

Jaypee University of Information Technology  
Waknaghat, Distt. Solan (H.P.)

## Learning Resource Center

CLASS NUM:

BOOK NUM.:

ACCESSION NO.: SP08101 / SP0812102

This book was issued is overdue due on the date stamped below. If the book is kept over due, a fine will be charged as per the library rules.

Due Date	Due Date	Due Date
12 NOV 2014		



# DIGITAL IMPLEMENTATION OF SOFTWARE DEFINED RADIO SUBSYSTEM

Project Report submitted in partial fulfillment of the requirement for the  
degree of

Bachelor of Technology.

in

**Electronics and Communication Engineering**

under the Supervision of

*Dr. Shruti Jain*

By

*Jatin Pahadia (081038)*

*Kshitiz Vaish (081104)*

*Nalin Yadav (081120)*

to



Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh



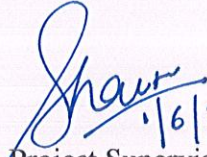


**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY**  
**WAKNAGHAT**  
**SOLAN, HIMACHAL PRADESH**

**Date: 17 May 2012**

**CERTIFICATE**

This is to certify that the work entitled “**DIGITAL IMPLEMENTATION OF RADIO RECEIVER**”, submitted by **Jatin Pahadia, Kshitiz Vaish and Nalin Yadav** in partial fulfillment for award of degree of Bachelor of Technology (Electronics and Communication Engineering) of Jaypee University of Information Technology has been carried out in my supervision. This work has not been submitted partially or wholly to any other university or institution for award of this or any other degree programme.

  
16/5/2012  
Project Supervisor

**Dr. Shruti Jain**  
**Sr. Lecturer (ECE), JUIT**



# Acknowledgement

We are highly grateful to **Prof.(Dr.) Sunil Bhooshan**, Head of Dept. Electronics and Communication, for providing us the opportunity to work on the project.

With great pleasure, we express our gratefulness to our guide and mentor, **Dr. Shruti Jain**, Senior Lecturer, Dept. of Electronics and Communication, for her valuable and sustained guidance and careful supervision during the project.

We express our sincere thanks to all the faculty members of JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY who have helped us directly or indirectly. Without their help and guidance we would not have been able to successfully complete our project.

## Group Members:

Jatin Pahadia (081038) *Jatin*

Kshitiz Vaish (081104) *Kshitiz*

Nalin Yadav (081120) *Nalin Yadav*



# TABLE OF CONTENTS

<b>Certificate</b>	ii
<b>Acknowledgement</b>	iii
<b>Abbreviations</b>	vii
<b>List of figures</b>	viii
<b>List of tables</b>	x
<b>Abstract</b>	xi
<b>Chapter 1 – Introduction</b>	
1.1 Objective	01
1.2 Overview	02
1.3 Software Defined Radio(SDR)	04
1.3.1 Architecture Of SDR	05
1.3.2 Features of SDR	07
1.3.3 Advantages of SDR	08
<b>Chapter 2 – AM/FM Receiver</b>	
2.1 Block Diagram	09
2.1.1 Anti-aliasing filter	10
2.1.2 Analog to digital Converter	11
2.1.3 Digital Local Oscillator	11
2.1.4 Digital Mixer	11
2.1.5 Digital Low Pass Filter	12
2.1.6 Digital Demodulator	14
2.1.6.1 AM Digital Demodulator	14
2.1.6.2 FM Digital Demodulator	16
2.1.6.2.1 FM Phase locked Demodulator	17
<b>Chapter 3 – Softwares used for AM/FM Receiver</b>	
3.1 MATLAB 7.6.0	20
3.1.1 Simulink 7.1	21
3.2 XILINX ISE System Generator 11.1	24
3.2.1 Field Programmable Gate Array	25



<b>Chapter 4 – Implementation of AM/FM Receiver</b>	
4.1 AM Receiver	28
4.1.1 Implementation of AM-LPF	28
4.1.1.1 Finite Impulse Response(FIR) filter	29
4.1.1.2 Digital Filter Design Process	30
4.1.1.3 Filter Specifications	31
4.1.1.4 Multistage filtering technique	32
4.1.1.5 Calculation of filter order	36
A) Implementation of AM-LPF on Simulink	37
B) Implementation of AM-LPF on XILINX System Generator	38
4.1.2 Implementation of AM Receiver	40
A) Using MATLAB Simulink	40
B) Using XILINX ISE System Generator	41
4.2 FM Receiver	45
4.2.1 Implementation of FM-LPF	45
4.2.1.1 Undersampling Technique	45
4.2.1.2 Filter Specifications	46
4.2.1.3 Calculation of filter order	47
A) Implementation of FM-LPF on Simulink	48
B) Implementation of FM-LPF on XILINX System Generator	49
4.2.3 Implementation Of Voltage Controlled Oscillator(VCO)	51
4.2.4 Implementation of FM Receiver	51
A) Using MATLAB Simulink	51
B) Using XILINX ISE System Generator	52
4.3 Implementation on FPGA	54
4.3.1 The System Generator Design Flow	54



4.3.2 System Testing in FPGA	55
4.3.2.1 Using XILINX System Generator Token tool	56
4.3.2.2 Using XILINX 11.1 Project Navigator	58
4.3.3 XILINX Implementation tools	58
<b>Chapter 5 – Conclusions</b>	<b>60</b>
<b>References</b>	<b>61</b>



## List of Acronyms

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
CDMA	Code Division Multiple Access
DAC	Digital to Analog Converter
DDC	Digital Down-converter
DSP	Digital signal Processor
DUC	Digital Up-converter
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
GSM	Global System for Mobile Communication
IIR	Infinite Impulse Response
LPF	Low Pass Filter
MAC	Multiply and Accumulate
PDR	Programmable Digital Radio
SDR	Software Defined Radio
TDMA	Time Division Multiple Access



# LIST OF FIGURES

Figure 1	Block diagram of SDR	5
Figure 2	System overview of digital radio receiver	9
Figure 3	Aliasing	10
Figure 4	Anti-aliasing filter	10
Figure 5	Input and output waveforms of A/D converter	11
Figure 6	Effect of Mixing	12
Figure 7	Effect of Filtering	13
Figure 8	AM spectrum	14
Figure 9	AM demodulator	15
Figure 10	Block diagram of Phase-locked loop FM demodulator	17
Figure 11	Phase comparator structure	18
Figure 12	Linearized PLL model	18
Figure 13	Schematic view of FPGA	27
Figure 14	Block diagram of FIR filter	28
Figure 15	Block diagram of IIR filter	28
Figure 16	Functional block diagram for an $N^{\text{th}}$ order FIR filter	28
Figure 17	Design flow for digital filters	30
Figure 18	Filter specifications	31
Figure 19	Multi-stage filtering	33
Figure 20	Simulink model of Low pass filter	37
Figure 21	9-stage low pass filter	37
Figure 22	Magnitude response of 9-stage AM low pass filter	38
Figure 23	XILINX System Generator model of 9-stage AM-LPF	38
Figure 24	Output of spectrum scope for a frequency of 3 kHz	39
Figure 25	Output of time scope for a frequency of 3 kHz	39



Figure 26	Implementation of AM Receiver on MATLAB Simulink	40
Figure 27	Implementation of AM Receiver using XILINX ISE System Generator	41
Figure 28	Implementation of DLO using SILINX ISE System Generator	42
Figure 29	Input Frequency spectrum for AM Receiver	43
Figure 30	Output Frequency spectrum of AM Receiver	43
Figure 31	Input and Output waveforms of AM Receiver	44
Figure 32	Under-sampling FM signal between DC and 20 MHz	45
Figure 33	Simulink model of FM-LPF	48
Figure 34	8-stage FM low pass filter	48
Figure 35	Magnitude response of FM low pass filter	49
Figure 36	XILINX system Generator model of 8-stage FM low pass filter	49
Figure 37	Output of spectrum scope for a frequency of 30 kHz	50
Figure 38	Output of time scope for a sine wave of frequency 30 kHz	50
Figure 39	Simulink model of VCO	51
Figure 40	Implementation of FM receiver on MATLAB Simulink	51
Figure 41	XILINX ISE System Generator implementation of FM receiver	52
Figure 42	Output of FM receiver for a sine wave of frequency 30 kHz	52
Figure 43	Input and output waveforms of FM receiver for a sine wave of 30 kHz	53
Figure 44	System Generator Design Flow	55
Figure 45	Selecting compilation mode and target part	57
Figure 46	HDL code generation	57
Figure 47	Project navigator window	58
Figure 48	Impact window	59
Figure 49	Device successfully programmed	59



## LIST OF TABLES

Table 1	Advantages of SDR	8
Table 2	AM filter specifications	28
Table 3	AM filter order	29
Table 4	N-stage filter design	30
Table 5	9-stage filter for AM digital receiver	33
Table 6	Filter specification for FM-LPF	43
Table 7	Filter order for FM-LPF	44
Table 8	Calculation of filter order for FM	44



## Abstract

Recent advancement in semiconductor technology has allowed traditional analog radio systems to be realized with today's digital circuit technology. Software Defined Radio (SDR) is a new digital technology wherein radio functions are implemented by signal- processing software running on generic hardware platforms. This makes the system very flexible and adaptive, and thus promises to solve the many problems faced by traditional hardware-based radio systems. Unlike traditional analog receivers, (which are built from analog components, digital receivers convert analog signal into digital representation, and employ digital signal processing techniques to process the digital data for extracting the desired information ) in SDR we use MATLAB-Simulink and Xilinx system generator to implement design on FPGA

MATLAB is a programming environment for algorithm development, data analysis, visualization, and numerical computation. Using MATLAB, we can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

Simulink is integrated with MATLAB, providing immediate access to an extensive range of tools that let you develop algorithms, analyze and visualize simulations, create batch processing scripts, customize the modeling environment, and define signal, parameter, and test data.

Xilinx System Generator is a system-level modeling tool that facilitates FPGA hardware design. It extends Simulink in many ways to provide a modeling environment that is well suited to hardware design. The tool provides high-level abstractions that are automatically compiled into an FPGA at the push of a button.

Current market drivers such as future-proof equipment, seamless integration of new services, multi-mode equipment and over-the-air feature insertion in commercial wireless networking industry have resulted in widespread interest in SDR technology.

In a nutshell, SDR is a promising technology that facilitates development of multi-band, multi-service, multi-standard, multi-feature consumer handsets and future-proof network infrastructure equipment.



# CHAPTER 1

## Introduction

### **1. 1 Objective**

The aim of this project is to develop an AM/FM Digital Radio Receiver using MATLAB-Simulink and Xilinx system generator to implement design on Field Programmable Gate Array (FPGA) as the hardware platform. Unlike traditional analog receivers, which are built from analog components, digital receivers convert analog signal into digital representation, and employ digital signal processing techniques to process the digital data for extracting the desired information. The extracted digital information (a radio channel in this case) is then converted back to analog format for listening purposes.

In developing the system, we aimed to investigate different techniques for designing digital radio receivers, from which suitable techniques would be chosen for implementing the AM/FM radio receiver. Several design models created in Simulink for implementing the receiver in FPGA would be developed and simulated, giving insight analysis into their performance as well as their efficiency in term of hardware usage.

Finally, these models would be translated into software (Hardware Description Language) used for synthesizing the system into a FPGA platform. The ultimate goal of this project will be having a fully working AM/FM Digital Receiver System residing in a FPGA board allowing users to tune in and listen to any of the available AM or FM channels.



## **1.2 Overview**

This project exemplifies to some extent the concepts, design and implementation of a broader area of digital technology called Software Defined Radio (SDR). This section presents a broad overview of Software Defined Radio including its background, applications, and a comparison of commonly used hardware platforms for implementing SDR.

Traditional analog radio receivers and transmitters consist of dedicated analog circuits for filtering, tuning, and demodulating/modulating a specific type of waveform. These hardware based radio systems are inflexible and hard to modify if changes are to be made to its fundamental characteristics such as demodulation/modulation types. To make the system more flexible, SDR technology facilitates implementation of some of these functions in software. This results in reconfigurable software radio systems where changes to its fundamental characteristics can be made simply by modifying the software, whereas a complete hardware based radio system would require hardware modification in order to change these parameters.

SDR technology has potential applications in all areas of radio communications and broadcasting. To highlight the significance of SDR technology, consider the following problems faced by the wireless communications industry due to implementation of network infrastructure and terminals completely in hardware:

- The constant evolution of wireless network standards from 2G to 3G and further to 4G causes problems for subscribers, network operators and equipment vendors. Users are forced to buy new handsets every time a new generation is deployed. Network operators face with the problem of migrating from one generation to another due to large number of subscribers using legacy handsets that may be incompatible with newer generation network.
- The air interface and link-layer protocols differ across different continents, for example, European wireless networks are mainly GSM/TDMA while US networks are predominantly CDMA. This problem has inhibited the deployment of global roaming facilities and thus causes inconvenience to subscribers who travel frequently between continents.



SDR technology promises to solve these problems as it enables implementation of radio functions in networking infrastructure and subscriber terminals as software modules running on generic hardware platforms. This relieves the cost and complexity of migrating network from one generation to another since the migration would only involve a software upgrade.

Furthermore, since radio functions are implemented in software, multiples software modules implementing different standards can co-exist in the equipment and handsets. An appropriate software module can be chosen to run (either by user or by the network) depending on the network requirements. This allows building of multi-mode handsets and equipment resulting in universal connectivity irrespective of the underlying network technology used.

As mentioned above, in Software Defined Radio technology, radio functions are implemented in software running on digital signal processing hardware platforms. The three most commonly used platforms are DSPs (Digital Signal Processor), ASICs (Application Specific Integrated Circuit), and FPGAs (Field Programmable Logic Array).

Digital Signal Processor does signal processing by fetching instructions and data from memory, does operations, and stores the results back to memory, just like a regular CPU. The difference between a DSP chip and a CPU chip is that a DSP chip usually has a block that does high-speed signal processing, especially a block called MAC (Multiply and Accumulate). By calling different routines in memory, a DSP chip can be reconfigured to perform various functions.

ASIC (Application-specific Integrated Circuit) is an integrated circuit that is designed to perform a fixed specific task. Examples of signal-processing specific ASIC's are DDC (digital down converter) chip, and digital filter chips. The disadvantage of ASIC is that its functionalities are fixed and thus cannot be changed by the user.

FPGA (Field Programmable Gate Array) is able to perform any task by mapping the task to the hardware. One of the advantages of FPGA is its reconfigurability capability that ASIC does not have. Reconfigurability is a feature, which enables



FPGA to realize any user hardware by changing the configuration data on a chip as many times as needed.

In summary, with its many advantages, FPGA has become key components in implementing high performance digital signal processing systems. In this project, we will be using FPGA as the targeted hardware platform for implementing an AM/FM Digital Radio Receiver.

### **1.3 Software defined radio (SDR)**

Software-Defined Radio (SDR) defines SDR technology as "radios that provide software control of a variety of modulation techniques, wide-band or narrow-band operation, communications security functions (such as hopping), and waveform requirements of current & evolving standards over a broad frequency range."

There are numerous definitions of Software Defined Radio in existence, all of which are not totally consistent with each other. The Federal Communications Commission (FCC) defines SDR as a "generation of radio equipment that can be reprogrammed quickly to transmit and receive on any frequency within a wide range of frequencies, using virtually any transmission format and any set of standards"

In contrary, the SDR Forum, as an international, non-profit organization promoting the development of SDR, offers a broader definition: "Software defined radio is a collection of hardware and software technologies that enable reconfigurable systems architectures for wireless networks and user terminals". One reason for the advent of several inconsistent definitions is probably due to the broad and complex nature of technology itself, and the variety of possible means for implementation of SDR systems.

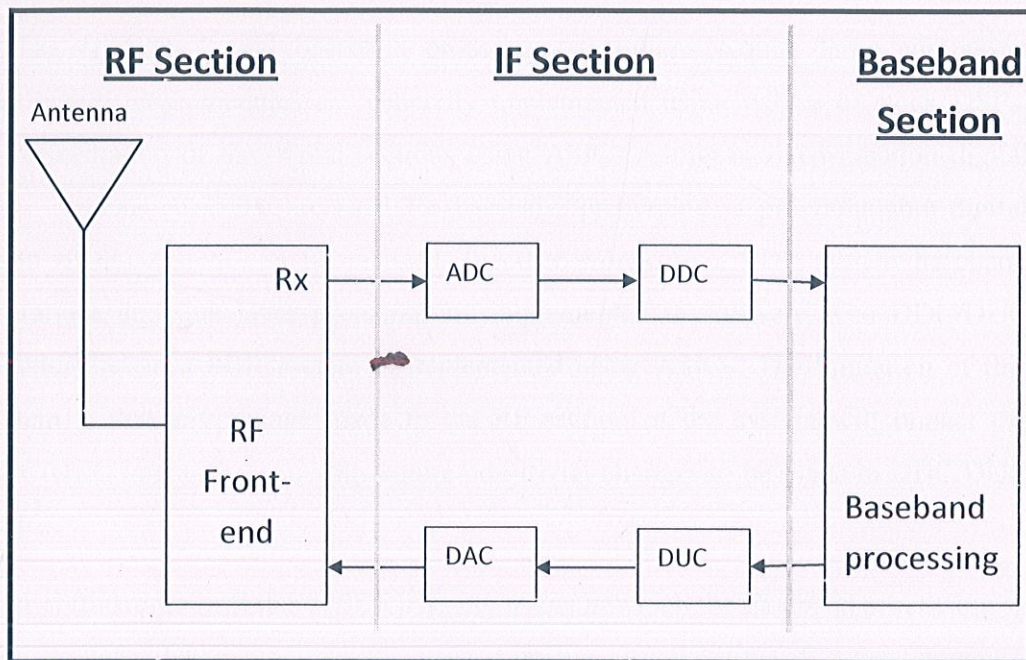
Software defined radio (SDR) is receiving enormous recognition as the next evolutionary stage of wireless technology, getting support from governmental agencies as well as civil and commercial entities. The numerous benefits provided by SDR have created widespread interest, and the triumphal procession of software-based radio systems now only remains a question of time.



### 1.3.1 Architecture of SDR

This section gives a brief overview of a basic conventional digital radio system and then explains how SDR technology can be used to implement radio functions in software. It then explains the software architecture of SDR. The various functional blocks in a generic digital radio transceiver (transmitter/receiver) system is depicted in figure 1.

The digital radio system consists of three main functional blocks: RF section, IF section and baseband section. The RF section consists of essentially analog hardware modules while IF and baseband sections contain digital hardware modules.



**Figure 1. Block Diagram of SDR**

The RF section (also called as RF front-end) is responsible for transmitting/receiving the radio frequency (RF) signal from the antenna via a coupler and converting the RF signal to an intermediate frequency (IF) signal. The RF front-end on the receive path performs RF amplification and analog down conversion from RF to IF. On the transmit path, RF front-end performs analog up conversion and RF power amplification.



The ADC/DAC blocks perform analog-to-digital conversion (on receive path) and digital-to-analog conversion (on transmit path), respectively. ADC/DAC blocks interface between the analog and digital sections of the radio system. DDC/DUC blocks perform digital-downconversion (on receive path) and digital-up-conversion (on transmit path), respectively. DUC/DDC blocks essentially perform modem operations, i.e., modulation of the signal on transmit path and demodulation (also called digital tuning) of the signal on receive path.

The baseband section performs baseband operations (connection setup, equalization, frequency hopping, timing recovery, correlation) and also implements the link layer protocol (layer 2 protocol in OSI protocol model).

The DDC/DUC and baseband processing operations require large computing power and these modules are generally implemented using ASICs or stock DSPs. Implementation of the digital sections using ASICs results in fixed-function digital radio systems. If DSPs are used for baseband processing, a programmable digital radio (PDR) system can be realized. In other words, in a PDR system baseband operations and link layer protocols are implemented in software. The DDC/DUC functionality in a PDR system is implemented using ASICs. The limitation of this system is that any change made to the RF section of the system will impact the DDC/DUC operations and will require non-trivial changes to be made in DDC/DUC ASICs.

A software-defined radio (SDR) system is one in which the baseband processing as well as DDC/DUC modules are programmable. Availability of smart antennas, wideband RF front-end, wideband ADC/DAC technologies and ever increasing processing capacity (MIPS) of DSPs and general-purpose microprocessors have fostered the development of multi-band, multi-standard, multi-mode radio systems using SDR technology. In an SDR system, the link-layer protocols and modulation/demodulation operations are implemented in software.

If the programmability is further extended to the RF section (i.e., performing analog-to-digital conversion and vice-versa right at the antenna) an ideal software radio systems that support programmable RF bands can be implemented.



### 1.3.2 Features of SDR

Following are the key features of SDR technology:

- **Reconfigurability:** SDR allows co-existence of multiple software modules implementing different standards on the same system allowing dynamic configuration of the system by just selecting the appropriate software module to run. This dynamic configuration is possible both in handsets as well as infrastructure equipment. The wireless network infrastructure can reconfigure itself to subscriber's handset type or the subscriber's handset can reconfigure itself to network type. SDR technology facilitates implementation of future-proof, multi-service, multi-mode, multi-band, multi-standard terminals and infrastructure equipment.

- **Universal Connectivity:** SDR enables implementation of air interface standards as software modules and multiple instances of such modules that implement different standards can co-exist in infrastructure equipment and handsets. This helps in realizing global roaming facility. If the terminal is incompatible with the network technology in a particular region, an appropriate software module needs to be installed onto the handset (possibly over-the-air) resulting in seamless network access across various geographies. Further, if the handset used by the subscriber is a legacy handset, the infrastructure equipment can use a software module implementing the older standard to communicate with the handset.

- **Interoperability:** SDR facilitates implementation of open architecture radio systems. End-users can seamlessly use innovative third-party applications on their handsets as in a PC system. This enhances the appeal and utility of the handsets.



### 1.3.3 Advantages of SDR

Table 1: Advantages of SDR

<b>Multi-operability</b>	Support of multiple standards through multimode, multiband radio capabilities
<b>Flexibility</b>	Efficient shift of technology and resources.
<b>Adaptability</b>	Faster migration towards new standards and technologies through programmability and reconfiguration
<b>Sustainability</b>	Increased utilization through generic hardware platforms
<b>Reduced Ownership Costs</b>	Less infrastructure, less maintenance, easier deployment



## CHAPTER 2

### AM/FM Receiver

#### 2.1 Block Diagram

Radio signals received by an AM/FM Antenna first go through an Anti-Aliasing Filter to remove all signals out of AM/FM bands. This analog filter is necessary in order to avoid aliasing problems in digital implementation. The analog input is then digitized into digital samples by an A/D (analog-to-digital) converter. From this point, all subsequent operations including mixing, filtering and demodulation is done using digital signal processing techniques to extract radio channels of interest. These operations are supported by key components of the digital receiver including Digital Mixer, Digital Local Oscillator, Digital Low Pass Filter and Digital Demodulation. After demodulation, digital samples of radio channels of interest are converted back to analog format using a D/A (digital-to-analog) converter. The radio signal is amplified and played by a loud speaker

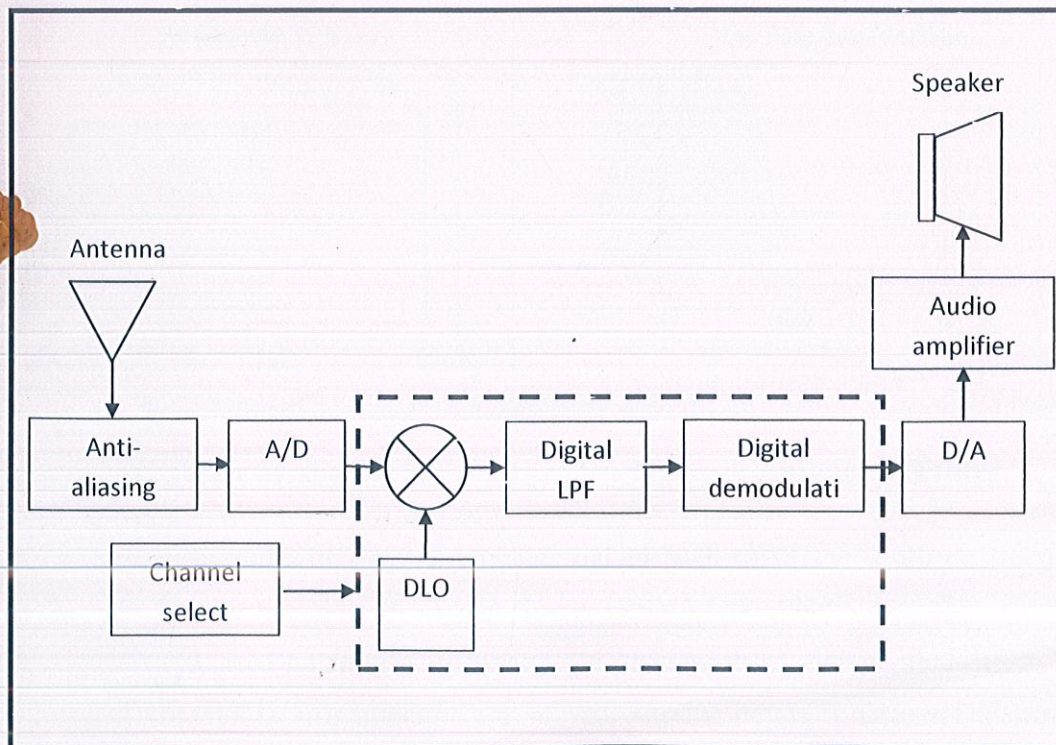


Figure 2: System Overview of a Digital Radio Receiver



### 2.1.1 Anti-aliasing filter

Anti aliasing filter is filter which removes all signals out of required band so that the signal satisfies the sampling theorem. Aliasing refers to an effect which causes different signals to become indistinguishable when sampled. Nyquist's theorem says that any signal can be represented by discrete samples if the sampling rate is at least twice the bandwidth of the signal. For example, if an A/D converter samples analog input at 50 MHz, then input signal must have bandwidth of less than 25 MHz. Let's consider what happens if we violate the Nyquist sampling theorem. Figure 3 shows the frequency spectrum of a system sampling at frequency  $f_s$ . Any signal below half the sampling frequency  $f_s/2$  such as the one at  $f_o$  can be represented by digital samples. However, if input signal has components at higher frequency than  $f_s/2$ , such as  $f_a$  then after sampling, this component will be folded back into the  $[0, f_s/2]$  region resulting in an image at frequency  $f_s - f_a$ . Obviously, this image component cannot be distinguished from a true frequency which may be present at that same frequency.

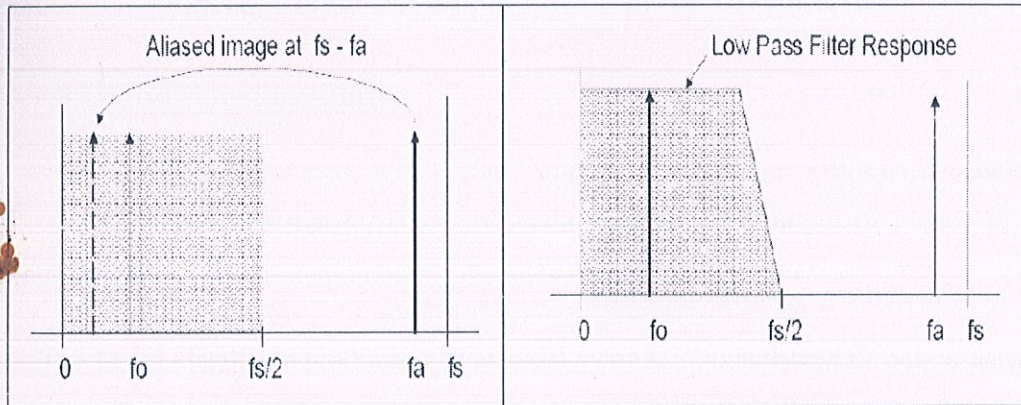


Figure 3: Aliasing

Figure 4: Anti-aliasing filter

This is an example of aliasing problem and in such cases the spectrum of the sampled data may not look at all like the spectrum of the continuous data. Once aliasing has occurred no further processing can distinguish an aliased signal from a true signal. Therefore, it is imperative to prevent aliasing before it occurs. The most common way of preventing aliasing is to use a filter before the A/D converter to remove all frequencies higher than half the sampling frequency. This filter is called an



anti-aliasing filter as shown in Figure 4. As can be seen, the signal at frequency  $f_a > f_s/2$  is now blocked from the input of the A/D converter.

### 2.1.2 Analog to digital converter

The analog input received is then digitized into digital samples by an A/D converter.

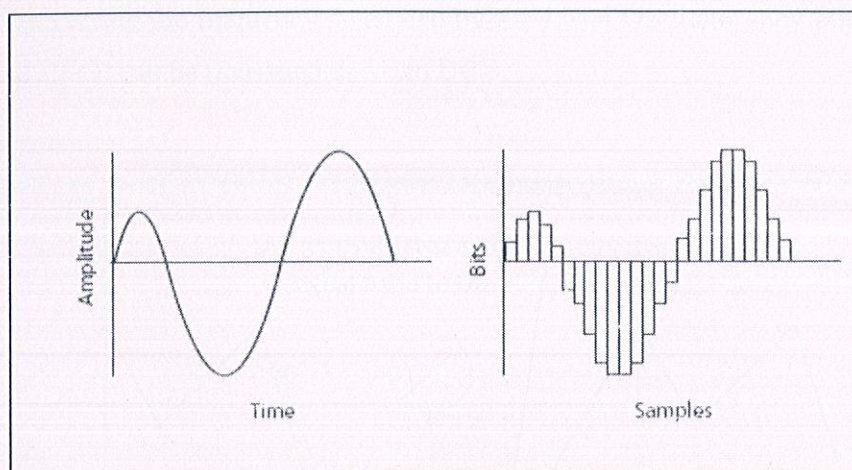


Figure 5: Input and output waveforms of A/D converter

### 2.1.3 Digital local oscillator

The Local Oscillator generates digital samples of a sine (or cosine) wave having the same frequency as that of the desired radio channel. The output frequency of the local oscillator is dependent upon the channel selected by the user.

The Local Oscillator produces a sinusoidal sample at exactly every output sample of the A/D converter, therefore it is driven by the same sampling clock of the A/D converter.

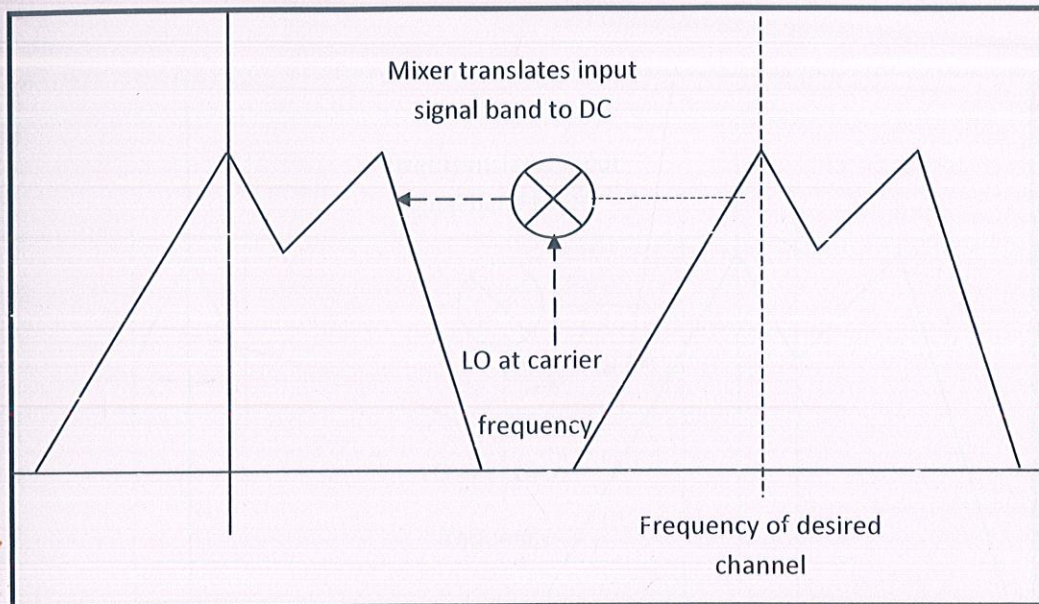
### 2.1.4 Digital Mixer

The Digital Mixer is simply a 2 inputs multiplier which outputs the product of two digital samples. Digital output samples from the A/D are mathematically multiplied with digital samples of a sine (or cosine) signal from the local oscillator. The input samples from the A/D and sine (cosine) samples from the Local Oscillator are generated at the same rate  $f_s$  (the sampling frequency of the A/D). The multiplication



is sample-by-sample and thus the mixer produces samples at the same rate of  $f_s$ . Note that unlike analog mixer which produces many unwanted mixer products, the digital mixer only produces the sum and the difference frequency signals.

Figure 6 illustrates the effect of mixing RF signal with a local oscillator signal. The mixer shifts the spectrum of the RF portion at the frequency of the local oscillator down to DC level. It should be pointed out here that thanks to the accuracy of the digital multiplication of the mixer, translation right down to 0 Hz is achieved. Therefore by varying the frequency generated by the Local Oscillator, any portion of the RF input signal can be translated down to DC.



**Figure 6: Effect of Mixing**

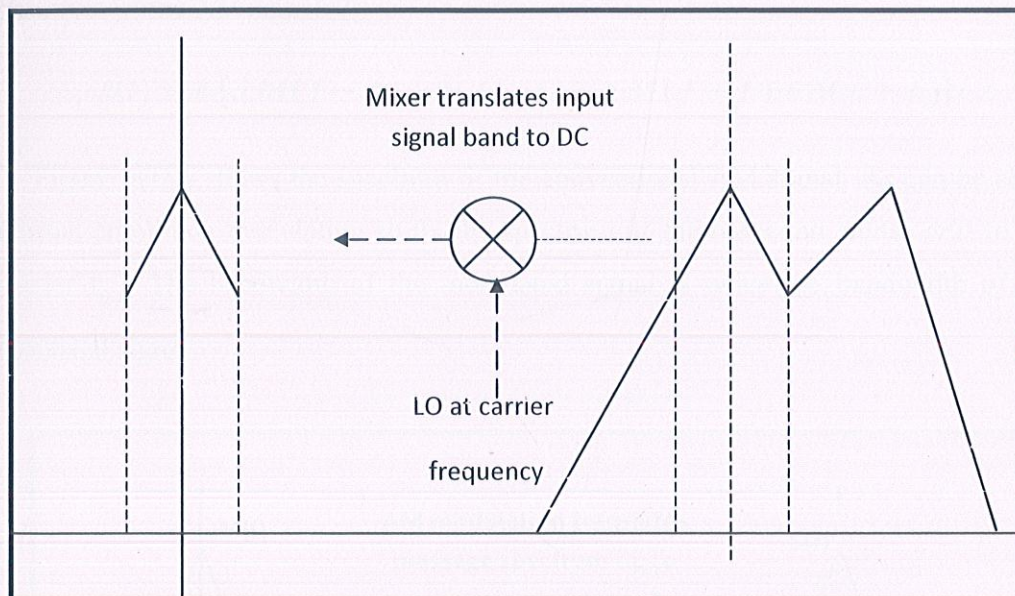
### **2.1.5 Digital low pass filter**

Once the RF signal has been shifted down to DC level, a low pass filter can be used to filter out unwanted high frequency components. The filter accepts digital samples from the output of the Mixer at the sampling frequency,  $f_s$ . It employs digital processing techniques to implement an FIR (finite impulse response) filter transfer function.



The filter passes all frequencies from 0 up to a cutoff frequency equal to the bandwidth of the message signal of interest, and rejects all frequency above that cutoff frequency.

Figure 7 illustrates the effect of band-limit filtering in the frequency domain. As can be seen, the filter only captures frequencies below its cutoff frequency and rejects all other. Note that the mixer has translated the input signal down to 0 Hz, thus allowing the filter to select only a narrow slice of the RF spectrum corresponding to a channel of interest.



**Figure 7: Effect of filtering**



## 2.1.6 Digital demodulator

### 2.1.6.1 AM digital demodulator

In amplitude modulation (AM), the message signal  $m(t)$  is impressed on the amplitude of the carrier signal  $A_c \cos(\omega_c t)$ . A conventional Amplitude Modulation (AM) signal is defined mathematically as

$$s(t) = A_c [1 + a.m(t)] \cos(2\pi f_c t)$$

Where  $f_c$  is the carrier frequency,  $|m(t)| < 1$ , and  $a$  is the modulation index. Function  $m(t)$  represents the *baseband* signal that contains the information to be conveyed. If  $m(t)$  is a deterministic signal with Fourier transform (spectrum)  $M(f)$ , the spectrum of the AM signal  $s(t)$  is

$$S(f) = A_c [a.M(f - f_c) + \delta(f - f_c) + a.M(f + f_c) + \delta(f + f_c)] / 2$$

Figure below shows the spectrum of the conventional AM signal. As can be seen, normal amplitude modulation shifts the spectrum of the baseband signal  $m(t)$  to the carrier  $f_c$ . The bandwidth of the modulated signal is twice the bandwidth of the baseband signal.

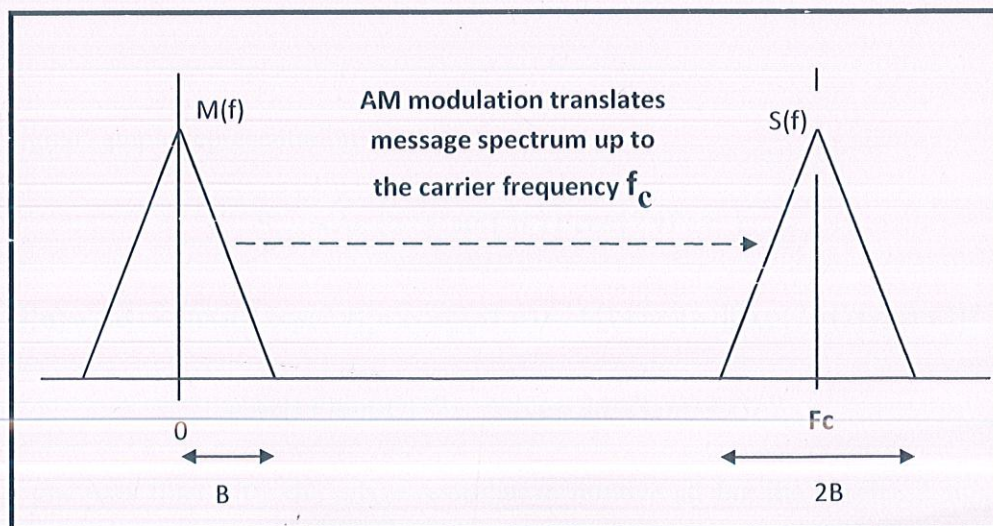


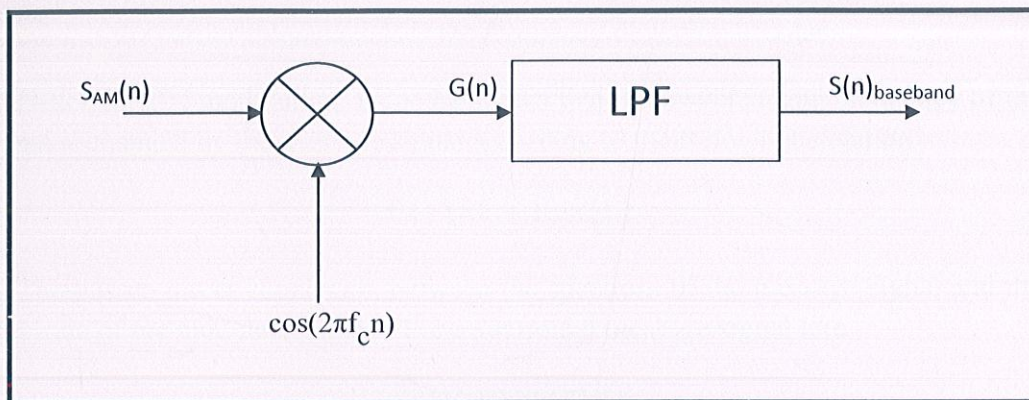
Figure 8: AM spectrum

AM band frequency ranges from 526.5 kHz to 1606.5 kHz. To avoid aliasing, the ADC (analog-to-digital converter) needs to sample signal at sampling rate at least



twice the highest frequency in the AM spectrum, thus a sampling rate of 4 MHz was chosen for the ADC to allow for some margin.

As mentioned previously, an ADC is used to digitize the analog signal received from the antenna into digital representation.  $S_{AM}(n)$  represents the digital samples of the modulated AM signal whose spectrum contains all AM channels. These samples are multiplied with digital sinusoidal samples ( $\cos(2\pi f_c n)$ ) of a local oscillator whose frequency is that of the channel of interest. In other words, the frequency of the sinusoidal signal generated by the local oscillator is dependent upon the channel selected by the user. The effect of this multiplication is to shift the spectrum of the AM signal down to DC level so that a low pass filter having a fixed cut-off frequency can be used to pass the baseband message of the radio channel of interest.



**Figure 9: AM Demodulator**

In digital sample representation, we have:

$$S_{AM} = A_c [1 + a_m(n)] \cos(2\pi f_c n)$$

$$G(n) = A_c [1 + a_m(n)] \cos(2\pi f_c n) \cos(2\pi f_c n) = A_c [1 + a_m(n)] [\cos(2\pi(2f_c)n) + 1] / 2$$

$$= A_c [1 + a_m(n)] / 2 + A_c [\cos(2\pi(2f_c)n) + 1] / 2$$

The low pass filter after the mixer is design to remove all but the baseband signals. Consequently, the output of the demodulator will be

$$S(n)_{baseband} = A_c [1 + a_m(n)] / 2$$



which has an AC component proportional to the baseband message signal  $m(t)$ . Thus we have successfully demodulated the AM signal.

### 2.1.6.2 FM Digital Demodulator

FM modulation is a type of angle modulated signal, a conventional frequency modulated signal is given by the following equation

$$S_{FM}(t) = A_c \cos(2\pi f_c t + \phi(t))$$

in which the message signal  $m(t)$  is transferred as the rate of change of the phase and is defined by

$$\phi(t) = 2\pi K_f \int_{-\infty}^t m(\tau) d\tau$$

where  $f_c$  is the carrier frequency,  $k_f$  is the frequency deviation constant.

Hence for FM modulation, it can be shown that the instantaneous frequency of the signal is changed by the message signal according to the following equation

$$f_i(t) = f_c + k_f m(t)$$

There is no direct connection between the spectrum of the message signal and the spectrum of the modulated signal. For a sinusoidal message signal like

$$m(t) = a \cos(2\pi f_m t)$$

The modulated signal can be expressed as

$$S_{FM}(t) = A_c \cos[2\pi f_c t + \{K_f a (\sin(2\pi f_m t)) / f_m\}] = A_c \cos(2\pi f_c t + \beta \sin(2\pi f_m t))$$

Where,  $\beta = K_f a / f_m$  is the modulation index. In general, the actual bandwidth of FM modulated signal is infinite. However, the amplitude of the spectrum declines fast outside bandwidth around the carrier frequency.

The effective bandwidth of a FM modulated signal, which contains 98 % of the signal power. where  $\beta$  is the modulation index and  $f_m$  is the highest frequency of the message signal.

$$B_c = 2(\beta + 1)f_m$$



### 2.1.6.2.1 FM Phase Locked Loop Demodulator

A Phase-Locked Loop (PLL) can be used to demodulate a FM modulated signal. The PLL is a feedback loop that is commonly used in analog demodulation of FM signal. Based on the original development of a Phase-lock loop for FM demodulator in continuous time domain, we have transferred this into an equivalent PLL demodulator in discrete time domain (suitable for digital implementation) as presented below.

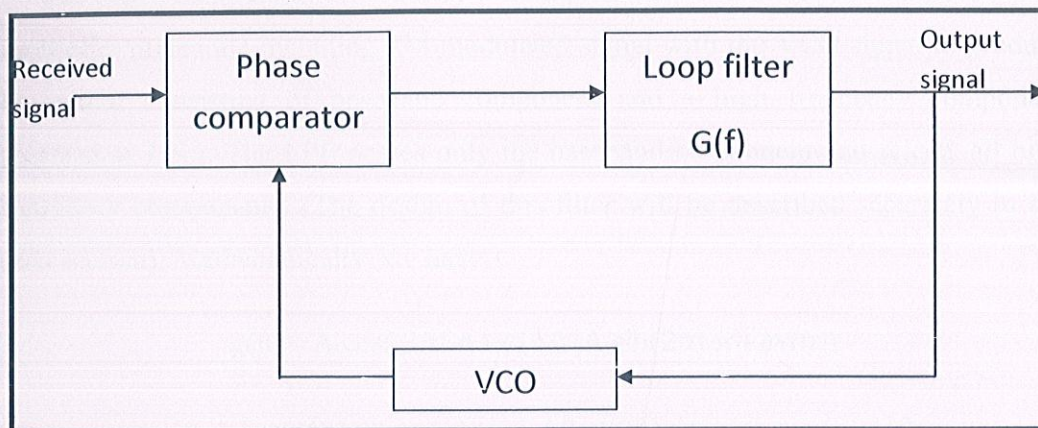


Figure 10: Block diagram of Phase-locked loop FM demodulator

The reference signal is the FM modulated signal

$$S_{FM}(n) = A_c \cos(2\pi f_c n + \varphi(n))$$

The VCO generates a sinusoidal of a fixed frequency, in this case the carrier frequency  $f_c$ , in the absence of an input control voltage. The voltage control to the VCO is the output of the loop filter (the demodulated signal), denoted as  $v(n)$ . Thus the instantaneous frequency of the VCO is

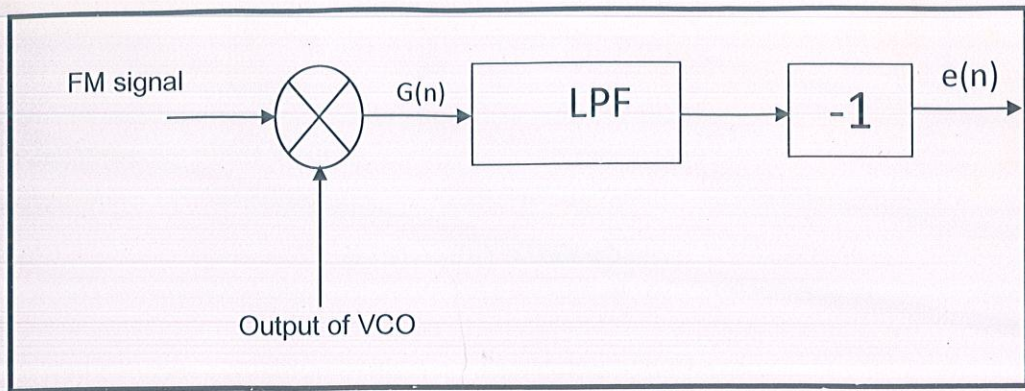
$$f_v = f_c + K_v v(n)$$

Where,  $k_v$  is a deviation constant. Therefore, the VCO output may be expressed as

$$y_v(n) = A_v \sin(2\pi f_c n + \varphi_v(n))$$

$$\varphi_v(n) = 2\pi K_v \sum_{i=0}^{n-1} v(i)$$



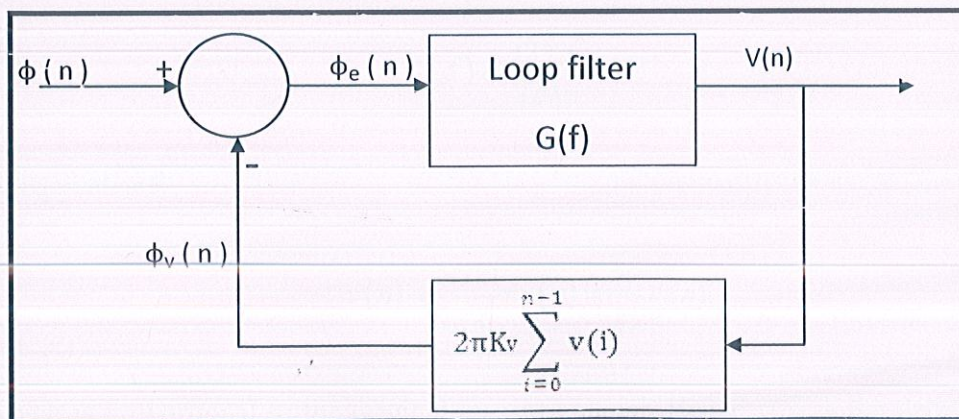


**Figure 11: Phase comparator structure**

The phase comparator is basically a multiplier and a Low Pass Filter (LPF). The multiplier mixes the incoming FM modulated signal with the VCO signal to produce an output consisting of baseband component and a high frequency component (centred at  $2f_c$ ). The LPF passes only the baseband component and rejects all high frequency components. (The design of this filter will be described separately in the next section). Mathematically, we have

$$\begin{aligned}
 g(n) &= A_c \cos(2\pi f_c n + \varphi(n)) \cdot A_v \sin(2\pi f_c n + \varphi_v(n)) \\
 &= A_c A_v [-\sin(\varphi(n) - \varphi_v(n)) + \sin(2\pi(2f_c)n + \varphi(n) + \varphi_v(n))] / 2 \\
 e(n) &= (-1)g(n) \cdot h_{\text{LPF}}(n) = A_c A_v [\sin(\varphi(n) - \varphi_v(n))] / 2
 \end{aligned}$$

where the difference  $\varphi(n) - \varphi_v(n) = \varphi_e(n)$  is the phase error. Let assume that the PLL is in lock, and thus  $\varphi_v(n)$  is very close to  $\varphi(n)$ . In that case, the phase error is very small and  $\sin(\varphi(n) - \varphi_v(n)) \approx \varphi(n) - \varphi_v(n) = \varphi_e(n)$ .



**Figure 12: Linearized PLL model**



We can write the phase error as

$$\varphi_e(n) = \varphi(n) - 2\pi k_v \sum_{i=0}^{n-1} v(i)$$

Differentiate above equation in discrete time domain, we have

$$\varphi_e'(n) + 2\pi k_v v(n) = \varphi'(n)$$

$$\varphi_e'(n) + 2\pi k_v \sum_{k=-\infty}^{\infty} \varphi_e g(n-k)$$

Taking the Discrete Fourier transform of above equation

$$(j2\pi f)\Phi_e(f) + 2\pi k_v \Phi_e(f)G(f) = (j2\pi f)\Phi(f)$$

And hence,

$$\Phi_e(f) = \Phi(f) / [1 + (k_v G(f) / jf)]$$

Suppose that we design  $G(f)$  such that

$$\left| \frac{k_v G(f)}{jf} \right| \gg 1$$

in the frequency band  $f < W$  of the message signal.  $V(f)$  can be approximated by

$$V(f) = \frac{j2\pi f}{2\pi k_v} \Phi(f)$$

or equivalently,

$$v(n) = \frac{1}{2\pi k_v} \Phi'(n) = \frac{k_f}{k_v} m(n)$$

which is proportional to the original message signal. Hence  $v(n)$  is the demodulated signal.



# SOFTWARES USED FOR AM/FM RECEIVER

### 3.1 MATLAB 7.6.0

Matrix Laboratory (MATLAB) is a programming environment for algorithm development, data analysis, visualization, and numerical computation. Using MATLAB, we can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

We can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. For a million engineers and scientists in industry and academia, MATLAB is the language of technical computing.

Key features of Matlab include –

- High-level language for technical computing.
- Development environment for managing code, files, and data.
- Interactive tools for iterative exploration, design, and problem solving.
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration.
- Tools for building custom graphical user interfaces.

MATLAB is the foundation for all products, including Simulink. We can extend MATLAB with add-on products for -

- Parallel Computing
- Math, Statistics, and Optimization
- Control System Design and Analysis



We can use the MATLAB product family for –

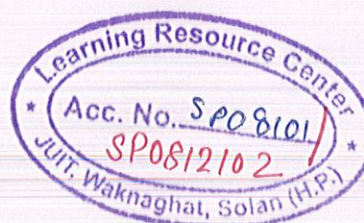
- Communications Systems
- Control Systems
- Digital Signal Processing
- Embedded Systems
- FPGA Design
- Image and Video Processing

### 3.1.1 Simulink 7.1

Simulink is an environment for multi-domain simulation and Model-Based Design for dynamic and embedded systems. It provides an interactive graphical environment and a customizable set of block libraries that let us design, simulate, implement, and test a variety of time-varying systems, including communications, controls, signal processing, video processing, and image processing. Simulink is integrated with MATLAB, providing immediate access to an extensive range of tools that let you develop algorithms, analyze and visualize simulations, create batch processing scripts, customize the modeling environment, and define signal, parameter, and test data.

Key features of Simulink include-

- Extensive and expandable libraries of predefined blocks
- Interactive graphical editor for assembling and managing intuitive block diagrams
- Ability to manage complex designs by segmenting models into hierarchies of design components
- Model Explorer to navigate, create, configure, and search all signals, parameters, properties, and generated code associated with your model





- Application programming interfaces (APIs) that let you connect with other simulation programs and incorporate hand-written code

We can extend the Simulink environment with add-on products for-

- Simulation Graphics
- Control System Design and Analysis
- Signal Processing and Communications
- Code Generation

We can use the Simulink product family for-

- Model-Based Design
- Control Systems
- Digital Signal Processing
- Communications Systems
- Image and Video Processing
- Embedded Systems

With Simulink, we can quickly create, model, and maintain a detailed block diagram of your system using a comprehensive set of predefined blocks. Simulink provides tools for hierarchical modeling, data management, and subsystem customization, making it easy to create concise, accurate representations, regardless of your system's complexity.



Simulink software includes an extensive library of functions commonly used in modeling a system. These include:

- Continuous and discrete dynamics blocks, such as Integration and Unit Delay
- Algorithmic blocks, such as Sum, Product, and Lookup Table
- Structural blocks, such as Mux, Switch, and Bus Selector

You can customize these built-in blocks or create new ones directly in Simulink and place them into our own libraries. Additional blocksets (available separately) extend Simulink with specific functionality for aerospace, communications, radio frequency, signal processing, video and image processing, and other applications. With Simulink, we can build models by dragging and dropping blocks from the library browser onto the graphical editor and connecting them with lines that establish mathematical relationships between the blocks. You can arrange the model by using graphical editing functions, such as copy, paste, undo, align, distribute, and resize.

The Simulink user interface gives us complete control over what you can see and use onscreen. We can add your commands and submenus to the editor and context menus. You can also disable and hide menus, menu items, and dialog box controls.

Simulink lets us organize your model into clear, manageable levels of hierarchy by using subsystems and model referencing. Subsystems encapsulate a group of blocks and signals in a single block. You can add a custom user interface to a subsystem that hides the subsystem's contents and makes the subsystem appear as an atomic block with its own icon and parameter dialog box. We can reuse the design components on multiple projects, easily maintaining audit and revision histories.

Organizing our models in this way lets us select the level of detail appropriate to the design task. For example, we can use simple relationships to model high-level specifications and add more detailed relationships as you move toward implementation.



### **3.2 Xilinx ISE System Generator 11.1**

System Generator allows device-specific hardware designs to be constructed directly in a flexible high-level system modeling environment. In a System Generator design, signals are not just bits. They can be signed and unsigned fixed-point numbers, and changes to the design automatically translate into appropriate changes in signal types. Blocks are not just stand-ins for hardware. They respond to their surroundings, automatically adjusting the results they produce and the hardware they become. System Generator allows designs to be composed from a variety of ingredients. Data flow models, traditional hardware design languages (VHDL, Verilog, and EDIF), and functions derived from the MATLAB programming language, can be used side-by-side, simulated together, and synthesized into working hardware. System Generator simulation results are bit and cycle-accurate. This means results seen in simulation exactly match the results that are seen in hardware. System Generator simulations are considerably faster than those from traditional HDL simulators, and results are easier to analyze.

System Generator for DSP™ is the industry's leading high-level tool for designing high-performance DSP systems using FPGAs.

- Develop highly parallel systems with the industry's most advanced FPGAs
- Provide system modeling and automatic code generation from Simulink® and MATLAB® (The Mathworks, Inc.)
- Integrates RTL, embedded, IP, MATLAB and hardware components of a DSP system
- A key component of the Xilinx DSP Targeted Design Platform

System Generator for DSP is part of both the DSP and System Editions of ISE® Design Suite. With System Generator for DSP, developers with little FPGA design experience can quickly create production quality FPGA implementations of DSP algorithms in a fraction of traditional RTL development times.



## Key Features

- **DSP modeling-**

Build and debug high-performance DSP systems in Simulink using the Xilinx Blockset that contains functions for signal processing (e.g., FIR filters, FFTs), error correction (e.g., Viterbi decoder, Reed-Solomon encoder/decoder), arithmetic, memories (e.g., FIFO, RAM, ROM), and digital logic.

- **Automatic code generation of VHDL or Verilog from Simulink-**

Implement behavioral (RTL) generation and target specific Xilinx IP cores from the Xilinx Blockset. System Generator also supports custom HDL through its HDL import flow.

- **Hardware co-simulation-**

A code generation option that allows you to validate working hardware and accelerate simulations in Simulink and MATLAB. System Generator supports Ethernet (10/100/Gigabit) and JTAG communication between a hardware platform and Simulink.

- **Hardware / software co-design of embedded systems-**

Build and debug DSP co-processors for the Xilinx MicroBlaze™ soft processor core. System Generator provides a shared memory abstraction of the HW/SW interface, automatically generating the DSP the bus interface logic, software drivers, and software documentation.

### **3.2.1 Field Programmable Gate Array (FPGA)**

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by the customer or designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL).

FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"—somewhat like many (changeable) logic gates that can be inter-wired in



(many) different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

There are 2 basic types of FPGAs: SRAM-based reprogrammable and One-time programmable(OTP). These two types of FPGAs differ in the implementation of the logic cell™ and the mechanism used to make connections in the device.

The dominant type of FPGA is SRAM-based and can be reprogrammed by the user as often as the user chooses. In fact, an SRAM FPGA is reprogrammed every time it is powered-up because the FPGA is really a fancy memory chip! (That's why we need a serial PROM or system memory with every SRAM FPGA).

Applications of FPGAs include digital signal processing, software-defined radio, aerospace and defence systems, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas.

Traditionally, FPGAs have been reserved for specific vertical applications where the volume of production is small. For these low-volume applications, the premium that companies pay in hardware costs per unit for a programmable chip is more affordable than the development resources spent on creating an ASIC for a low-volume application. Today, new cost and performance dynamics have broadened the range of viable applications.

FPGAs are increasingly used in conventional high performance computing applications where computational kernels (Fast Fourier transform, convolution etc) are performed on the FPGA instead of a microprocessor. FPGA implementations of these kernels offer order of magnitude performance improvements over microprocessors. Other benefits are in terms of power used: an FPGA implementation of FFT or convolution is expected to consume lesser power than a microprocessor.



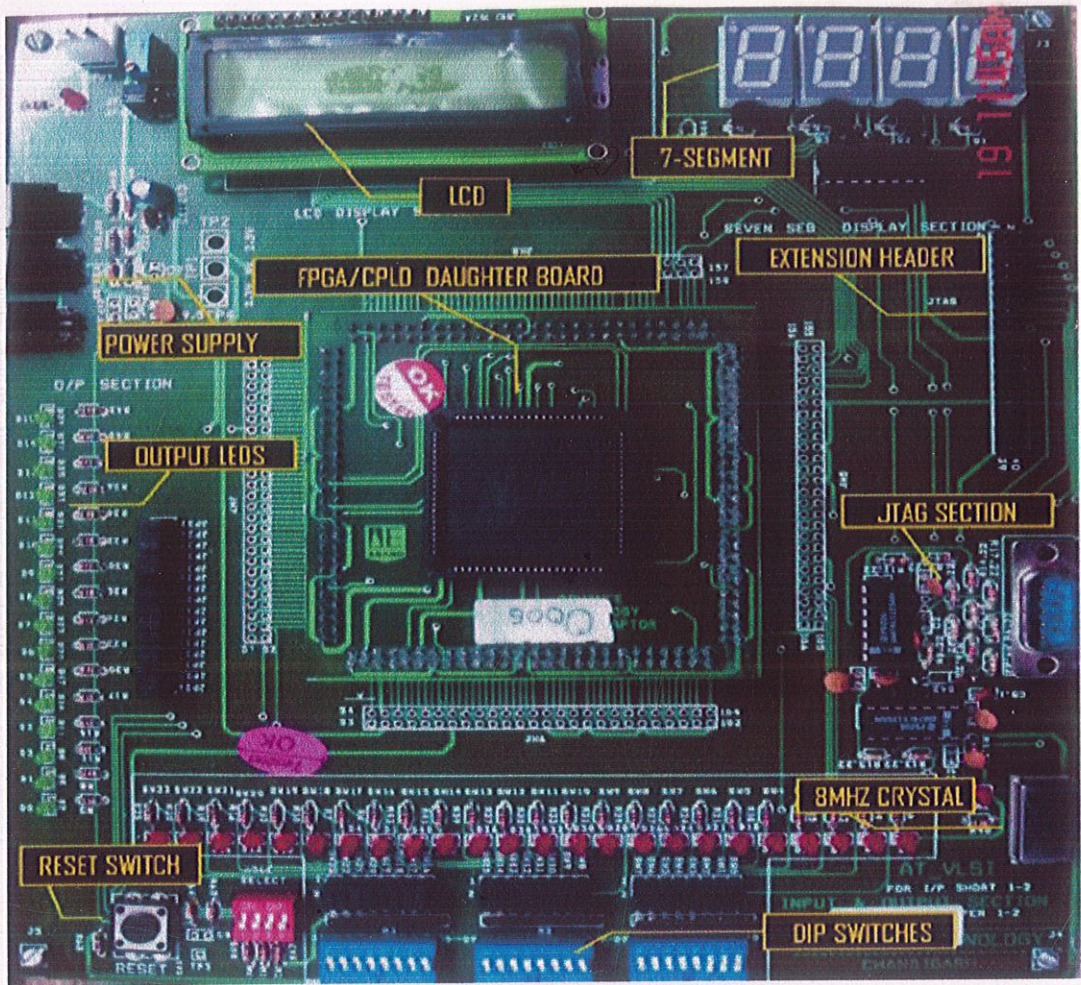


Figure 13: Schematic view of FPGA



## CHAPTER 4

### IMPLEMENTATION OF AM/FM RECEIVER

#### 4.1 AM RECIEVER

##### 4.1.1 IMPLEMENTATION OF AM-LPF

Digital filters transform digital representation of analog signals to remove noise and other unwanted signal components and to shape the spectral characteristics of the resulting signal. It operates on a finite precision digital representation of a signal. Two common architectures for the linear digital filters are

- a) Finite implulse response(FIR) filter
- b) Infinite impulse response(IIR) filter.

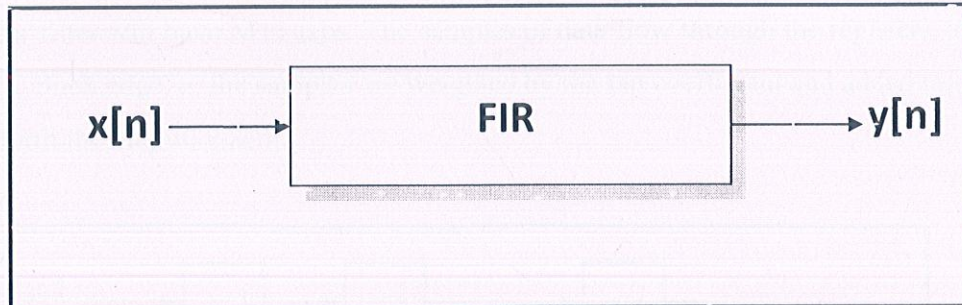


Figure 14: Block diagram of FIR filter

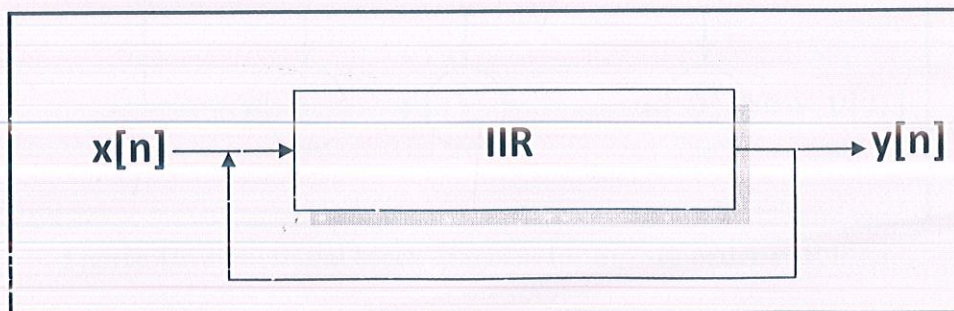


Figure 15: Block diagram of IIR filter



Digital filters operate in the time-sequence domain, accepting a sequence of discrete, finite-length words to produce an output sequence. The sequence of inputs,  $x[n]$  may be the output of an analog to digital converter or the output of some other functional unit. A FIR filter forms its output as a weighted sum of its input; an IIR filter forms its output from a weighted sum of its input and past value of its output.

#### 4.1.1.1 Finite Impulse Response(FIR) filter

A FIR digital filter forms its output as a weighted sum of present and past samples of its inputs, as described by the feed-forward difference equation written below. FIR filters are called moving average filters because their output at any time index depends on a window containing only the most recent M samples of the input.

$$y_{\text{FIR}}[n] = \sum b_k x[n-k], 0 < k < N$$

A FIR filter can be described by the z-domain functional block diagram as shown below, where each box labelled with  $z^{-1}$  denotes a register cell having a delay of one clock cycle. Each stage of the filter holds a delayed sample of the input, and the connection at the input and connections at the tap coefficients of the filter. An Nth order filter will have M+1 taps. The samples of data flow through the registers, and at each clock edge, n, the samples are weighted by the tap coefficient and added together to form the output,  $y_{\text{FIR}}[n]$ .

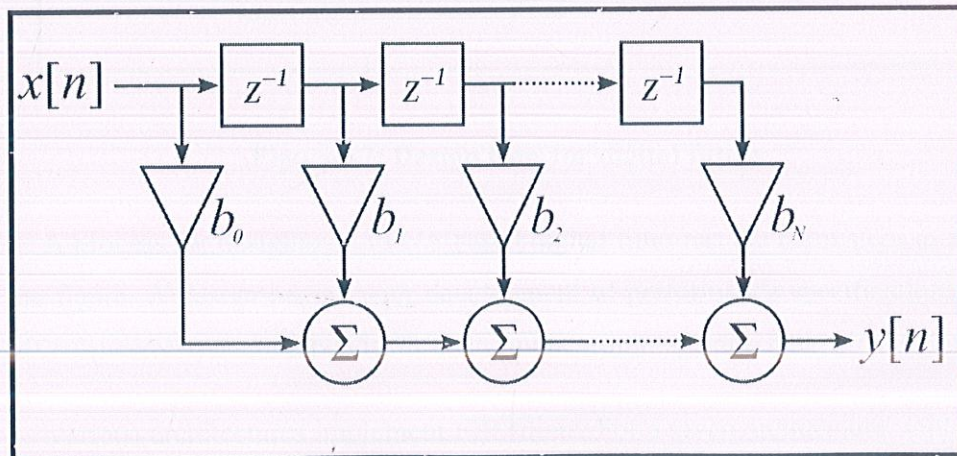
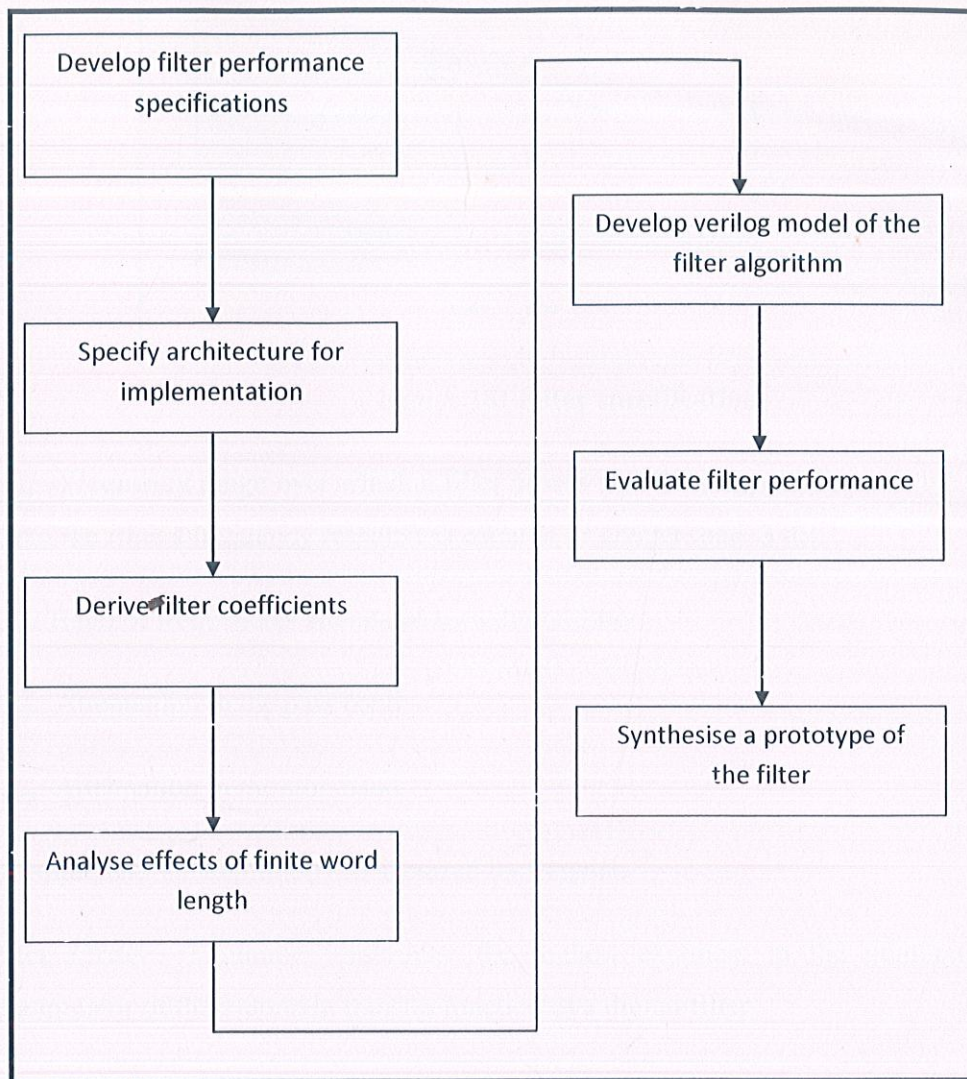


Figure 16: Functional block diagram for an Nth order FIR filter



#### 4.1.1.2 Digital Filter Design process



**Figure 17: Design flow for digital filters**

A process for designing a FPGA based digital filter has the main steps as shown in the figure. A design begins with development of performance specifications for cut-off frequency, transition bands, in-band ripple, minimum stop band attenuation, etc.

Various architectures implement FIR filters. For a given architecture, tools such as MATLAB can be used to determine filter coefficients that implement a filter that satisfy specification of the design.



### 4.1.1.3 Filter Specifications

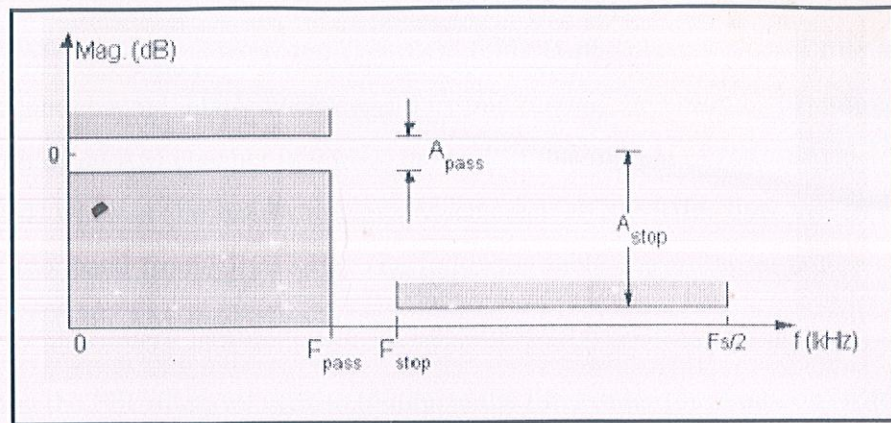


Figure 18: Filter specifications

$F_{pass}$ : Frequency range over which a filter passes signal energy. Defined as the range where the filter's frequency response is equal to or greater than -3 dB.

$F_{stop}$ : Band of frequencies attenuated by a digital filter.

$A_{pass}$ : Attenuation in the pass band.

$A_{stop}$ : Attenuation in the stop band.

**Attenuation**: an amplitude loss incurred by a signal.

**Filter Order** - a number describing the highest exponent in the numerator or denominator of the z-domain transfer function of a digital filter.

Table 2: AM Filter specifications

AM Filter
$F_{pass} = 3 \text{ kHz}$
$F_{stop} = 4.5 \text{ kHz}$
$A_{pass} = 3 \text{ dB}$
$A_{stop} = 30 \text{ dB}$



#### 4.1.1.4 Multistage Filtering Technique

For both AM and FM receivers, the Low Pass Filter (LPF) plays a critical role in isolating adjacent channel interferences, and thus minimizing the noise at the output. The bandwidth of an AM channel is 9 kHz, thus a LPF having pass band from 0-4.5 kHz is desired, whereas the bandwidth of FM channel is 200 kHz, thus the filter should have pass band from 0-100 kHz. The following table shows the summary of filter requirements for AM and FM receivers.

In designing the FIR filter, we seek to minimize the filter order (or number of filter taps) while satisfying requirements specified above. FIR filter theory states that the filter order is approximately proportional to the input sampling frequency and inversely proportional to the width of the transition band. That is the higher the input sampling frequency, the higher the filter order, and the narrower the transition band, the higher the filter order.

$$\text{Filter order } M \approx \frac{F_s}{\Delta F} = \frac{F_s}{F_{stop} - F_{pass}}$$

Calculating filter order using the following specifications, we get

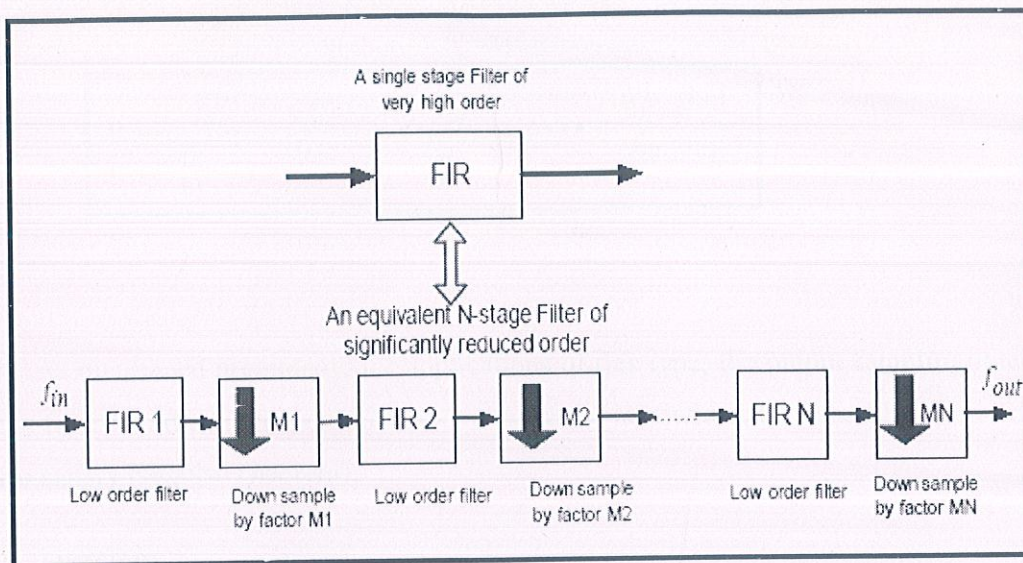
**Table 3: AM Filter order**

Specification	AM Filter
Sampling frequency	4000 kHz
$F_{pass}$	3 kHz
$F_{stop}$	4.5 kHz
<b>Order</b>	<b>2670</b>

This order is astronomically large due to very high input sampling rate and the narrow transition band. Therefore, a single FIR filter satisfying our requirements would be impossible to be realized in hardware. In order to overcome this problem,



we employ a common strategy of multistage filtering to reduce the number of filter taps as shown in the figure below.



**Figure 19: Multistage Filtering**

It is based on the observation that the output of the filter is the baseband message of bandwidth around 4 kHz, the output sample rate needs not to be maintained at 4000 kHz but could be reduced to around twice the bandwidth of the baseband message, about 8 kHz, just to satisfy Nyquist theorem. In this technique a single stage filter of very high order is replaced by a chain of N stages of smaller order filters and down-sample operations. At each stage, the cut-off frequency of the filter is at the desired cut-off frequency of the original single filter, but its stop band frequency is set at half the new sampling rate (to the next stage). Therefore, the filter at each stage has lower input sample rate and broader transition band, resulting in a much smaller number of taps at each stage.

**Table 4: N-stage filter design**

Filter	Cutoff frequency	Stopband frequency
FIR 1	$f_c$	$f_{in}/4$
FIR 2	$f_c$	$f_{in}/8$
FIR 3	$f_c$	$f_{in}/16$
FIR k	$f_c$	$f_{in}/2^{k+1}$



Using the approximation for filter order, we can estimate the total number of filter taps (denoted M) required in this case as follows

$$M = \frac{f_{in}}{\frac{f_{in}}{4} - f_c} + \frac{f_{in}/2}{\frac{f_{in}}{8} - f_c} + \frac{f_{in}/4}{\frac{f_{in}}{16} - f_c} + \dots + \frac{f_{in}/2^{k-1}}{\frac{f_{in}}{2^{k+1}} - f_c}$$

As mentioned previously, for applications of this type, the output sampling rate of the filter could be reduced to just above twice the bandwidth of the baseband message (to satisfy Nyquist Theorem).

Therefore,

$$\begin{aligned} f_{out} &= \frac{f_{in}}{2^N} \geq 2f_c \\ f_c &\leq \frac{f_{in}}{2^{N+1}} \end{aligned}$$

Thus the maximum number of taps (when  $F_c = F_{in}/2^{N+1}$ ) for stage  $k$  can be written as

$$M_k = \frac{f_{in}/2^{k-1}}{\frac{f_{in}}{2^{k+1}} - f_c} = \frac{f_{in}/2^{k-1}}{\frac{f_{in}}{2^{k+1}} - \frac{f_{in}}{2^{N+1}}} = \frac{4}{1 - \frac{1}{2^{N-k}}}$$

Examining the above expression, we can see that the number of taps for stage  $k$  is approximately 4 if  $1/2^{N-k}$  is small. In fact  $M_k \approx 4$  for  $1/2^{N-k} \leq 0.2$ .

Thus,  $M_k \approx 4$  (for  $k \leq N-3$ ).

In other words, the estimated filter order for the first  $(N-3)$  stages of the  $N$ -stage filter is  $4(N-3)$ .



Therefore, the total number of filter taps in the N-stage filter can be rewritten as,

$$\begin{aligned}
 M &\approx 4(N-3) + M_{N-2} + M_{N-1} + M_N \\
 &= 4(N-3) + \frac{4}{1 - \frac{1}{2^{N-(N-2)}}} + \frac{4}{1 - \frac{1}{2^{N-(N-1)}}} + M_N \\
 &= 4(N-3) + 5 + 8 + M_N \\
 &= 4N + 1 + M_N
 \end{aligned}$$

Where  $M_N$  is the filter order of the last stage, which can be estimated to be about 10, depending on the actual value of  $f_c$  relative to  $f_{out}$ . Therefore, we have shown that the total number of taps of the N-stage filter is approximately given by

$$M \approx 4N = 4 \log_2 \left( \frac{f_{in}}{f_{out}} \right)$$

Compared with the estimate of the number of filter taps for the single filter, it can be seen that the N-stage filter has dramatically reduced the number of filter taps. In fact, the filter order of the N-stage filter is of logarithmic order of the input sampling rate whereas the order of the single stage filter is the order of the input sampling rate.

An N-stage filter achieves a reduction in the number of filter taps by a factor of  $(F_s / (4 * \log_2 F_s))$  compared to a single stage filter, where  $F_s$  is the input sampling rate of the input data.



#### 4.1.1.5 Calculation of filter order

In the design of the filter for the AM Digital Receiver, the input sampling rate is 4000 kHz and the baseband message is of 4 kHz bandwidth, therefore we decided to reduce the output sampling rate to around 15 kHz to allow for some margin.

The number of stages of a N-stage filter would be  $N = \log_2(4000/15) = 8$ , wherein the sampling rate is reduced by a factor of 2 at each stage as discussed above. The following table summarises the design. The filter order was calculated for each stage using the formula:

$$\text{Filter order} = \frac{F_s}{F_{stop} - F_{pass}} = \frac{4000}{1000 - 4} \approx 4$$

Similarly, the order for each stage was calculated and is summarized in table 5.

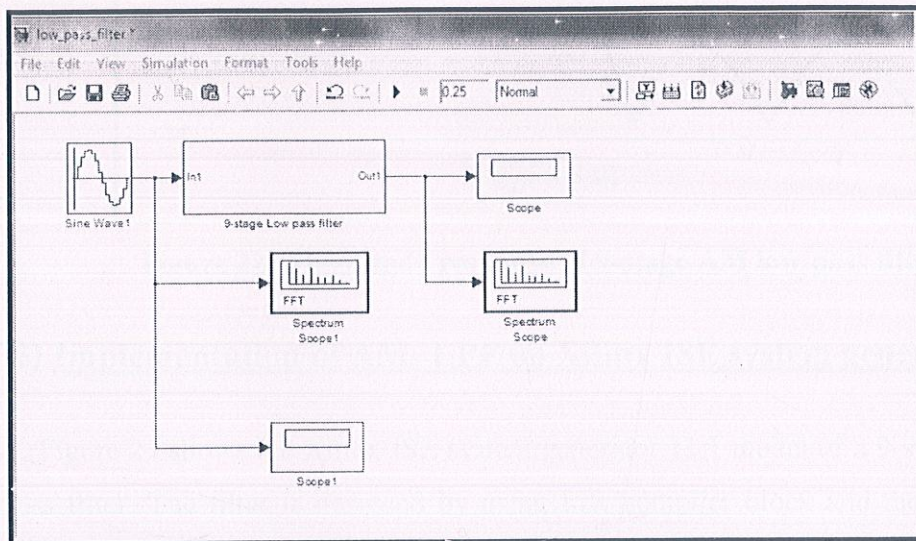
**Table 5: A 9-stage filter for AM digital receiver**

Stage	Sampling Frequency(kHz)	Passband Frequency(kHz)	Stopband Frequency(kHz)	Filter Order
1	4000	0-4	1000	4
2	2000	0-4	500	4
3	1000	0-4	250	4
4	500	0-4	125	4
5	250	0-4	62.5	4
6	125	0-4	31.25	5
7	62.5	0-4	15.625	6
8	31.25	0-4	7.8125	9
9	15.625	0-3	4.5	11
<b>Total number of filter taps</b>				<b>51</b>



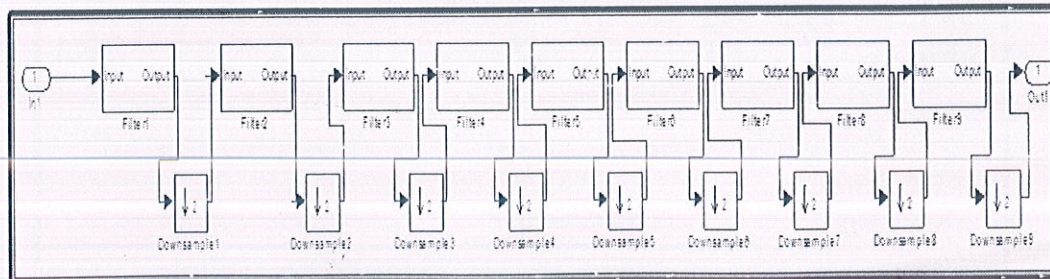
## A) Implementation of AM- LPF on Simulink

The simulink model of low pass filter was realized using simulink blocks and a sine wave was generated and the output was seen through spectrum scope..



**Figure 20: Simulink model of Low pass filter**

Figure 21 shows the 9-stage low pass filter realized in MATLAB Simulink. The sampling frequency was down-sampled at every stage by 2 and the stop band was also halved at each stage. Figure 22 shows the magnitude response of the AM-low pass filter.



**Figure 21: 9-stage low pass filter**



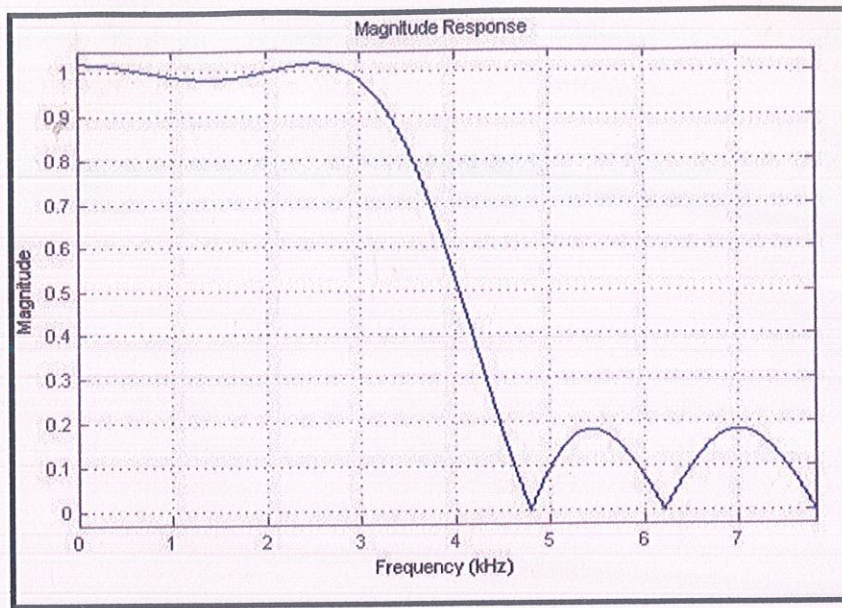


Figure 22: Magnitude response of 9-stage AM low pass filter

### B) Implementation of AM- LPF on Xilinx ISE system generator 11.1

Figure 23 shows the Xilinx ISE system generator 11.1 model of a 9-stage AM low pass filter. The filter is designed by using FIR compiler block and the coefficients were calculated using Filter Design and Analysis (FDA) tool.

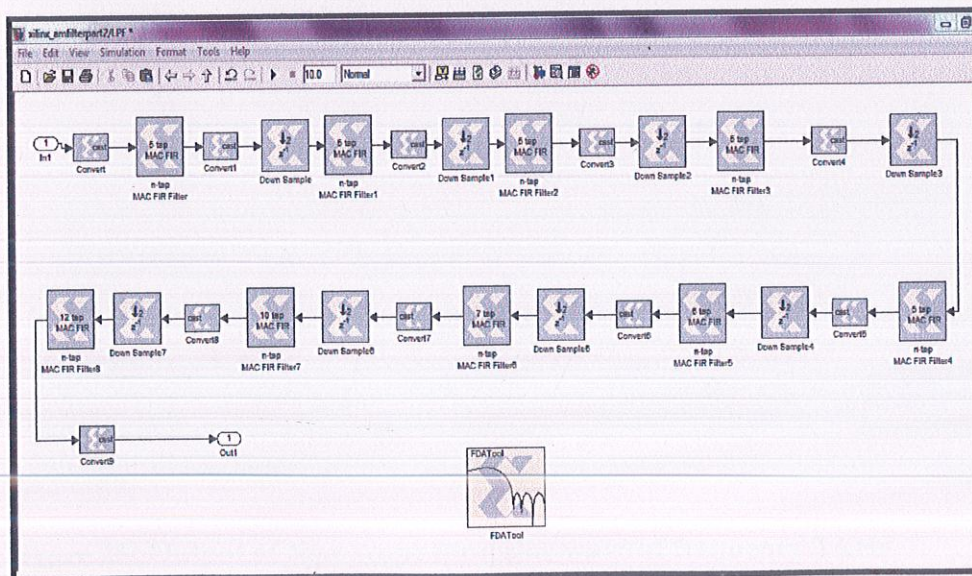


Figure 23: Xilinx System generator model of 9-stage AM-LPF



A sine wave of frequency 3 kHz was passed through the low pass filter and the output was seen through spectrum scope and time scope as shown in figure 24 and figure 25

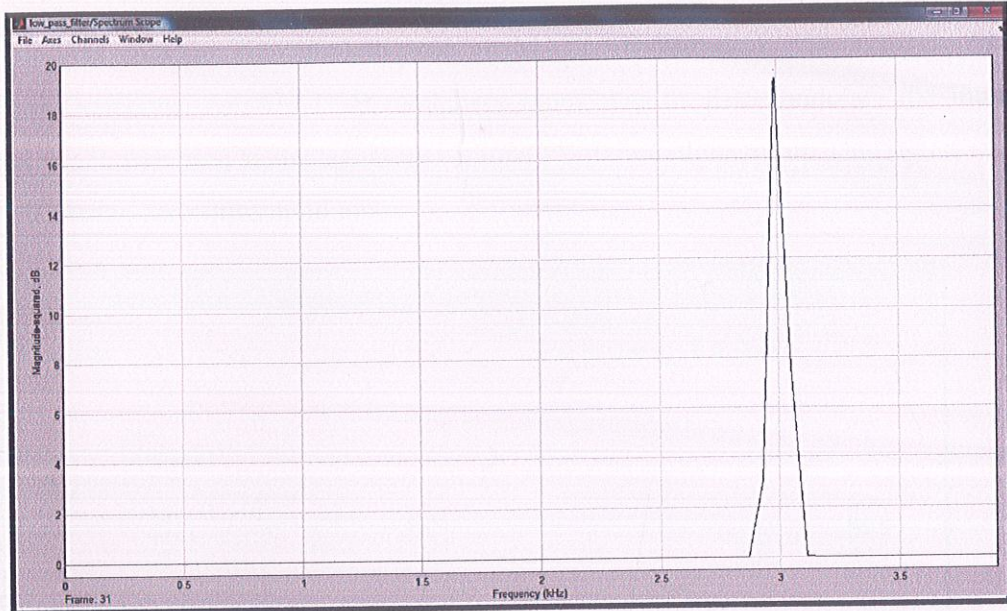


Figure 24: Output of spectrum scope for a frequency of 3 kHz

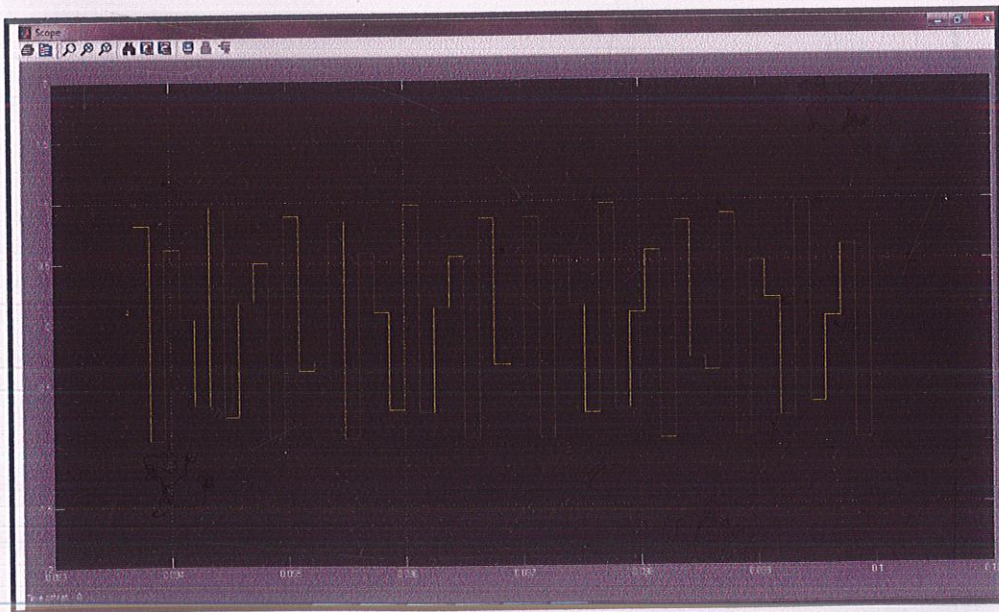


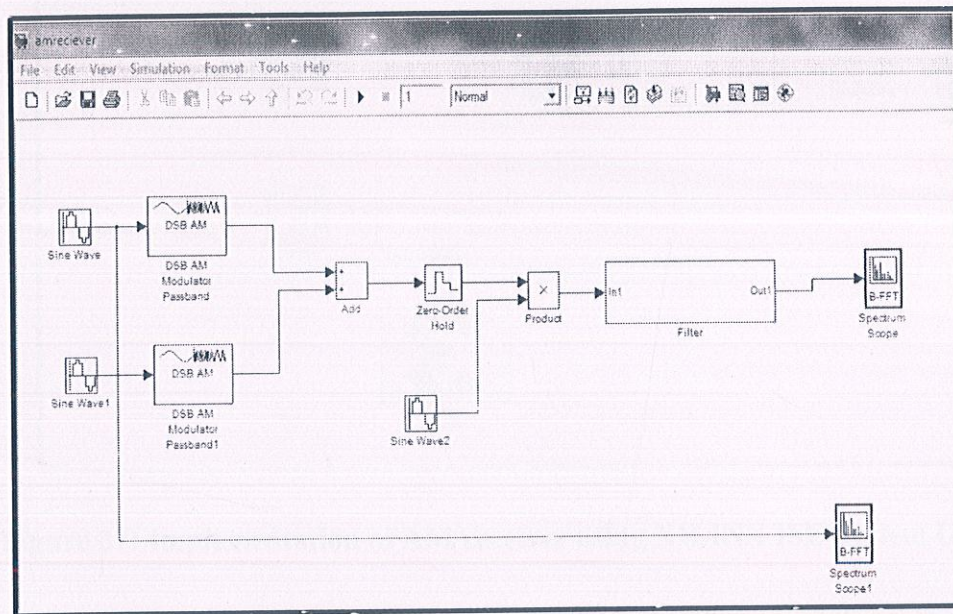
Figure 25: Output of time scope for a sine wave of frequency 3 kHz



## 4.1.2: Implementation of AM receiver

### A) Using MATLAB Simulink 7.1

The AM receiver was implemented on MATLAB Simulink and XILINX ISE System Generator. An AM-DSB modulator block was used to modulate the input signal which was used as the signal transmitted from the radio station. This signal was sampled using zero-order hold block.



**Figure 26: Implementaion of AM receiver on MATLAB Simulink**

The mixer is used to translate the message from high frequency to DC level. The 9-stage low pass filter is used to remove the unwanted high frequency components. The block is imported from the LPF simulink model realized above and the output of was observed through the spectrum scope.



## B) Using XILINX ISE System Generator 11.1

The AM Receiver model is constructed using Xilinx Blocksets. As can be seen, its structure is the same as the AM model built using MATLAB Simulink, except that all blocks are taken from the Xilinx Blocksets library.

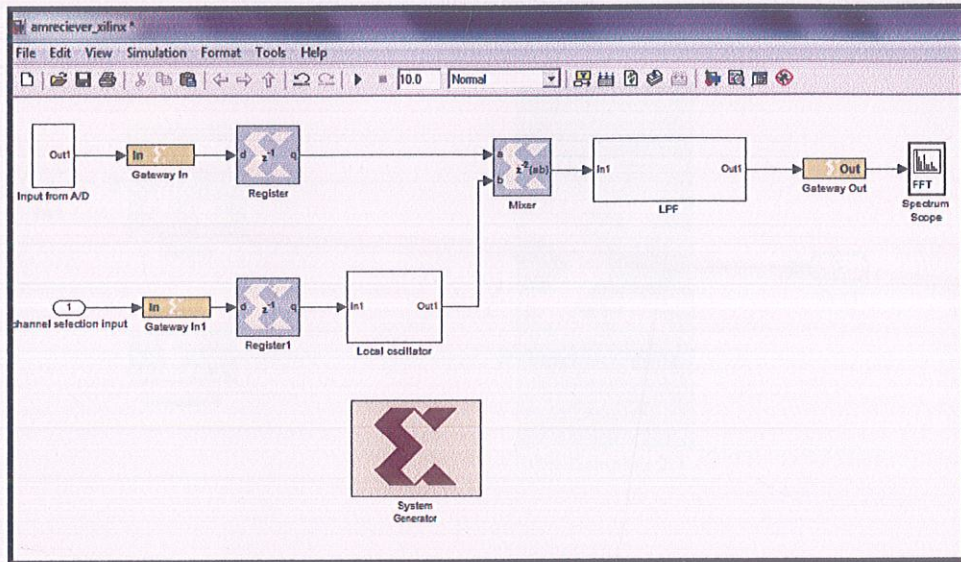


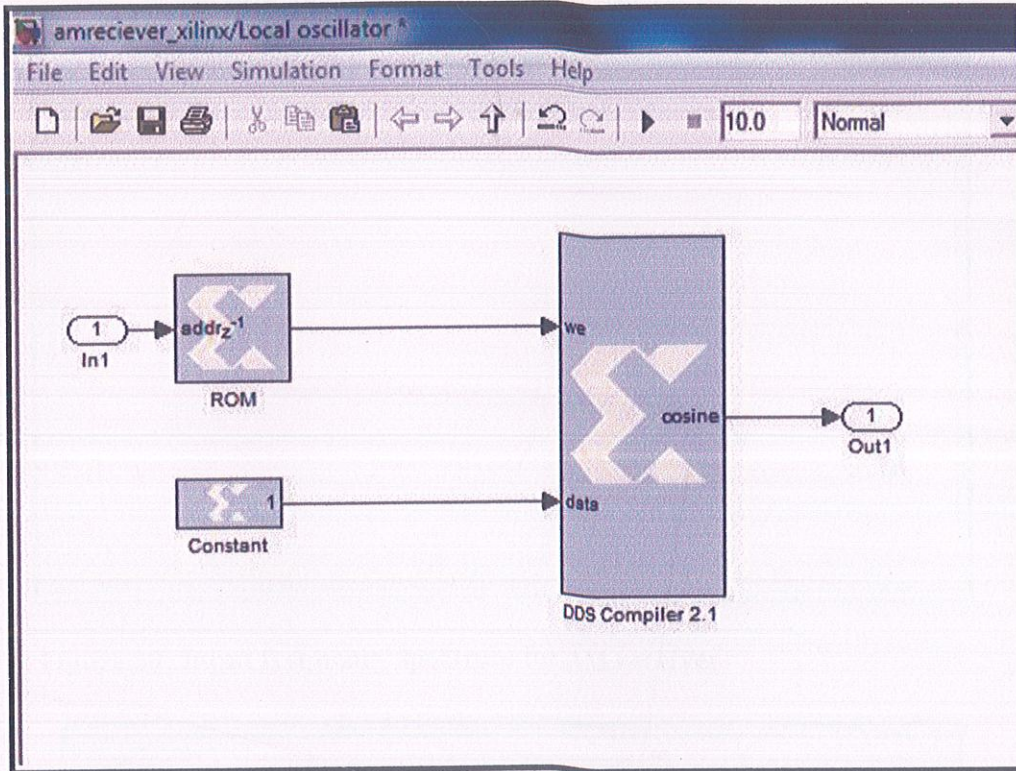
Figure 27: Implementation of AM receiver using XILINX ISE System Generator

Gateway In1 is the input signal sampled data, and is thus connected to the output of the ADC. Gateway In2 is the channel selection input, and is thus connected to user switches. Gateway Out is the demodulated signal, and connects to the input of the DAC.

The Low Pass Filter is the 9-stage filter as designed previously. It is constructed from Xilinx FIR and Downsample blocks. The filter coefficients are taken from the filter design in MATLAB FDA tool.



DIGITAL LOCAL OSCILLATOR: The Local Oscillator is built from a ROM and a Direct Digital Synthesizer (DDS) blocks as shown in Figure 28.



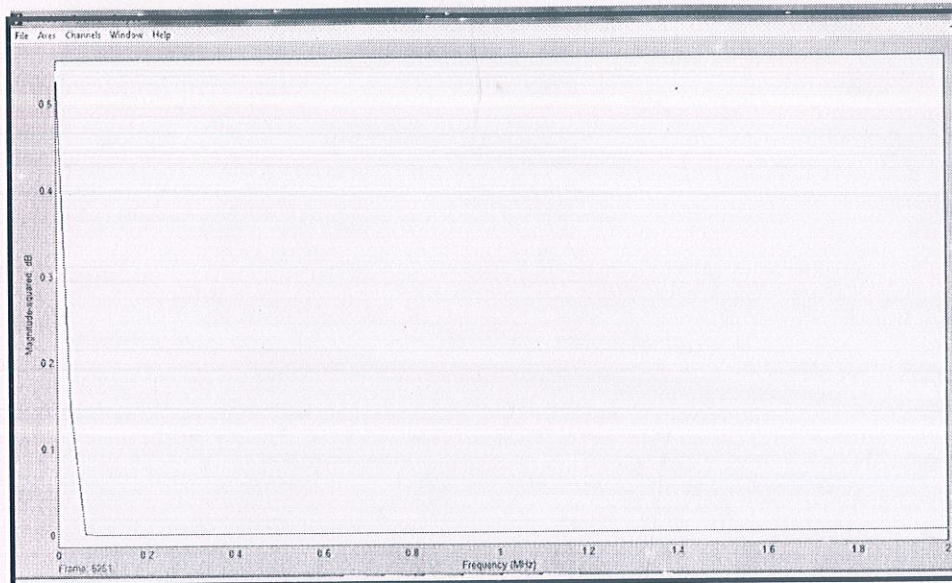
**Figure 28: Implementation of DLO using XILINX ISE System Generator**

The ROM structure stores all AM carrier frequencies from address 1 to address 120. The user selection input is used as the address from which to read the corresponding carrier frequency value. This value is the output of the ROM and is used by the DDS to generate a sampled sinusoidal signal.

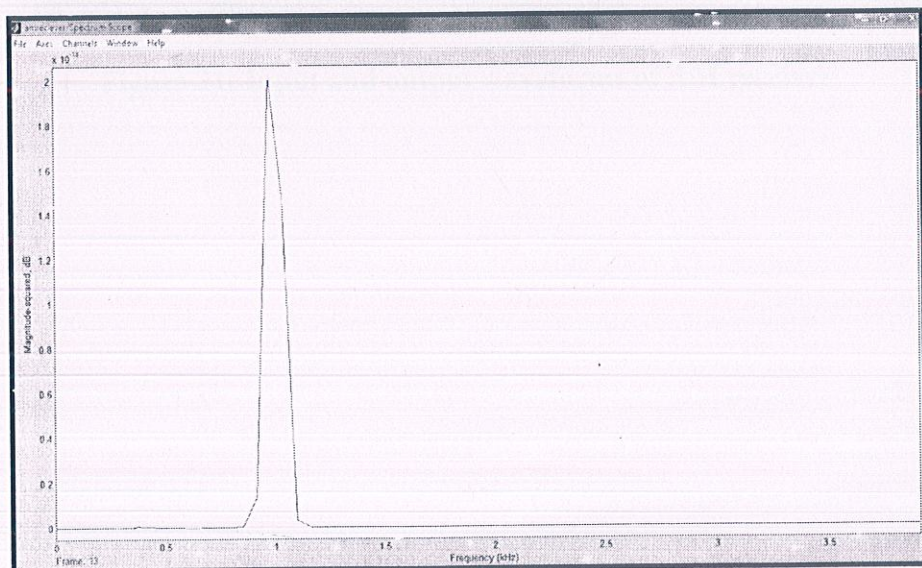
The DDS is used to generate a sinusoidal signal (cosine) of different frequencies depending on the input from the ROM. The output of the DDS is digital samples of the sinusoidal signal at frequency of the user selected channel.



For an input wave of frequency 1 kHz, the following input and output waves are observed on spectrum scope and time scope as shown in figure 29 and figure 30.

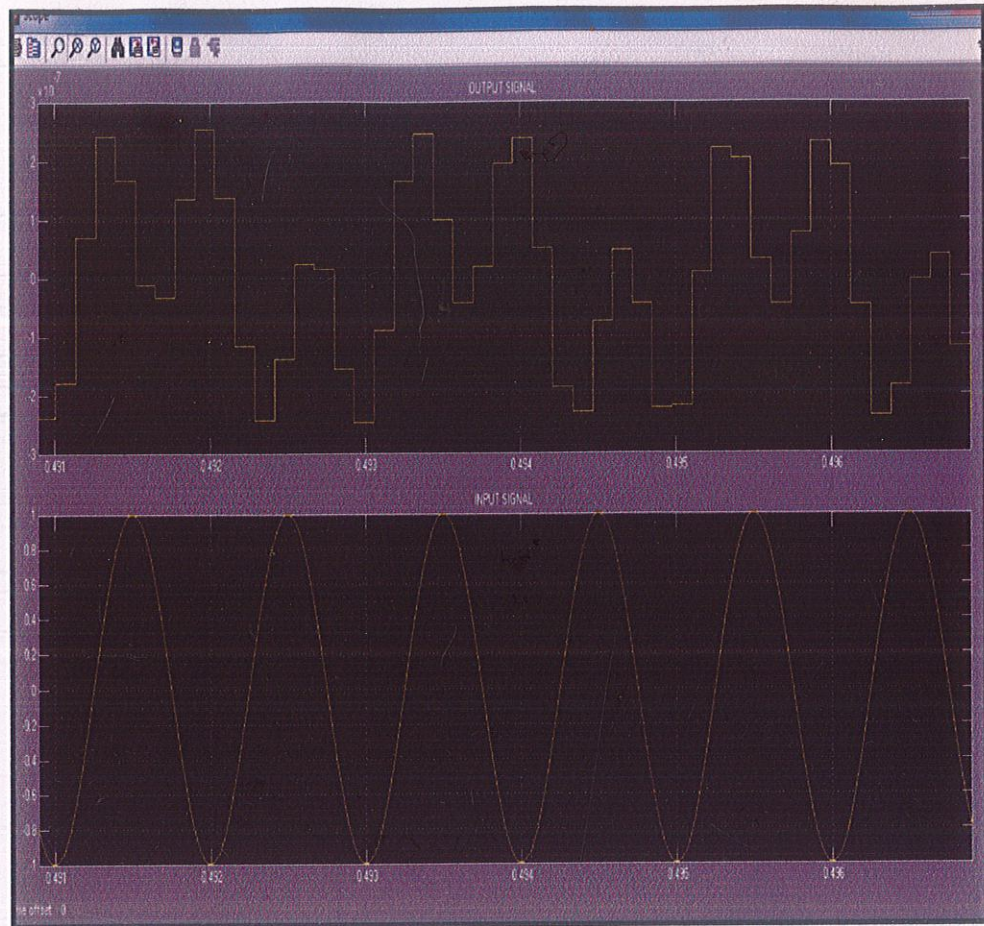


**Figure 29: Input frequency spectrum for AM receiver**



**Figure 30: Output frequency spectrum of AM receiver**





**Figure 31: Input and output waveforms of AM receiver**



## 4.2 FM Receiver

### 4.2.1 Implementation of FM-LPF

#### 4.2.1.1 Under-sampling technique

Since FM spectrum has the highest carrier frequency at 108 MHz, to satisfy Nyquist criterion, the signal must be sampled at around 216 MHz to avoid aliasing. However, this sampling rate is too fast for current ADCs technology (the ADS807 used in this project has maximum sampling rate of 53 MHz). Therefore, unlike the AM Receiver, for the FM Receiver, due to the high frequency signal, a different approach to sampling is needed.

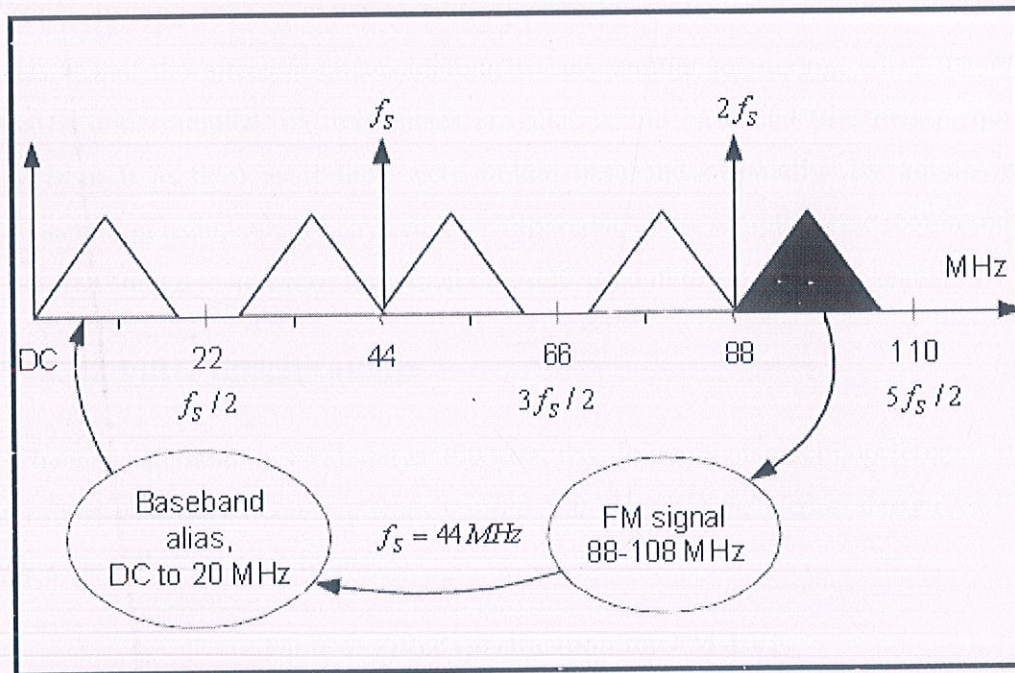


Figure 32: Under-sampling FM signal (88 MHz – 108 MHz) between DC and 20 MHz

Literature suggests a technique called “Undersampling”, which only samples signal at about twice its bandwidth (not twice the highest frequency in the signal). This technique applies for signals which do not extend to DC (bandpass signals), and for such signals the minimum sampling rate depends on the bandwidth of the signal as well as its position in the frequency spectrum. The sampling rate  $f_s$  is chosen such that the entire band of the bandpass signal lies within a single region of length  $f_s/2$ , not multiples of  $f_s/2$ . Above figure illustrates this technique by considering the FM signal



(88 MHz to 108 MHz) being sampled at 44 MHz. For  $f_s = 44$  MHz, the entire FM band lies within  $2 f_s$  and  $5 f_s / 2$ . After sampling, the actual FM signal between 88 and 108 MHz is aliased around multiples of  $f_s$ . Notice that any one of the alias components is an accurate representation of the original signal. In particular, the component lying in the baseband region from DC to 20 MHz is the output spectrum from the ADC, and is also the accurate representation of the original signal. In summary, the effect of undersampling is shifting the original spectrum down to baseband level, while keeping the frequency content of the signal intact. As a result of undersampling at 44 MHz, all carrier frequencies in the FM signal are now shifted down to baseband by an amount of 88 MHz.

Therefore, the original carrier frequencies at 88.1, 88.3, ... 107.9 MHz now appear at 100 kHz, 300 kHz, ... 19.9 MHz in the sampled data of the ADC. It can be observed that the effect of undersampling is very similar to mixing, when the FM signal is undersampled, its frequencies are aliased into baseband (the first Nyquist zone from 0 to  $f_s/2$ ) as if they were in the baseband originally. By employing undersampling technique, we have successfully reduced the sampling rate requirement of the ADC down to 44 MHz, for which the ADC used in this project is capable of.

#### 4.2.1.2 Filter Specifications

Channel separation in FM band is 200 kHz, thus to avoid channel interference, the filter must pass only frequency from 0 up to 100 kHz. We decided to design a filter having the following specifications.

**Table 6: Filter specification for FM-LPF**

FM Filter
$F_{\text{pass}} = 90$ kHz
$F_{\text{stop}} = 100$ kHz
$A_{\text{pass}} = 3$ dB
$A_{\text{stop}} = 30$ dB



Calculating filter order using the above specifications and formula for filter order in section 4.1.1.4, we get

**Table 7: Filter order for FM-LPF**

Specification	AM Filter
Sampling frequency	44000 kHz
$F_{pass}$	90 kHz
$F_{stop}$	100 kHz
<b>Order</b>	<b>4400</b>

#### 4.2.1.3 Calculation of filter order

Similar to AM filter, multi-stage filtering technique is utilized to build the FM filter. The filter order was calculated using the formula:

$$Filter\ order = \frac{F_s}{F_{stop} - F_{pass}} = \frac{44000}{11000 - 90} \approx 4$$

Similarly, the order for each stage was calculated and is summarized in table 8.

**Table 8: Calculation of filter order for FM**

Stage	Sampling Frequency (kHz)	Passband Frequency (kHz)	Stopband Frequency (kHz)	Filter Order
1	44000	0-90	11000	4
2	22000	0-90	5500	4
3	11000	0-90	2750	4
4	5500	0-90	1375	4
5	2750	0-90	687.5	5
6	1375	0-90	343.75	6
7	687.5	0-90	171.875	9
8	343.75	0-80	100	18
<b>Total number of filter taps required</b>				<b>54</b>



## A) Implementation of FM- LPF on Simulink

The simulink model of low pass filter was realized using simulink blocks and a sine wave was generated and the output was seen through spectrum scope.

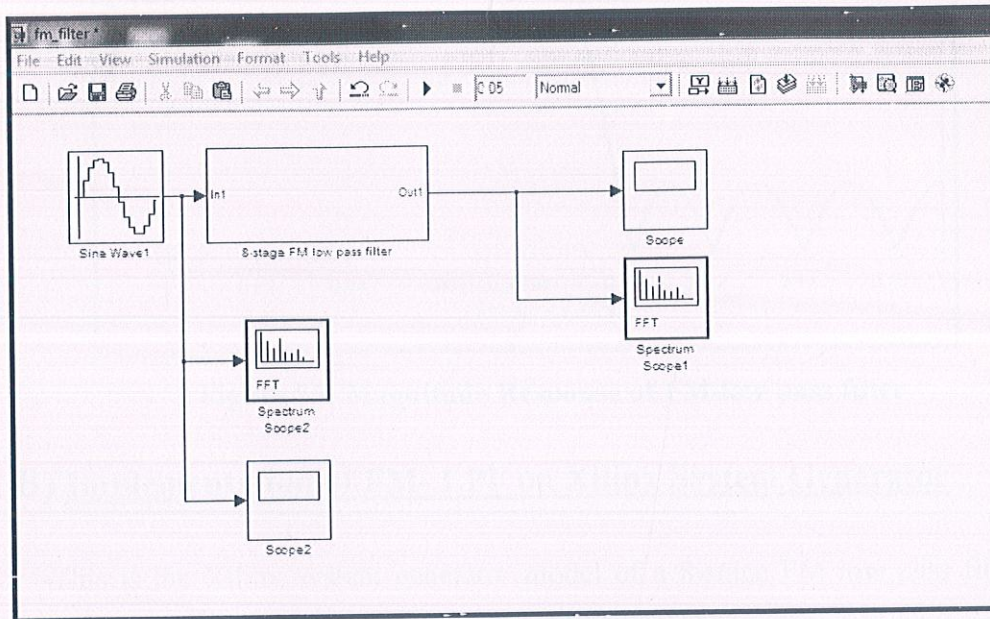


Figure 33: Simulink model of FM-LPF

This is a 8-stage low pass filter realized in MATLAB Simulink. The sampling frequency was down-sampled at every stage by 2 and the stop band was also halved at each stage.

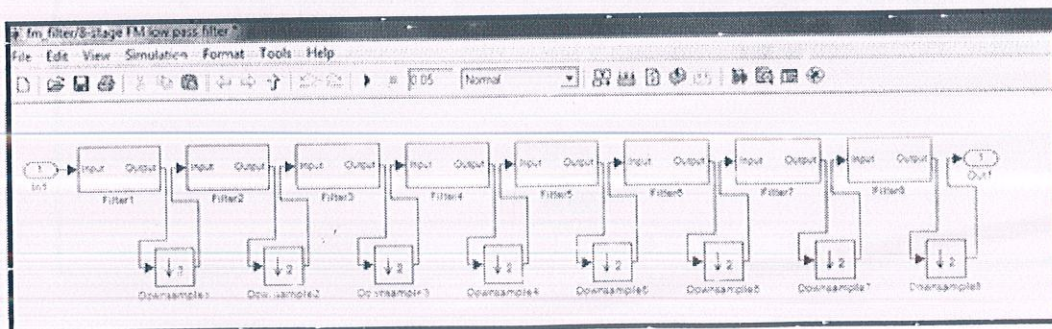


Figure 34: 8-stage FM-Low pass filter



This is the magnitude response of the 8-stage FM low pass filter.

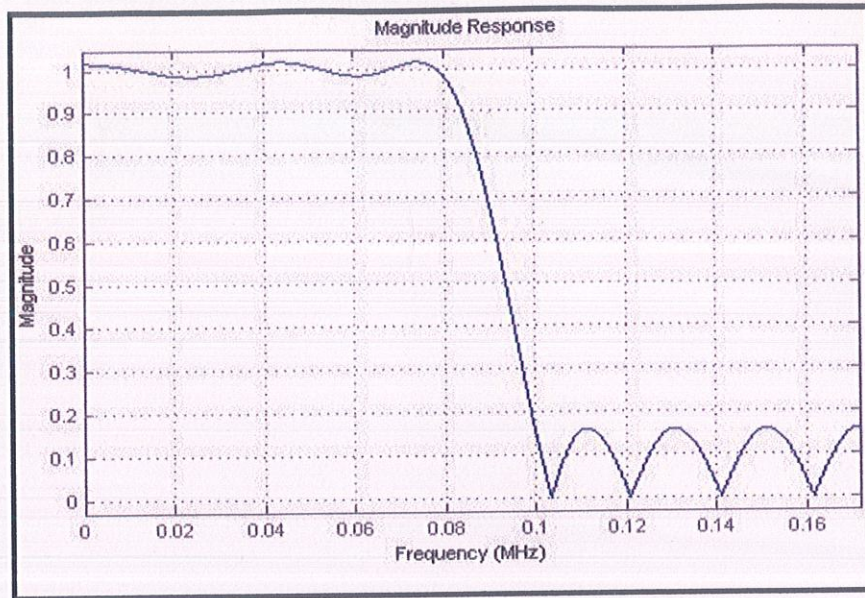


Figure 35: Magnitude Response of FM-low pass filter

## B) Implementation of FM- LPF on Xilinx System Generator

This is the Xilinx system generator model of a 8-stage FM low pass filter. The filter is designed by using FIR compiler block and the coefficients were calculated using FDA tool.

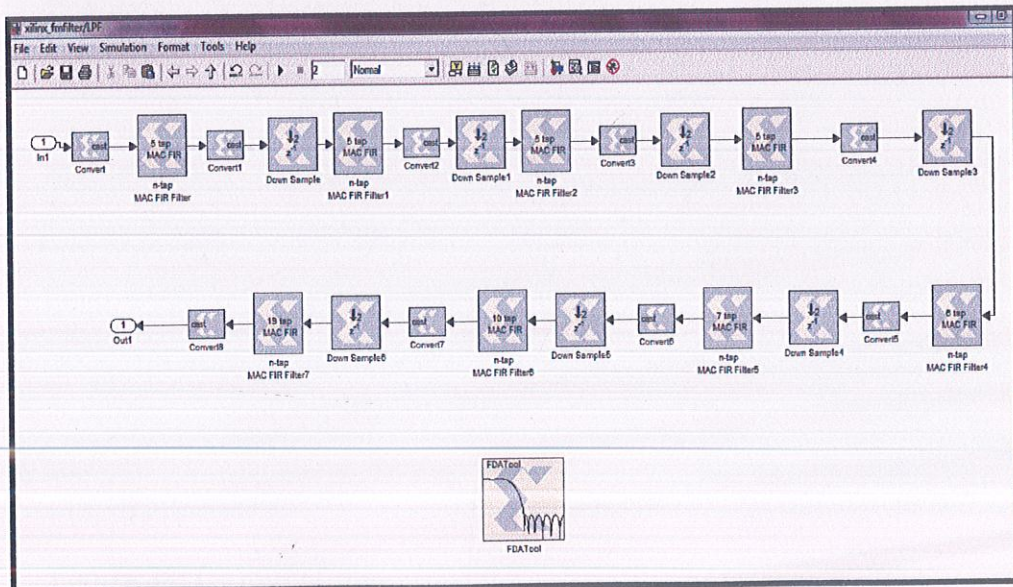
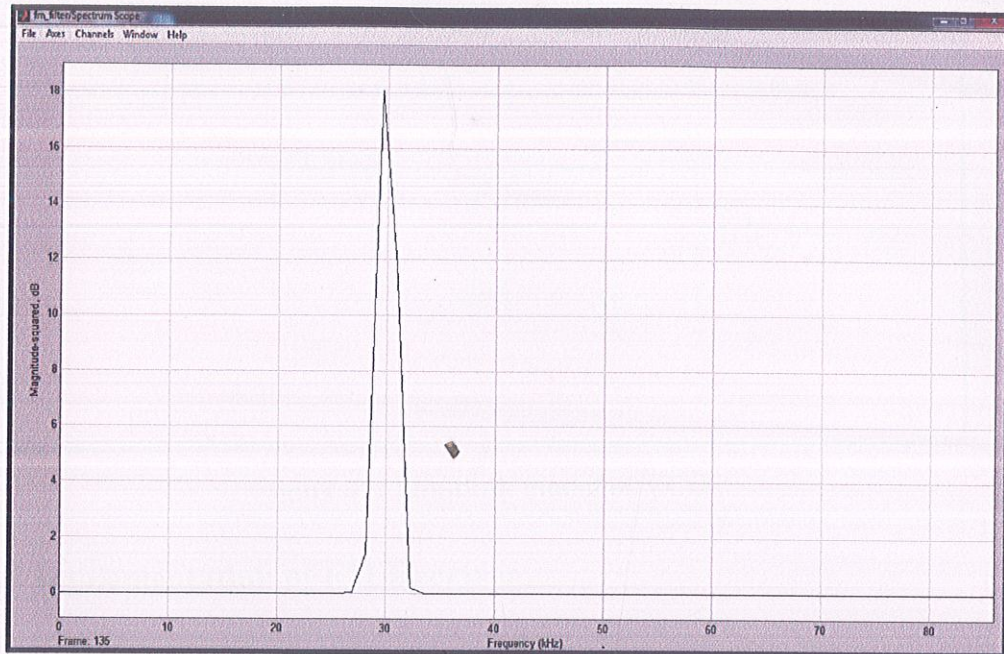


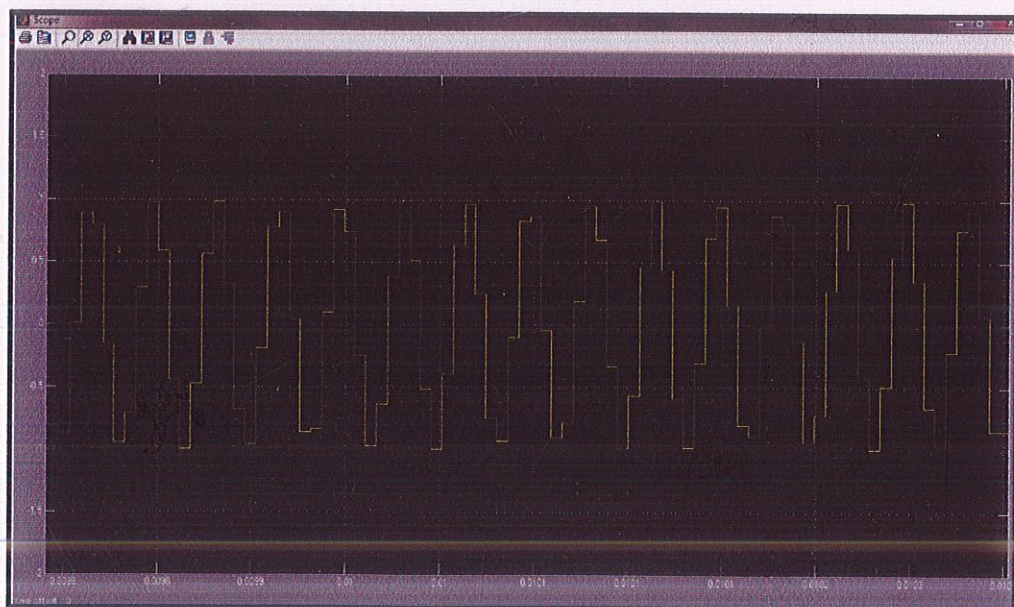
Figure 36: Xilinx System Generator model of 8-stage FM low pass filter



A sine wave of frequency 30 kHz was passed through the low pass filter and the output was seen through spectrum scope and time scope as shown in figure 37 and figure 38.



**Figure 37: Output of spectrum scope for a frequency of 30 kHz**



**Figure 38: Output of time scope for a sine wave of frequency 30 kHz**



### 4.2.3 Implementation of Voltage Controlled Oscillator(VCO)

The VCO generates a sinusoidal of a fixed frequency, in this case the carrier frequency  $f_c$ , in the absence of an input control voltage.

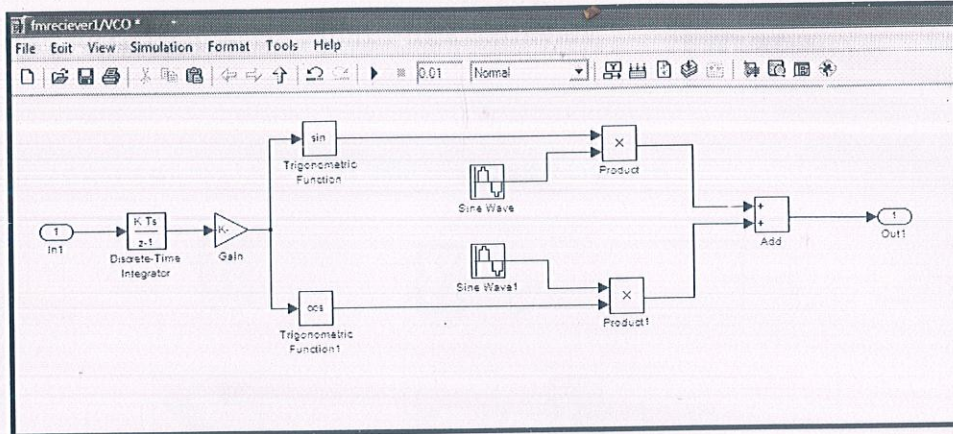


Figure 39: Simulink model of VCO

### 4.2.4 Implementation of FM Receiver

#### A) Using MATLAB Simulink

The FM receiver was implemented on MATLAB Simulink and XILINX ISE System Generator. A FM-passband modulator block was used to modulate the input signal which was used as the signal transmitted from the radio station.

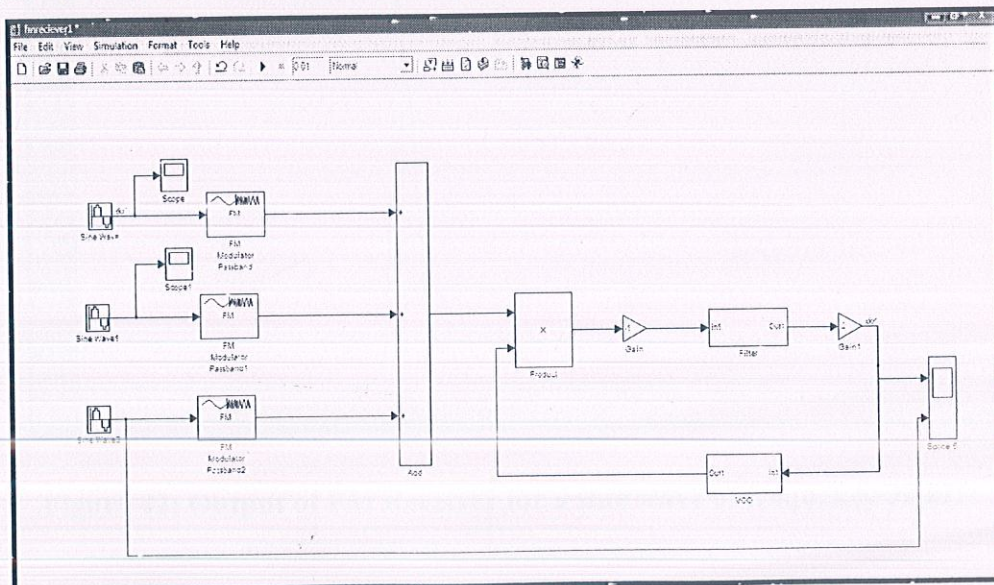
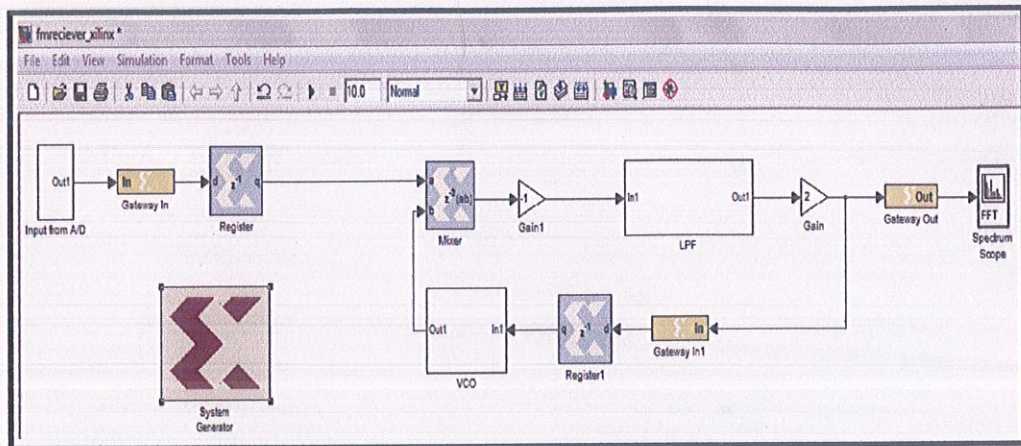


Figure 40: Implementation of FM receiver on MATLAB Simulink



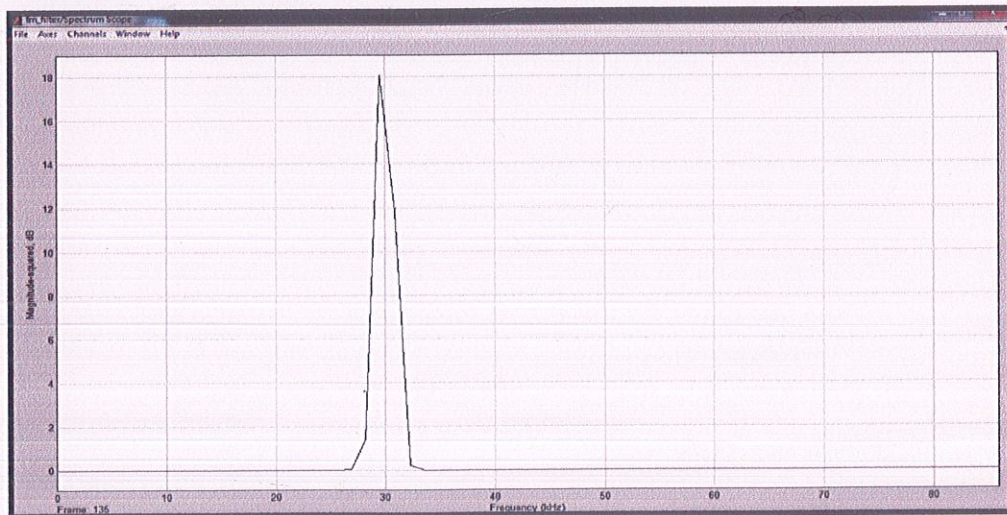
## B) Using XILINX ISE System Generator 11.1

The FM Receiver model is constructed using Xilinx Blocksets. As can be seen, its structure is the same as the FM model built using MATLAB Simulink, except that all blocks are taken from the Xilinx Blocksets library.



**Figure 41: XILINX ISE System Generator implementation of FM Receiver**

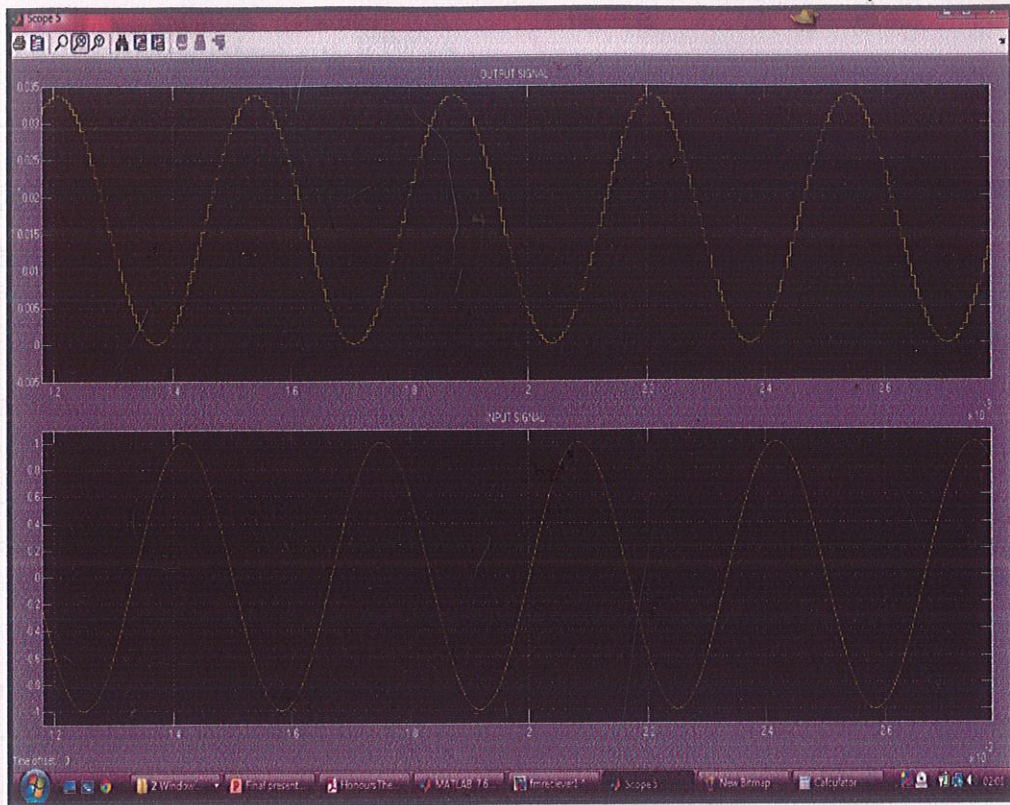
For an input wave of frequency 30 kHz, the following input and output waves are observed on spectrum scope and time scope as shown in figure 42 and figure 43.



**Figure 42: Output of FM Receiver for a sine wave of frequency 30 kHz**

The output and input for a sine wave of amplitude 1V and frequency 30 kHz as observed through time scope are shown below.





**Figure 43: Input and output waveforms of FM Receiver for a sine wave of 30 kHz**



### **4.3 Implementation on FPGA**

This section presents the implementation in FPGA of the AM and FM Receiver models developed in section 4.1 and 4.2. The AM/FM Digital Receiver is implemented on a XILINX Spartan XC 2S50 FPGA Development Board.

To implement the system in FPGA, it is necessary to describe our design models developed in Section 4.1 in Hardware Description Language (HDL). The software can then be used to synthesize the design into the FPGA platform.

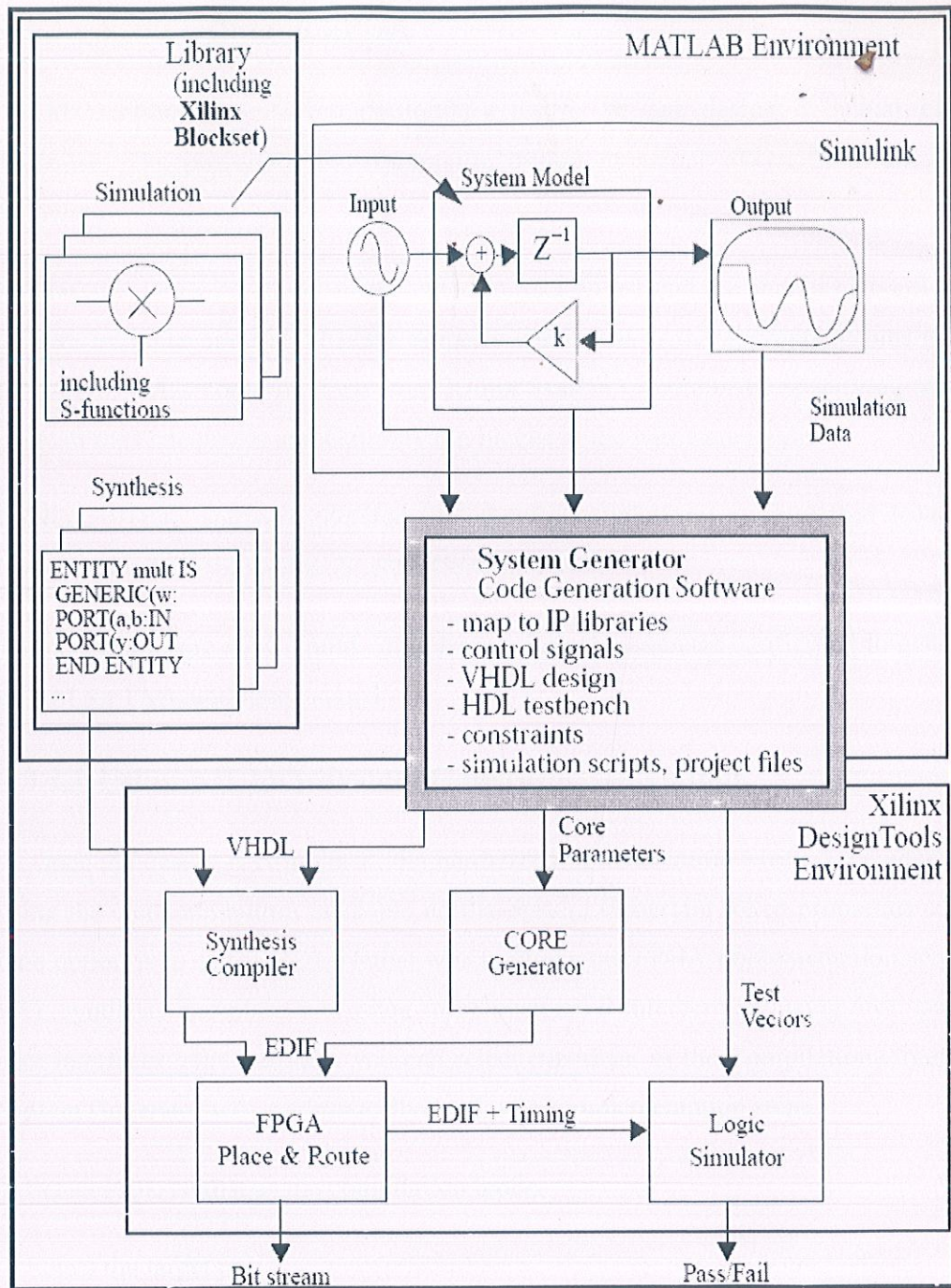
In this project, we chose not to describe our design directly in HDL, but used Xilinx System Generator software to speed up the development cycle. This tool allows us to describe the design in high level abstraction and the tool automatically maps the system to a faithful hardware implementation. System Generator allows us to model a system graphically from a set of common signal processing blocks such as adder, multiplier, filter etc.

#### **4.3.1 The System Generator Design Flow**

Simulink is a component of MATLAB software; it provides a graphical environment for creating and modelling dynamical systems. System Generator consists of a Simulink library called the Xilinx Blockset, and software to translate a Simulink model into a hardware realization of the model. System Generator maps system parameters defined in Simulink (e.g. as mask variables in Xilinx Blockset blocks), into entities and architectures, ports, signals, and attributes in hardware realization.

In addition, System Generator automatically produces command files for FPGA synthesis, HDL simulation, and implementation tools, so that the user can work entirely in graphical environments in going from system specification to hardware realization [7]. Figure 29 shows the System Generator Design Flow.





**Figure 44: System Generator Design Flow**

Note that in Section 4.1 and 4.2, we have developed Simulink models for both the AM and FM Receivers, thus the models were rebuilt using equivalent Xilinx Blocksets along with deciding on a suitable the number of bits to be used for digital representation.



### **4.3.2 System testing in FPGA**

This section presents the preliminary testing of our design in actual FPGA hardware. The procedure for testing the design in actual FPGA hardware is as following.

1. First, the hardware model as presented in Section 4.1 and 4.2 should be configured to have its input and output ports properly connected to the appropriate pins of the FPGA chip. This could be done from Xilinx System Generator by specifying the pin name in the Gateway in and Gateway out blocks in the hardware model.
2. Next, VHDL software targeted into the FPGA platform is generated from the hardware model using Xilinx System Generator tool.
3. The input to the ADC could come from a Signal Generator configured to generate an AM (or FM) modulated signal.

#### **4.3.2.1 Using XILINX System Generator Token tool**

Once the design is completed, the hardware implementation files can be generated using the Generate button available on the System Generator token properties editor. One option is to select HDL Netlist which allows the FPGA implementation steps of RTL synthesis and place and route to be performed interactively using tool specific user interfaces. Alternatively, you can select Bitstream as the Compilation target and System Generator will automatically perform all implementation steps.

1. Select HDL Netlist as the compilation mode.
2. Select the target part.
3. Select HDL language.
4. Set the FPGA clock period.
5. Generate HDL code.



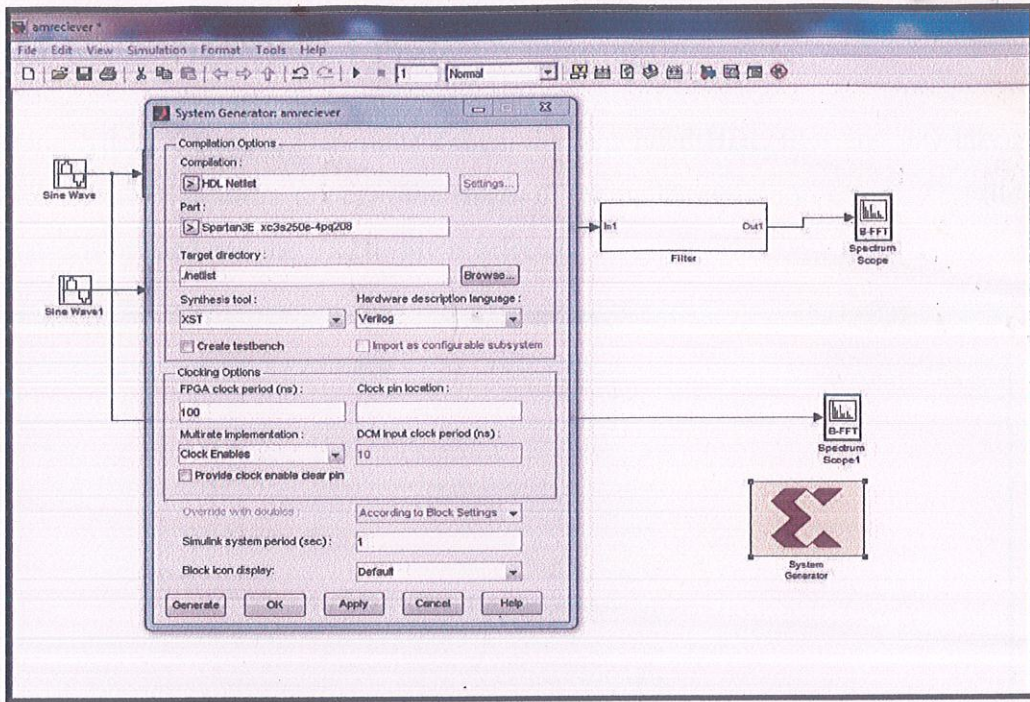


Figure 45: Selecting compilation mode and target part

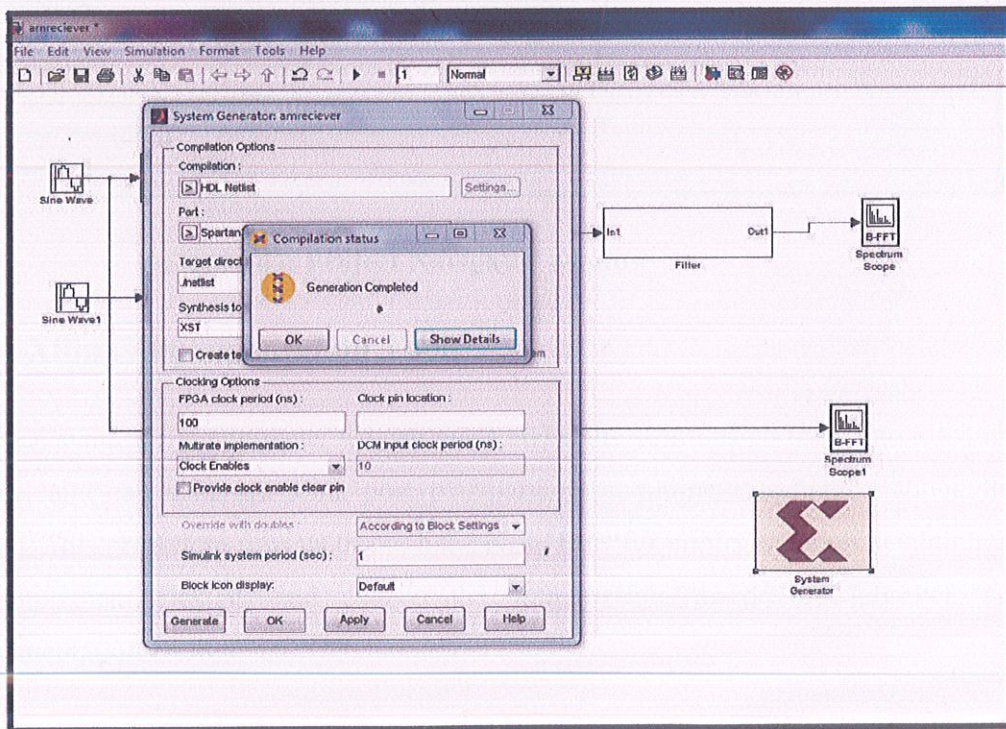


Figure 46: HDL code generation



### 4.3.2.2 Using XILINX 11.1 Project Navigator

Using XILINX System Generator Token, the generated HDL files are now taken into XILINX ISE in order to begin the stages of taking the design to FPGA. The file generated is opened and the VHDL file is selected.

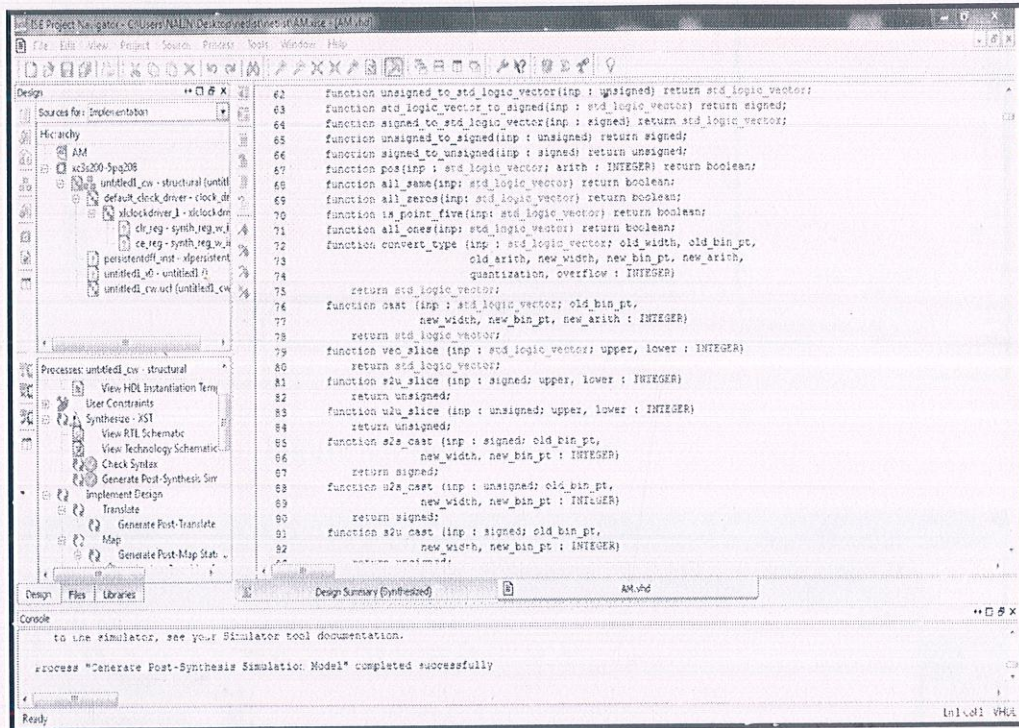


Figure 47: Project Navigator window

### 4.3.3 Xilinx implementation Tools

Once synthesis is complete, we can place and route your design to fit into a Xilinx device, and can also get some post place-and-route timing information about the design. This procedure runs us through the basic flow for implementation. Right-click on Implement Design, and choose the Run option, or double left-click on Implementation Design.



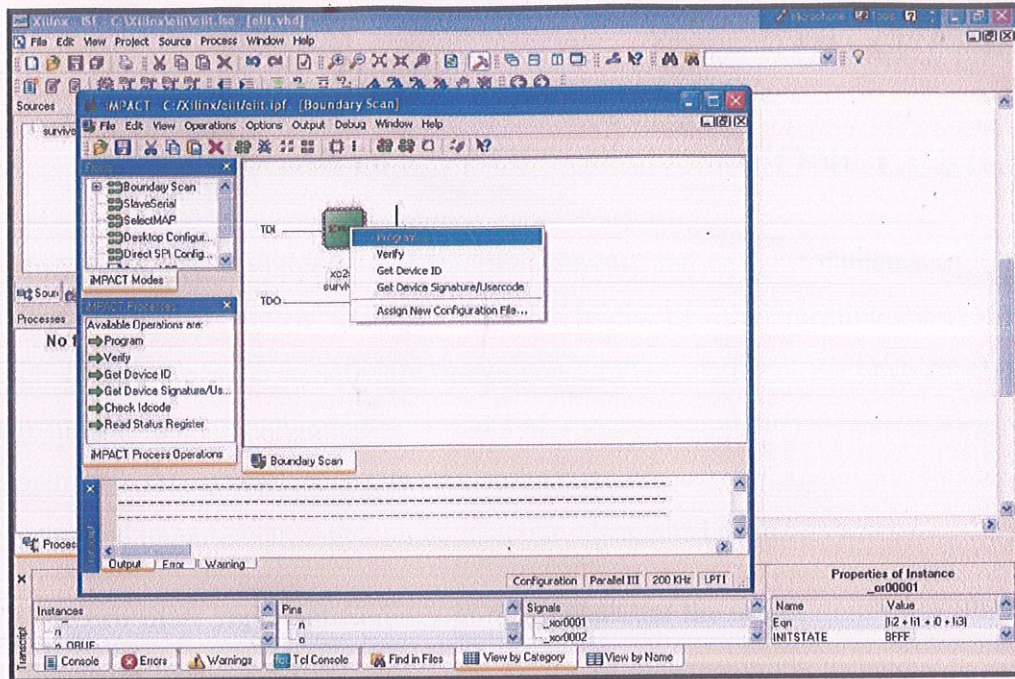


Figure 48: Impact window

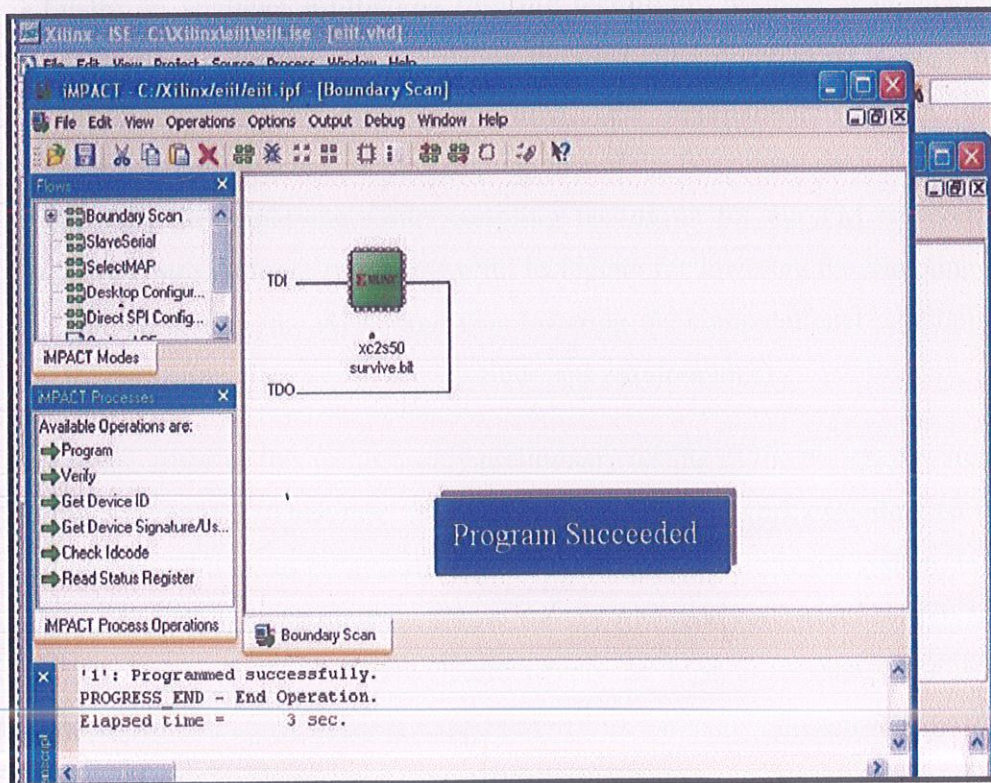


Figure 49: Device successfully programmed



## CHAPTER 5

### CONCLUSION

In this project, we have looked into the concepts and design techniques of a new digital technology called Software Defined Radio by implementing an AM/FM Digital Radio Receiver in software running on a FPGA platform. We have developed two models for demodulating AM and FM signals. Simulation of our models has shown that the general concepts and techniques of software radio are feasible in general and in particular to the implementation of an AM/FM radio receiver.

The two most important findings in this project are the use of multistage filtering technique to reduce the filter order, and the use of undersampling technique to lower the sampling rate requirement for the ADC.

- Firstly, in applying **multistage filtering technique**, we have mathematically formulated a formula for estimating the total filter order in an N-stage filter which reduces the number of filter taps approximately by a factor of  $(F_s/4\log_2 F_s)$  where  $F_s$  is the sampling rate of the filter input data.
- Secondly, in employing **undersampling technique** for our FM receiver, we have confirmed that it is a powerful technique for lowering the sampling rate requirement for the ADC, and thus lowering the computational requirements for the entire system (e.g. lowering filter taps requirements).

Due to the scope of this project, only preliminary testing of the designed system in FPGA was carried out. Therefore, the most important future work extending on from this project would have to be a comprehensive testing of the designed system. In this project we have designed **Low Pass Filter**, for our AM receiver system as well as for FM receiver system. These filters should ideally be high order filters but due to techniques specified above order is reduced which increases its efficiency.

Overall, this project have been a worthwhile effort in a sense that it has been a good learning process. This project posed an open-ended problem, for which we had to make our own choices, and use our own initiative to find suitable solutions. We thought this project has left us better prepared for real-world projects.



## REFERENCES

1. Wipro Technologies, Software-Defined Radio- A technology overview, White Paper, August 2002, 10 pages
2. John G. Proakis and Dimitris G. Manolakis, Digital Signal Processing-Principles, Algorithms and Application, Third Edition, Prentice-Hall International Inc., 1996, 1033 pages.
3. Micheal D. Cilleti, Advanced Digital Design with the Verilog HDL, Second Edition, Prentice Hall, 2011
4. Thanh H. Le, AM/FM Software Radio Receiver Implementation in FPGA, 2004
5. "*Xilinx System Generator v2.1 Reference Guide*", [www.mathworks.fr/applications/dsp\\_comm/xilinx\\_ref\\_guide.pdf](http://www.mathworks.fr/applications/dsp_comm/xilinx_ref_guide.pdf)
6. System Generator for DSP-Getting started guide, release 10.1, March 2008, Xilinx
7. Peter Thorwartl, Implementation of a Digital Receiver with FPGA's, third edition, 2003.
8. Andrew S. Tanenbaum, Computer Networks, Prentice Hall, 2003, 384 pages.