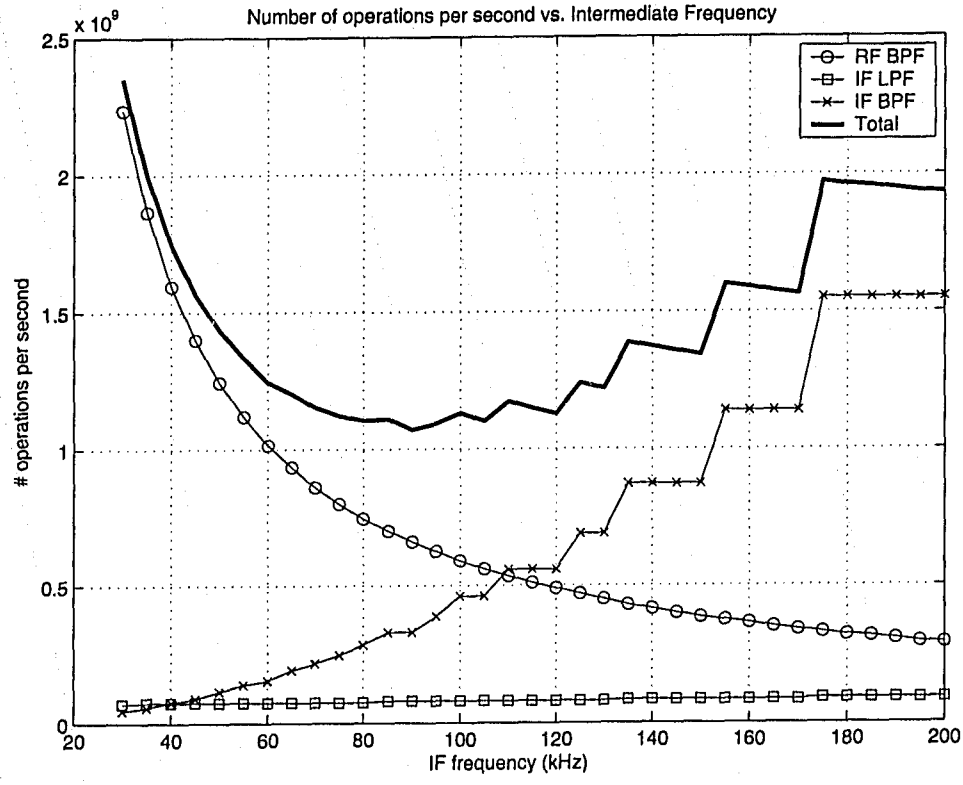

Figure 3.6: Number of operations per second vs. intermediate frequency.

MATLAB.

EXAMPLE: Low Pass Filter design in MATLAB
We want to build a low pass filter with a passband from 0 to 4500 Hz and a stop band starting at 5 kHz. The maximum band ripple is 0.01 and the sample rate is 200 kSPS.

```
>> [N,Wn,beta,typ]=kaiserord([4.5e3 5e3],
                          [1 0],[0.01 0.01],200e3);
>> B=fir1(N,Wn,typ,kaiser(N+1,beta),'noscale');
```

In the above command, the vector B will contain the FIR coefficients for the low pass filter.

The same procedure can be used to create bandpass filters. Detailed description of this procedure can be found in MATLAB help pages. Note that we have chosen FIR filters (as opposed to IIR filters) due to their stability and linear phase properties.

3.3.3 Coefficient Quantization

The next implementation related issue we need to consider is the use of fixed point arithmetic. For an N-tap FIR filter, the difference equation can be expressed as:

$$y[n] = b_0x[n] + b_1x[n-1] + ... + b_{N-1}x[n-M+1] \quad (3.4)$$

$$= \sum_{i=0}^{N-1} b_ix[n-i] \quad (3.5)$$

where $x[n]$ and $y[n]$ are the input and output data samples at time n respectively. The coefficients in an FIR filter is represented by b_i. Since in practical implementations b_i's have finite word length, we need to scale them in order to minimize the coefficient quantization error. It can be shown in [43] that in order to minimize the coefficient quantization error, we need to find b_i', the scaled version of b_i such that:

$$b_i' = \frac{B_i}{2^w} \quad (3.6)$$

where

$$B_i = round(b_i * 2^w) \tag{3.7}$$

and w is the number of binary digits used to represent the fractional part of b_i.

Clearly, due to the irreversible round operation, we lose some precision in this step. The parameter w is chosen such that the biggest coefficient b_i in the FIR filter can still be represented without truncation. The maximum value of w such that this condition is satisfied can be found as [43]:

$$w = \lfloor log_2 \left(\frac{(2^{M-1} - 1)}{max(|b_i|)} \right) \rfloor \tag{3.8}$$

where M is the total number of bits available to represent one filter coefficient. Eq. (3.8) tells us how to represent a floating point coefficient in fixed point representation. In the AM receiver, we use 8 bits to represent each filter coefficient.

3.3.4 Data Word Length

We also have to determine the data word length in our filter. The upper limit is imposed by the input A/D converter, which is 12 bits in our case. However, if we could use fewer bits and still be able to demodulate the radio channel, then we could save on hardware resources. Since the majority of applications use 8 bits to represent an audio signal (e.g. PCM), we assume that 8 bits per sample is our lower bound. The exact word length will depend on the subjective audio quality that we want. This parameter has been set to 8 bits and it can be increased later if needed.

3.3.5 Summary

Table 3.1 shows the parameters obtained through simulation for each filter that appears in Figure 3.3.

Figures 3.7 to 3.11 show the magnitude of the frequency response of the five filters listed in Table 3.1. The filters are set to demodulate a carrier frequency of 600 kHz. All the filters have an input datawidth of 8 bits. Both the output sample and the filter coefficients are quantized to 8 bits.

Table 3.1: Filter parameters for the AM receiver.

Filter	Passband	Sample Rate	# Taps
Prefilter LPF	0 to 1.7 MHz	50 MSPS	19
RF BPF	$F_c \pm 50$ kHz	5 MSPS	140
IF LPF	0 to 52.5 kHz	5 MSPS	9
IF BPF	50 kHz ± 2.5 kHz	227 kSPS	144
Detector LPF	0 to 5 kHz	227 kSPS	250

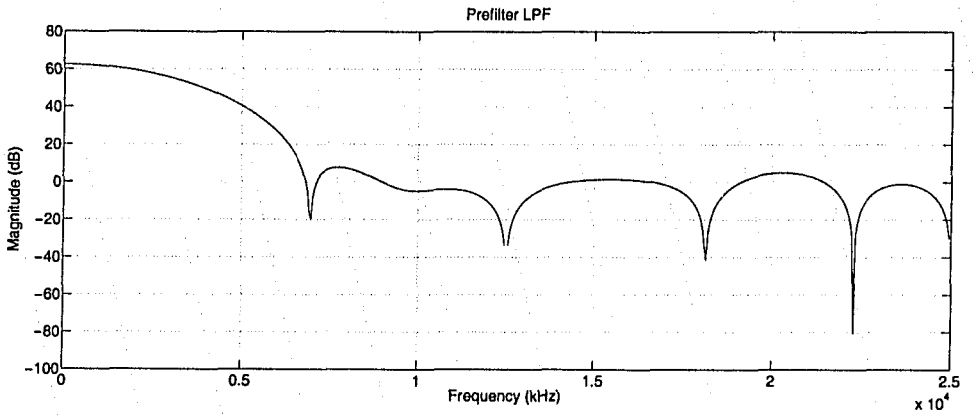

Figure 3.7: Magnitude of the frequency response of prefilter LPF.

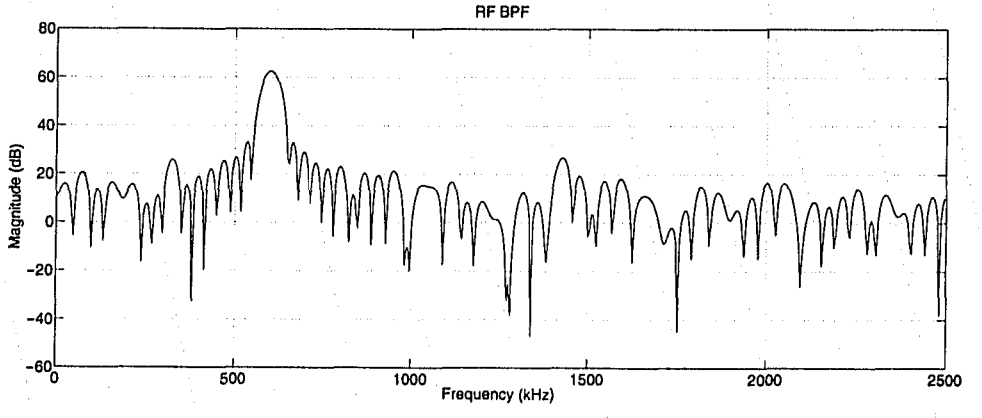

Figure 3.8: Magnitude of the frequency response of RF BPF.

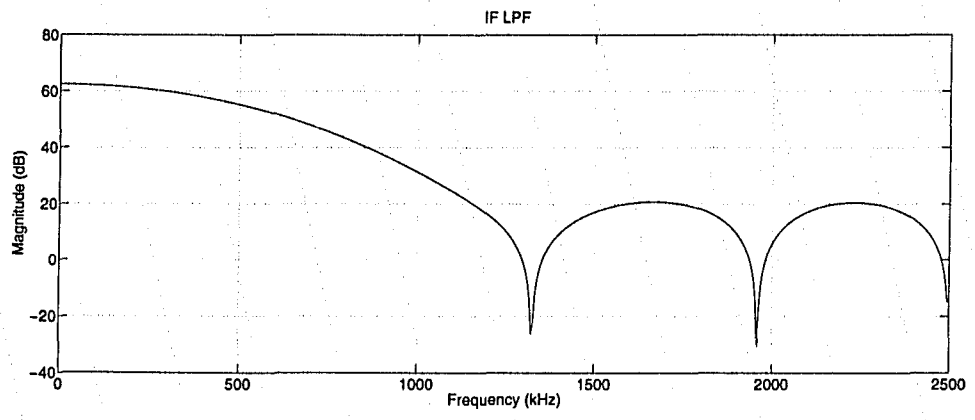

Figure 3.9: Magnitude of the frequency response of IF LPF.

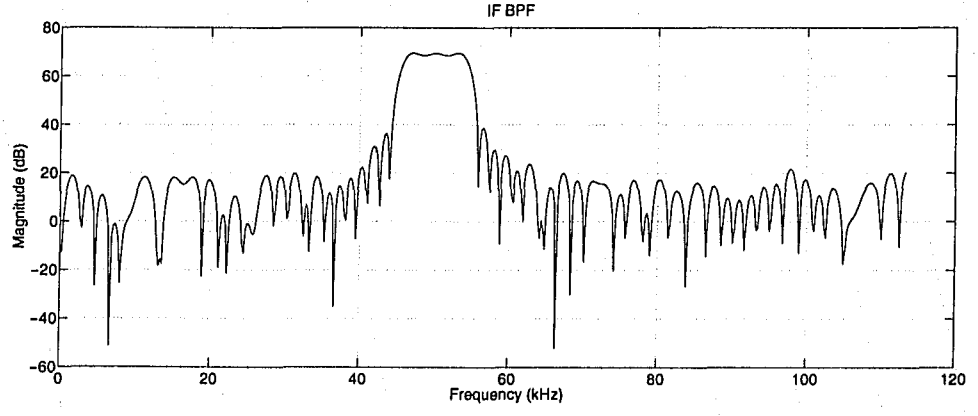

Figure 3.10: Magnitude of the frequency response of IF BPF.

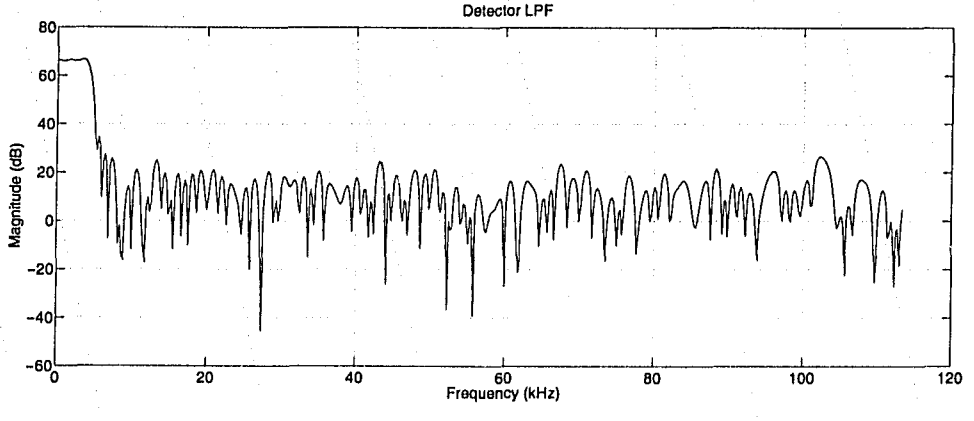

Figure 3.11: Magnitude of the frequency response of detector LPF.

3.4 Hardware Implementation

In this section we present the hardware implementation of the Tier 3 software AM radio receiver described in Section 3.3. The receiver architecture showing the antenna and data converters is shown in Figure 3.12.

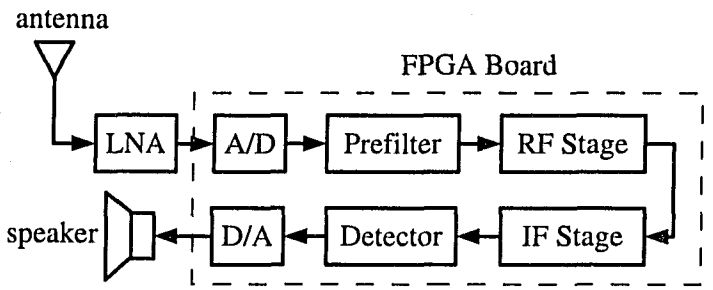

Figure 3.12: AM receiver architecture.

The received signal from the antenna is first amplified using a low noise amplifier (LNA). This amplifier provides the necessary 1 Vpp signal level to the 50 Ω input of the ADC. The amplifier is designed so that the output signal is band limited to 10 MHz, effectively acting as an anti-aliasing filter to the ADC.

Once the signal is converted into the digital domain, the signal is passed through four stages, namely, *prefilter, RF stage, IF stage* and *detector stage*. All these four blocks are implemented inside an FPGA device.

3.4.1 Prefilter Stage

Since the incoming signal is highly oversampled, the function of the prefilter stage is to decimate the incoming signal by a factor of 10. This is shown in Figure 3.13. The cut-off frequency of the LPF is 1.7 MHz because this is the highest frequency of interest in the AM band. This filter is implemented as a *Distributed Arithmetic* (DA) FIR filter [44] and a readily available IP Core from Xilinx was used for this purpose. Since the samples are arriving at the system clock rate, a fully parallel implementation of this filter was chose because the LPF has to produce one output for each system clock cycle.

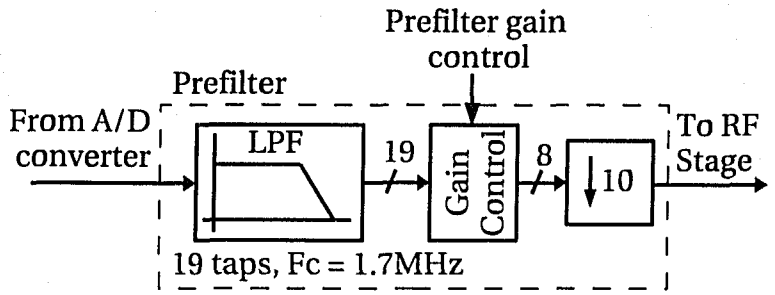

Figure 3.13: Block diagram of the prefilter stage.

The output of the LPF has a width of 19 bits and this value is determined automatically by the IP Core in order to avoid overflow or loss of precision. In order to reduce the number of bits back to 8 bits, we use a *gain control block* to select the number of LSBs that we wish to drop. This gain control block, along with its control interface, will be described in detail in Section 3.4.5. The decimator is simply a counter that counts from 0 to 9 and enables the output register whenever the counter reaches 9.

At this stage, the sampling rate has been reduced from 50 MSPS to 5 MSPS. This gives subsequent stages a total of 10 clock cycles to operate on each data, which allows for serial implementations of FIR filters.

3.4.2 RF Stage

The BPF in the RF stage is shown in Figure 3.14.

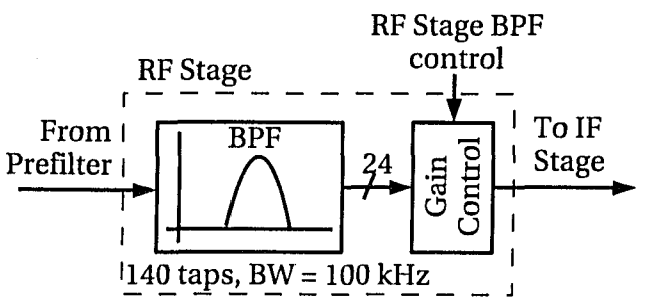

Figure 3.14: Block diagram of the RF stage.

Since the IF value is 50 kHz, this BPF has a bandwidth of 100 kHz. This is achieved with a FIR filter of length 140. Again, in order to reduce the resulting data

width, another gain control block with an input width of 24 bits is used here.

3.4.3 IF Stage

The IF stage is composed of a mixer, a low pass filter, a decimator and a bandpass filter. For each filter, there is a gain control block in order to reduce the datawidth back to 8 bits. The block diagram of the IF stage is shown in Figure 3.15.

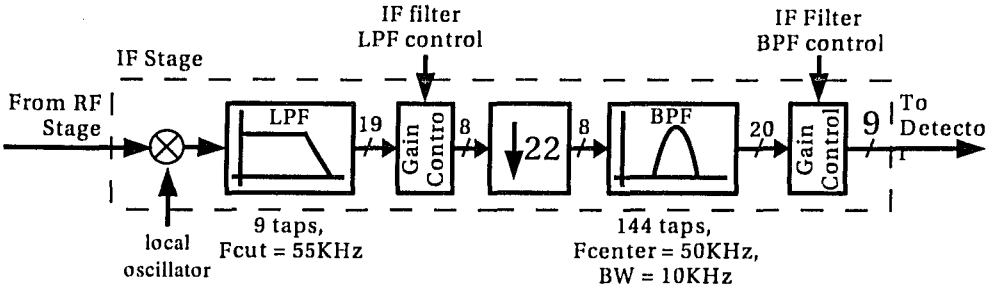

Figure 3.15: Block diagram of the IF stage.

The output of the mixer is simply the product of the input signal with the local oscillator. In order to implement the local oscillator, a direct digital synthesizer (DDS) [45] is used. The multiplication operation, together with the local oscillator forms the *digital down converter* (DDC) and a readily available IP Core from Xilinx is used [46].

The width of the BPF is 10 kHz because that is the bandwidth allocated for AM broadcast transmissions. A 144-tap BPF is used for this purpose. A gain control block is attached to the output of each filter as in previous stages. Notice, however, that the last gain control block has an output width of 9 bits instead of 8 bits. The reason for will become clear in the next section.

3.4.4 Detector Stage

The detector stage is in charge of recovering the baseband signal that forms the envelope of the IF signal. The block diagram is shown in Figure 3.16.

The first block in the detector stage is the absolute value block. As the name implies, it takes the absolute value of the input data. Note that datapath width

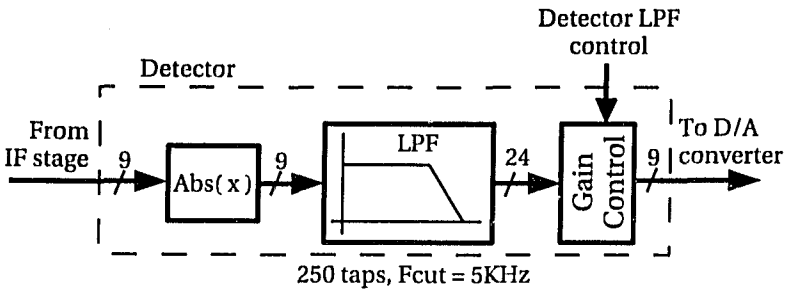

Figure 3.16: Block diagram of the detector stage.

has been increased from 8 bits to 9 bits[1]. This is because the MSB at the output of the absolute value module is always 0 for positive numbers in 2's complement representation. In order to keep an effective resolution of 8 bits, we need a datapath of 9 bits since the MSB is always 0.

The magnitude of the signal is then passed to a 250-tap LPF with a cutoff frequency of 4.5 kHz. This filter effectively recovers the baseband signal but with a DC component.The DC component in the recovered signal is not treated here because the on-board DAC is in single ended configuration and the output can swing from 0 to V_{DD} only.

3.4.5 Gain Control Block

For each of the FIR filters used in this receiver, there is a gain controller block placed immediately at the output of the filter. The function of this gain control block is to reduce the datapath width back to 8 bits (or 9 bits in the detector LPF stage) in order to be compatible with the other processing blocks in the system. The input and output signals for the gain controller block used in the prefilter stage is shown in Figure 3.17.

The input port `din` has a width of 19 bits which corresponds to the output datawidth of the LPF at the prefilter stage (see Figure 3.13). The output has a width of 8 bits as expected. The signal `nd` (new data) indicates the arrival of a new sample. This signal is then registered and made available at the output `rdy` after

[1]It is possible to maintain a datawidth of 8 bits if we process the numbers as *unsigned*. However, the LPF and the gain control block would need to be modified in order to handle this special case.

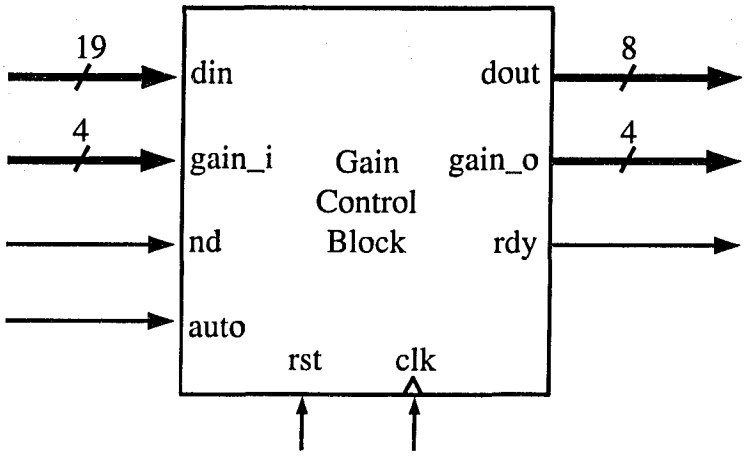

Figure 3.17: Block diagram of the gain controller.

one clock cycle. Depending on the `auto` signal, the gain control block behaves in two different modes: *automatic mode* or *manual mode*.

3.4.5.1 Automatic Mode

If `auto` is asserted, the gain control block will determine automatically the number of bits to truncate in order to avoid overflow. This is accomplished by observing a window of data samples and determine the maximum value during this window of samples. The length of this window is set as a parameter during synthesis time and it cannot be changed afterward.

For example, if the 19-bit wide input data stream has a maximum value of 0x03F0F ($0011\_1111\_0000\_1111_b$) during the observation interval, then the gain control block will drop the seven least-significant bits and allow bits {18, 13, 12, 11, 10, 9, 8, 7} to pass to the output (assuming bit 18 is the MSB and bit 0 is the LSB). This is to guarantee that all 8 bits in the output are used without causing an overflow. To reflect the fact that seven LSB were dropped from the input, the output port `gain_o` will have a value of 7 (ie. 0111_b). In the automatic mode, the input port `gain_i` is ignored.

3.4.5.2 Manual Mode

As the name implies, when the signal `auto` is low, the user can control the gain of this block manually. Whenever a new sample is present (`nd` is high), the number of LSB dropped will be determined by the signal `gain_i`.

Following the same example, if the input signal is 0x03F0F (ie. $0011\_1111\_0000\_1111_b$) and the input `gain_i` has a value of 9, then the 9 LSB bits are ignored. This means that the output will have a value of 0x1F (0001_1111_b).

Since there are many gain control blocks in the whole receiver and all of them have different input datawidths, a Perl script has been written to generate these gain control blocks. The script takes as argument the input datawidth and the output datawidth. It then generates synthesizable Verilog code ready to be included in the receiver.

3.5 Control Logic

There are two units that need to be updated whenever the user selects a new radio station: the RF BPF needs to change its center frequency to match the newly selected carrier frequency and the local oscillator frequency needs to be updated to satisfy Eq. (3.2). This is accomplished with the control units shown in Figure 3.18.

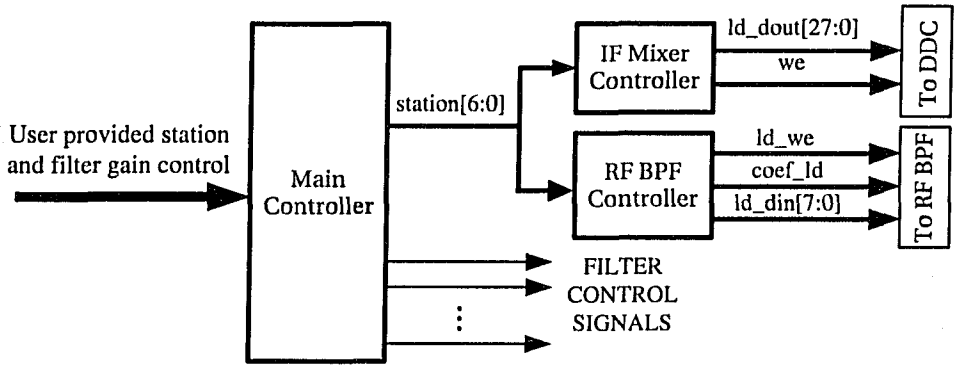

Figure 3.18: Block diagram of the control path for the AM receiver.

The user provides the desired station and the gain control signals to the *main controller* unit by writing to its *write-only* registers. This user interface is described

later in Chapter 5. This unit then passes the new station information to the *IF Mixer controller* and the *RF BPF controller* units.

The IF Mixer controller is basically a ROM that contains the phase increment value for each specific radio station. The input signal `station` serves as the address to the ROM. The output phase increment value, denoted by the signal `ld_dout`, is written to the appropriate register in the local oscillator unit by asserting `we`.

In order to change the carrier frequency of the RF BPF, we need to load a new set of coefficients for each radio station. The signal `coef_ld` is asserted for one clock cycle to signal the start of a coefficient reload operation. Each coefficient is then loaded in parallel to the BPF through the `ld_din` signal. The signal `ld_we` serves as the write enable signal for each coefficient. Since the BPF is symmetric, only half of the coefficients need to be stored in the DDC controller and loaded into the filter.

Notice that the main controller also provides control signals to each one of the gain control blocks used in the receiver. This interface will be described later in Chapter 5.

3.6 Resource Utilization

The AM receiver described in Chapter 3 was synthesized and implemented using Xilinx ISE 6.2.03i tools. Table 3.2 shows the resource utilization. As we can see, only 6% of the 4-input LUTs are needed when the AM receiver is implemented on a Virtex-E 2000 FPGA. This design has an equivalent gate count of 687,744 gates.

In terms of power consumption, the development board draws approximately 1.5 Amps of current at a supply voltage of 5 Volts. This translates into a power consumption of 7.5 Watts.

Table 3.2: Resource utilization for the AM receiver on a Virtex-E 2000 device.

	Used	Total	Percentage
Slice FF	3,960	38,400	10 %
4 input LUT	2,678	38,400	6 %
Occupied Slices	2,639	19,200	13 %
Bonded IOBs	60	404	14 %
Block RAMs	32	160	20 %
Equivalent gate count	687,744		

3.7 Testing and Performance Measurement

Two methods were used to test the receiver's performance. The first method injects a single amplitude modulated sine wave directly into the ADC and the output is observed through the oscilloscope and spectrum analyzer as shown in Figure 3.19.

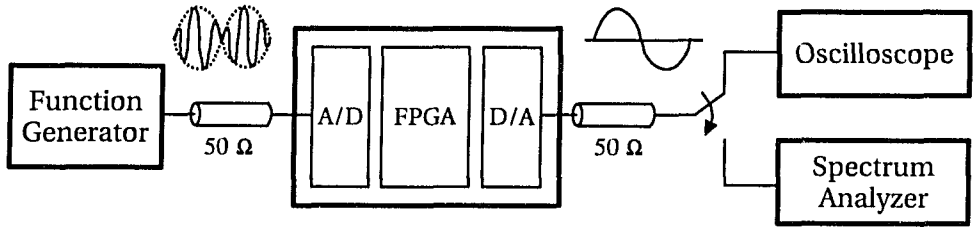

Figure 3.19: Receiver test environment with signal generator, oscilloscope and spectrum analyzer.

The function generator produces an AM signal with a carrier frequency of 600 kHz, AM modulation index of 1.0 and a sinusoidal baseband signal of 2500 Hz. The spectrum of this input signal is shown in Figure 3.20.

From Figure 3.20 we can see that the input signal has a *signal-to-noise ratio* (SNR) of 65 dB for the carrier and 60 dB for the baseband signal.

When the receiver is tuned to the carrier frequency, we obtained a 2500 Hz sine wave from the oscilloscope as expected. The spectrum of the demodulated signal is shown in Figure 3.21. From the graph, we can see that the SNR in this case is approximately 50 dB for a datapath width of 8 bits. This SNR is relatively high considering that the quantization noise for a 8-bit system sampling at Nyquist rate

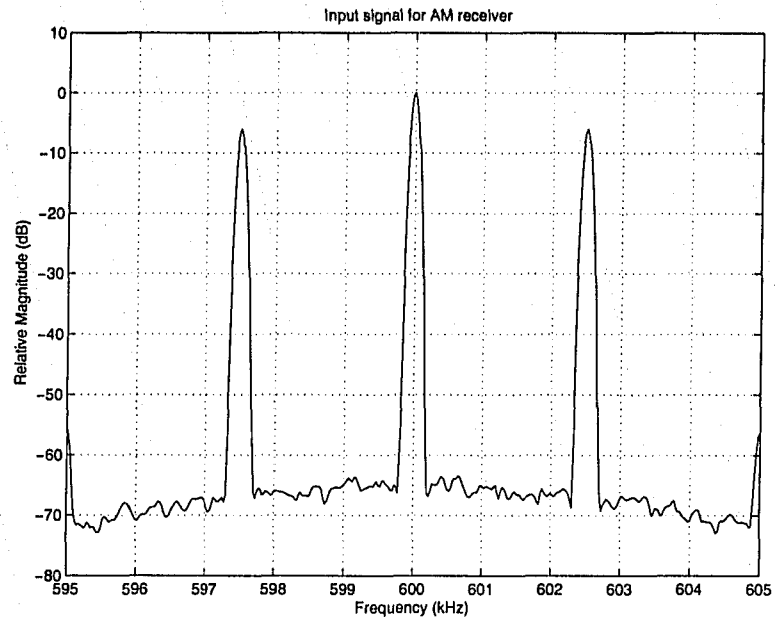

Figure 3.20: Spectrum of the input signal for the AM receiver. The carrier frequency is 600 kHz and the baseband signal is 2500 Hz.

is given by:

$$SNR_{max} = 6.02 \times 8 \text{ bits} + 1.76 = 49.92 \text{ dB} \tag{3.9}$$

However, since the sampling rate in the AM receiver is much higher than the Nyquist rate, according to [47], the maximum attainable SNR should be:

$$SNR_{max} = 6.02N + 1.76 + 10\log(OSR) \tag{3.10}$$

where N is the number of bits in the system and the *oversampling ratio*, OSR, is defined as:

$$OSR \equiv \frac{F_s}{F_{Nyquist}} \tag{3.11}$$

In our particular case, the sampling rate $F_s = 50$ MSPS and the Nyquist rate is $1.6 \times 2 = 3.2$ MHz, giving an oversampling ratio of 15.63. This effectively adds 11 dB to the theoretical maximum SNR for this kind of receiver due to oversampling.

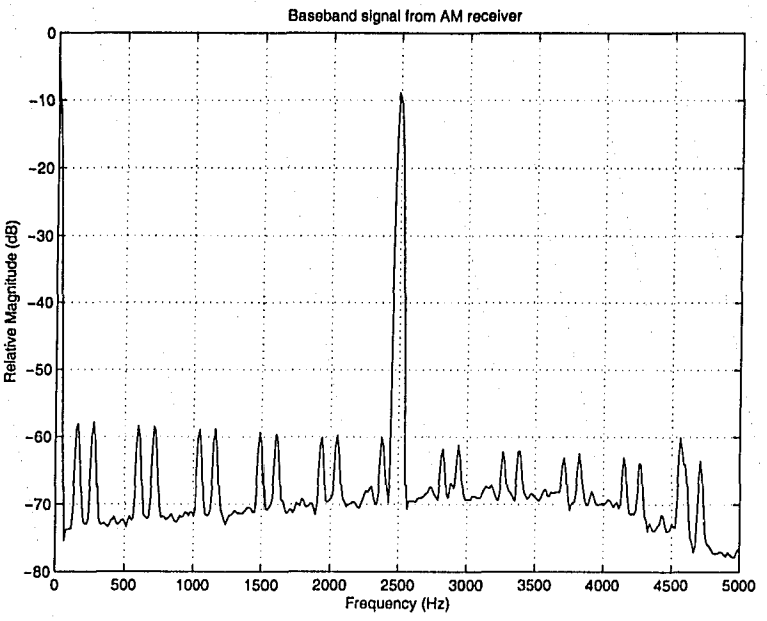

Figure 3.21: Spectrum of the demodulated signal when the AM receiver is tuned to 600 kHz.

When the receiver is tuned to the an adjacent channel (e.g. 590 kHz), the spectrum is shown in Figure 3.22. In this case, we see that the peak located at 2500 Hz has been reduced from -8 dB to -80 dB, giving an total attenuation of 72 dB.

The second method involves a real AM broadcast signal. A simple loop antenna (discussed in Section 5.1.1) in series with the LNA is connected at the input of the ADC. The receiver is tuned to a known local AM radio station and the gain controller units are adjusted to the proper levels for good audio quality. The receiver successfully tuned into the desired radio channel and reproduced the baseband audio signal. The audio quality was good and we could understand what the reporter was saying. Sometimes there was a high pitch ringing noise in the background but in general the audio quality was considered acceptable.

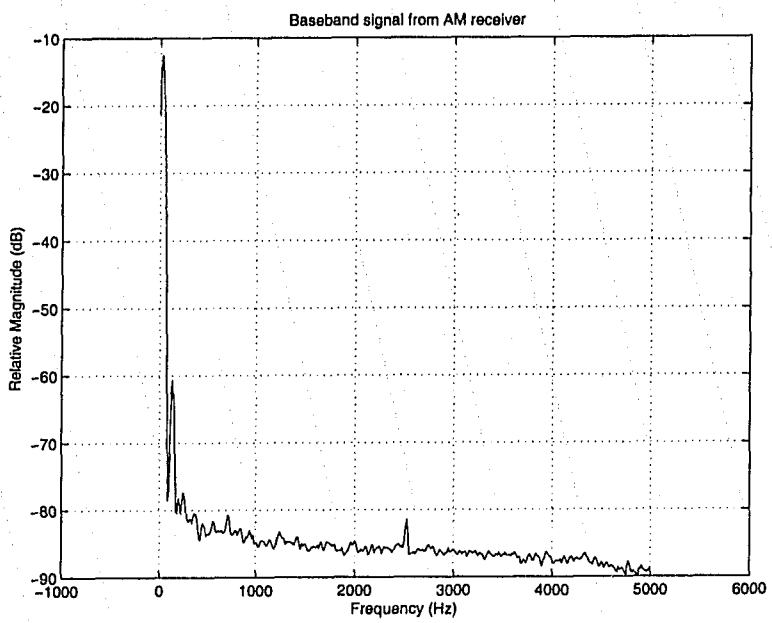

Figure 3.22: Spectrum of the demodulated signal when receiver is tuned to 590 kHz.

NO TEXT

Chapter 4

Short Wave Receiver

Due to the high oversampling rate of the AM receiver, the same architecture can be extended to operate in a higher frequency range. In this chapter we describe a receiver operating in the lower Short Wave range, namely, from 3-5 MHz. The receiver is implemented in the same target platform as in the AM receiver. The Short Wave receiver is designed to demodulate the same type of signal as in the AM receiver (ie. DSB signals with carrier with each carrier separated by 10 kHz). First, we introduce the principle of a *superheterodyne* receiver in Section 4.1. Then we describe the simulation results in Section 4.2. The actual hardware implementation details are provided in Section 4.3. Finally, the test and performance measurement of the receiver are provided in Sections 4.5 and 4.6 respectively.

4.1 Superheterodyne Receiver

In a *superheterodyne* receiver architecture, instead of having one intermediate frequency as in the case of a *heterodyne* receiver, we have two intermediate frequencies, namely $IF1$ and $IF2$. Figure 4.1 shows the architecture of a generic superheterodyne receiver.

In Figure 4.1 we can see that the incoming signal undergoes two downconversion operations. The first oscillator running at $F_{local1} = F_c + IF1$ shifts the incoming signal from the carrier frequency F_c to the first intermediate frequency $IF1$. The preceding BPF eliminates image stations prior to the downconversion.

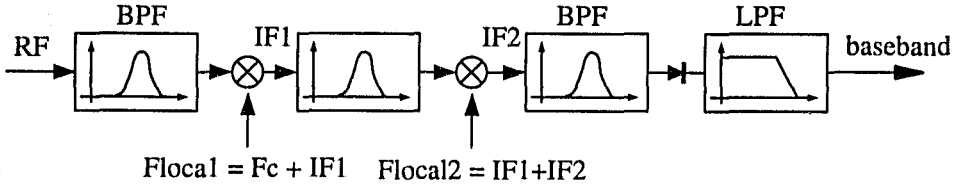

Figure 4.1: Block diagram of a generic superheterodyne receiver.

Once the desired signal is centered at $IF1$, we repeat the same procedure in order to shift the signal to $IF2$. The second mixer with a local oscillator running at $F_{local2} = IF1 + IF2$ effectively downconverts the desired signal to $IF2$. Finally, the detector stage recovers the baseband signal through envelope detection.

Recall that in the AM receiver we divided the receiver into four functional stages: *prefilter, RF, IF* and *detector* stages (refer to Section 3.2.4). If we were to perform the same block mappings for the Short Wave receiver, we obtain the grouping shown with dashed lines in Figure 4.2.

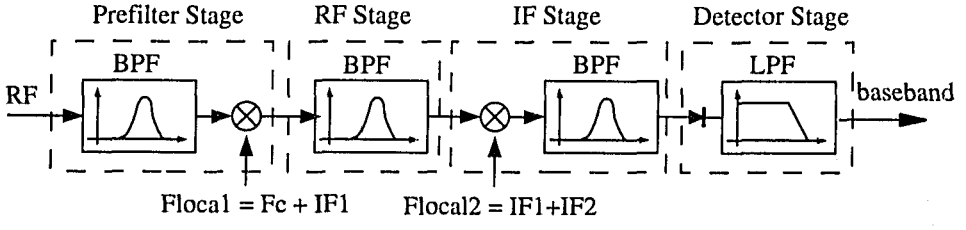

Figure 4.2: Block diagram of the superheterodyne receiver showing the four stages: prefilter, RF, IF and detector stages.

From Figure 4.2 we can point out some interesting characteristics of the superheterodyne architecture:

1. The incoming RF signal is shifted to a lower frequency $IF2$ in two stages. In general, this procedure requires fewer hardware resources compared to doing the downconversion in a single stage. This is because the transition band of the image rejection BPF used in the downconversion process is larger which results in a smaller filter. Moreover, the decimation operation

that takes place in between the two downconversion processes allows for an even further saving in hardware resources.

2. The only blocks that need to be changed depending on the carrier frequency are the first BPF and the first local oscillator. Both of these blocks are in the prefilter stage. Once the signal is downconverted to $IF1$, all the remaining processing blocks are fixed regardless of the carrier frequency.

3. Since $F_c > IF1 > IF2$, it is possible and also desirable to reduce the sampling rate as the operating frequency gets lower.

4. The block division shown in Figure 4.2, though arbitrary, shows that the only difference between the AM and Short Wave receiver lies in the *Prefilter* stage. If $IF1$ lies within the AM carrier frequency range, ie:

$$560 \text{ kHz} < IF1 < 1600 \text{ kHz}$$

then we could potentially treat the downconverted Short Wave signal centered at $IF1$ as just an AM radio station with a carrier frequency of $IF1$. This means we could re-use the same hardware block as in the AM receiver and only replace the *prefilter* stage.

Item 3 implies that we can introduce decimation blocks after each downconversion since the frequency of interest has been lowered. This modification is shown in Figure 4.3 with three decimation blocks and the preceding anti-aliasing low pass filters.

Item 4 implies that it is possible to *share* hardware between the AM receiver and the Short Wave receiver. More specifically, we could use the same RF, IF and detector stages as in the AM receiver. Figure 4.4 illustrates this point.

The architecture shown in Figure 4.4 is valid if we are interested in having both radio receivers in the same chip and be able to tune to *either* radio stations *one at a time*. It is not possible to have *both* receivers working at the same time unless we duplicate the shared hardware. In this work our eventual target is to have both

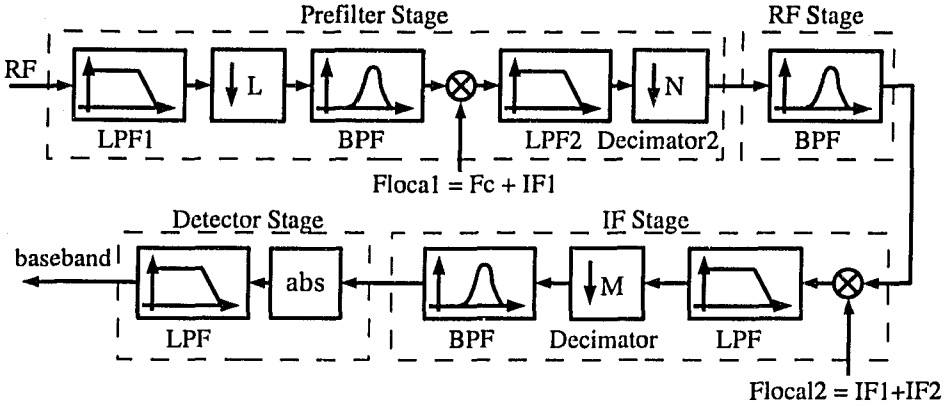

Figure 4.3: Modified block diagram of superheterodyne receiver showing decimation blocks.

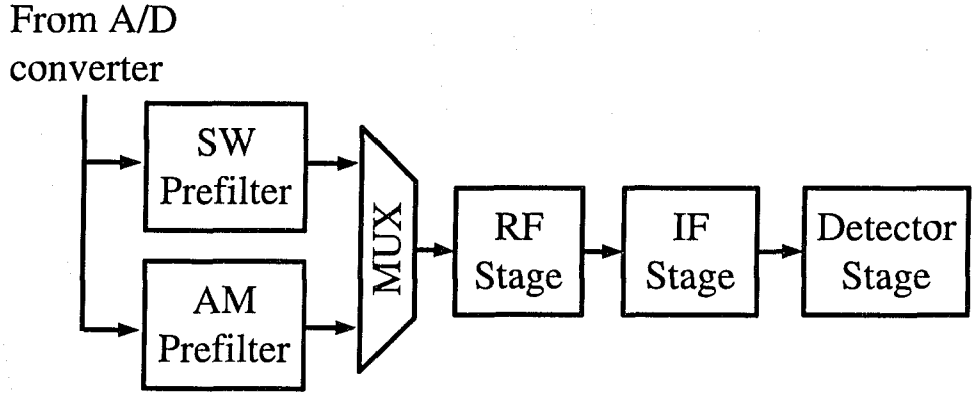

Figure 4.4: Resource sharing between Short Wave and AM radio receivers.

receivers working at the same time so this kind of hardware sharing technique is not employed.

4.2 Receiver Simulation

The Short Wave receiver is also modelled in MATLAB first in order to estimate the intermediate frequency values and the filter coefficients. In this case, we have to determine two intermediate frequencies and the parameters for the seven filters shown in Figure 4.3. In an attempt to improve the SNR of the Short Wave receiver with respect to the AM receiver, the datapath width is increased from 8 bits to 9 bits.

Also, the filter coefficients have a width of 9 bits instead of 8 bits.

4.2.1 Selection of IF1

Based on the Prefilter stage for the AM receiver, we know that input sampling rate has been decimated by a factor of 10 yielding an output sampling rate of 5 MSPS. Therefore, the first intermediate frequency should be low enough in order to be sampled at 5 MSPS without aliasing. Through trial and error, we settle for a first intermediate frequency $IF1$ of 800 kHz. This value is low enough for the required sampling rate and high enough to avoid excessively long filters. The decimation process is performed in two successive stages where the first decimation factor $L = 2$ and the second decimation factor $N = 5$ (refer to Figure 4.3).

4.2.2 Selection of IF2

Since the value of $IF1$ falls in the range of the AM receiver described in Chapter 3, it is possible to choose the same IF value as in the AM receiver for the second intermediate frequency (ie. setting $IF2 = IF = 50$ kHz). However, this value is not optimal if we consider the Short Wave receiver independently as a standalone receiver. In fact, we could take advantage of knowing the value of $IF1$ and calculate the optimal value of $IF2$ that would minimize the overall hardware resources as we did in Section 3.3.1 for the AM receiver.

The choice of the second intermediate frequency $IF2$ involves a trade-off between the BPF of the RF stage and the BPF at the IF stage. We compute the overall filter length for different values of $IF2$ and choose the value of $IF2$ that would minimize the overall filter length. A plot of the above computation is shown in Figure 4.5.

From Figure 4.5 we see that when $IF2 = 25$ kHz the overall length is minimized. Figure 4.6 shows the choice of decimation factor for different intermediate frequencies. We can see that when $IF2 = 25$ kHz, we can achieve a decimation factor of 30.

On the other hand, if we were to minimize the total number of operations per

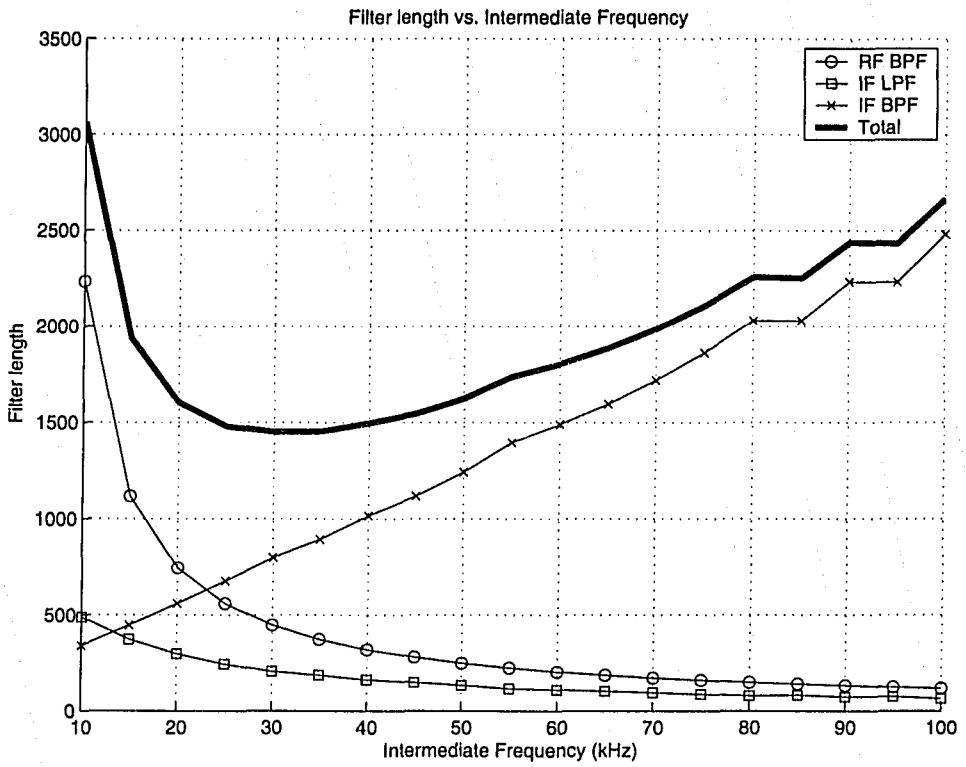

Figure 4.5: Filter length vs. intermediate frequency 2.

second, we need to plot the number of operations per second versus $IF2$. This is shown in Figure 4.7. In this case, the total number of operations per second is minimized when $IF2 = 70$ kHz. Again, we decide to minimize the hardware resources and choose the value of 25 kHz for $IF2$.

4.2.3 Summary

With the chosen values of $IF1, IF2, L, N$ and M, we can use MATLAB to estimate the filter parameters. Table 4.1 shows the parameters obtained through simulation for each filter that appears in Figure 4.3.

Figures 4.8 to 4.14 show the magnitude of the frequency response of the seven filters listed in Table 4.1. The filters are set to demodulate a carrier frequency of 4.5 MHz. All the filters have an input of 9 bits. Both the output data and filter coefficients are quantized to 9 bits.

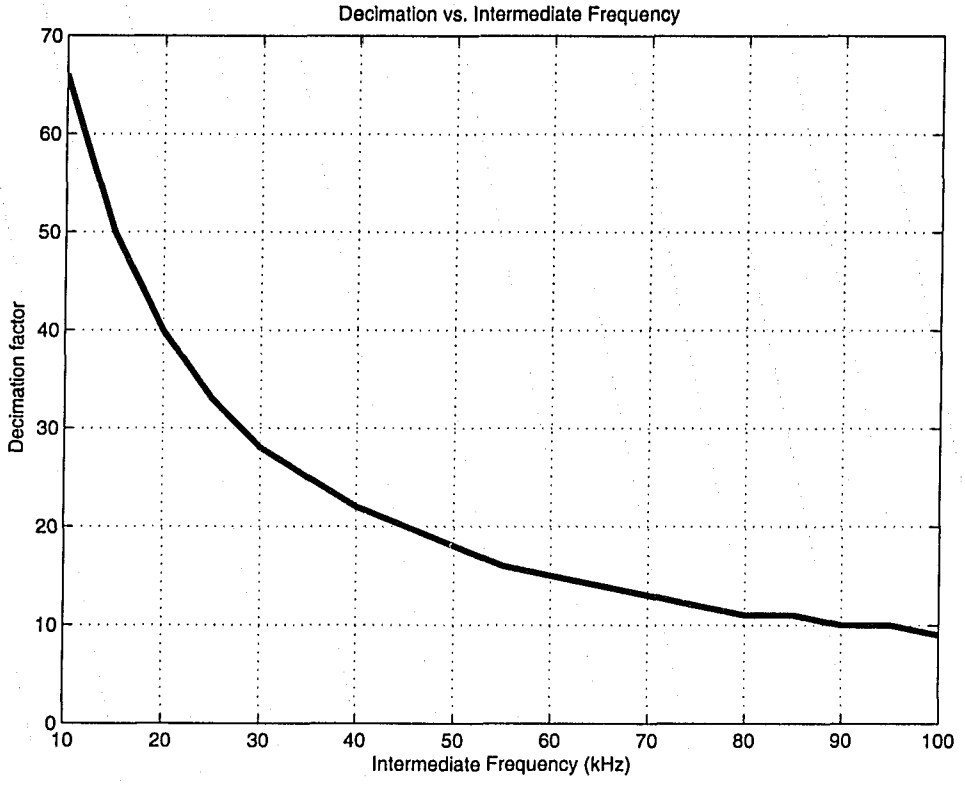

Figure 4.6: IF stage decimation factor vs. intermediate frequency 2.

Table 4.1: Filter parameters for the SW receiver.

Filter	Passband	Sample Rate	# Taps
Prefilter LPF1	0 to 5.005 MHz	50 MSPS	16
Prefilter BPF	$F_c \pm 800$ kHz	25 MSPS	141
Prefilter LPF	0 to 800 kHz	25 MSPS	34
RF BPF	800 kHz $\pm$ 25 kHz	5 MSPS	560
IF LPF	0 to 30 kHz	5 MSPS	211
IF BPF	25 kHz $\pm$ 2.5 kHz	166 kSPS	421
Detector LPF	0 to 5 kHz	166 kSPS	421

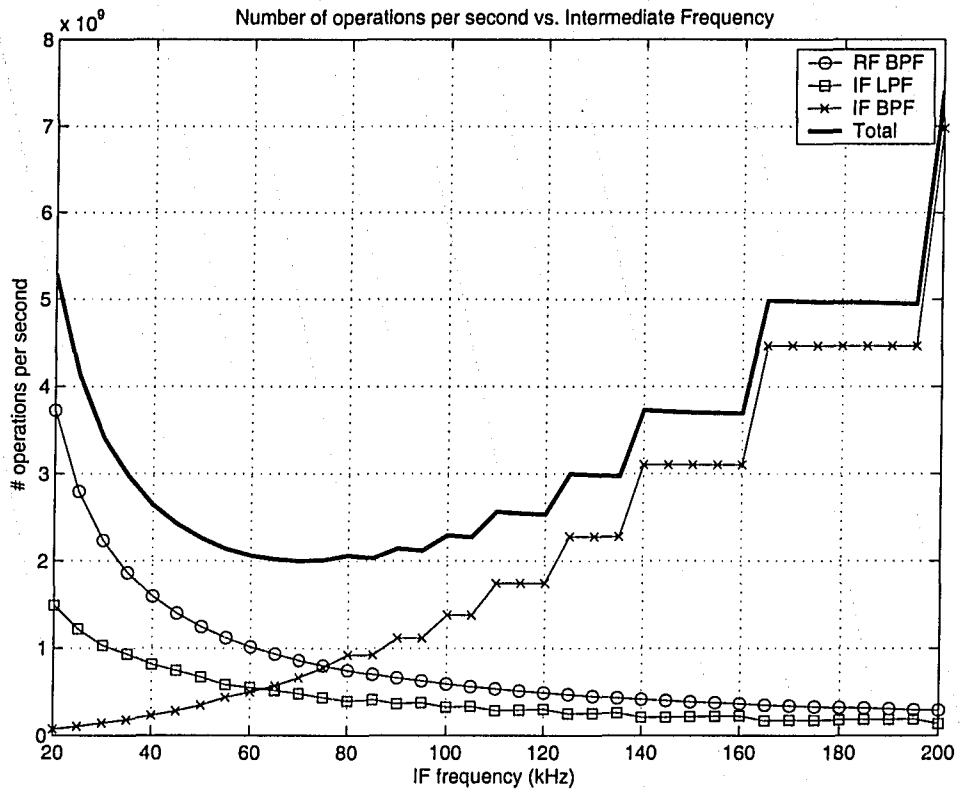

Figure 4.7: Operations per second vs. intermediate frequency 2.

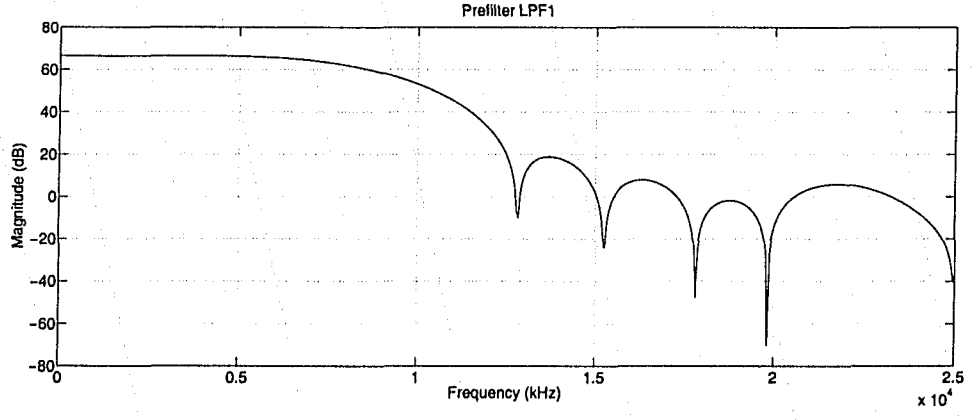

Figure 4.8: Magnitude of the frequency response of prefilter LPF1.

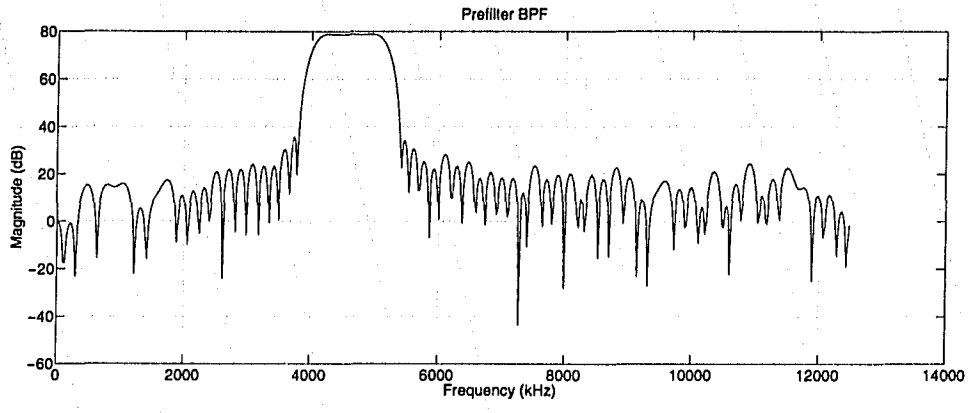

Figure 4.9: Magnitude of the frequency response of prefilter BPF.

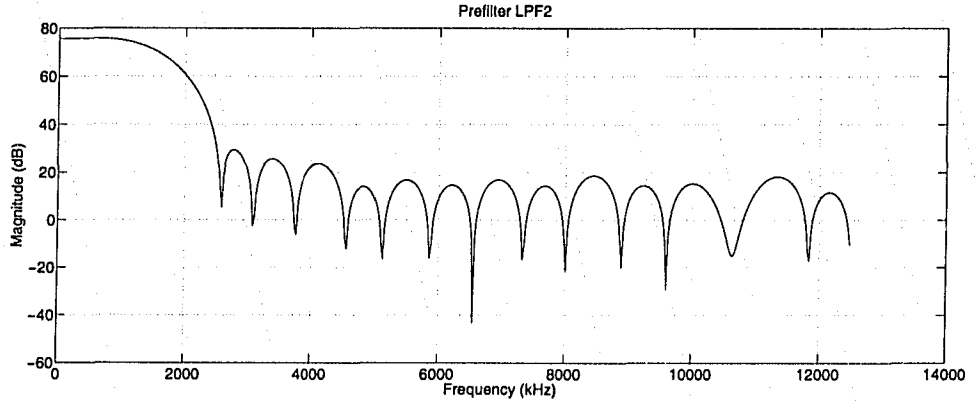

Figure 4.10: Magnitude of the frequency response of prefilter LPF2.

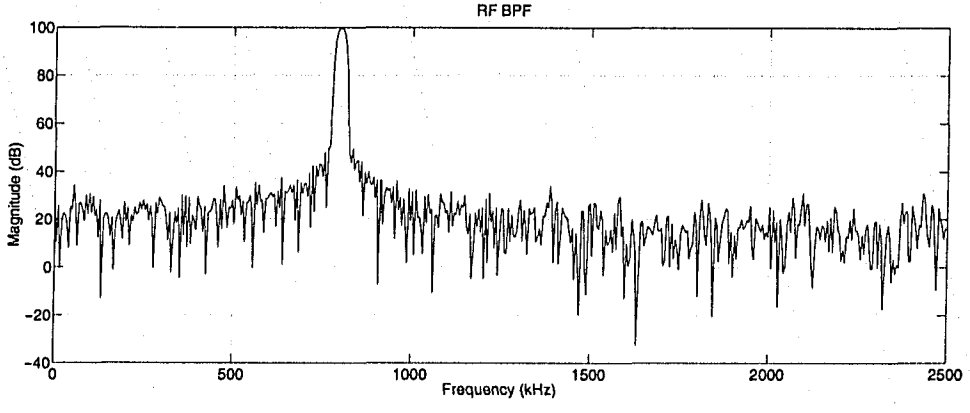

Figure 4.11: Magnitude of the frequency response of RF BPF.

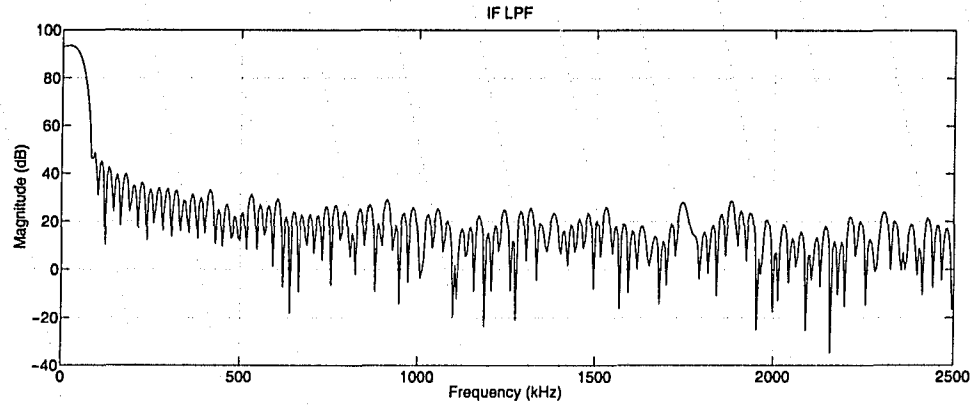

Figure 4.12: Magnitude of the frequency response of IF LPF.

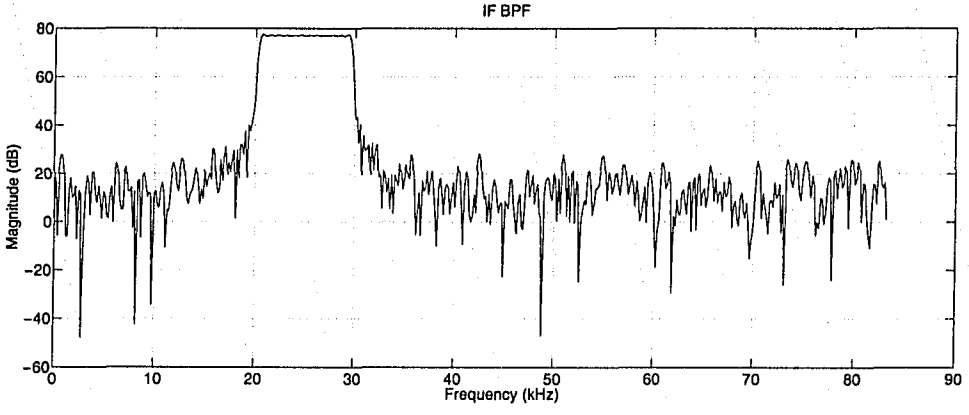

Figure 4.13: Magnitude of the frequency response of IF BPF.

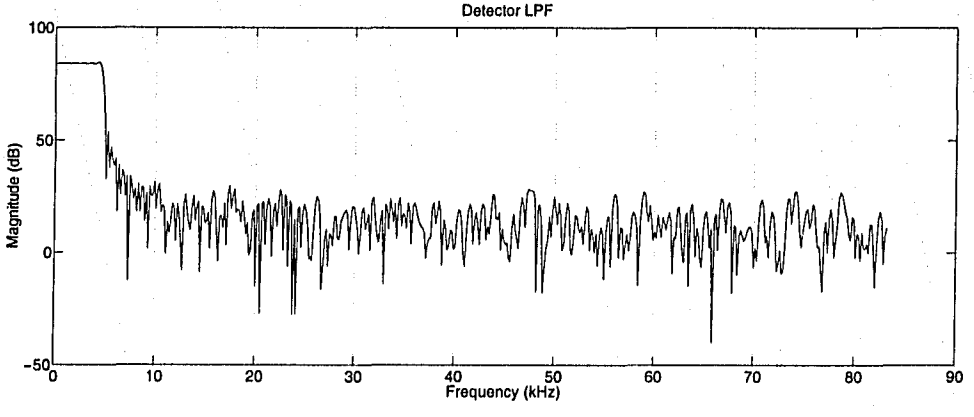

Figure 4.14: Magnitude of the frequency response of detector LPF.

4.3 Hardware Implementation

In this section we describe the receiver proposed above for the intended FPGA device. The SW receiver architecture is shown in Figure 4.15.

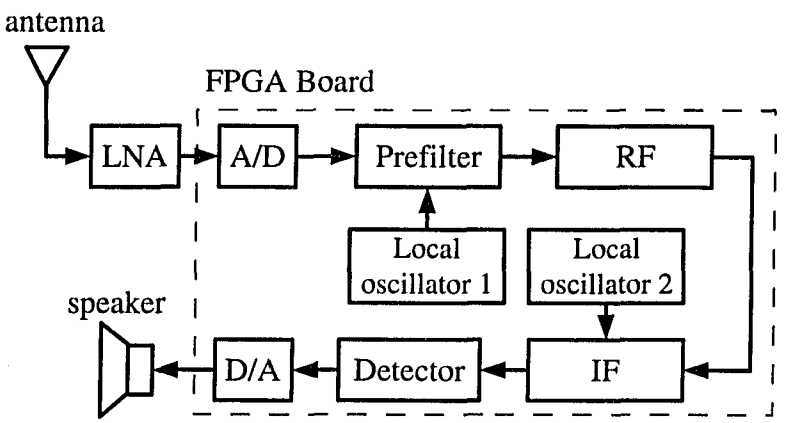

Figure 4.15: Short Wave receiver architecture.

As we can see from Figures 3.12 and 4.15, the main difference between the AM receiver and the SW receiver is the addition of one extra local oscillator since we have two intermediate frequencies. Most of the receiver complexity is found in the prefilter and IF stages which we will describe in the following sections.

4.3.1 Prefilter Stage

The block diagram of the prefilter stage is shown in Figure 4.16.

The *prefilter* performs two main functions: 1) Lower the sampling rate of the incoming signal, and 2) shift the desired channel to the first intermediate frequency $IF1$. An overall decimation of 10 is carried out in two steps. We first decimate by 2 and then decimate by 5. This approach reduces the total length of the low pass filters required to carry out the decimation since the number of taps needed in an anti-aliasing filter increases considerably with the sampling rate. The downconversion is performed by a *Digital Direct Downconverter* (DDC) with a local oscillator frequency of

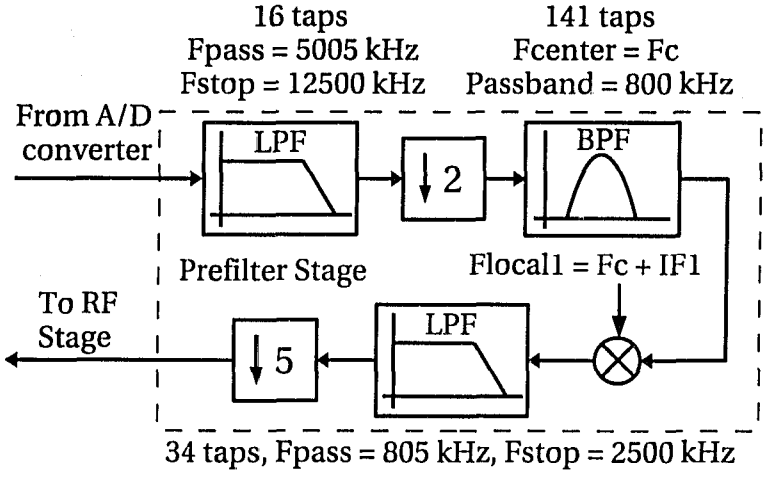

Figure 4.16: Prefilter stage of the SW receiver.

$$F_{local1} = F_c + IF1 \tag{4.1}$$

where F_c is the carrier frequency and $IF1$ = 800 kHz as mentioned before.

The first low pass filter is *fully parallel* because it has to process one data sample per system clock cycle. Because of this, it is imperative to make this filter as small as possible because extra hardware resources are needed in order to have a throughput of one sample per clock cycle.

The bandpass filter's center frequency varies depending on the carrier frequency. This is because the inequality described by Eq. (3.1) has to be satisfied in order to reject image stations. Therefore, this BPF is implemented as a *reloadable* FIR filter and the different filter coefficients are loaded depending on the range of carrier frequencies.

Figure 4.17 shows the bandpass filters used for different carrier frequency ranges. For example, if the carrier frequency is F_c = 3.5 MHz, we select the 2^{nd} bandpass filter since $F_c < 3.6$ MHz. This bandpass filter has a flat passband from 3.00 to 3.8 MHz as shown in Figure 4.17. The same selection algorithm is applied to other carrier frequencies and the required control logic is implemented with a very simple finite state machine inside the FPGA. This BPF has to produce one output sample

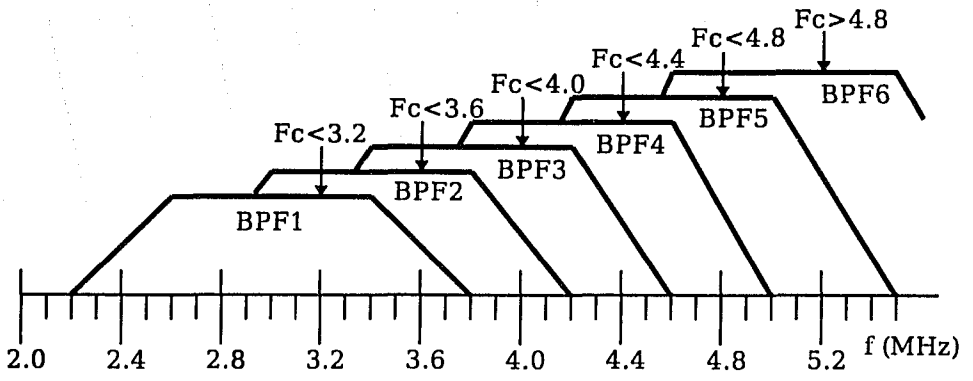

Figure 4.17: Prefilter bandpass filter selection with respect to carrier frequency.

for every two clock cycles because the input data rate is 25 MSPS and the system clock rate is 50 MHz.

Note that the output of every FIR filter is wider than 9 bits as in the case of the AM receiver. A gain controller block is appended at the output of each FIR filter to rescale the output to 9 bits. These gain controller blocks are not shown in the figures for simplicity.

4.3.2 RF Stage

The RF stage is similar to its AM receiver counterpart and it is composed of only one bandpass filter. However, since the input signal is fixed at a frequency of $IF1$, this bandpass filter is not reloadable and it is centered at $IF1$ with a passband of $2 \times IF2 = 50$ kHz. The bandpass filter is implemented with a fully serial FIR filter of length 560 since we have 10 clock cycles to operate on each data sample.

4.3.3 IF Stage

The block diagram of the IF filter is shown in Figure 4.18.

The purpose of the IF Filter stage is to downconvert the signal to $IF2$ and reject adjacent channels. In this case, the second local oscillator has a fixed frequency of

$$F_{local2} = IF1 + IF2 = 825 \text{ kHz} \tag{4.2}$$

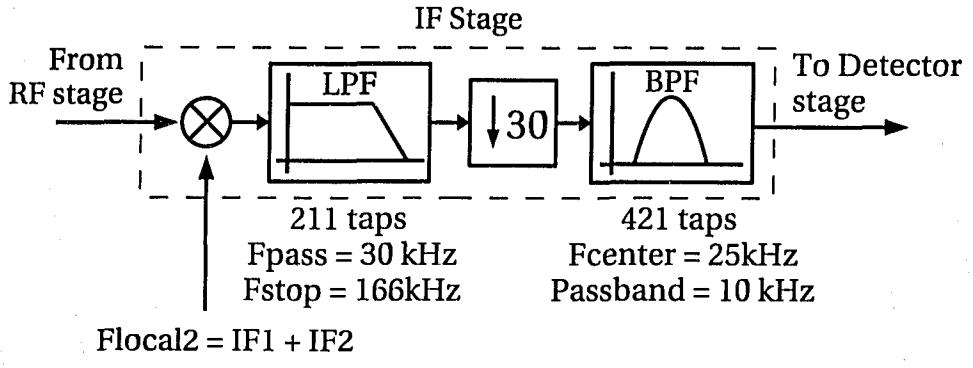

Figure 4.18: IF filter stage of the Short Wave receiver.

As we can see from Figure 4.18, the signal undergoes the following operations: downconversion to $IF2$, low pass filtering, decimation by 30 and adjacent channel rejection. The resulting signal contains only the desired radio station at a carrier frequency of 25 kHz. The LPF and BPF have length of 211 and 421 taps respectively.

4.3.4 Detector Stage

The detector has the same architecture as shown in Figure 3.16. In this case, the datapath is also increased by one bit in order to account for the fact that the MSB after the absolute value block is always 0. This is why Figure 4.19 shows a datapath of 10 bits.

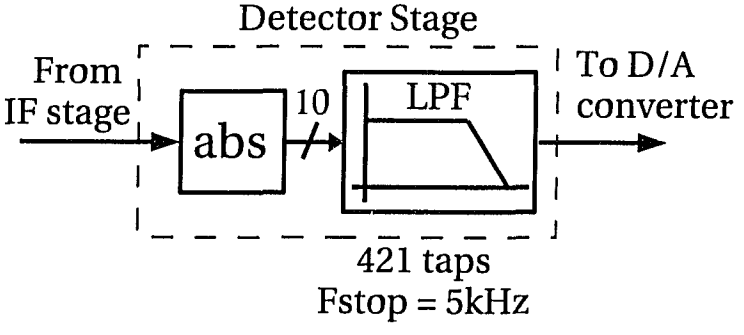

Figure 4.19: Detector stage of the Short Wave receiver.

Table 4.2: Resource utilization for the SW receiver on a Virtex-E 2000 device.

	Used	Total	Percentage
Slice FF	14,075	38,400	36 %
4 input LUT	9,823	38,400	25 %
Occupied Slices	8,948	19,200	46 %
Bonded IOBs	66	404	16 %
Block RAMs	6	160	3 %
Equivalent gate count	777,792		

4.4 Control Logic

The control logic in the SW receiver is similar to the control logic in the AM receiver. The two blocks that need to be updated whenever the user chooses another radio station are the bandpass filter and the digital downconverter found in the prefilter stage (refer to Fig. 4.16). The same control blocks shown in Figure 3.18 are implemented in the SW receiver. Again, the BPF coefficients and the DDC phase increment values are stored in block RAM inside the FPGA. A main controller exposing a set of write-only registers provide the desired radio station. This information is then passed to the DDC and BPF controllers in order to update these two blocks. Detailed description of the control path can be found in Chapter 5.

4.5 Resource Utilization

Table 4.2 shows the resource utilization of the SW receiver described in Chapter 4.

As we can see from Table 4.2, the Short Wave receiver consumes 25% of available 4-input LUTs versus 6% for the AM counterpart. If we compare Table 4.2 with Table 3.2, we can see that the SW receiver consumes more hardware resources compared to the AM receiver. This increase in resource utilization is due to several reasons:

- The datapath in the SW receiver is 9 bits wide versus 8 bits wide in the AM receiver.

- The SW receiver operates with a much higher carrier frequency, resulting in the need for longer filters.
- The transition band of filters used in the SW receiver are generally narrower than the filters used in the AM receiver.

Note that while the SW receiver requires more resources in general, it does utilize fewer block RAMs compared with the AM receiver (3% for SW receiver versus 20% for AM receiver). This is mainly due to the fact that fewer sets of BPF coefficients are needed in the SW receiver. In fact, the SW receiver needs to store only 6 sets of BPF coefficients for the prefilter block (shown in Figure 4.17) while the AM receiver stores 200 sets of BPF filter coefficients for the RF stage.

In terms of power consumption, the SW receiver consumes more power than the AM receiver. Approximately 3 Amps are drawn from a power supply of 5 Volts, which translates into a power consumption of 15 Watts. This increase in power consumption is mainly due to a higher device utilization in the SW receiver compared to the AM receiver. A heat sink was installed on the FPGA device in order to help dissipate the heat.

4.6 Test and Performance Measurement

In order to test the performance of the SW receiver, we proceed in the same way as before. The complete receiver system along with the signal generator and spectrum analyzer is shown in Figure 4.20.

The function generator produces an amplitude modulated signal with a carrier frequency of 4 MHz, AM index: 100% and a sinusoidal baseband signal of 2500 Hz. The spectrum of this input signal is shown in Figure 4.21. Again, we have an input SNR of approximately 60 dB for the sidebands.

When the receiver tunes into the same carrier frequency, the output is a clean 2500 Hz sine wave. The spectrum of the demodulated signal is shown in Figure 4.22. From the graph, we can see that the SNR in this case is approximately

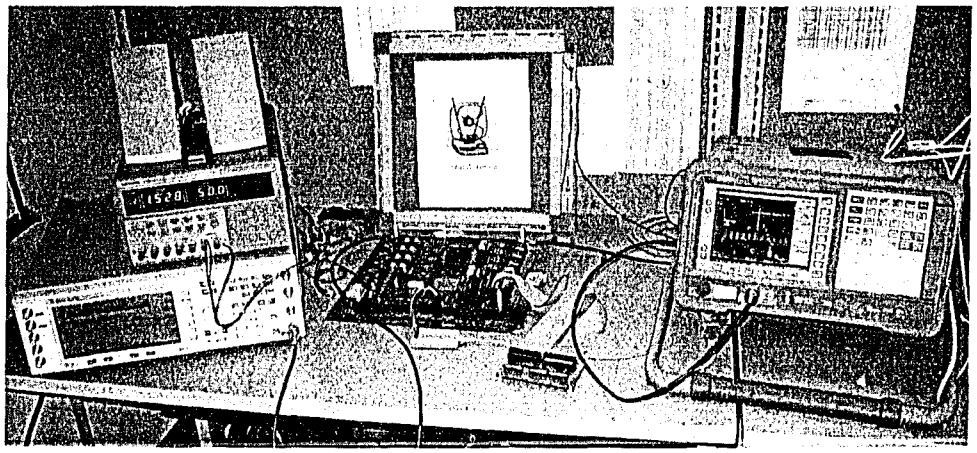

Figure 4.20: Complete SDR receiver system showing the antenna, the signal generator, the spectrum analyzer and the pair of output speakers.

55 dB. This value is expected to be higher than the AM receiver since we are spending one extra bit in the datapath compared to the AM receiver.

When the receiver is tuned to a different frequency, the baseband signal disappears and the spectrum is shown in Figure 4.23. Again, original baseband signal has been attenuated by approximately 70 dB.

In the case of real audio signals, we could not proceed in the same way as in the AM case because most commercial Short Wave broadcast stations use a *Single Side Band* (SSB) modulation scheme. However, the setup shown in Figure 4.24 enabled us to test the performance of this receiver under real audio signals. The setup consists of an *Enhanced Signal Generator* (ESG) that is capable of producing arbitrary waveforms. This generator produces the test signal for the Short Wave receiver.

An random 90-second talk show audio recording was obtained from on-line sources and converted to WAV format. The file is sampled at 44.1 kHz with a resolution of 16 bits. This audio file is then formatted according to [48] and downloaded the vector signal generator through FTP. The vector signal generator then uses the *Arbitrary Waveform* feature to generate an amplitude modulated signal at a carrier frequency between 3 to 5 MHz. This modulated signal is then used to test the Short

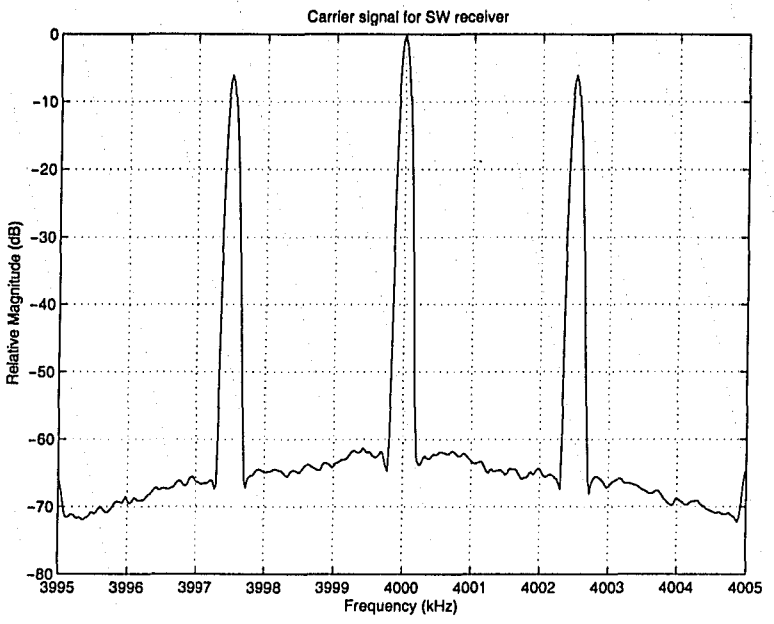

Figure 4.21: Spectrum of the input signal for the SW receiver. The carrier frequency is 4 MHz and the baseband signal frequency is 2500 Hz.

Wave receiver at different carrier frequencies. The resulting demodulated audio recording was intelligible and the audio quality acceptable.

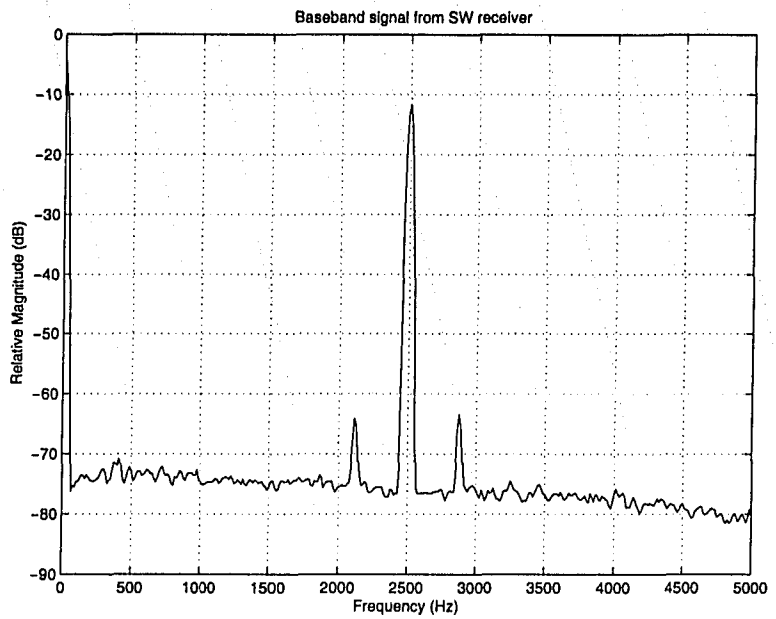

Figure 4.22: Spectrum of demodulated signal when the receiver is tuned to 4 MHz.

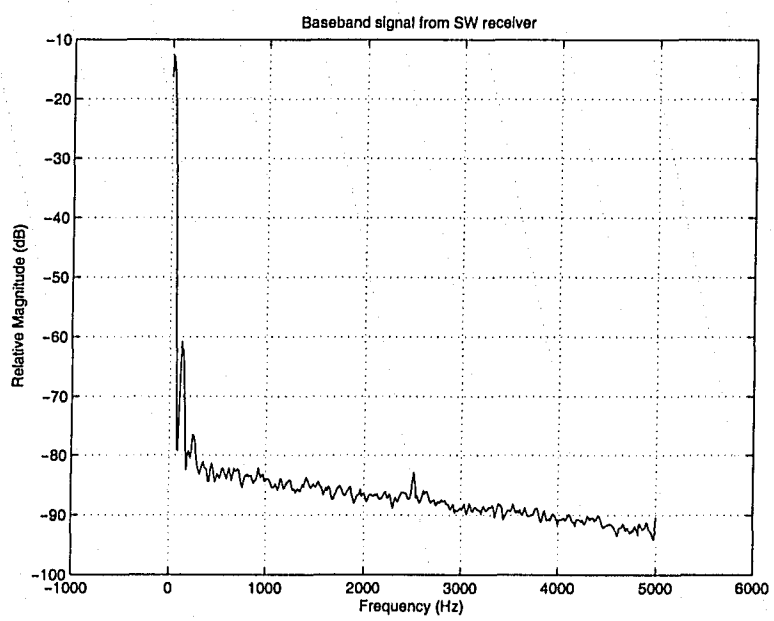

Figure 4.23: Spectrum of demodulated signal when tuned to 3.090 MHz.

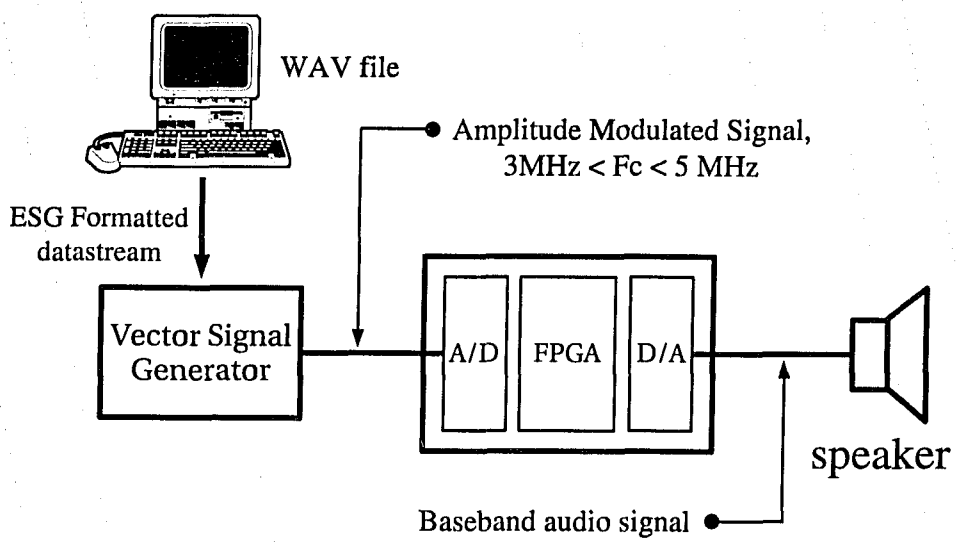

Figure 4.24: Test setup for Short Wave receiver using a function generator.

Chapter 5

System Integration

> All parts should go together without forcing. You must remember that the parts you are reassembling were disassembled by you. Therefore, if you can't get them together again, there must be a reason. By all means, do not use a hammer.
>
> IBM maintenance manual (1925)

Having described both the AM and Short Wave receivers, we are ready to assemble the two SDR receivers together in a single device. In order to do that, we first describe the set of peripherals that are needed in order to communicate with the two receivers. This set of peripherals are described in Section 5.1. In Section 5.2 we describe in detail the user interface and the control path in order to control both receivers in an uniform and user-friendly manner. Finally we give an overview of what has been accomplished in Section 5.3.

5.1 Peripherals

As we can see from Figure 5.1, the entire software defined radio platform includes an antenna, a *low noise amplifier* (LNA) or also referred as the *RF Frontend*, the FPGA board itself, an SMA-to-audio converter and a set of audio speakers.

5.1.1 Loop Antenna

The antenna itself is a simple loop antenna adapted from [49, 50]. The antenna is built by wrapping 100 feet of AWG 24 cable around a square made with plastic

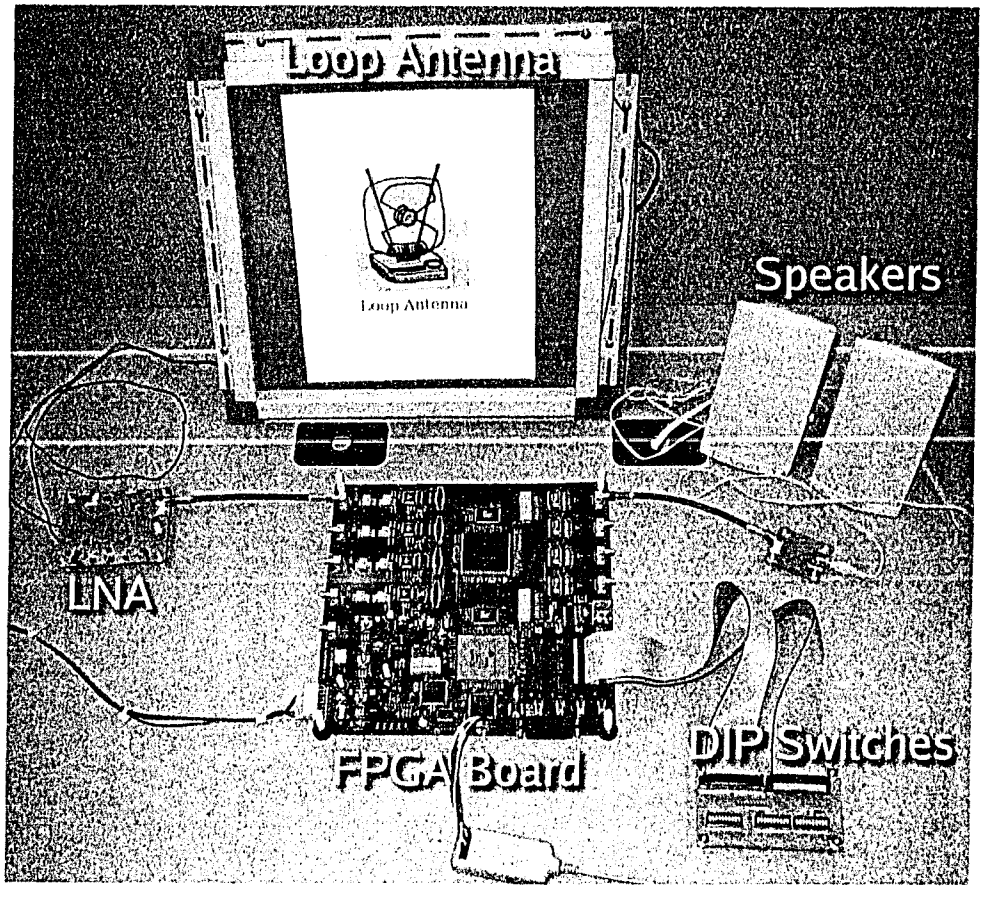

Figure 5.1: Whole receiver system showing the antenna, low noise amplifier, FPGA board, backend and speaker.

rails with U-shaped cross section. The square measures 12 1/2" on each side. The plastic square is fixed on a piece of cardboard to maintain its shape.

5.1.2 Low Noise Amplifier

The purpose of the *Low Noise Amplifier* (LNA) is to amplify the signal generated by the antenna to a level that is suitable for the A/D converter. The maximum dynamic range of the A/D converters is $1V_{pp}$. With a resolution of 12 bits, the minimum detectable signal level is approximately 0.48 mV

A simple low noise amplifier adapted from [51] is shown in Figure 5.2. The first stage of this *2-stage* amplifier consists of a NPN bipolar transistor (2N3904) in

common emitter configuration. The base of this transistor is biased by R1 and R2 to a DC voltage of around 2V. The input signal is AC coupled through C2. Note that capacitor C2 and the resistor R1||R2 form a high pass filter that eliminates unwanted low frequency noise such as 60 Hz power line fluctuations that may be picked up by the loop antenna. The variable resistor R4 at the collector sets the gain of the circuit. The second stage consists of a PNP transistor (2N3906) in common collector configuration and it serves as an impedance matching stage by providing a low output impedance. The large capacitor C1 is used as a bypass capacitor to eliminate noise from the power supply.

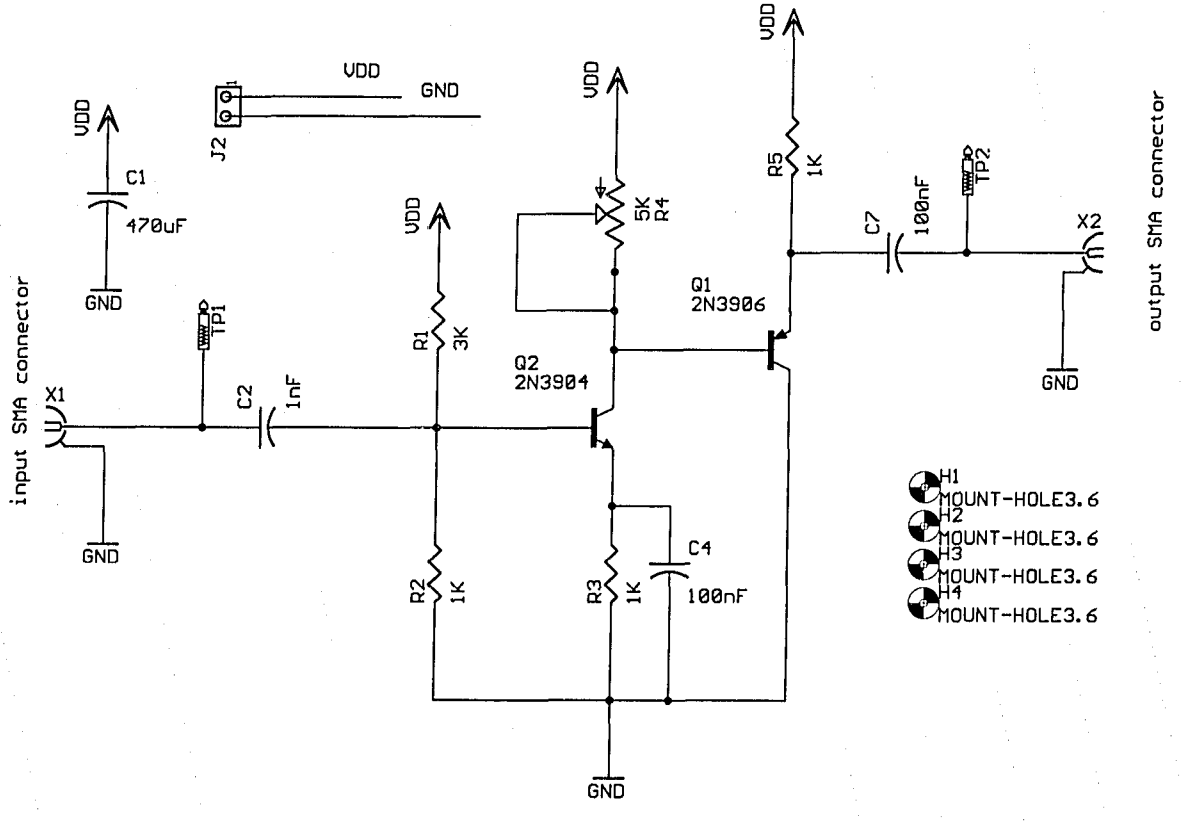

Figure 5.2: Schematic of the low noise amplifier.

To verify the frequency response of this amplifier, we used gEDA [52] for schematic capture and simulated the circuit using NGSpice [53]. Figure 5.3 shows the input and output response in dB versus frequency in log scale. As we can see,

the amplifier provides enough attenuation at low frequencies, unity gain at around 2-3 kHz and a flat gain of $\approx$ 18 dB from 50 kHz to 10 MHz.

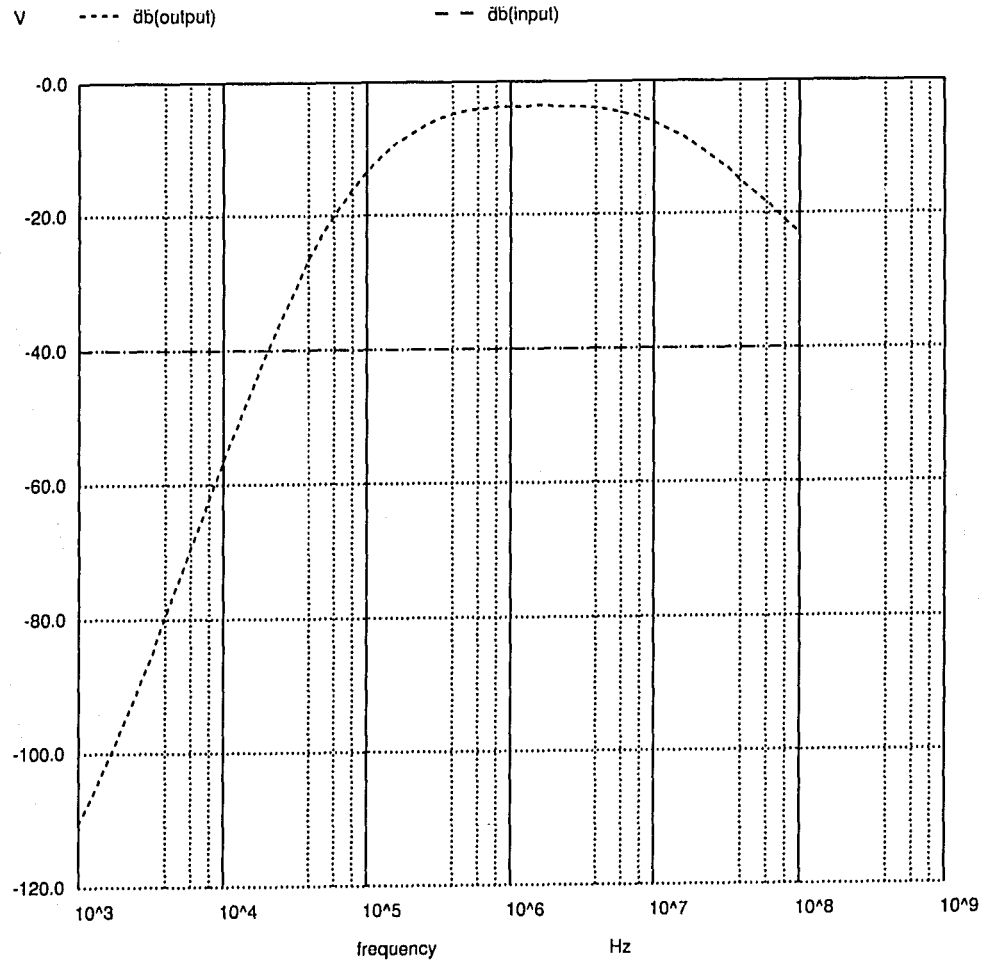

Figure 5.3: Frequency response of the low noise amplifier.

A picture of the actual low noise amplifier is shown in Figure 5.4. Tests performed with a function generator and spectrum analyzer confirm that this amplifier actually works as expected from the simulation results.

5.1.3 FPGA Board

The FPGA board is shown in Figure 5.5. In the middle of the board we can see two Xilinx Virtex-E 2000 devices. The top device, denoted as the AC FPGA by the manufacturer, has a heat sink installed in order to help dissipate heat. The bottom

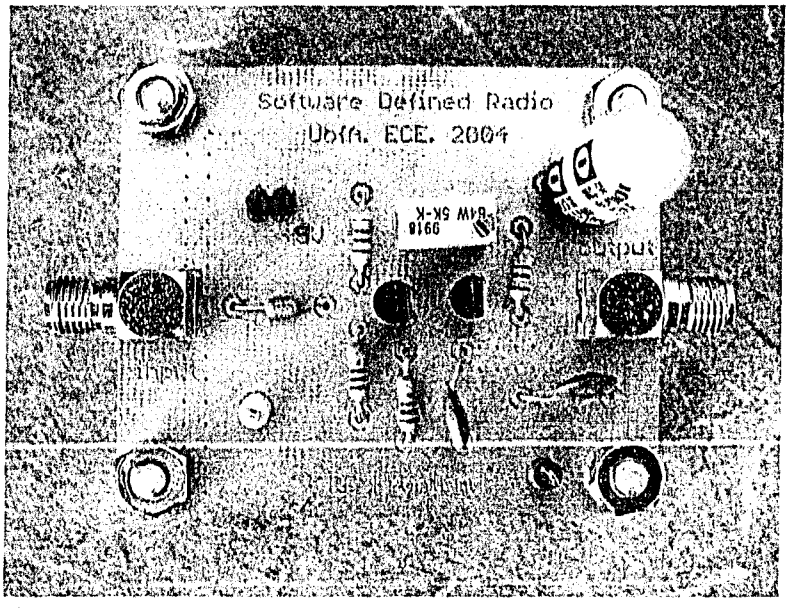

Figure 5.4: Board for the low noise amplifier.

device, denoted the DP FPGA, provides the user interface and its purpose will be described in Section 5.2.

To the left of the board we have the four ADC channels capable of sampling at 100 MSPS with 12 bits of precision (AD9432). The four DAC channels are located in the right hand side of the board and use the AD9762 chip.

5.1.4 SMA-to-Audio Connector Converter

The goals of the SMA-to-Audio-Connector converter are:

1. Remove the DC component of the output signal.

2. Convert from SMA female connector to an audio female connector.

It is necessary to remove the DC component in the analog domain because the AD9762 DAC used in the FPGA board is in *single-ended* configuration and hence it does not allow negative voltages [54]. The above goals are accomplished with a capacitor in series with the output. The schematic of the board and the actual implementation are shown in Figures 5.6 and 5.7.

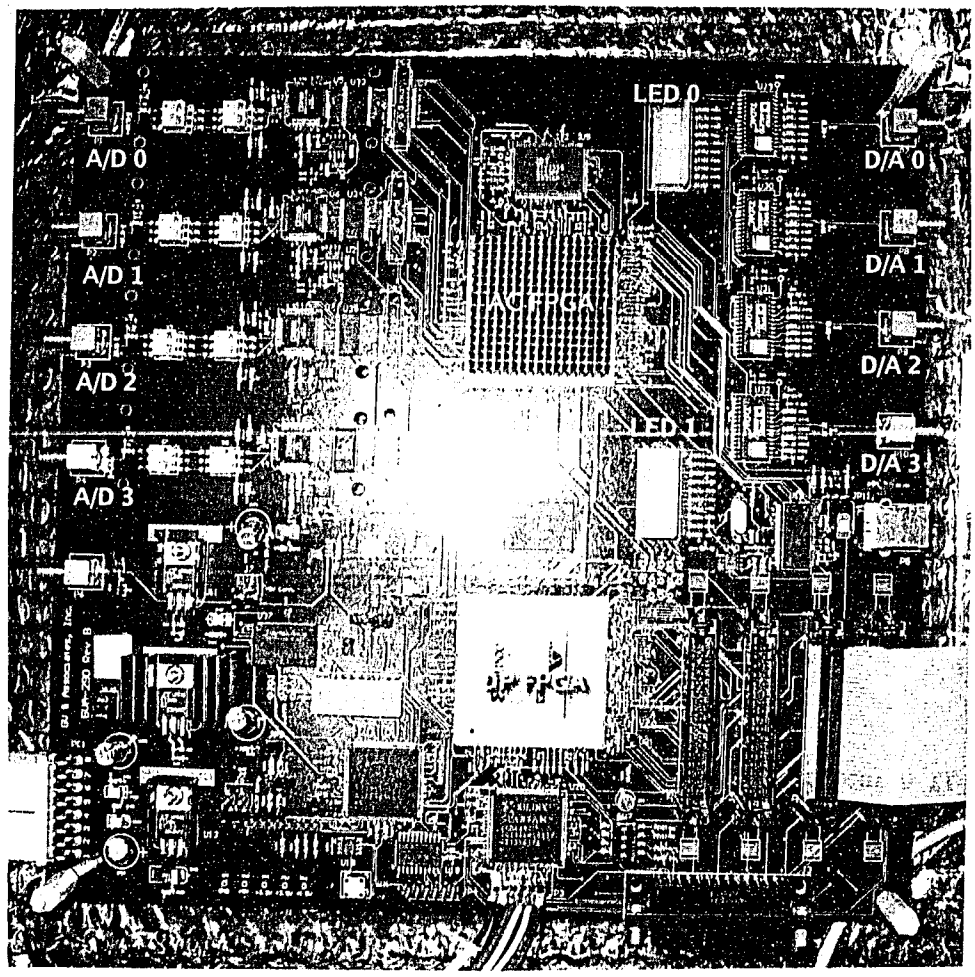

Figure 5.5: GVA-290 FPGA board showing the A/D and D/A converters and the two Virtex-E 2000 FPGAs.

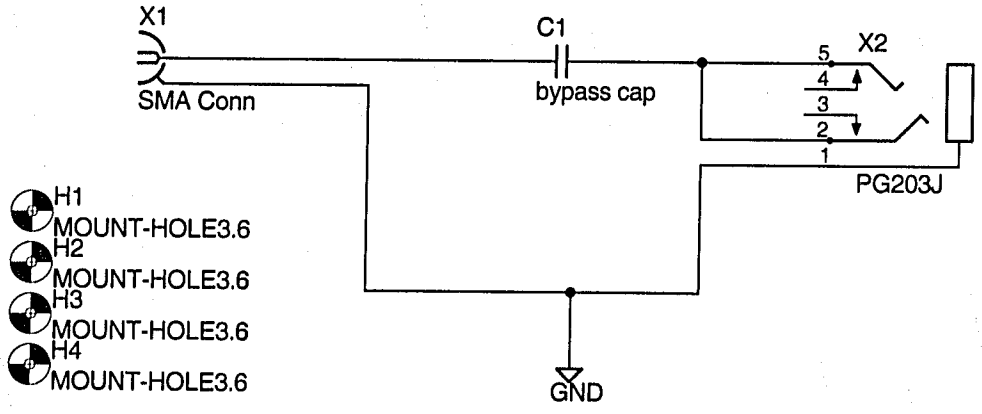

Figure 5.6: Receiver backend schematic.

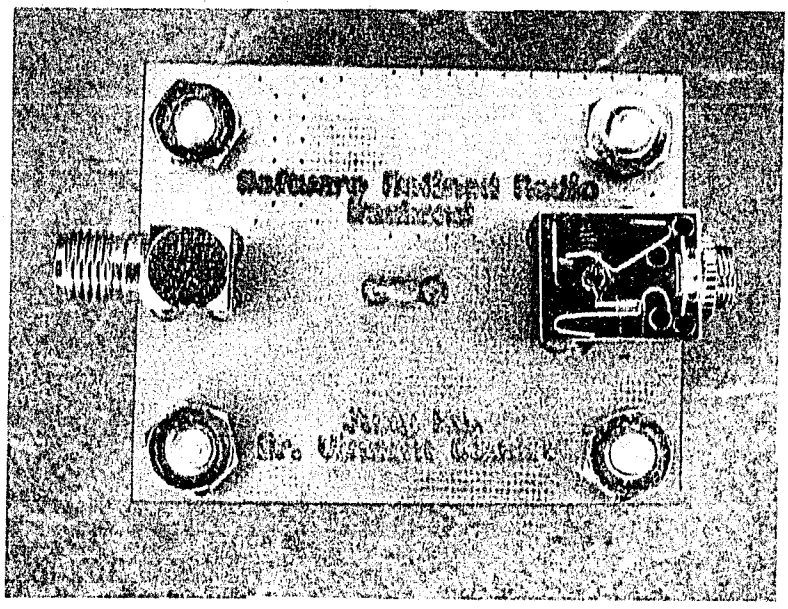

Figure 5.7: Receiver backend board.

5.2 Control Path and User Interface

5.2.1 Overview

In this section we present the control logic necessary to control *both* AM and Short Wave receivers in the same FPGA device. A soft core microprocessor from Xilinx, Microblaze [55], is used to control the two receivers and it provides a user interface to the operator through a serial port. Figure 5.8 gives an overview of how the system is controlled by the microprocessor.

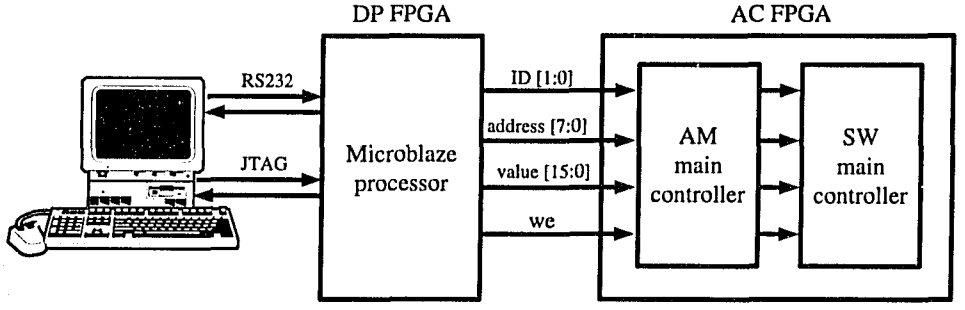

Figure 5.8: Overview of the user interface.

There are three main components shown in Figure 5.8: the user interface provided by the PC, the Microblaze processor implemented in the DP FPGA and the two SDR receivers implemented in the AC FPGA. In the following sections we will describe the AC FPGA interface and the user interface between the PC and the DP FPGA. Implementation details of the Microblaze processor can be found in Appendix A.

5.2.2 AC FPGA

In the AC FPGA we implement the two SDR receivers discussed in Chapters 3 and 4. The control block that receives the external input from the user is called the *main controller* and it was presented in Sections 3.5 and 4.4 for the AM and Short Wave receivers respectively. The purposes of the main controller are:

1. Allow the user to choose the carrier frequency (ie. tune into a different

radio station).

2. Debug the receiver by redirecting the output of intermediate stages to the output DAC.

3. Set the gain value of each gain control blocks in the receiver, or enable the automatic gain control feature for each filter.

4. Observe the gain setting of each gain control block.

Figure 5.9 shows how the AM and Short Wave receivers are implemented inside the AC FPGA.

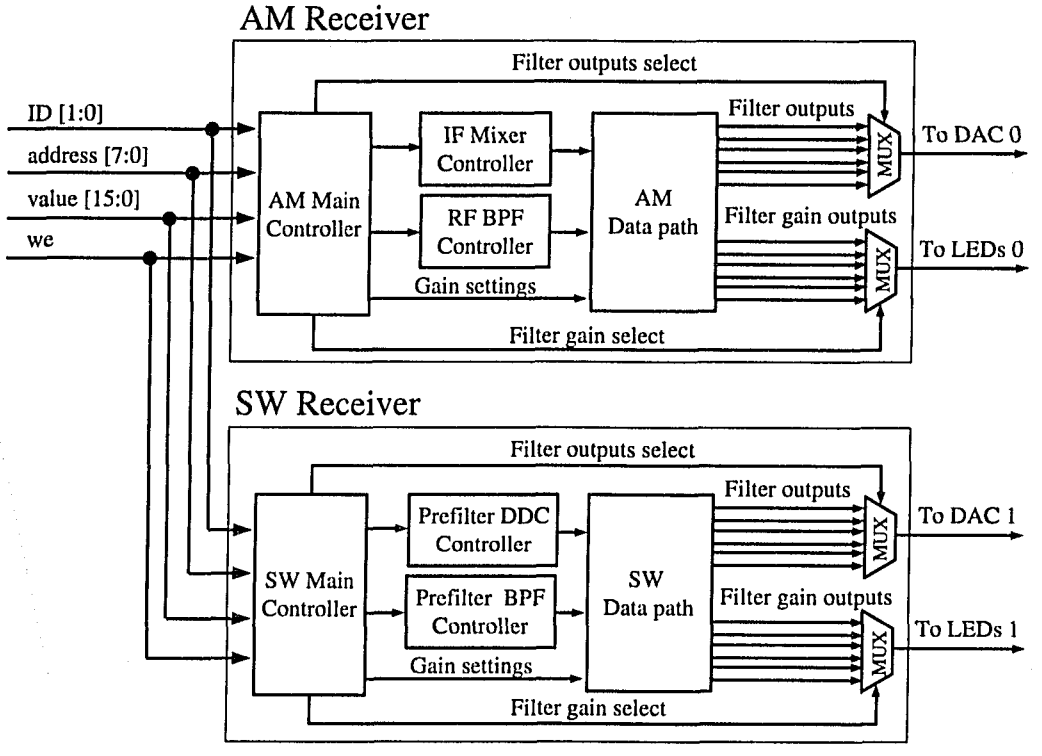

Figure 5.9: AM and Short Wave receivers implemented inside the AC FPGA.

As we can see from Figure 5.9, the user interacts with the two main controllers through a shared bus. A two-bit `ID` bus uniquely identifies between the AM and the Short Wave main controllers. The user can write to the desired register by

specifying the `address` and `data` buses and assert the `we` signal for one clock cycle.

Based on the width of the main controller's input signals, we can conclude that this configuration can address up to 4 different receivers, 256 internal registers and register values up to $2^{16} - 1 = 65535$. This address space should be more than enough for the present work including future expansions.

In order to observe the output, two bus multiplexers are implemented in each receiver to choose which outputs should be observed at a given time. The first multiplexer chooses which intermediate output should be routed to the output DAC. The second multiplexer does the same thing but for the AGC gain values, which are routed to the on-board LED segment. The select signals are also provided by the main controller block and each select signal has its own register in the main controller. This same concept is extended for other control signals such as the selection of the carrier frequency. The reader is referred to Appendix B for a detailed register mapping of the two main controllers and some examples on how to write to these internal registers.

5.2.3 User Interface

The user interface between the PC and the Microblaze processor is provided by a C program running inside the embedded processor. The user controls each receiver by typing commands in a command-line driven interface. These commands are then sent to the processor through a serial link as shown in Figure 5.8. A complete list of the available commands and their descriptions can be found in Appendix C.

The user can use terminal programs such as *minicom* [56] or *Hyperterminal* in order to communicate with the user interface. Figures 5.10 and 5.11 illustrate the user interface under a *Hyperterminal* session.

As we can see, this interface allows the user to control each one of the software receivers independently. As long as a given receiver contains a main controller that has the same communication interface, it is possible to control and debug using this user interface. This user interface turned out to be very useful in the testing and

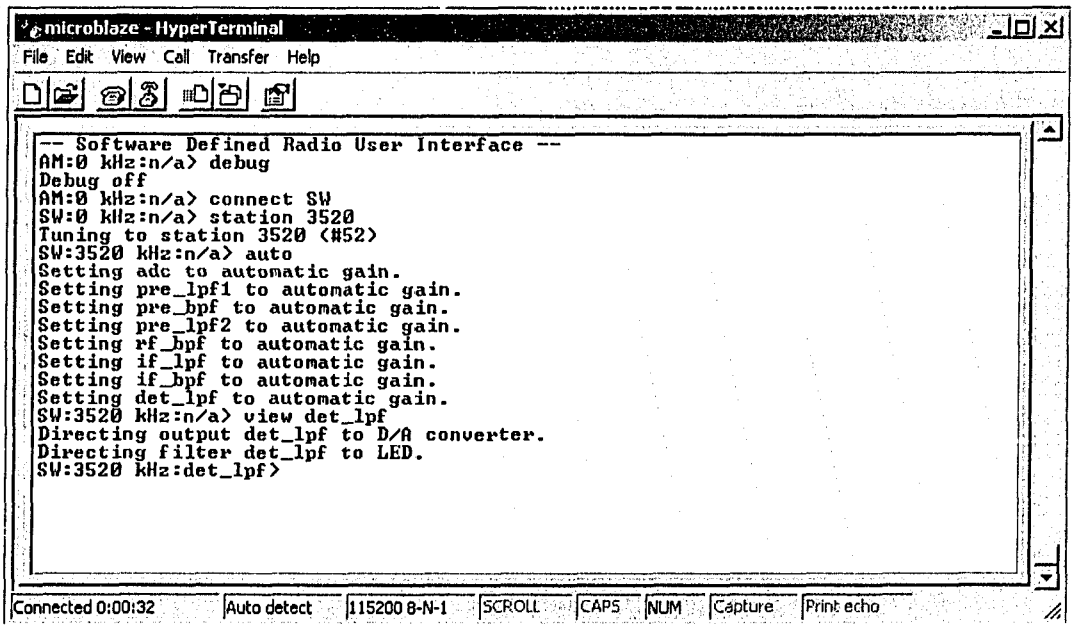

Figure 5.10: Snapshot 1 of the user interface under Hyperterminal.

development process of the SDR receivers.

5.3 System Overview

From an end-user point of view, we created a 100% reconfigurable software defined radio platform that is capable of demodulating almost *any* kind of modulation schemes below 5 MHz. The main advantage of this ideal software defined radio lies in the fact that the end-user can change the modulation scheme by simply loading a new bitstream into the FPGA device *without any hardware modifications.*

As a proof-of-concept design, we designed an AM and a Short Wave radio receiver. With the standalone version of the two receivers, we can switch from one receiver to another by simply reconfiguring the FPGA with the new bitstream. We also implemented both receivers in the same device and achieved simultaneous demodulation of AM and Short Wave signals using the same antenna and low noise amplifier.

This platform can be easily extended to support new modulation schemes. For

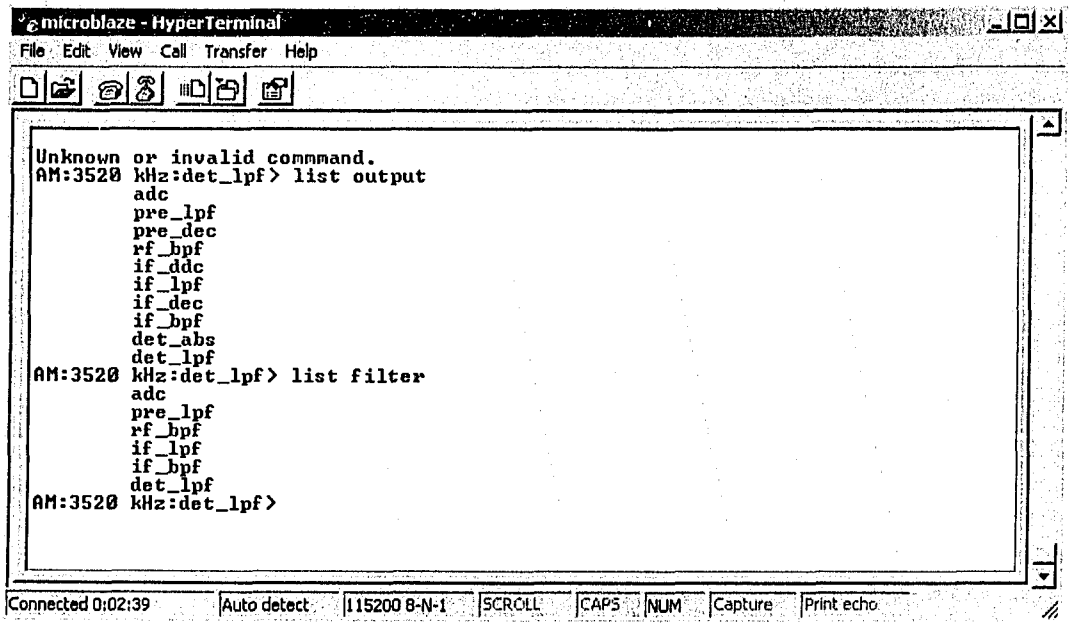

Figure 5.11: Snapshot 2 of the user interface under Hyperterminal.

instance, the datapath of the new modulation scheme can be implemented in HDL and loaded into the FPGA device. The control path of the new modulation scheme can be easily implemented in C and loaded into the Microblaze processor. Finally, the user interface could be written in any high level languages running on the host computer. The communication interface currently used is the serial port but it can be easily modified to use the parallel port, Ethernet or even the USB port. Provided that there are enough hardware resources, a newly created receiver could co-exist with the AM and Short Wave receivers.

With the prototype that we built, we successfully studied the feasibility of building a Tier 3 SDR receiver. This design could be further improved in the future to handle more complex protocols.

Chapter 6

Conclusion and Future Directions

> Where a calculator on the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers in the future may have only 1,000 vacuum tubes and perhaps weigh 1.5 tons.
>
> Anonymous. Popular Mechanics (March 1949)

Integrated circuit technology has progressed a lot since the invention of the first transistor. Several predictions on the future of integrated circuits are indeed quite accurate, while others fail by several orders of magnitude. In spite of this, we will still attempt to forecast the future of ideal software defined radio in light of the what has been accomplished in this work.

6.1 Main Contributions

In this work we introduced the concept of *Ideal Software Defined Radio* and presented, in the author's best knowledge, the *first ever* implementation and integration of Tier 3 Software Defined AM and Short Wave radio receivers. The two receiver prototypes successfully demodulate AM and Short Wave signals in realtime and they are integrated into a single device.

The device utilization of 4-input LUTs is 6% for the AM receiver and 25% for the Short Wave receiver using a Xilinx Virtex-E 2000 FPGA. The AM receiver operates in the frequency range of 540 kHz to 1600 kHz while the Short Wave receiver operates in the lower SW range from 3 MHz to 5 MHz. The output SNR is 50 dB for the AM receiver and 55 dB for the Short Wave receiver. The two receivers use 8 and 9 bits of quantization respectively.

A loop antenna and a low noise amplifier has been constructed as the RF front-tend of the receiver. A pair of computer loud speaker were used as the output device. In terms of control, a soft core microprocessor is used to control the two receivers. The user interface is implemented as a command-line driven program and the two devices worked independently in a single chip without interference.

6.2 Discussion

The main advantage of Tier 3 receivers over Tier 2 receivers is that the RF stage is implemented in the digital domain and it is not tied to any particular modulation scheme. As long as there is enough processing power, virtually *any* kind of modulation scheme can be implemented within the frequency and resolution range of the A/D converter.

The above flexibility does come with some major drawbacks. First of all, a quick glance at some of the Tier 2 prototypes found in recent literature (see Chapter 2) shows that all the Tier 2 prototypes listed in Table 2.1 are able to work with much more complex demodulation schemes. The carrier frequency of these prototypes is usually one or two orders of magnitude higher than the SDR receivers presented in this work. Moreover, these modulation schemes and frequency ranges are much more attractive from a commercial point of view compared with AM and Short Wave modulation schemes.

One of the main reasons why Tier 2 devices can operate in such a high frequency range is because the most computationally intensive operation – the RF stage, is implemented in the analog domain. In such a scheme, the A/D converter can sample at a much lower sampling rate by using the sub-sampling technique or performing the downconversion in the analog domain prior to sampling. Hence, the A/D converter is no longer the main bottleneck as it is in our present Tier 3 architecture.

The second drawback lies in the relatively high power consumption of the SDR receivers presented in this work. Although the prototypes surveyed in Table 2.1 did not report their power consumption, it is still fair to say that a power consumption of 15 Watts is quite high for an AM and Short Wave radio receiver if we consider

that inexpensive portable AM receivers can operate with two AA batteries for many hours! This relatively high power consumption is mainly due to three reasons: 1) the large number of computations that we need to perform in order to bring the RF signal down to baseband if we want to do it entirely in the digital domain; 2) the choice of reconfigurable hardware such as an FPGA device further increases the power consumption since FPGA devices are known to consume more power than its ASIC counterpart; and 3) the overall receiver was optimized for computational resources rather than power. Clearly, we are confronted with a trade-off between power consumption and receiver flexibility.

Although an ideal software radio working as an AM or Short Wave receiver has little immediate commercial interest and/or practical value, this work does serve as a "proof-of-concept" for Tier 3 ideal radios. Our work shows that it is indeed possible to perform all the demodulation and baseband processing entirely in the digital domain. Moreover, all these processing blocks can be implemented in a reconfigurable hardware and modified even after deployment.

This work also provides some guidelines in terms of device utilization, receiver complexity, design challenges, etc. for anyone who is interested in actual implementations of ideal software radios. In fact, a lot of effort has been put in making the receiver as modular and user friendly as possible. This truly creates an interesting development and testing platform for future SDR projects.

As an extension to the above idea, the SDR platform presented in this work is ideal for students who is learning about telecommunication protocols in general and more specifically, superheterodyne receiver architectures. In fact, the SDR receivers have been used as simple lab demonstrations for undergraduate students taking introductory telecommunication classes in several occasions.

6.3 Possible Applications

The main advantage of an ideal software defined radio is the ability to re-use the same hardware for different modulation schemes. This technology could be applied to places where a number of fast-evolving wireless standards have to share the same

RF spectrum. An example of this could be in large metropolitan areas where there is a large monetary incentive to have one 100% reconfigurable base station serve different wireless standards at the same time. Another example is in military applications. Being able to change the wireless protocol on-the-fly could circumvent malicious jamming attempts from the enemy.

The holy grail of Tier 3 SDR receiver would be in the deployment of reconfigurable mobile handsets since this is possibly the largest wireless market in the world. However, *power consumption* will be the main obstacle to overcome before this holy grail can be reached. Many generations of low power FPGA devices *might* make Tier 3 hand-held SDR device a possibility, but it is certainly not achievable in the near future.

A more realistic application of ideal software defined radio could be in amateur radio devices or wireless routers where power consumption is not the most important optimization goal. A universal, generic, reconfigurable transceiver could be built that can adapt to different communication protocols. The change in protocol could be made on-the-fly and become transparent to the end-user. Of course, for this to become a reality, Tier 3 SDR devices should be able to work in the GHz frequency range and be more cost-effective.

6.4 Possible Improvements and Future Work

As ADCs improve in speed and resolution, we could expect to see more ideal software radio implementations in the future. Provided that the processing power of reconfigurable devices continue to increase, commercially-off-the-shelf ADC converters sampling at 500 MSPS should be fast enough for the implementation of FM radio receivers. The implementation of more interesting receivers should be technically feasible as long as both ADC and DSP engines continue to increase their sampling and processing speed.

In this work we did not consider the case of hardware resource sharing among different receivers. Future work in this area could concentrate on this issue. For example, the AM and the Short Wave receiver could share the same IF and Detector

stages if proper time-sharing control signals are implemented. Moreover, it would be interesting to see some sort of *partial reconfiguration* of FPGA devices. This kind of receivers would be able to adapt their modulation scheme *on-demand* while other functionalities in the receiver remain unchanged.

The proposed architecture could be further optimized in terms of hardware resources. In theory, we are not limited to using only one or two intermediate frequencies. We could have a *double* or even *triple* superheterodyne architecture. In this scenario, the incoming RF signal is first downconverted to $IF1$, then to $IF2$, then to $IF3$ and so on until it is finally demodulated to the desired baseband. This could potentially lead to better overall hardware resource utilization.

Mature ideal software defined radio receivers that can be used in commercial products are still far from being a reality. However, just as computers were able to shrink from a 30-ton ENIAC machine to the state-of-the-art laptop weighing only few pounds, it may not be so unrealistic to imagine a world where all radios are ideal software radios. *Hopefully, the present work is a tiny step toward this goal.*

NO TEXT

Bibliography

[1] J. Mitola, "The software radio architecture," *IEEE Commun. Mag.*, pp. 26–37, May 1995.

[2] H. W. Tuttlebee, "Software-defined radio: Facets of a developing technology," *IEEE Personal Commun. Mag.*, pp. 38–44, April 1999.

[3] (2004, February) Definition and semantics. Software Defined Radio Forum. [Online]. Available: http://www.sdrforum.org/tech_comm/definitions.html

[4] E. Buracchini, "The software radio concept," *IEEE Commun. Mag.*, pp. 138–143, Sept 2000.

[5] N. Nakajima, "Research and developments of software-defined radio technologies in Japan," *IEEE Commun. Mag.*, pp. 146–155, Aug 2001.

[6] H. Yoshida, H. Tsurumi, and Y. Suzuki., "Broadband RF front-end and software execution procedure in software-defined radio," *IEEE Trans. Veh. Technol.*, pp. 2133–2137, 1999.

[7] J. R. MacLeod, T. Nesimoglu, M. A. Beach, and P. A. Warr, "Enabling technologies for software defined radio transceivers," *MILCOM 2002. Proceedings*, vol. 1, pp. 354–358, Oct 2002.

[8] D. M. Pearson, "SDR Systems Defined Radio: How do we get there from here," *IEEE Military Communications Conference*, pp. 571–575, 2001.

[9] D. Greifendorf, J. Stammen, S. Sappok, M. van Ackeren, and P. Jung, "A novel hardware design paradigm for mobile 'software defined radio' terminals," *IEEE* 7^{th} *Symposium on Spread-Spectrum Technology & Applications*, pp. 828–831, 2002.

[10] L. Zhigang, L. Wei, Z. Yan, and G. Wei, "A multi-standard SDR base band platform," *IEEE International Conference on Computer Networks and Mobile Computing*, pp. 461–464, 2003.

[11] S. Srikanteswara, R. Palat, J. Reed, and P. Athanas, "An overview of configurable computing machines for software radio handsets," *IEEE Commun. Mag.*, pp. 134–141, 2003.

[12] J. R. Hauser and J. Wawrzynek, "Garp: a MIPS processor with a reconfigurable coprocessor," *Proc. 5th Annual IEEE Symp. FPGAs for Custom Comp. Machines*, pp. 12–21, April 1997.

[13] J. Brakensiek *et al.*, "Software radio approach for reconfigurable multi-

standard radios," *13th IEEE Int'l. Symp. Pers. Indoor and Mobile Radio Commun.*, pp. 110–114, 2002.

[14] P. M. Heysters *et al.*, "A reconfigurable function array architecture for 3G and 4G wireless terminals," *Proc. 2002 World Wireless Cong.*, pp. 399–404, May 2002.

[15] M. B. Taylor *et al.*, "The raw microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE Micro*, pp. 25–35, Mar 2002.

[16] S. Srikanteswara *et al.*, "Soft radio implementations for 3G and future high data rate systems," *GLOBECOMM*, Nov 2001.

[17] A. Haghighat, "A review on essentials and technical challenges of software defined radio," *MILCOM Proceedings*, vol. 1, pp. 377–382, Oct 2002.

[18] K. C. Gupta, J. Li, R. Ramadoss, C. Wang, Y. C. Lee, and V. M. Brightl, "MEMS-based reconfigurable slot antennas," *IEEE ACCRS2000 Conference*, 2000.

[19] K. C. Gupta, J. Li, R. Ramadoss, C. Wang, Y. C. Lee, and V. M. Brightl, "Design of frequency-reconfigurable rectangular slot ring antennas," *IEEE APS2000 International Symposium*, 2000.

[20] K. Wang and C. Nguyen, "High-order micromechanical electron filter," *IEEE International Micro Electromechanical Systems Workshop*, Jan 1997.

[21] R. J. Richards and H. J. De Los Santos, "Mems for RF/microwave wireless applications: The next wave part I & II," *Microwave Journal*, Mar 2001.

[22] "AD12500 – 12-bit, 500 MSPS A/D converter," Analog Devices, Norwood, MA.

[23] "2 GHz PMC ADC digitizer board," Delphi Engineering Group, Inc., Costa Mesa, CA.

[24] (2005, Mar) 6 bit 6 GS/s analog to digital converter. Rockwell Scientific. [Online]. Available: http://www.rockwellscientific.com/highspeed/adc.html

[25] O. Mukhanov, V. K. Semenov, I. V. Vernik, A. M. Kadin, T. V. Filippov, D. Gupta, D.K.Brock, I. Rochwarger, and Y. A. Polyakov, "High-resolution ADC operation up to 19.6 GHz clock frequency," *Superconductor Science and Technology*, pp. 1065–1070, 2001.

[26] L. Pucker, "Paving paths to software radio design," *Communication System Design Magazine*, June 2001.

[27] H. R. Karimi, N. W. Anderson, and P. McAndrew, "Digital signal processing aspects of software definable radios," *IEE Colloquium on Adaptable and Multistandard Mobile Radio Terminals*, pp. 3/1–3/8, 1998.

[28] L. B. Michael, M. J. Mihaljevic, and S. Haruyama, "A framework for secure download for software-defined radio," *IEEE Commun. Mag.*, pp. 88–96, 2002.

[29] J. Witkowsky and G.-J. van Rooyen, "A hardware emulator testbed for soft-

ware defined radio," *IEEE Africon Conference in Africa*, pp. 383–388, 2002.

[30] Y.-D. Chuang, Y.-S. Chang, R.-H. Wu, S.-C. Lin, and S.-M. Yuan., "SDRsim: A PC-based simulator of software defined radio," *IEEE International Conference on Telecommunications*, pp. 1251–1258, 2003.

[31] J. K. Hwang, "Innovative communication design lab based on PC sound card and Matlab: A software-defined-radio OFDM modem example," *IEEE Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. III–761–4, 2003.

[32] (2004, November) The GNU software radio. Free Software Foundation, Inc. [Online]. Available: http://www.gnuradio.org

[33] K. Araki, "Prehistory of the SDR studies in Japan – a role of ARIB study group," *IEICE Trans. Commun.*, no. 6, pp. 1183–1188, Jun 2000.

[34] T. Yokoi *et al.*, "Software receiver technology and its applications," *IEICE Trans. Commun.*, no. 6, pp. 1200–1209, Jun 2000.

[35] H. Harada, Y. Kamio, and M. Fujise, "Multimode software radio system by parameter controlled and telecommunication component block embedded digital signal processing," *IEICE Trans. Commun.*, no. 6, pp. 1217–1228, Jun 2000.

[36] R. Kohno *et al.*, "Universal platform for software defined radio," *IEEE Int'nl. Symp. Intelligent Sig. Processing and Commun. Sys.*, pp. 523–526, Nov 2000.

[37] Y. Suzuki *et al.*, "Universal radio base and personal station prototypes," *IEICE Trans. Commun.*, no. 6, pp. 1261–1268, Jun 2000.

[38] H. Tsurumi *et al.*, "Broadband and flexible receiver architecture for software defined radio terminal using direct conversion and low-if principle," *IEICE Trans. Commun.*, no. 6, pp. 1246–1253, Jun 2000.

[39] J. Glossner, D. Iancu, J. Lu, E. Hokenek, and M. Moudgill, "A software-defined communications baseband design," *IEEE Commun. Mag.*, pp. 120–128, Jan 2003.

[40] "GV–290 hardware accelerator manual," GV&Associates, Inc., Ramona, CA.

[41] P. H. Young, *Electronic Communication Techniques*. Merrill Publisher, 1990, pp. 158–160.

[42] B. P. Lathi, *Modern Digital and Analog Communications Systems*, 3rd ed. Oxford University Press, 1998, pp. 189–190.

[43] R. Yates, "Practical considerations in fixed-point FIR filter implementations," Digital Sound Labs, Tech. Rep., 2000. [Online]. Available: http://www.s3.kth.se/signal/edu/projekt/DSPsupport/documents/practical_%fp.pdf

[44] "Distributed arithmetic FIR filter v8.0," Xilinx, Inc., San Jose, CA, Mar 2003.

[45] "Direct digital synthesizer v4.2," Xilinx, Inc., San Jose, CA, Mar 2003.

[46] "Digital down converter v1.0," Xilinx, Inc., San Jose, CA, Mar 2002.

[47] D. A. Johns and K. Martin, *Analog Integrated Circuit Design.* John Wiley & Sons, Inc., 1997, p. 536.

[48] "Agilent Technologies ESG Vector Signal Generator programming guide," Agilent Technologies, Palo Alto, CA, pp. 158–159, May 2004.

[49] (2005, February) AM loop antennas. [Online]. Available: http://www.mindspring.com/~loop_antenna/

[50] (2005, February) Dave's radio loops. [Online]. Available: http://www.schmarder.com/radios/misc-stuff/loops.htm

[51] (2005, February) Active antenna with gain. [Online]. Available: http://www.designnotes.com/CIRCUITS/antenna.htm

[52] (2005, February) GPL'd suite of electronic design automation tools. gEDA. [Online]. Available: http://www.geda.seul.org/

[53] (2005, February) Ngspice the free* circuit simulator. gEDA. [Online]. Available: http://ngspice.sourceforge.net/

[54] "AD9762 – 12-bit, 100 MSPS A/D converter," Analog Devices, Norwood, MA.

[55] "Getting started with EDK," Xilinx, Inc., San Jose, CA, Aug 2004.

[56] (2005, March) Minicom. Debian. [Online]. Available: http://alioth.debian.org/projects/minicom/

Appendix A

Microblaze Processor

In this section we provide a brief discussion of the Xilinx Microblaze microprocessor and how it is used to control the software receivers discussed so far. Detailed description of this processor is beyond the scope of this work and the reader is referred to [55] for more information.

In order to write to the registers inside the main controllers of each SDR receiver, we use a Microblaze processor implemented inside the DP FPGA. This microprocessor communicates with the SDR receivers through a dedicated 27 bits wide bus as shown in Figure 5.9. On the other hand, the user controls this microprocessor through an RS232 serial port and the JTAG port as shown in Figure 5.8

The 32-bit soft core RISC microprocessor has a Harvard architecture where separate buses are used for data and instructions. Internally, the Microblaze processor is attached to different peripherals through five different buses as shown in Figure A.1. This set of peripherals are reconfigurable through the Embedded Development Kit (EDK) provided by Xilinx.

The five buses shown in Figure A.1 are:

- ILMB: Instruction Local Memory Bus. This bus is used to fetch instruction located in the on-chip block RAM.
- DLMB: Data Local Memory Bus. This bus is used to read and write data in the on-chip block RAM.
- FSL: Fast Simplex Link. This optional bus is used to download software into the microprocessor.

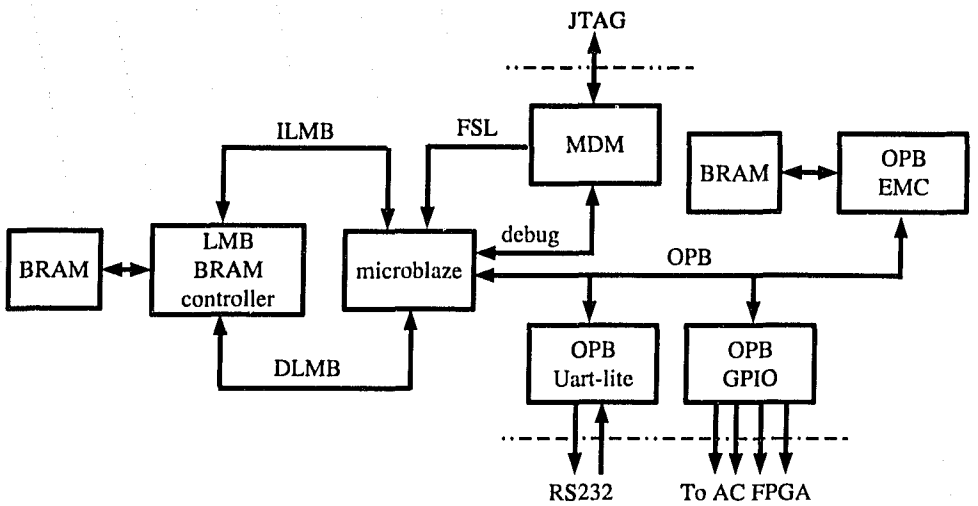

Figure A.1: Microblaze processor peripheral configuration.

- debug: This bus is used to debug the microprocessor.
- OPB: On-chip Peripheral Bus. Several peripherals are attached to this bus and it is used by the processor to communicate with the outside world.

The peripherals attached to the processors are:

- MDM: Microblaze Debug Module. This module is used to debug the microprocessor. It can read/write into individual registers, step instructions, reset the processor, etc. The user connects to this module through a JTAG cable.
- EMC: External Memory Controller. This peripheral provides the necessary connection to external memories such as SRAM or DRAM interfaces. In our case, we used 65 Kb of internal block RAM as the main memory interface.
- GPIO: General Purpose Input/Output. This peripheral controls generic input/output signals. In this case, we use it to control the bus interface between this processor and the SDR receivers.
- Uart-lite. This peripheral provides the bridge between the OPB bus and the RS232 protocol. This peripheral is designed as the standard input and output of the processor.

Table A.1: Microblaze peripheral memory mapping.

Peripheral	Bus	Base Address	High Address	Size (bytes)
debug module	OPB	0x80400000	0x804000ff	256
instruction	LMB	0x00000000	0x00003fff	16384
data	LMB	0x00000000	0x00003fff	16384
EMC	OPB	0x80700000	0x8070ffff	65536
LEDs	OPB	0x80500000	0x805001ff	512
Uart	OPB	0x80800000	0x808000ff	256
main controller bus	OPB	0x80900000	0x80900ff	256

Each of the peripherals shown above are memory mapped to the processor. The bus type, base address, high address and size of this peripheral-to-memory mapping are shown in Table A.1.

NO TEXT

Appendix B

Main Controller Register Mapping

B.1 AM Main Controller

The AM main controller exposes the registers shown in Table B.1. As we can see, the AM receiver has a unique ID = 0x00 assigned to it.

For instance, if we wish to set the gain setting of the LPF in the prefilter stage, we should write the new gain value into the register at address 0x01. To change

Table B.1: Register mapping of the AM main controller.

Receiver ID	0x00	
Register Address	**Register Name**	**Register Width**
0x00	ADC gain	4
0x01	Prefilter LPF gain	4
0x02	RF BPF gain	4
0x03	IF LPF gain	4
0x04	IF BPF gain	4
0x05	Detector LPF gain	4
0x06	Output MUX select	4
0x07	AM carrier frequency	7
0x08	LED MUX select	3
0x09	ADC auto gain	1
0x0A	Prefilter LPF auto gain	1
0x0B	RF BPF auto gain	1
0x0C	IF LPF auto gain	1
0x0D	IF BPF auto gain	1
0x0E	Detector LPF auto gain	1

Table B.2: AM main controller output multiplexer input/output mapping.

MUX Select	Input
0x00	ADC
0x01	Prefilter LPF
0x02	Prefilter Decimator
0x03	RF BPF
0x04	IF DDC
0x05	IF LPF
0x06	IF Decimator
0x07	IF BPF
0x08	Detector ABS
0x09	Detector LPF

Table B.3: AM main controller LED multiplexer input/output mapping.

MUX Select	Input
0x00	ADC gain
0x01	Prefilter LPF gain
0x02	RF BPF gain
0x03	IF BPF gain
0x04	IF BPF gain
0x05	Detector LPF gain

the carrier frequency, we need to update the register `AM carrier frequency` located at address 0x07.

In order to view the output at intermediate stages of the processing, the user should write to the `Output MUX select` register located at address 0x06. This configures a 10-to-1 output bus multiplexer with a input/output mapping as shown in Table B.2. The chosen intermediate signal is then routed to the output DAC.

To view the gain setting of each gain control blocks, we need to change the value in the `LED MUX select` register located at address 0x08. This register provides the `select` signal of a 6-to-1 bus multiplexer with an input/output mapping shown in Table B.3. The chosen filter gain value is routed to a set of LEDs available in the FPGA board.

Table B.4: Register mapping of the SW main controller.

Receiver ID	0x01	
Register Address	**Register Name**	**Register Width**
0x00	ADC gain	4
0x01	Prefilter LPF 1 gain	4
0x02	Prefilter BPF gain	4
0x03	Prefilter LPF 2 gain	4
0x04	RF BPF gain	4
0x05	IF LPF gain	4
0x06	IF BPF gain	4
0x07	Detector LPF gain	4
0x08	Output MUX select	4
0x09	SW carrier frequency	8
0x0A	LED MUX select	3
0x0B	ADC auto gain	1
0x0C	Prefilter LPF 1 auto gain	1
0x0D	Prefilter BPF auto gain	1
0x0E	Prefilter LPF 2 auto gain	1
0x0F	RF BPF auto gain	1
0x10	IF LPF auto gain	1
0x11	IF BPF auto gain	1
0x12	Detector LPF auto gain	1

B.2 Short Wave Main Controller

The main controller of the Short Wave receiver is essentially the same as the AM receiver. The only difference is that the Short Wave main controller has to control more filters and hence exposes more write-only registers to the outside world. The register mapping of the SW receiver is shown in Table B.4. The Short Wave receiver has been assigned an unique ID of 0x01.

The output multiplexer is controlled by the `Output MUX select` register located at address 0x08. This configures a 14-to-1 output bus multiplexer with a input/output mapping as shown in Table B.5.

To view the gain setting of each gain control blocks, we need to change the value in the `LED MUX select` register located at address 0x0A. This in turn

Table B.5: SW main controller output multiplexer input/output mapping.

MUX Select	Input
0x00	ADC
0x01	Prefilter LPF 1
0x02	Prefilter Decimator 1
0x03	Prefilter BPF
0x04	Prefilter DDC
0x05	Prefilter LPF 2
0x06	Prefilter Decimator 2
0x07	RF BPF
0x08	IF DDC
0x09	IF LPF
0x0A	IF Decimator
0x0B	IF BPF
0x0C	Detector ABS
0x0D	Detector LPF

Table B.6: SW main controller LED multiplexer input/output mapping.

MUX Select	Input
0x00	ADC gain
0x01	Prefilter LPF 1 gain
0x02	Prefilter BPF gain
0x03	Prefilter LPF 2 gain
0x04	RF BPF gain
0x05	IF LPF
0x06	IF BPF
0x07	Detector LPF

will provide the `select` signal of an 8-to-1 bus multiplexer with a input/output mapping shown in Table B.6.

Appendix C

User Commands

The user interface that runs inside the microprocessor is written in C language and compiled for the Microblaze processor. The program implements a command-line driven interface. The supported commands and their respectively description are shown in Table C.1.

Table C.1: Supported commands in the user interface.

Command Usage	Description
`help`	The `help` command provides an explanation for each supported command explained below.
`list`	The `list` command displays the name of the supported receivers. Currently this command will output `AM` and `SW`.
`connect` receiver name	The `connect` command connects to the given receiver. Currently the receiver name can be either `AM` or `SW`. The program will remember what is the current receiver so that subsequent command will be directed to the intended receiver.
`station` frequency	This command tunes the current receiver to the given *frequency* in kilohertz.
`list` output | filter	The `list` command takes one argument that is either `output` or `filter`. The former argument will list all the available outputs for the currently chosen receiver. The latter argument will list all the available filters in the currently chosen receiver.
`set` filter_name gain	This command takes two arguments, the filter name and the gain value. The filter name must be one of the names listed by the command `list filter` shown above. The gain is either an integer denoting the number of LSB to drop or the word *auto* to set this filter in automatic gain mode.
`view` output_name	This command will direct the given output to the DAC. The name of the output must be one listed by the command `list output` described above.
`led` filter_name	This command will show the gain setting of the chosen filter to the output LEDs.
`auto`	The `auto` command will set *all* the filters in the currently connected receiver into automatic gain mode.
`debug`	This command enables or disables the debug mode. In debug mode, most of the commands described above will output extra debug information.

Appendix D

System Setup

To summarize the steps needed to setup the two software receivers, a flow diagram is shown in Figure D.1. From this figure we can see that three programs are used in sequence in order to initialize the receivers.

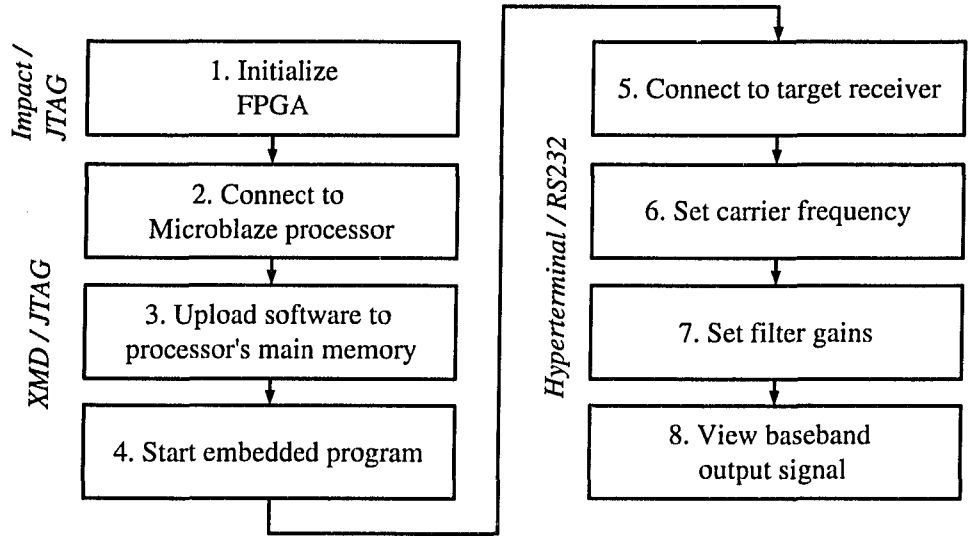

Figure D.1: Steps needed to setup the two software receivers.

First of all, the user needs to initialize all the FPGA devices present on the board. This is accomplished with the program *Impact* provided by Xilinx and the FPGA bitstream is uploaded through the JTAG interface (step 1). Secondly, the user needs to invoke *Xilinx Microblaze Debugger* (XMD) tool in order to connect to the processor. It is necessary to upload the software to the processor's main memory

because our microprocessor uses a volatile main memory. The software inside the microprocessor can then be started with the debugger (steps 2-4). All these communication between the debugger and the microprocessor are done through the JTAG port. Finally, we can control the two radio receivers through the RS232 interface (steps 5-8).