Systems for Fractional Sampling Rate Conversion

by

Naina Rao Nagrale

A Dissertation
Submitted to the Program in Electronics
Department of Electronics
in partial fulfillment of the requirements for the degree
of

MASTER IN SCIENCE SPECIALIZED IN ELECTRONICS

at the

National Institute of Astrophysics, Optics, and Electronics

July-2007
Tonantzintla, Puebla, MEXICO

Supervised by:

Dr. Gordana Jovanovic Dolecek
Dr. Jorge Francisco Martinez Carballido

Abstract

This thesis treats the design of decimation filters for fractional sampling rate conversion system, applicable for Software Radio (SR).

We first present a general introduction of Software Radio technology along with the brief description of radio system architecture, multirate techniques, and programmable logic devices (PLDs). Then, the basic principles of Sampling Rate Conversion (SRC) is presented, including various methods for decimation and their impacts on the filter complexity.

Further, a review of the Finite Impulse Response (FIR) filters has been done, along with the methods of rounding and sharpening for efficient filter design. The raising technique, block filtering and time varying polyphase structures are also revised.

Overview of existing sampling rate conversion filters has been presented, which includes Cascaded Integrated Comb (CIC) filter, Polynomial filter and Time Varying CIC filter.

Using multirate techniques and polyphase representation, a simple method of designing fractional sampling rate conversion system based on the Interpolated Finite Impulse Response (IFIR) filter has been proposed.

Finally, the proposed algorithm is simulated in MATLAB and implemented in the SPARTAN 3 family of the Xilinx's Field Programmable Gate Array (FPGA) using project navigator Xilinx ISE 8.2i. The input and output of the implemented structure is verified in Symphony EDA Sonata 3.1. All the proposed MATLAB functions and VHDL programs are included in Appendix.

Resumen

La Presente tesis trata el diseño de filtros de decimación para sistemas de conversión de la razón de muestreo fraccional utilizable en Software Radio.

En principio se presenta una introducción general de la tecnología Software Radio incluyendo arquitectura, técnicas de multirazon y dispositivos lógicos programables.

Posteriormente, se muestran los principios básicos de conversión de muestreo así como los métodos de decimación y su impacto en la complejidad de filtros.

Además, se presenta una revisión de los filtros, FIR (Respuesta al impulso Finita), incluyendo los métodos redondeo y afilado para el diseño eficiente de filtros, técnicas de elevación y metodos block filtering.

Una vista general de filtros de razón de muestreo existentes es presentada, la cual contiene los filtros Cascaded Integrated Comb (CIC) Filtro, filtro Polynomial, and Time varying CIC FIlter.

Se propone un método simple de diseño de un sistema de conversión de la razón de muestreo fraccional basada en filtros Interpolated Finite Impulse Response (IFIR) usando técnicos multirazon y representación polifase.

Finalmente, el algoritmo propuesta es simulado en MATLAB e implementado mediante SPARTAN 3 familia de Field Programmable Gate Array (FPGA) usando navegador Xilinx ISE 8.2i. La entrada y la salida de la estructura implementada es verificada en Symphony EDA Sonata 3.1.

Todas las funciones MATLAB propuestas y las programas VHDL son incluidos en el Apéndice.

Acknowledgements

This thesis is conducted by the means of scholarship awarded by the Government of Mexico through the Ministry of Foreign Affairs.

The work is a part of CONACYT project number 49640 multirate digital signal processing for Software Radio.

I want to express my deep gratitude to the National Institute of Astrophysics, Optics, and Electronics (INAOE) for giving me the opportunity to study Master in Science with specialization in Electronics.

I am very thankful to Dr. Gordana Jovanovic Dolecek, for her valuable support in constructing the work and to Dr. Jorge Francisco Martinez Carballido, for teaching me VHDL language and implementation in SPARTAN 3.

Thanks to Dr. Jose Alexandro Diaz Mendez, Dr. Miguel Angel Garcia Andrade, and Dr. Roberto Rosas Romero, for reviewing and giving comments to improve the thesis.

Special acknowledge to Dr. Alfonso Fernández Vázquez for his precious time and suggestions he gave for discussing the problems.

Deeply thanks to all who are in INAOE and have given me a family environment.

Dedications

To the God, His creations and My Mother

Preface

The principal goal of this thesis is to design an efficient fractional sampling rate conversion system applied in the Software Radio (SR) Technology.

In the following we discuss the importance of this research work and the methodology we used to fulfill the desired goal. With an exponential blowup of cellular mobile systems and plethora of analog and digital standards attempting to crowd the radio spectrum all over the world [20], Software Radio concept is emerged as a potential pragmatic solution for an interoperability across the diverse communications bands. The term Software Radio stands for radio functionalities defined by software, which implies the use of digital signal processors (DSPs) to execute the necessary software which implements radio interfaces and upper layer protocols, in real time [6].

The main property of software radio is to design an independent terminal irrespective of various air interface communication standards. These communication standards are generally based on different master clock rates and therefore employ different bit/chip rates. An efficient solution for different master clock rates is to run the terminal on a fixed clock rate and perform digital sample rate conversion controlled by software [11].

In general sampling rate conversion is well known as multirate techniques. A key characteristic of the multirate algorithms is their high computational efficiency. In many cases, these algorithms are the prime reason for an application to be implemented economically using modern digital signal processors. In this work, we consider multirate based design for the fractional sampling rate conversion systems.

Going towards the desired goal, the following conditions has to be satisfied:

- The stopband attenuation of the filter should be high enough to reject the adjacent channel interferences.
- Shifting the filter from high sampling rate region towards low sampling rate region.
- Try to decrease the filter complexity by eliminating multipliers.
- Power consumption should be minimum.
- FPGA implementation with minimum gates utilization.

Chasing these conditions, a brief introduction of the Software Radio (SR) technology, Sampling Rate Conversion (SRC), and Programmable Logic Devices (PLDs) has been given in Chapter 1.

Various multirate algorithms have been reviewed in Chapter 2.

Next, techniques for improving the filter characteristics such as rounding and sharpening techniques have been discussed in Chapter 3 along with time varying filter analyzing methods.

Overview of various existing filters for fractional sampling rate conversion has been presented in Chapter 4.

The proposed algorithm has been given in Chapter 5 at low sampling rate as well as at high sampling rate.

The implementation of proposed algorithm in Xilinx SPARTAN 3 family of FPGA has been done in Chapter 6. This chapter describes the implementation cost through device utilization summary generated by Xilinx ISE 8.2i. The input-output waveforms have been simulated in Symphony EDA Sonata 3.1.

Finally, the comparative study of proposed structures at low and high sampling rate is presented. The implementation of proposed structures has also been compared with respect to the FPGA equivalent gates utilization with and without multipliers.

MATLAB functions for the proposed algorithm are listed in Appendix A. VHDL programs details are provided in Appendix B.

The list of publication relating to the thesis is presented in Appendix C.

Contents

1

Introduction of Software Radio

This chapter presents the basics of Software Radio. The necessity of Software Radio technology along with the advantages and unsolved problems have been described. Radio system architecture for Software Radio is explained. Finally, introduction of Sampling Rate Conversion (SRC) process and Programmable Logic Devices (PLDs) have been demonstrated.

1.1. What Software Radio is and why we should use it

1.1.1. Definitions

The term Software Radio (SR) was first introduced by Joe Mitola in May 1992 [20], to refer multiservice, multistandard, multiband, reconfigurable, and reprogrammable radio system. This radio system possess the objective of liberating traditional radio based services from hardware dependency on frequency band, channel bandwidth, and channel coding scheme. Mitola explained Software Radio as a combination of techniques that include multiband antennas, Radio Frequency (RF) conversion, wideband Analog to Digital (A/D) and Digital to Analog (D/A) conversion, and implementation of Intermediate Frequency (IF), baseband (BB), and bitstream processing functions in general-purpose programmable processors.

Later, many researchers had presented different definitions to describe Software Radio more completely. For example Enrico Burrachini [6] has defined Software Radio system as the dynamic adaptation of user terminal to various radio environment irrespective of

location, and defining the radio functionalities through software by utilizing DSPs. After knowing completely what is Software Radio, the questions which remain are Why we need Software Radio and What makes researchers to work for Software Radio. Here, are some of the most burning factors for the evolution of Software Radio

- After the migration of radio systems from analog to digital, the exponential blowup of cellular mobile systems lead to the jockeying of telecommunications giants in the international and national standards. The most adverse effect was face by the military and aerospace. As a result, in mid 70's the U.S. Advanced Research Projects Agency (ARPA), Army, Air Force, and others in Europe began to develop Software Radio technology for interoperability across diverse communication bands. Recently, the commercial market also demanded Software Radio for free mobility across international boundries.

- Another most important factor for such a wide revolution of Software Radio is, the industrial competition among different continents. This lead to the demand of a common worldwide standard for future mobile systems which has been justified will raise trading benefit among them.

1.1.2. Multidimensional view of Software Radio

Considering all these factors, recent advances in Software Radio include an interesting multidimensional view of Software Radio presented by Stephen M. Blust of *Cingular Wireless* [29], as a model of integration of software with hardware for communication in the worldwide network (see Fig. 1.1). This multidimensional aspect of SR has following four main planes

1. Radio implementer plane,
2. Network operator plane,
3. Service providers plane, and
4. User applications plane.

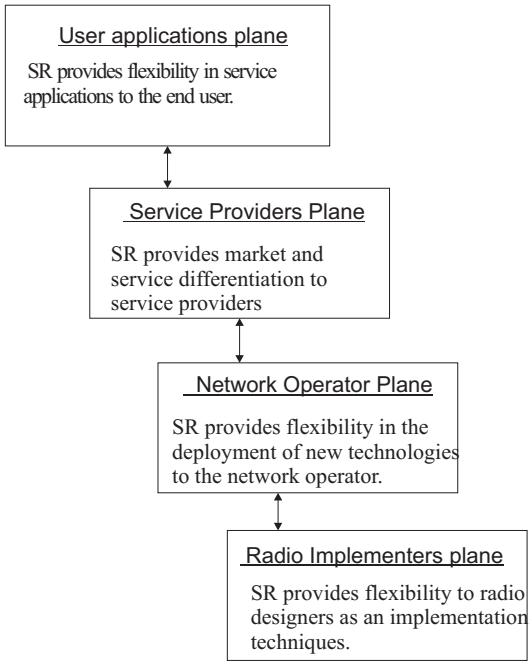


Figure 1.1: The Multidimensional perspectives of software based radio.

Each of these four planes has a specific task to perform and an interactive interface among them will make a complete Software Radio system. A brief explanation of all these four planes is given below

1. **Radio Implementer Plane**: In this plane, the traditional methods of implementing transmitters and receivers have been replaced by multimode, multistandard, adaptive, reconfigurable, and reprogrammable radio system. Mitola described this plane as *channel processing streams* [21]. Radio Implementer Plane has been divided into five main segments, these are:

   - *Antenna Segment* : This segment contains multiple antennas per band or octave bandwidth antennas and an acute frequency reference in the RF segment. The main objective of this segment is to reduce interference from large number of interferers using one or no auxiliary antenna. The structure of the antenna arrays determines the number of distinct physical and logical signal processing paths in the RF conversion and IF processing segments. Thus, the antenna segment of Software Radio must be capable of directional selectivity, multipath compensation and interference suppression.

- *RF conversion segment* : RF conversion includes output power generation, preamplification, and conversion of RF signals to and from standard intermediate frequencies (IFs) suitable for wideband A/D/A conversion. The main problems of this segments are the need of amplifier linearity and efficiency across the access band, introduction of processor clock harmonics, and electromagnetic interference (EMI). These problems can be resolved to some extent by placement of A/D/A converters before final IF section and filters.
- *IF Processing segment* : The function of IF processing segment is to map the transmitted and received signals between modulated baseband and IF sections. The IF receiver processing segment includes wideband digital filtering to select a service band from those available. Furthermore, IF filtering recovers medium band channels (e.g., a 200 kHz TDMA channel in GSM) and/or wideband subscriber channels (e.g., a 2 MHz CDMA channel) and converts the signal to baseband.
- *Baseband Processing Segment* : The baseband segment imparts the first level of channel modulation onto the signal (and conversely demodulates the signal in the receiver). Predistortion for nonlinear channels would be included in baseband processing. Trellis coding and software decision parameter estimation also occur in the baseband processing segment. The complexity of this segment therefore depends on the bandwidth at baseband, the complexity of the channel waveform, and the complexity of related processing.
- *Bitstream Segment* : Digital multiplexing of source coded bitstreams from multiple users is being done by the bitstream segment. It imparts forward error control (FEC) onto the bitstream, including bit interleaving, frame alignment, bit-stuffing, and radio link encryption. Signaling, control and operations, administration and maintenance functions are also provided in the bitstream segment. These functions are even-driven and typically impart an order of magnitude less computational demand than baseband processing.

2. **Network operator plane** : It works to make the wireless network dynamic for

existing and new standards by downloading software. Due to the frequent mobility of user terminal, software download in the mobile terminal must be as fast as possible. There are two methods of software download :

- *Smart card loading* : Under this method, the user purchases a smart card from the sell point of the network operator. The software download is performed when the smart card is inserted in the terminal. This method is error-free and fast, but in case of change of location, user has to change the smart card with respect to the network area.
- *Air interface download* : In this method, the network operator performs the software download effort when the user terminal changes its location. It is intelligent updating of user terminal when user is traveling through an area served by different cellular systems. This is comparative complicated method with high storage requirement and occurrence of errors that make it slow.

3. **Service Provider Plane** : Software radio allows service providers a mechanism for service differentiation, and a cost-effective means for easily upgrading the network infrastructure through software downloads to the network and handset components.
4. **User Application Plane** : It provides user's perspectives to software radio advantages, by the ability to excess the hardware-software combination at any moment.

Thus, these four planes of software radio architecture can be viewed as the critical enabling foundation for the potentials and benefit of software radio.

1.1.3. Features, Goals, and the Benefits

By the efforts made by different investigators, the obvious encountered features of SR can be summarized as

- System reconfigurability.
- Increased hardware-lifetime.
- Novel service quality levels.

- Multiband and Multimode applications.
- Reduced hardware size, weight and power consumption.
- Cost effective and high capacity chips.
- CDMA and TDMA applications compatibility.
- Multimedia.
- Integrated personal communications services (PCS), land mobile and satellite mobile services.
- Dynamic adaptation.

To effectively achieve all these characteristics, Software Radio has to obtain some principle goals like

- To move the border between the analog and digital world as much as possible towards radio frequency (RF) by adapting analog-to-digital (A/D) and digital-to-analog (D/A) wideband conversion as near as possible to the antenna.
- To replace application-specific integrated circuits (ASICs) (dedicated hardware) with DSPs for baseband signal processing, in order to define as many radio functionalities in software.
- To download air interface standards dynamically.
- To make software realization of frequency band, channel bandwidth, channel modulation, coding scheme, and user applications.
- To design flexible, controllable and programmable architecture.

Having SR with all these properties will bring benefits to many actors involved in the telecommunications market like: manufacturers, operators, and users [6].

- For the manufactures, there is the possibility to concentrate on Research & Development efforts on a reduced hardware platform set. The device can be applicable

to every cellular system and market, irrespective of national or regional bounderies. As a consequence, mass production would allow lowered cost. Another relevant advantage would be the possibility to improve the software in successive steps, and the correction of software errors and bugs discovered during operation.

- In case of operators, SR helps to rapidly roll out new services tailored to the needs of each user and differentiated from those of other operators. With the same terminal it will be possible to provide all services even if supported by different communication standards. In addition, there is the possibility to implement multistandard base stations.

- And with SR, the users can enjoy the possibilities to roam to other cellular systems and take the advantage of worldwide mobility and coverage. Including this, users can also configure their terminal according to their preferences.

1.1.4. Pitfalls

Besides providing all these flexibilities, Software Radio is accompanied with various problems. Most of these pitfalls can be avoided, and continued research in these difficulties will further reduce costs and time to market [6]. Some of these problems are detected as

- Designing of wideband low loss antennas and RF converters, which is difficult to accomplish for fullband digitization.

- It is also difficult to accurately estimate processing demands of applications and processing capacity of reprogrammable DSP/CPU configurations.

- Data rates requirements across interprocessor interfaces is problematic.

- Open architecture standards for high capacity wideband signal buses have not yet emerged.

- Ability to mix and match real time software tools and modules from different software suppliers is yet to be achieve.

1.1.5. Applications

Software Radio with such number of favorable characteristics can be widely applicable to [21]

- Air and sea traffic management : The incredible complexity of port management, whether for ships or aircrafts, requires highly sophisticated applications. Software Radio along with satellite global positioning system (GPS) technology can be used for the evaluation of supersophisticated navigation and avionics systems used to maneuver land military and commercial air, and sea craft, without having to rely on human intervention [1].

- Trunk radios : In a trunk radio system, all users share a pool of frequencies from five up to a maximum of twenty-eight. Users are assigned a "*groupid*" and field radios are programmed to only pick-up transmissions for that group [2]. Software Radio can be applied for easy programmatic allocation of frequency bands to various users of same or different groups. Trunk radios are broadly used by police and fire departments.

- Peer networks : The network in which the component nodes are connected to each other with no distinction between client and server is known as peer network [1]. The main benefits of peer-to-peer system are scalability, fault-tolerance, and the lack of resource bottlenecks in servers. Software Radio can be useful for peer network in terms of decreasing the complexity, and making self organized and decentralized network architecture.

1.2. Architecture of Software Radio

1.2.1. Definition of Radio Architecture

Radio Architecture can be defined as the comprehensive, and consistent set of functions, components and design rules according to which systems of interest may be organized, designed, and constructed [21]. A specific architecture entails a partitioning of

functions and components such that functions are assigned to components, and interfaces among components correspond to interfaces among functions.

When such functions and interfaces are defined in formal design rules via a public forum, the resulting architectures are called 'open architecture'. An architecture can be said as powerful or weak depending on its ability to increase or decrease the system development complexity.

Radio architecture can be directly related to the network hierarchy and channel data rate. As network hierarchy is rapidly moving from early point-to-point towards chaotic peer networks and more hierarchical structures with improved service quality. Similarly, channel data rate continue to increase through multiplexing and spectrum spreading. Thus, the complexity of functions, components, and design rules of radio architecture continues to increase with each generation.

Software Radio has been emerged to increase service quality through an environment that has active RF bands, channel access modes, data rates, bit error rates (BERs), power, and functionality. At the same time, software radio architecture simplifies hardware component tradeoffs and provide new ways of managing the complexity of rapidly emerging standards.

1.2.2. Software Radio Transceiver

In order to design demanding multipersonal capable, less complex, and low power consuming transceivers, an ideal software radio transceiver tries to move the border between the analog and digital world in the transmitter and receiver, as much as possible towards radio frequency. Considering the receiver, the only analog components are the antenna, the bandpass filter, and the low noise amplifier (see Fig. 1.2). However, an ideal software radio transceiver is far from the realization, due to wide bandwidth ranging of receiving signal, intermodulation and jitter effect [6].

The present acceptable software radio transceiver is similar to wideband transceiver. It has RF stage as completely analog, and IF stage is digital (see Fig. 1.3).

In this transceiver after the ADC in the RF section, there are three main operations which make a combination of IF and BB stage [6]. These operations are :

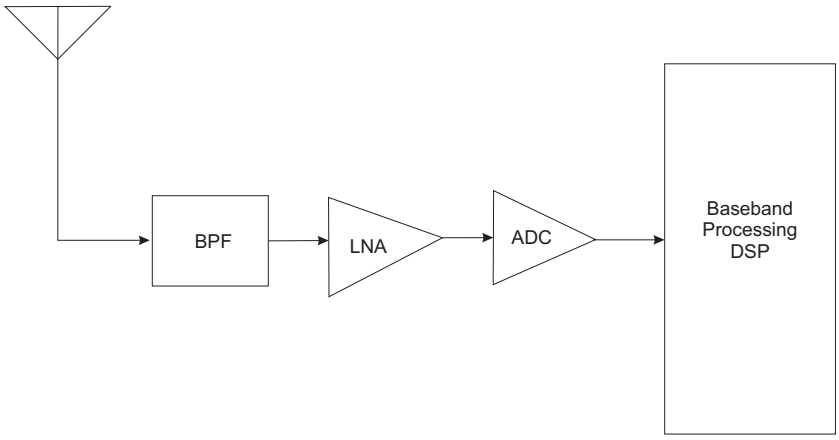


Figure 1.2: An ideal software radio receiver.

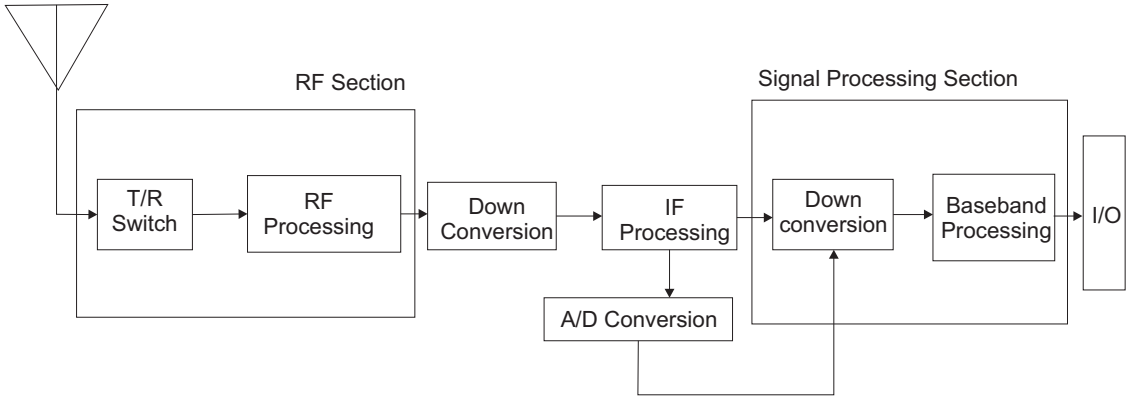


Figure 1.3: Software Defined Radio single mode Receiver.

- **Downconversion**: The process of converting the IF signals to BB signals digitally is called downconversion. This operation involves the use of a lookup table which contains the samples of sinusoidal carrier signal. This lookup table replaces the local oscillator used in an analog downconverter.

- **Channelization**: After obtaining the baseband signal, the next step is to select the carrier and channel which is to be elaborated by digital filtering. This operation in analog receivers is performed by an analog filter, with very stringent requirements, before BB conversion.

- **Sample rate adaptation**: Finally, the undersampling of the signal output of the channelization filter is performed to match the sample rate of the selected channel

bandwidth, which is a narrowband signal. This operation exploits the fundamentals of multirate systems to perform processing of digital signals.

Studies shows that all these operations come under *radio implementer plane* of Software Radio Architecture. Therefore, to make an integration of software with these operations the *radio implementer plane* is divided into two major areas [29]

- Radio Front End : Hardware plays the dominant role in this section.
- Radio Back End : In this section hardware exists along with software as the supporting role.

These subdivisions are for both base station and user terminal. With the progress of technology, Software Radio can be completely digitized at or very near to the antenna with software as the dominant driver residing in high-speed digital signal processing elements. Thus, with technology advancement, the Software Radio transceiver can be represented as the evolutionary stages, which shows the transition from traditional single mode, single band radio interfaces towards multicapability, multimode, and multiband digital radio global phone.

1.2.3. Resource Requirements

Resources critical to the software radio architecture include I/O bandwidth, memory and processing capacity [21]. Good estimates of the demand for these resources result in a well informed mapping of the above segments to appropriate hardware. Depending on the details of the hardware, the critical resources may be memory, bus, or I/O bandwidth, or a particular embedded processor. When the critical resources are identified early in the design process , order of magnitude shortfalls in performance can be avoided. Identification of the critical resources can be accomplished quickly using below given techniques.

- *Standardized Measures of Demand and Capacity*: Since many contemporary processors include pipelined floating point arithmetic sections or single instruction butterfly operations, like million instructions per seconds (MIPS), and mega floating point operations per second (MFLOPS) which are not interchangeable. Both these types

of operations require processor clock cycles, allowing one to express demand in a common measure of millions of operations per seconds (MOPS) where an operation is the amount of work that can be accomplished by a given resource in a single clock cycle of a standard width. Thus, software demand may be translated rigorously to equivalent MOPS for each resource using queueing theory.

- *Estimate Demand in the Context of the Canonical Data Flow*: Each segment of radio implementer plane has its associated resource demand. Although, these demands may exceed the capacity available with a given generation of devices. Performing capacity estimation by determining the number of operations required per point operation and multiplying by the critical parameter, one can quickly arrive at demand estimates that frame the related implementation decisions.

- *Facility Utilization Accurately Predicts Performance*: The most significant design parameter of the mapping of processing demand to processor capacity is resource utilization of the CPU, DSP chip, memory, bus, etc. Resource utilization is the ratio of offered demand to available capacity. In general,the average number of items waiting for resource services varies as a function of utilization.

1.3. Importance of Sampling Rate Conversion

1.3.1. Various Communication Standards

In a multimode software defined radio, a system filter should select a bandwidth covering all of the services that the software radio should support or a part of them. These processes of selection are called *full band digitization* and *partial-band digitization*, respectively [11]. Within this bandwidth, there are signals of different air interfaces communication standards used by several operators. This is explained in Fig. 1.4, where three different communication standards are considered. These are the global system for telecommunications (GSM) as a single carrier system based on Frequency Division Multiple Access (FDMA), the Interim Standard-95 (IS-95) as a spread spectrum Code Division Multiple Access (CDMA) system, and the Pacific Digital Communication (PDC) as a multichannel

system based on Time Division Multiple Access (TDMA). In case of FDMA and TDMA , the bandwidth of each service is split into several channels, while in case of CDMA whole bandwidth is occupied by spread spectrum channel.

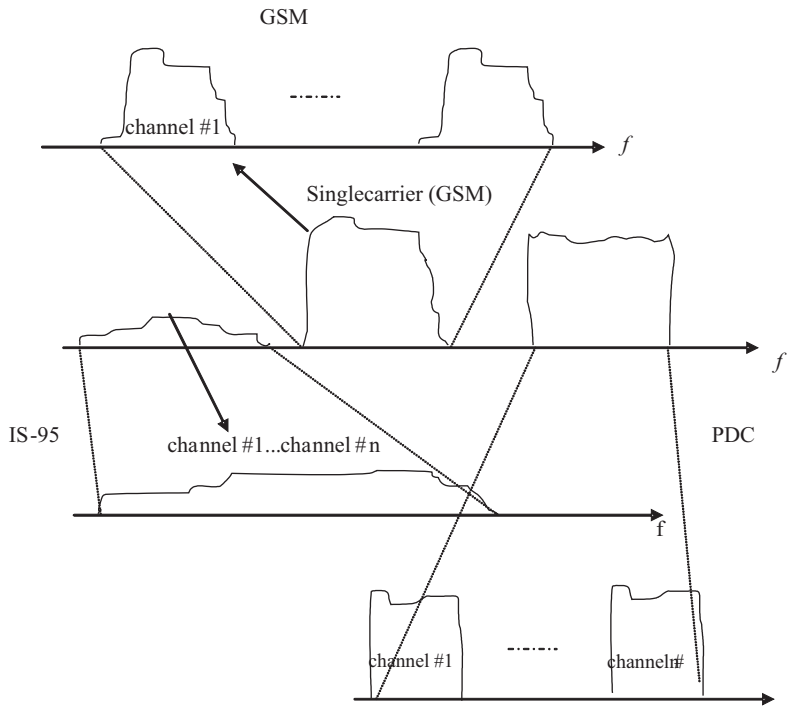


Figure 1.4: Software Radio Service band.

Many mobile radio standards have been developed for wireless systems throughout the world, and more standards are likely to emerge [6]. Table 1.1 lists the most common cellular communications standards used in North America, Europe, and Japan [25].

Table 1.1: Major Mobile Radio Standards in all around World.

Characteristics	GSM	IS-95	PDC
Year of Introduction	1990	1993	1993
Multiple Access	FDMA	TDMA	CDMA
Frequency Band (MHz)	890-960	824-894	890-1501
Modulation	GMSK	QPSK/BPSK	$\pi/4$ DQPSK
Channel Bandwidth (kHz)	200	1250	25
Symbol/chip rate	270.83 kSps	1.2288 Mcps	42 kSps

For operating all the three different communication standards given in Table 1.1, a multimode radio system is required. This radio system is being programmed by software, which makes the selection of desired frequency range according to service provider location. The multimode SR architecture for GSM, IS-95, and PDC is shown in Fig. 1.5.

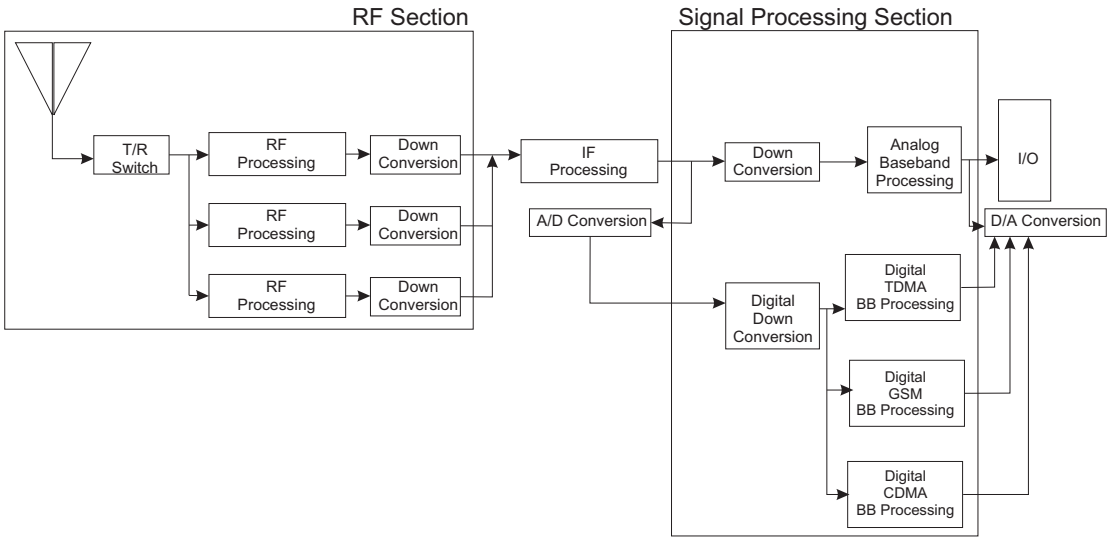


Figure 1.5: Software defined radio multimode receiver.

To make the same hardware to work for different frequency ranges, the conversion of sampling rate is required which is explained in the next section.

1.3.2. Sampling Rate Conversion Definition

The main property of software radio is to design an independent terminal irrespective of various air interface communication standards. These communication standards are generally based on different master clock rates and therefore employ different bit/chip rates. A solution to cope with the diversity of master clock rates in one terminal is to have a dedicated master clock for each standard of operation. But this kind of solution is costly and realizable only for a specific master clock rates. An efficient solution for different master clock rates is to run the terminal on a fixed clock rate and perform digital sample rate conversion controlled by software [11].

The process of digitally converting the sampling rate of a signal from a given rate $f = 1/T$ to a different rate f'$=1/T$', where T and T' are sampling periods, is called *sampling*

rate conversion (SRC). SRC involves the basic operations of decimation and interpolation defined as

- Decimation is the method of decreasing the sample rate by performing downsampling of input bandlimited signal.

- Interpolation is the process to increase the sample rate of input signal by upsampling the input signal and then performing anti-aliasing.

An interpolator and a decimator are dual digital systems [9]. The basic steps of decimation and interpolation are downsampling and upsampling, respectively.

Downsampling is the process of removing M-1 samples from the input digital signals, performed by a downsampler or expander of factor 'M'. This will consequently lead to the decreasing of the signal frequency. The main drawback of downsampling is overlapping effect known as *aliasing*. To prevent aliasing, a band limiting filter is applied before downsampler known as *anti-aliasing filter*. This complete cascade of anti-aliasing filter and downsampler is called *decimator* shown in Fig. 1.6.

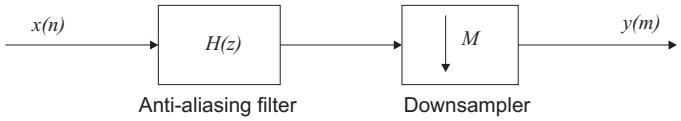


Figure 1.6: Decimator.

Upsampling is performed by an upsampler or compressor of factor 'L' introduces L-1 zeros between each pair of input signal samples. The zeros in time-response will cause the formation of images in the frequency response and signal spectrum gets compressed by $2\pi/L$. Hence, an anti-iamging filter is required to eliminate all these images. The cascade of upsampler followed by an anti-imaging filter is called *interpolator*. Fig. 1.7 shows an interpolator.

Thus, Software Radio terminals can process various communication standards simply by converting the sampling rate to desired sampling rate.

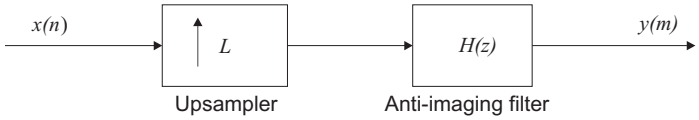


Figure 1.7: Interpolator.

1.4. Programmable Logic Devices for Software Radio

PLDs are integrated circuits (ICs) which can be programmed in house or on the field. The design and implementation of an application on these devices can be achieved with the help of software tools. Hardware descriptive languages such as Verilog and VHDL are widely used for this purpose. The codes written in these languages can also be synthesized using a third party Electronic Design and Automation (EDA) tool or the software tool provided by the vendor. With the help of these tools it is also possible to optimize the design for speed or space.

The PLDs have gradually grown to immense prominence in the field of digital signal processing. The gates have grown from a group of AND/OR gates accomplishing simple 'sum of products' operations to millions of gates on a single substrate with additional devices like memory elements, multiplier and also in- built processors (Spartan 3 from Xilinx Inc.). Various types of programmable logic devices are discussed below

Programmable logic array (PLA) are one of the first programmable devices developed. They consist of a layer of AND gates and OR gates. They accomplish the Sum of Product (SOP) operation. Hence, the number of inputs cannot exceed the number of AND gates.

Programmable Array Logic (PAL) devices are similar to PLA. They were introduced by Monolithic Memories (now a part of Advanced Micro Devices). These devices also have only AND and OR gates. However, unlike PLA, only AND gates are programmable and every OR gate is connected to a bunch of AND gates. Thus, the maximum number of minterms allowed for an OR gate is equal to the number of inputs to the OR gate. The logic function of higher minterms can be implemented by routing the output of one OR gate to the input of another minterm. Fig.1.9

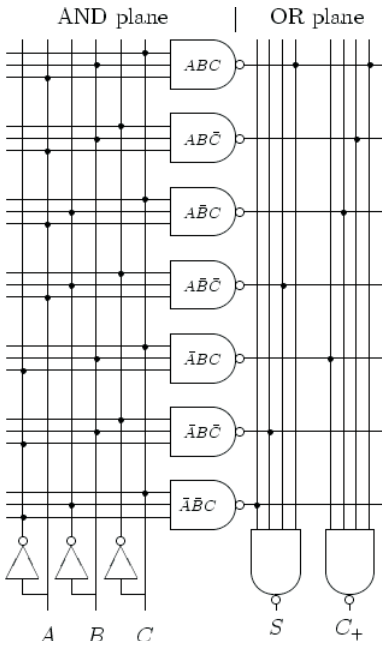


Figure 1.8: PLA implementation of a full-adder.

shows PAL device architecture.

Complex Programmable Logic Device (CPLD) as the name suggests, is more complex than the previously discussed architectures. Firstly, they are large granularity devices and consist of a group of arrays of logic elements or logic cell, which are connected through wide buses (called Programmable Interconnect Array (PIA) by Altera) with short delays. The logic cells typically have 8 to 10 inputs, 3 to 4 outputs, and support around 20 product terms. The data path is not unidirectional from input to output of the IC; instead outputs of all the arrays are fed back to the PIA. The output of the IC that is required to be fed as input into another cell is first routed back to the common interconnect lines and then connected to designation logic. By combining the bus and the fixed IC timing, it is possible to provide predictable and short pin-to-pin delays in CPLDs. Fig.1.10 shows CPLD Architecture

Field Programmable Gate Array (FPGA) have similar structure to gate arrays however they have programmable elements. Traditional gate arrays contain a number of building blocks or primitive cells etched on a single silicon substrate. The connec-

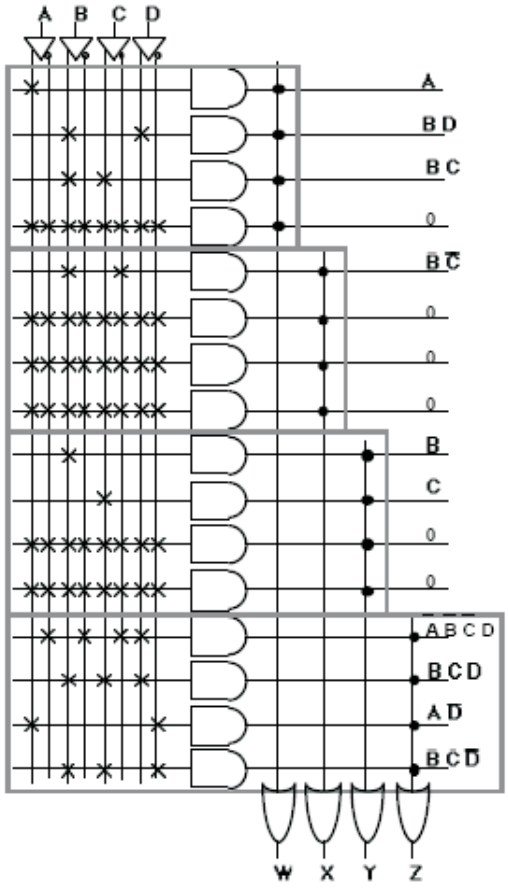


Figure 1.9: Programmable Array Logic.

tions between cells are permanent and made later. These are non-reprogrammable high-density devices containing about 5 millions gates. The programmable cell of FPGA is called Logic Element (LE) in case of Altera device and Configurable Logic Block (CLB) in Xilinx devices. FPGA use the Complementary Metal Oxide Semiconductor SRAM technology and are thus reset at power off.

Software Radio technology has gained momentum as engineers every where are developing radio architectures that include minimum hardwired analog components [10]. The ability to perform intermediate frequency (IF), bandwidth modulation, coding schemes and other radio functions is the appeal for such widespread interest. Current technology in field-programmable gate array (FPGA) technology have enabled high-speed processing in a compact footprint, while retaining the flexibility and programmability of Software Radio

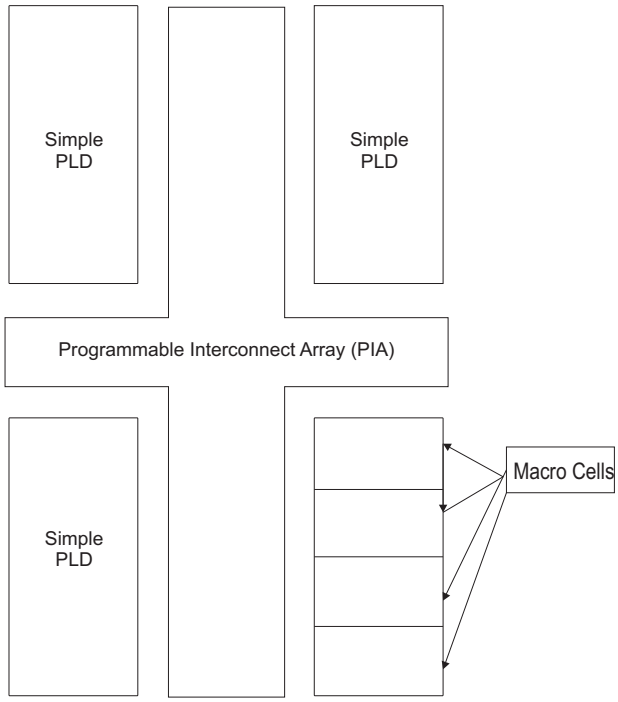


Figure 1.10: Complex Programmable Logic Device.

technology. The hallmarks of FPGA are

- FPGAs are very popular for high-speed, compute-intensive, reconfigurable applications like Fast Fourier Transform (FFT), Finite Impulse Response (FIR) filters, and other multiply-accumulate operations.

- FPGAs have evolved from being flexible logic design platforms to signal processing engines. They are now an essential component of Software Radio due to their flexibility and real time processing capabilities. Increasingly, system designers are porting more and more signal processing functionalities in FPGAs. The flexibility of having the ability to integrate logic design with signal processing is pushing designers to replace traditional digital signal processors (DSPs) with FPGAs.

- FPGAs are also inherently suited for high-speed parallel multiply and accumulate functions. Current generation FPGAs can perform 18 x 18 multiplication operation at speeds in access of 200 MHz. This makes FPGAs an ideal platform for operations such as FFT, FIR, digital downconverters (DDC), digital upconverters (DUC), correlators and pulse compression.

re Radio

2

Basic Principles of Sampling Rate Conversion

The objective of this chapter is to assemble the concept of analog-to-digital conversion (ADC) and sampling rate conversion (SRC) over software radio (SR) platform. The concept of discrete sampling, which is the first step of SRC is described. An analog interpretation and digital approach of SRC is presented to understand various aspects of SRC process. Types of SRC process are explained along with the conversion factor as the decisive parameter for SRC, with the idea of separating integer factor and fractional factor. The last section describes the factors on which hardware effort of an SRC system depends.

2.1. Analog-to-Digital Conversion

Digital signal processing involves the methods of processing the signals in discrete-time domain. Thus, the conversion of continuous-time (analog) signal into discrete-time (digital) signal is among the initial steps of SRC. This process of converting continuous-time signal into discrete-time sequences is known as analog-to-digital conversion (ADC). Various specific work are performing by the area of analog design to make advanced ADCs with respect to technological demands, like Sigma-Delta converters are among the new ADCs with advanced features. In general, the analog-to-digital conversion process takes place in two steps i.e. *sampling* and *quantization* [24].

Sampling is the process of selecting amplitude values of the continuous-time signal at certain time instance T. When the sampling period T is constant between two

samples then this type of sampling is called *equidistant sampling*. In this context, equidistant sampling has been considered as sampling. The mathematical model of sampling is the multiplication of analog signal with a periodic impulse train. Let $x_C(t)$ be a continuous function of the continuous time variable t. Sampling of $x_C(t)$ at the uniform rate of

$$t = nT, \quad -\infty < n < \infty \tag{2.1}$$

is obtained by multiplying $x_C(t)$ with a sampling function $s(t)$. This multiplication process is also known as *pulse amplitude modulation* (PAM). The sampled signal $x(n)$ is given by

$$x(n) = x_C(t) \cdot s(t) \tag{2.2}$$

where, t changes with respect to sampling period T, and

$$s(t) = \sum_{l=-\infty}^{\infty} u_0(t - lT) \tag{2.3}$$

and, where $u_0(t)$ denotes an ideal unit impulse function. Fig. 2.1 shows an example of a signal $x_C(t)$ and the associated sampled signal $x(n)$.

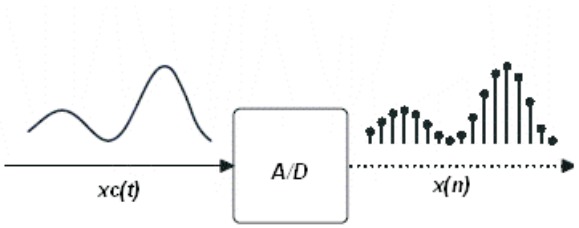


Figure 2.1: Analog-to-Digital Converter.

Taking the Fourier transform of the input signal $x_C(t)$ will give $X_C(j\omega)$ defined as

$$X_C(j\omega) = \int_{-\infty}^{\infty} x_C(t) e^{-j\omega t} dt \tag{2.4}$$

where, ω denotes the analog frequency (in radians/sec).

Similarly, the Fourier transform of the sampling function *s(t)* can be defined as

$$S(j\omega) = \int_{-\infty}^{\infty} s(t) e^{-j\omega t} dt \tag{2.5}$$

and it can be shown that by applying Eq.(2.3) to Eq.(2.5), $S(j\omega)$ has the form

$$S(j\omega) = \frac{2\pi}{T} \sum_{l=-\infty}^{\infty} u_0 \left[\omega - \frac{2\pi l}{T}\right]. \tag{2.6}$$

By defining

$$F = \frac{1}{T} \tag{2.7}$$

$$\omega = 2\pi f \tag{2.8}$$

and

$$\omega_F = 2\pi F. \tag{2.9}$$

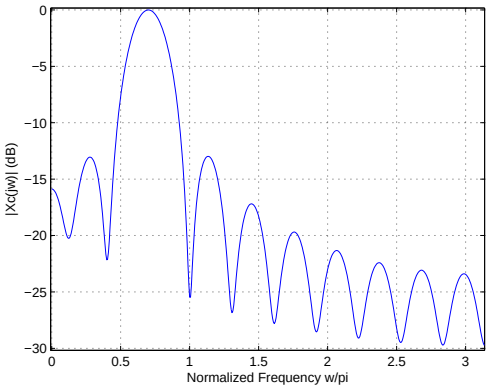
A uniformly spaced impulse train in time, *s(t)*, transforms to a uniformly spaced impulse train in frequency, $S(j\omega)$.

Since, multiplication in the time domain is equivalent to convolution in the frequency domain, we have

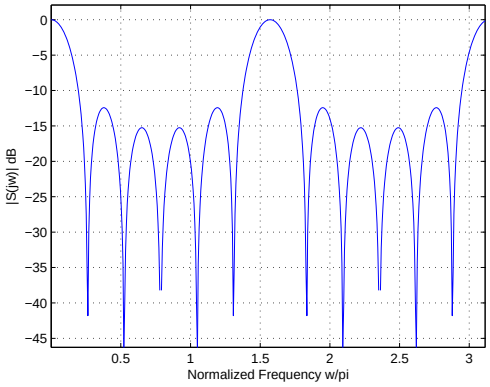
$$X_C(j\omega) * S(j\omega) = \int_{-\infty}^{\infty} [x_C(t)s(t)]e^{-j\omega t}dt \tag{2.10}$$

where $*$ denotes the linear convolution of $X_C(j\omega)$ and $S(j\omega)$ in frequency. Fig. 2.2 shows typical plots of $X_C(j\omega)$, $S(j\omega)$ and the convolution $X_C(j\omega)*S(j\omega)$, where it is assumed that $X_C(j\omega)$ is bandlimited and its highest frequency component $2\pi F_C$ is less than one-half of the sampling frequency. Thus, sampling the signal $x_C(t)$ at time period T causes repetition of the spectrum $X_C(j\omega)$ at integer multiples of $1/T$. This effect is called *imaging*, and the spectral copies are called *images*. It can also be observed from the figure that as long as the sampling rate is larger than twice the highest frequency component of $X_C(j\omega)$, there is no overlap of the images. This criteria is called *sampling theorem* and the sampling rate is called *Nyquist rate*. The overlapping of the images is called *aliasing*. Aliasing is an irreversible effect occur due to the violation of the sampling theorem. In order to recover the analog signal from it's sampled signal, aliasing should be avoided.

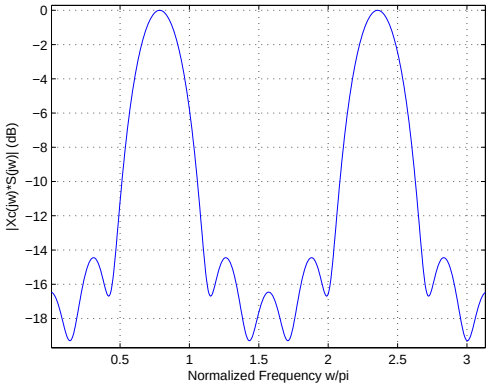
Quantization is a nonlinear operation of mapping the signal values to the quantized values. Quantization is said to be uniform when the quantization step ΔV will be



(a)



(b)



(c)

Figure 2.2: Sampled and Modulated spectra.

constant independent of the current signal value, and ΔV can be calculated as:

$$\Delta V = \frac{V_{pp}}{q} = \frac{V_{pp}}{2^{\beta}} \quad with \quad q = 2^{\beta} \tag{2.11}$$

where, V_{pp} is the peak-to-peak voltage of the quantizer, and q is the number of quantization steps, here regarded as being the βth power of two. Thus, the amplitudes of the quantized signal samples can be coded with β bits.

Quantization process causes distortions of the sampled signal. These distortions are described by the error signal *e(n)* which is the difference between the quantized signal $x_Q(n)$ and the sampled signal *x(n)*

$$e(n) = x_Q(n) - x(n)\cdot \tag{2.12}$$

The quantization error is modeled as additive white noise that is uncorrelated with *x(n)*. Thus, *e(n)* represents white noise, called *quantization noise*. The most interesting figure resulting from modeling the quantization error as additive white noise is the signal-to-quantization noise ratio (SNR) of a quantizer, thus, of an ADC. SNR can be defined as

$$SNR = 10 \log \frac{P_x}{P_e} \tag{2.13}$$

where, P_x and P_e are the respective signal power and quantization noise power.

Because the quantization noise is assumed to be white, the noise power is uniformly distributed in the frequency interval $[-f_s/2, f_s/2]$, where f_s is the sample rate. If the quantized signal covers only a fraction of this interval , namely a bandwidth $B < f_s$, then only the noise power within this bandwidth B distorts the signal. This is called *oversampling* and the oversampling ratio *(OSR)* is defined as

$$OSR = \frac{f_s}{B}\cdot \tag{2.14}$$

If the signal at the input of quantizer is a multichannel mobile communications signal which occupy the available bandwidth $[-f_s/2, f_s/2]$ fully, then there is no conventional oversampling. However, a certain channel of available bandwidth b (i.e. the channel of interest) covers only a fraction of the available bandwidth which illustrates the *generalized oversampling ratio* ($GOSR$) of the signal. Thus, the channel

of interest is oversampled by the factor

$$GOSR = \frac{f_s}{b}. \tag{2.15}$$

The difference between *OSR* and *GOSR* is that the power of channel of interest is *GOSR* times lower than the total signal power.

In the context of SRC, *OSR* and *GOSR* are very important. The *GOSR* after the SRC process directly determines the relative bandwidth of the potential aliasing components that have to be attenuated by the SRC filter. The higher the *GOSR* is, the smaller the passband and the stopband of the filter. Hence, a high *GOSR* (after SRC) relaxes the design constraints and resulting to simpler filter structures. A consequence of this is that SRC is advantageously implemented on a cascaded multirate system with relaxed requirements at high sample rates where the *GOSR* of the signal of interest is high, and strong requirements at low sample rates.

2.2. Discrete Sampling

Sampling as discussed in the previous section, is the process of converting a continuous-time signal into discrete-time signal. But, when a discrete-time signal *x(n)* is sampled, such that every M-th value of *x(n)* is retained and all remaining values are set to 0, then this process is called *discrete sampling* [9]. To sample a discrete-time signal, a *discrete sampling function* $\omega_M(n)$ is required which can be expressed as :

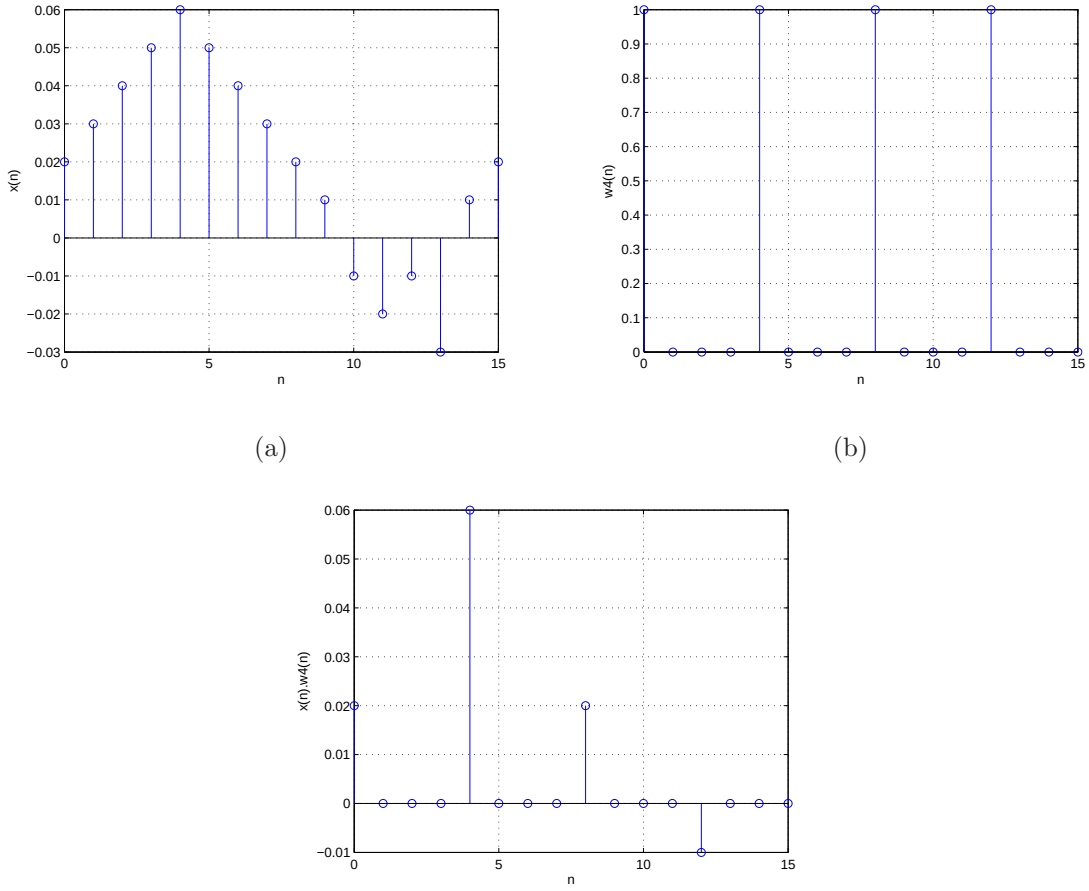
$$\omega_M(n) = \frac{1}{M}\sum_{v=0}^{M-1} W_M^{-vn} = \begin{cases} 1 & \text{for} \quad n = mM, \quad m \quad \text{is} \quad \text{an} \quad \text{integer.} \\ 0 & \text{otherwise.} \end{cases} \tag{2.16}$$

This means that the *discrete sampling function* $\omega_M(n)$ will be 1 if n is a multiple of M, otherwise 0. Here, W_M is the complex number representation of discrete signals defined as :

$$W_M = e^{-\frac{j2\pi}{M}} = \sqrt[M]{1}. \tag{2.17}$$

Multiplication of an arbitrary complex number z with W_M will change the argument of z on the complex plane, but its magnitude $|z|$ remains unchanged. For this reason, multiplication with W_M is referred to as a rotation operation.

Fig. 2.3 shows the *discrete sampling* of discrete sequence *x(n)* by *discrete sampling function* $\omega_M(n)$ for $M = 4$. It can be noted from these figures that the sampling rate has not been changed by the discrete sampling.



(a) (b)

(c)

Figure 2.3: Discrete Sampling.

Discrete sampling can also be carried out with a phase offset λ, which retains every $n = mM + \lambda$ values of *x(n)*. To achieve this a discrete sampling function with a phase offset λ is required , which is expressed by

$$\omega_M(n-\lambda) = \frac{1}{M}\sum_{v=0}^{M-1} W_M^{-v(n-\lambda)} = \begin{cases} 1, n = \lambda + mM, & m \quad \text{is} \quad \text{an} \quad \text{integer.} \\ 0, & \text{otherwise.} \end{cases} \tag{2.18}$$

Using discrete sampling with phase offset, M different discretely sampled signals can be obtain from *x(n)*. These M different discrete signals for $\lambda = 0, 1, 2, \cdots, M-1$ are called

polyphase components denoted as $x_\lambda^{(p)}$. The original discrete signal *x(n)* can be recovered by adding all these polyphase components. This representation of *x(n)* is called *polyphase representation*. The general expression of polyphase representation in time domain is

$$x(n) = \sum_{\lambda=0}^{M-1} x_\lambda^{(p)}(n) = \sum_{\lambda=0}^{M-1} x(n) \cdot \omega_M(n-\lambda). \tag{2.19}$$

For $M = 4$, *polyphase representation* will be

$$x(n) = x_0^{(p)}(n) + x_1^{(p)}(n) + x_2^{(p)}(n) + x_3^{(p)}(n). \tag{2.20}$$

and, in terms of discrete sampling function $\omega_M(n-\lambda)$

$$x(n) = x(n) \cdot \omega_4(n) + x(n) \cdot \omega_4(n-1) + x(n) \cdot \omega_4(n-2) + x(n) \cdot \omega_4(n-3). \tag{2.21}$$

In *z-transform*, discrete signal *x(n)* is

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}. \tag{2.22}$$

Therefore, general expression of polyphase representation in z-transform for $n = mM + \lambda$ is

$$X(z) = \sum_{m=-\infty}^{\infty} \sum_{\lambda=0}^{M-1} x(mM+\lambda) z^{-mM+\lambda}. \tag{2.23}$$

in terms of polyphase components

$$X(z) = \sum_{\lambda=0}^{M-1} z^{-\lambda} X_\lambda^{(p)}(z^M). \tag{2.24}$$

where $X_\lambda^{(p)}$ is the z-transform of polyphase components $x_\lambda^{(p)}(n)$ for $n = mM + \lambda$

$$X_\lambda^{(p)}(z^M) = \sum_{m=-\infty}^{\infty} x(mM+\lambda) z^{-mM}. \tag{2.25}$$

Polyphase representation defined by Eq.(2.19) and Eq.(2.23) is called *type 1 polyphase representation* or *standard representation*. From *type 1 polyphase representation* two more polyphase representations can be derived. Replacing λ by $M-1-\lambda$ creates *type 2 polyphase representation*. The general expression of *type 2 polyphase representation* is

$$X(z) = \sum_{\lambda=0}^{M-1} z^{-(M-1-\lambda)} X_\lambda^{(p2)}(z^M). \tag{2.26}$$

where,

$$X_{\lambda}^{(p2)}(z^M) = \sum_{m=-\infty}^{\infty} x_{\lambda}^{(p2)}(m) z^{-mM}. \tag{2.27}$$

and,

$$x_{\lambda}^{(p2)}(m) = x(mM + M - 1 - \lambda). \tag{2.28}$$

On replacing λ by $-\lambda$ in *standard polyphase representation*, *type 3 polyphase representation* will be obtain. The general expression of *type 3 polyphase representation* is

$$X(z) = \sum_{\lambda=0}^{M-1} z^{-\lambda} X_{\lambda}^{(p3)}(z^M) \tag{2.29}$$

where,

$$X_{\lambda}^{(p3)}(z^M) = \sum_{m=-\infty}^{\infty} x_{\lambda}^{(p3)}(m) z^{-mM} \tag{2.30}$$

and,

$$x_{\lambda}^{(p3)}(m) = x(mM - \lambda). \tag{2.31}$$

Decomposing of discrete signal *x(n)* into its polyphase components is useful to recompose *x(n)* by using commutator and it is memory saving. Polyphase representation also permits to shift the polyphase components from high frequency rate to low sampling rate by using multirate identities. Thus, an efficient filter can be implemented using polyphase representation, which occupy less area and consumes less power. Polyphase representations are widely applicable for filter banks.

2.3. Sampling Rate Conversion - Various Aspects

The process of sampling rate conversion is the method of converting the discrete-time signal *x(n)*, which is obtained by sampling the continuous-time signal $x_c(t)$ with a period T, to another discrete-time signal *y(m)* obtained by sampling $x_c(t)$ with another period *T'*. This process involves the reconstruction of $x_c(t)$ from its samples and then perform resampling of bandlimited reconstructed $x_c(t)$ with period *T'* to give *y(m)*. This approach is also known as resampling after reconstruction [27] as illustrated in Fig. 2.4. Here, the analog filter $h_c(t)$ is assumed to have finite impulse response. Hence, the reconstructed

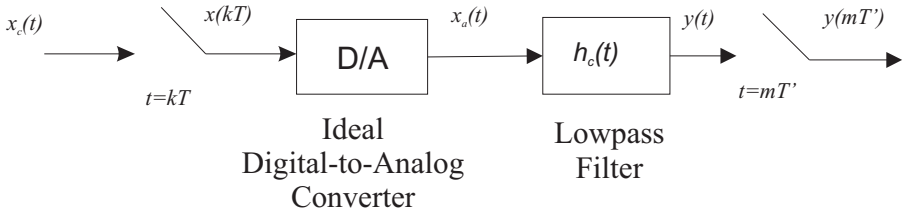


Figure 2.4: Resampling After Reconstruction.

signal $x_c(t)$ is derived from the finite set of discrete signal *x(nT)*. Then, the value of *y(mT')* for any m can be expressed as

$$y(mT') = x_c(t)|_{t=mT'} = \sum_{n=N_1}^{N_2} x(nT)h_c(mT' - nT) \tag{2.32}$$

where, N_1 and N_2 are range of values of n involved in computation of *y(mT')*. Such that

$$N_1 = \lceil \frac{mT' - t_2}{T} \rceil \quad \text{and} \quad N_2 = \lfloor \frac{mT' - t_1}{T} \rfloor \tag{2.33}$$

here, t_1 and t_2 are end points of filter where $h_c(t)$ is not equal to 0.

In sampling rate conversion, the evolution of *y(mT')* depends upon distinct set of samples of *x(nT)* and $h_c(t)$ for each value of m. Hence, by introducing the change of variables

$$k = \lfloor \frac{mT'}{T} \rfloor - n \tag{2.34}$$

the Eq.(2.32) can be modified to another form as:

$$y(mT') = \sum_{k=K_1}^{K_2} x\left[\lfloor \frac{mT'}{T} \rfloor - k\right] h_c\left[mT' - \lfloor \frac{mT'}{T} \rfloor T + kT\right] \tag{2.35}$$

where,$\lfloor u \rfloor$ defines the largest number less than or equal to u. Rearranging *y(mT')* gives the desired form as:

$$y(mT') = \sum_{k=K_1}^{K_2} h_c((k + \delta_m)T)x\left[\lfloor \frac{mT'}{T} \rfloor - k\right] \tag{2.36}$$

where, δ_m and K_1, K_2 are defined as :

$$\delta_m = \frac{mT'}{T} - \lfloor \frac{mT'}{T} \rfloor \tag{2.37}$$

$$K_1 = \lceil \frac{t_1}{T} - \delta_m \rceil = \lceil \frac{t_1}{T} - \frac{mT'}{T} \rceil + \lfloor \frac{mT'}{T} \rfloor \tag{2.38}$$

$$K_2 = \lceil \frac{t_2}{T} - \delta_m \rceil = \lceil \frac{t_2}{T} - \frac{mT'}{T} \rceil + \lfloor \frac{mT'}{T} \rfloor. \tag{2.39}$$

Here, the ratio of $\frac{T'}{T}$ can be represented as the ratio of $\frac{M}{L}$, where M and L are integers. Therefore

$$\delta_m = \frac{mM}{L} - \lfloor \frac{mM}{L} \rfloor. \tag{2.40}$$

From this relation it is seen that δ_m can take only L unique values 0,1/L,2/L,...,(L-1)/L, for all values of M. Thus, there are only L unique sets of samples of $h_c(t)$ that are used in computing *y(m)* from *x(n)*.

In many signal processing applications, it is desirable to perform SRC directly by digital computation without an intermediate analog stage [27] as illustrated in Fig. 2.5. For direct digital approach, it is need to derive a linear digital system $g_m(n)$ such that *x(n)* can be processed by $g_m(n)$ to give *y(m)* directly as shown in Fig. 2.6.

Figure 2.5: A Direct Digital Conversion for SRC

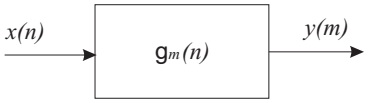


Figure 2.6: Extended Interpretation of SRC Digital Approach.

A close examination of the proposed structure of Fig. 2.6 shows that the systems for digital-to-digital sampling rate conversion are inherently linear time-varying system. Since the system is linear, each output sample *y(m)* can be expressed as a linear combination of input samples *x(n)*. The general expression of *y(m)* is

$$y(m) = \sum_{n=-\infty}^{\infty} g_m(n) x[\lfloor \frac{mM}{L} \rfloor - n] \tag{2.41}$$

where, $g_m(n)$ can be expressed as

$$g_m(n) = \hat{h}((n + \delta_m)T)\cdot \tag{2.42}$$

Since $g_m(n)$ can take on L distinct sets of values, it is periodic in m; that is

$$g_m(n) = g_{m+rL}(n), r = 0, \pm 1, \pm 2, ... \tag{2.43}$$

Thus, the system $g_m(n)$ is a linear, digital, periodically time-varying system. The formulation of above digital periodically time-varying system is particularly suitable for defining models and deriving practical structures for decimators and interpolators.

2.4. Sampling Rate Reduction - Decimation

The process of reducing the sampling rate of *x(n)* by an integer factor M is known as Integer Factor *Decimation*. For this process, the ratio of sampling period of input and output sequence is given by

$$\frac{T'}{T} = \frac{M}{1}\cdot \tag{2.44}$$

Then, the new sampling rate is

$$F' = \frac{1}{T'} = \frac{1}{MT} = \frac{F}{M}\cdot \tag{2.45}$$

In order to avoid aliasing, it is necessary to filter the signal *x(n)* with a digital lowpass filter. The ideal characteristics of this filter will be

$$\tilde{H}(e^{j\omega}) = \begin{cases} 1, |\omega| \leq \frac{2\pi F'T}{2} = \frac{\pi}{M} \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.46}$$

The sampling rate reduction is then achieved by forming the sequence *y(m)* by retaining every M^{th} sample of the filtered output. The complete process along with block diagram is illustrated in Fig. 2.7, Fig. 2.9 and Fig. 2.10

The output *w(n)* of the lowpass filter is given by

$$w(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \tag{2.47}$$

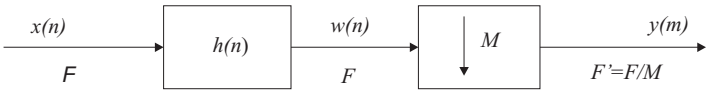
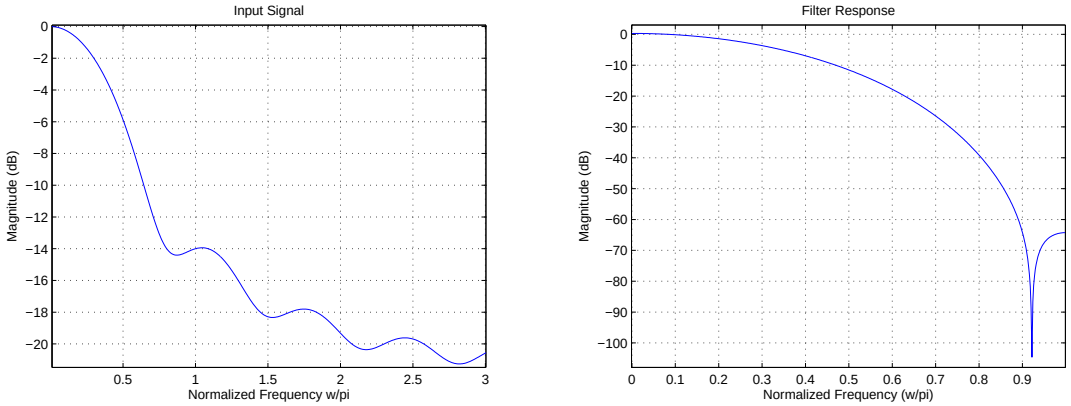
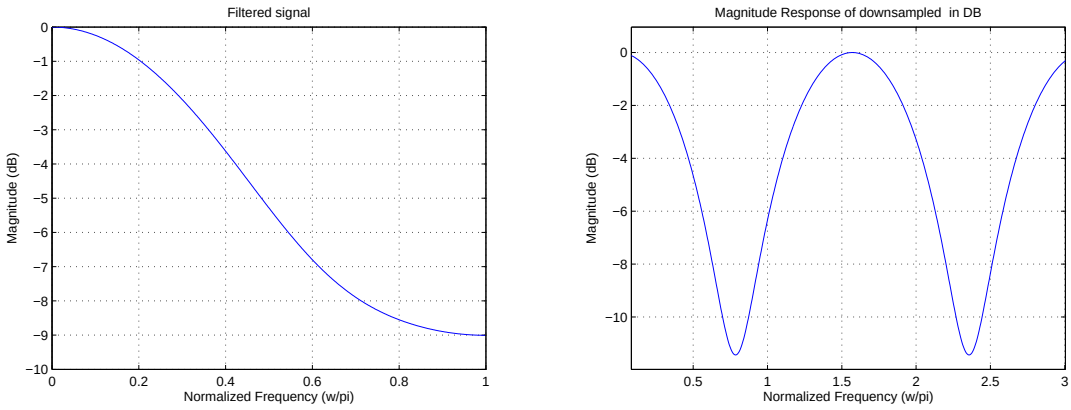


Figure 2.7: Decimation by integer factor $M = 4$.



(a) (b)

Figure 2.8: Input Signal and Anti-aliasing filter Response.



(a) (b)

Figure 2.9: Bandlimited and Downsampled Signal.

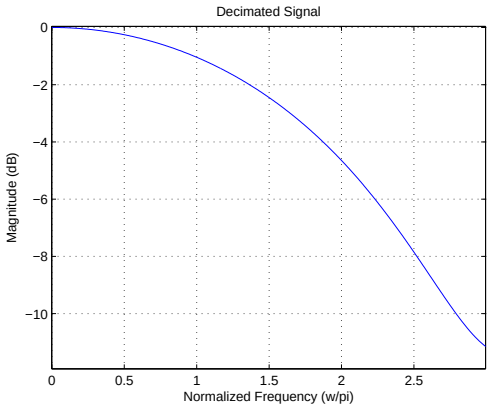


Figure 2.10: Decimated Spectra.

where, *h(k)* is the impulse response of lowpass filter and *x(n-k)* is the time shifted input impulse response. The final output *y(m)* is then obtained by downsampling *w(n)* thus

$$y(m) = w(mM)\cdot \tag{2.48}$$

The block diagram symbol of a down-arrow with an integer corresponds to decreasing the sampling rate and referred to as *downsampler* or *compressor*. Downsampling contains two steps

1. Discrete Sampling of input sequence as we discussed in section 3.1.
2. Remove all the zero valued samples from the discrete sampled sequence.

The consequence of downsampling is aliasing or overlapping of spectrum obtained by removing the zero valued data. Aliasing will not appear only when the input signal is bandlimited.

By combining Eqs. (2.47) and (2.48) the relation between *y(m)* and *x(n)* is of the form:

$$y(m) = \sum_{k=-\infty}^{\infty} h(k)x(mM-k)\cdot \tag{2.49}$$

Since, decimation is a time-varying system, it signifies that a shift in input sequence will not be same shift at output sequence. But there are spectrum, as at every M^{th} multiple shift of input sequence the output sequence has the same shift. It can be expressed as

$$x(n-\delta) \rightarrow y(m-\frac{\delta}{M}) \quad \text{unless} \quad \delta = rM. \tag{2.50}$$

here, r is an integer, To obtain the relation between the z-transforms of *y(m)* and *x(n)*, *w(n)* is redefined as

$$w'(n) = \begin{cases} w(n), n = 0, \pm M, \pm 2M, ... \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.51}$$

That is, *w'(n)* = *w(n)* at the sampling instants of *y(m)*, but is zero otherwise. A convenient and useful representation of *w'(n)* is then

$$w'(n) = w(n) \left[\frac{1}{M} \sum_{l=0}^{M-1} e^{\frac{j2\pi ln}{M}} \right], -\infty < n < \infty \tag{2.52}$$

where, the term in bracket corresponds to a discrete Fourier series representation of a periodic impulse train with period of M samples. Thus, the relation between *y(m)* and *w(n)* will become

$$y(m) = w'(mM) = w(mM), -\infty < n < \infty\cdot \tag{2.53}$$

The input/output z-transform relation will be :

$$Y(z) = \sum_{m=-\infty}^{\infty} y(m) z^{-m} = \sum_{m=-\infty}^{\infty} w'(mM) z^{-m}\cdot \tag{2.54}$$

and since w'(m) is zero except at integer multiples of M, Eq.(2.54) becomes

$$\begin{aligned} Y(z) &= \sum_{m=-\infty}^{\infty} w'(m) z^{-m/M} \\ &= \left[\frac{1}{M} \sum_{l=0}^{M-1} e^{\frac{j2\pi ln}{M}} \right] z^{\frac{-m}{M}} \\ &= \frac{1}{M} \sum_{l=0}^{M-1} \left[\frac{1}{M} \sum_{m=-\infty}^{\infty} w(m) e^{\frac{j2\pi ln}{M}} z^{\frac{-m}{M}} \right] \\ &= \frac{1}{M} \sum_{l=0}^{M-1} W(e^{\frac{-j2\pi ln}{M}} z^{\frac{1}{M}})\cdot \end{aligned} \tag{2.55}$$

Since,

$$W(z) = H(z)X(z) \tag{2.56}$$

we can express *Y(z)* as

$$Y(z) = \frac{1}{M} \sum_{l=0}^{M-1} H(e^{\frac{-j2\pi ln}{M}} z^{\frac{1}{M}}) X(e^{\frac{-j2\pi ln}{M}} z^{\frac{1}{M}})\cdot \tag{2.57}$$

Evaluating Y(z) by substituting $z = e^{j\omega'}$, leads to the Fourier transform of the output signal *y(m)*, given as

$$Y(e^{j\omega'}) = \frac{1}{M}\sum_{l=0}^{M-1} H(e^{\frac{j(\omega'-2\pi l)}{M}})X(e^{\frac{j(\omega'-2\pi l)}{M}}) \tag{2.58}$$

where,

$$\omega' = 2\pi f T'\cdot \tag{2.59}$$

The purpose of the lowpass filter $H(e^{j\omega'})$ is to filter *x(n)* sufficiently so that its spectral components above the frequency $\omega = \pi/M$ are negligible. Thus, it serves as an *anti-aliasing* filter.

The advantageous features of polyphase representation as discussed in the previous section can also be applied for decimation. The decimated signal *y(m)* of polyphase decimator in time domain can be obtain by replacing $k = r \cdot M + \lambda$ as

$$y(m) = \sum_{k=-\infty}^{\infty}\sum_{\lambda=0}^{M-1} h(rM+\lambda)x([m-r]M-\lambda). \tag{2.60}$$

The general structure of polyphase decimator with M branches and sampling rate reduction by a factor of M is shown in Fig. 2.11 As, the output decimated signal *y(m)* are the

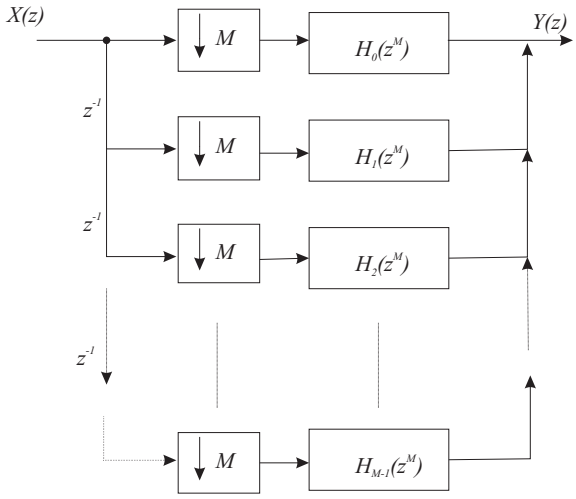


Figure 2.11: Polyphase Decimator.

samples of input signal *x(n)* only when $n = m \cdot M$. Thus, there are some unnecessary calculations of *y(m)* when $n \neq m \cdot M$. This can be prevented by shifting the polyphase filter

after the downsampler utilizing the identities of multirate systems [9] and called as efficient structure. The efficient structure of polyphase decimator is shown in Fig. 2.12. Thus,

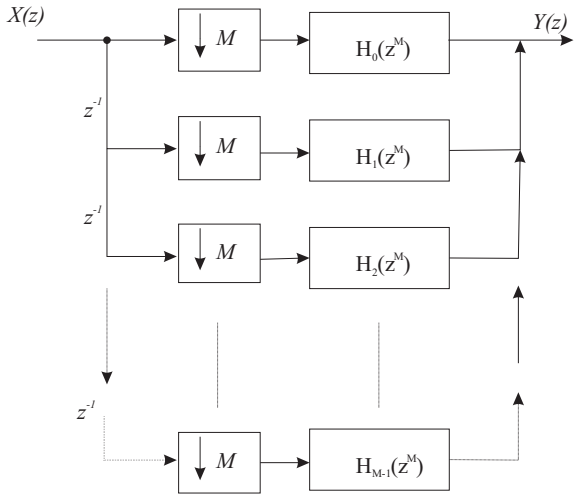


Figure 2.12: Efficient Polyphase Decimator.

the efficient structure will provide less calculation and low memory space by performing only the necessary calculations.

2.5. Sampling Rate Amplification - Interpolation

The process of increasing the sampling rate of a signal *x(n)* by L is called *Interpolation* by an integer factor. In this process L -1 new sample values are interpolated between each pair of samples of *x(n)*. If the sampling rate is increased by an integer factor L, then the new sampling period, T' of output signal *y(m)*, will be

$$\frac{T'}{T} = \frac{1}{L} \tag{2.61}$$

and the new sampling rate F' is

$$F' = LF \tag{2.62}$$

Fig. 2.13 and Fig. 2.15 below illustrates an example of interpolation by a factor $L = 4$. The input signal *x(n)* is "filled in"with L-1 zero-valued samples between each pair of samples of *x(n)*, giving the signal,

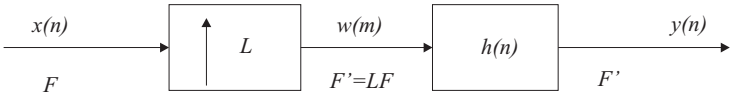
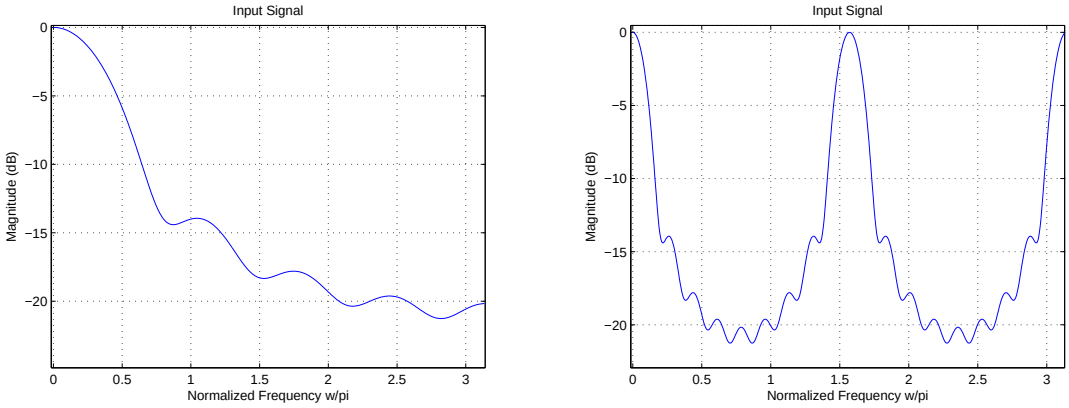
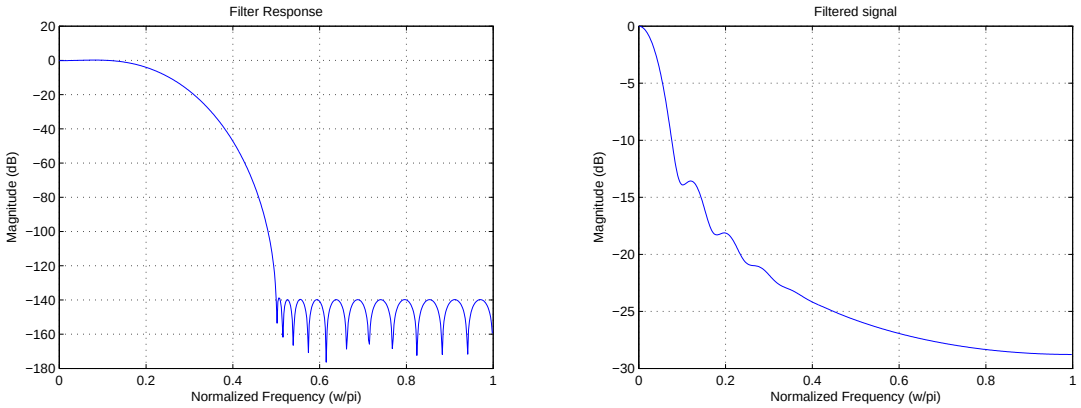


Figure 2.13: Interpolation by an integer factor $L = 4$.



(a) (b)

Figure 2.14: Input and Upsampled Signal.



(a) (b)

Figure 2.15: Anti-imaging filter Response and Interpolated Spectra.

$$w(m) = \begin{cases} x(\frac{m}{L}), & m = 0, \pm L, \pm 2L, ... \\ 0, & \text{otherwise.} \end{cases} \tag{2.63}$$

As with the decimation operation, the block diagram symbol of an up-arrow with an integer corresponds to increasing the sampling rate and referred to as *upsampler* or *expander*. The resulting signal *w(m)* has the z-transform

$$W(z) = \sum_{m=-\infty}^{\infty} w(m)z^{-m} = \sum_{m=-\infty}^{\infty} x(m)z^{-mL} = X(z^L) \tag{2.64}$$

evaluating Fourier transform of *W(z)* we have

$$W(e^{j\omega'}) = X(e^{j\omega' L}) \tag{2.65}$$

where,

$$\omega' = 2\pi f T' \cdot \tag{2.66}$$

As illustrated in the spectral interpretation in Fig. 2.14(b), the spectrum *w(m)* contains not only the baseband frequencies of interest but also *images* of the baseband centered at harmonics of the original sampling frequency $\pm 2\pi/L, \pm 4\pi/L, ...$ To recover the baseband signal of interest and eliminate the unwanted image components, it is necessary to filter the signal *w(m)* with a digital lowpass filter also known as *anti-imaging* filter which approximates the ideal characteristics

$$\tilde{H}(e^{j\omega'}) = \begin{cases} G, |\omega'| \leq \frac{2\pi F T'}{2} = \frac{\pi}{L} \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.67}$$

To ensure the amplitude of *y(m)* is correct, the gain G of the filter , must be L in the passband. Letting $H(e^{j\omega'})$ denotes the frequency response of an actual filter, the frequency response of output signal will be given by

$$Y(e^{j\omega'}) = H(e^{j\omega'})X(e^{j\omega' L}) \tag{2.68}$$

and within the approximation of Eq.(2.67),

$$\tilde{Y}(e^{j\omega'}) = \begin{cases} GX(e^{j\omega' L}, |\omega'| \leq \frac{\pi}{L} \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.69}$$

If *h(m)* denotes the unit impulse response of $H(e^{j\omega'})$, then *y(m)* can be expressed as

$$y(m) = \sum_{k=-\infty}^{\infty} h(m-k)w(k) \tag{2.70}$$

combining with Eq.(2.63), the time domain input-to-output relation of the interpolator is given by:

$$y(m) = \sum_{k=-\infty}^{\infty} h(m-k)x(\frac{k}{L}) = \sum_{k=-\infty}^{\infty} h(m-rL)x(r)\cdot \tag{2.71}$$

Here r is expressed as

$$r = \lfloor \frac{m}{L} \rfloor - n \tag{2.72}$$

where,$\lfloor u \rfloor$ again denotes the integer less than or equal to u. Applying to Eq.(2.71) we have

$$y(m) = \sum_{r=-\infty}^{\infty} h(m-rL)x(r)\cdot \tag{2.73}$$

This equation represents the output *y(m)* in terms of the input *x(n)* and the filter coefficient *h(m)*.

Similar to polyphase decimator, an interpolator can also be implemented as polyphase structure. The general and efficient structure of polyphase interpolator is shown in Fig. 2.16 and Fig. 2.17.

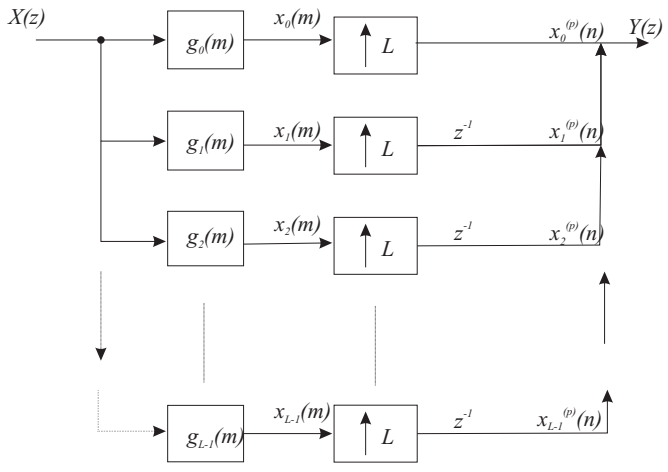


Figure 2.16: Polyphase Interpolator.

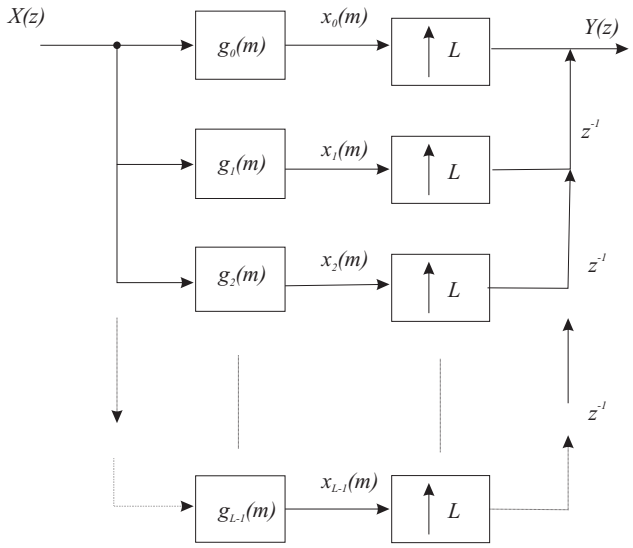


Figure 2.17: Efficient Polyphase Interpolator.

2.6. Sampling Rate Conversion by Rational Factor

In the previous two sections the cases of decimation by an integer factor M and interpolation by an integer factor L are considered. In this section, the general case of conversion by the ratio

$$\frac{T'}{T} = \frac{M}{L} \tag{2.74}$$

or

$$F' = \frac{L}{M}F\cdot \tag{2.75}$$

This conversion can be achieved by a cascade of the two processes of integer conversion discussed above by first increasing the sampling rate by L and then decreasing it by M. Fig. 2.18 illustrates this process. It is important to recognize that the interpolation by L must precede the decimation process by M so that the width of the baseband of the intermediate signal $s(k)$ is greater than or equal to the width of the baseband of $x(n)$ or $y(m)$. It can be seen from Fig. 2.18 that the two filters $h_1(k)$ and $h_2(k)$ are operating in cascade at the sampling rate LF. Thus, the more efficient implementation of the overall process can be achieved if the filters are combined into one composite lowpass filter as shown in Fig. 2.19. Since the digital filter $h(k)$, must serve the purposes of both the decimation and interpolation operations described in the preceding sections, it is clear from the Eqs.(2.46)

and (2.67) that it must approximate the ideal digital lowpass characteristics as

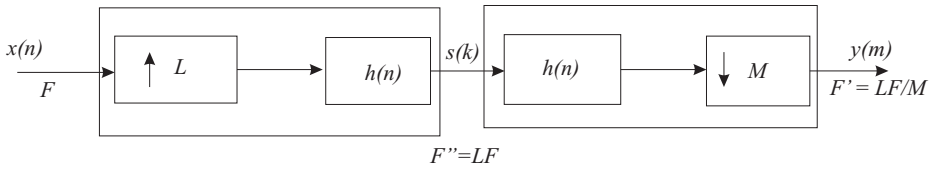


Figure 2.18: Cascade of an integer interpolator and an integer decimator for achieving rational factor sampling rate conversion.

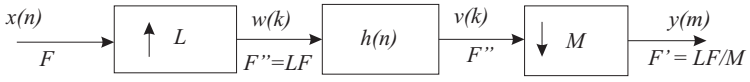


Figure 2.19: Rational Factor Sampling Rate Conversion.

$$\tilde{H}(e^{j\omega''}) = \begin{cases} L, |\omega''| \leq min|\frac{\pi}{L}, \frac{\pi}{M}| \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.76}$$

where,

$$\omega'' = 2\pi f T'' = 2\pi f \frac{T}{L}\cdot \tag{2.77}$$

This means that the ideal cutoff frequency must be the minimum of the two cutoff frequency requirements of the decimator and interpolator, and the sampling rate of the filter is $F'' = LF$. The time domain input-to-output relation for the general conversion circuit of Fig. 2.19 can be derived by considering the integer interpolation and decimation relation derived in section 2.5 and 2.6; then from Eq.(2.71) *v(k)* can be expressed as

$$v(k) = \sum_{r=-\infty}^{\infty} h(k - rL)x(r) \tag{2.78}$$

and from Eq.(2.48) *y(m)* can be expressed in terms of *v(k)* as

$$y(m) = v(mM)\cdot \tag{2.79}$$

Combining Eqs.(2.71) and (2.79) we arrive to the desired result

$$y(m) = \sum_{r=-\infty}^{\infty} h(mM - rL)x(r)\cdot \tag{2.80}$$

Alternatively, by making the change of variable

$$r = \lfloor \frac{mM}{L} \rfloor - n \tag{2.81}$$

and applying it to Eq.(2.82), we get

$$y(m) = \sum_{r=-\infty}^{\infty} h(\left\lfloor \frac{mM}{L} \right\rfloor L + nL)\, x(\left\lfloor \frac{mM}{L} \right\rfloor - n) \cdot \tag{2.82}$$

It is seen that Eq.(2.82) corresponds to the general form of the time-varying digital-to-digital conversion system described by Eqs.(2.41) and (2.42).

By considering the transform relationships of the individual integer decimation and interpolation systems, the output spectrum $Y(e^{j\omega'})$ can be determined in terms of the input spectrum $X(e^{j\omega})$ and the frequency response of the filter $H(e^{j\omega''})$. From Eq.(2.68) it is seen that $V(e^{j\omega''})$ can be expressed in terms of $X(e^{j\omega})$ and $h(e^{j\omega''})$ as

$$V(e^{j\omega''}) = H(e^{j\omega''})X(e^{j\omega'' L}) \cdot \tag{2.83}$$

Then from Eq.(2.55) $Y(e^{j\omega'})$ can be expressed in terms of $V(e^{j\omega''})$ as

$$Y(e^{j\omega'}) = \begin{cases} \frac{L}{M} X(e^{j\omega' L}, |\omega'| \leq min[\pi \frac{M}{L}] \\ 0, \quad \text{otherwise.} \end{cases} \tag{2.84}$$

Thus, we have developed the general system for sampling rate conversion of lowpass signal by arbitrary rational factors, L/M. It was shown that the process of sampling rate conversion could be modeled as a linear, periodically time-varying system.

2.7. The Conversion Factor

In order to reduce the cost of analog components in mobile communications terminals, the ADC should be clocked at a fixed rather than tunable rate [11]. Therefore, the conversion between the digitization rate and the variable target rate (symbol/chip rate) should be realized digitally. This process is referred to as SRC. As both the digitization rate and the target rate can be expressed as an integer number of samples per unit of time, so the rate change factor is a rational number. The ratio of output sample rate f_2 and the input sample rate f_1 is

$$\frac{f_2}{f_1} = \frac{T_1}{T_2} = \frac{L}{M} \tag{2.85}$$

with, L and M being relative prime. Although they are similar, integer factor SRC, and fractional SRC also have differences, especially in implementation. Therefore, it is sensible to separate fractional SRC and integer factor SRC. To do this the rate change factor can be factorized to a fractional part $(L/M)_{frac}$ and an integer part L_{int} or $1/M_{int}$

$$\frac{L}{M} = \begin{cases} (L/M)_{frac} \cdot L_{int} & \text{effective increase of the sample rate.} \\ (L/M)_{frac} \cdot \frac{1}{M_{int}} & \text{effective reduction of the sample rate.} \end{cases} \tag{2.86}$$

It should be noted that the fractional factor $(L/M)_{frac}$ is limited to the interval

$$0 \cdot 5 < \left(\frac{L}{M}\right)_{frac} < 2\cdot \tag{2.87}$$

if the overall conversion is an effective reduction of the sample rate. This factor can be limited further to

$$0 \cdot 5 < (\frac{L}{M})_{frac} < 1 \tag{2.88}$$

or in case of an effective increase of the sample rate to

$$1 < (\frac{L}{M})_{frac} < 2\cdot \tag{2.89}$$

However, the order of arranging the fractional and integer part is an open question. Two parameters that can help to decide this are number of coefficients N and the rate of multiplication of coefficients with the clock rate R_{mult}. The relation of these two parameters with respect to the $GOSR$ obeys the following proportionality

$$N \sim \frac{GOSR}{GOSR - 1} \tag{2.90}$$

$$R_{mult} \sim b\frac{GOSR^2}{GOSR - 1} \tag{2.91}$$

where, $\sim$ stands for the proportionality between the right-hand and left-hand side of equation. Hence, for a fixed bandwidth 'b', if $GOSR$ approaches to 1, the number of coefficients explodes where as on increasing the $GOSR$ the filter order decreases and the multiplication rate is high. Thus,it can be concluded that the hardware effort can be minimized when placing fractional SRC before integer factor SRC where the $GOSR$ of the signal is high. In case of minimizing the multiplication rate, it is recommendable to place the integer SRC before the fractional SRC. A comparative chart for justifying the above statements is demonstrated in the example below.

Example 2-1.

Realize a cascade of integer factor and fractional SRC for the parameters given in Table(2.1), where MSps stands for Mega Samples per second and kSps means kilo Samples per second .

Table 2.1: Parameters for SRC.

Input Sampling Rate	80 MSps
Signal Bandwidth b	200 kHz
Target rate (after SRC)	270.83 kSps
Conversion Factor	$\frac{13}{1920} = \frac{13}{2^7 \cdot 3 \cdot 5}$

Hence, from the conversion factor, the integer factor and fractional conversion factor can be derived as

Table 2.2: The splitted Conversion Factor.

Integer Conversion Factor	$\frac{1}{128}$
Fractional Conversion Factor	$\frac{13}{15}$

The cascading of these two conversion factors can be done in two ways as shown in Fig.2.20 and Fig.2.21.

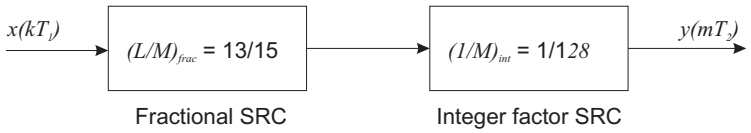


Figure 2.20: Case 1:Fractional SRC before Integer factor SRC.

The respective changes in GOSR after each stage in both the cases is calculated by using the Eq.(2.15) and is shown in Table 2.3. From this, it can be notice that the changes in GOSR after both the stages in Case 2 are same. While in Case 1 the GOSR is very high after fractional SRC in comparison to Case 2 where integer factor SRC is before fractional

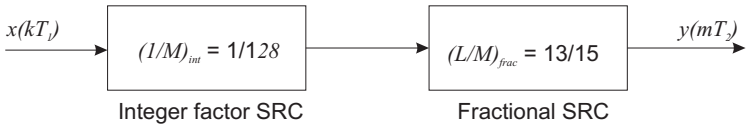


Figure 2.21: Case 2:Integer factor SRC before Fractional SRC.

Table 2.3: Comparision Table .

GOSR	Case 1	Case 2
GOSR after 1st stage	≈ 347	≈ 3.1
GOSR after 2nd stage	≈ 2.7	≈ 2.7

SRC. Thus, the cascading of conversion factor can be concluded with the advantages and disadvantages of fractional SRC as

Advantages

- Placing the fractional SRC at high sampling rate minimizes the filter order N by increasing GOSR (see Eq.(2.90)).
- The hardware efforts get reduces by less complexity and relaxed specifications.

Disadvantages

- From Eq.(2.91) the multiplication rate for high GOSR is high, and so the system has to work at high clock rate.
- There is a possibility of designing different filters for different communication standards.
- Placing the fractional SRC in front will require high attenuation of aliasing components for dynamic range multichannel signals.

3

Review of Finite Impulse Response (FIR) Filters

In this chapter, introduction of FIR filters are provided along with brief notes on different methods of designing FIR filters and various types of filter structures. Then, rounding and sharpening techniques applied for designing efficient FIR filters are presented. With the implementation of rounding and sharpening techniques, the designed filter will be multiplierless and hence will save the power consumption with the desired magnitude response. Similarly, utilization of raising techniques and block filtering will help to make filter analysis simple as well as easier. Finally, the time varying polyphase structures, to realize higher rational sampling rate factor based filter without facing high input frequency is explained.

3.1. Introduction

FIR filters are the digital filters whose impulse response are within the finite limits of time interval and have linear as well as non-linear phase response [18]. There are various methods of designing FIR filter, like

- Windowing Method: This method is applicable to design a FIR filter by truncating an infinite duration impulse response with the help of a suitable window function.
- DFT Method: In this method the impulse response is determined from the desired magnitude specifications and an appropriate phase response.
- Frequency-sampling method: When windowing and DFT method are combined to

find the filter of desired specification from finite impulse response, the design procedure is referred to as frequency-sampling method.

- Uniform Approximation Method: This method designs FIR filter with minimum error of deviation from desired amplitude frequency response. The approximation theorem is applied using iterative method of *Parks-McClellan* based on *Remez Exchange algorithm*.

FIR filters are the interconnection of three simple elements *adders*, *multipliers*, and *delays*. These elements are grouped together to make the filter structures that show the processing of a signal in a filter. The filter structure is further implemented to an integrated circuit. There are various filter structures which provide an efficient filter, like

- Transversal Structures: There two types of transversal structures, namely: direct form, and transposed direct form structures. These structures are further categorized for symmetrical impulse response.

- Lattice Structures: Standard lattice structures and QMF lattices are two widely applicable filter structures for filter banks.

- Structure Passivity: These structure is proposed by Vaidyanathan and Mitra [30], as the efficient filter structure which provide very low sensitive at passband and stopband.

There are various advantages and disadvantages of FIR filter. The advantages can be counted as

- FIR filters are stable filters.

- FIR filter can be designed for magnitude response specifications other than piecewise constant.

- FIR filter can has no phase distortion.

The disadvantages of FIR filter are

- It requires more computation.

- It has less computational complexity comparing with the infinite impulse response (IIR) filter with the same specifications.
- It requires more memory.

In the following sections, the methods of designing efficient FIR filters with less multipliers and which can provide desired specifications are demonstrated. For the time-varying filters, two recent techniques presented by Tim Hentschel [11] is also described in the continuing section.

3.2. Methods of designing Efficient FIR filters

3.2.1. Rounding

To improve FIR filter's efficiency, several techniques have been developed. Rounding Procedure as an alternative method has been proposed by [5] in which the filter impulse response is manipulated to reduce its complexity while retaining the magnitude response of the filter within set specifications through iterative process. The Rounding Procedure can be explained in the following steps:

- **Step 1**: In this step, *Parks-McClellan* algorithm is employed to design a basis filter with edge frequencies.
- **Step 2**: This is followed by the generation of impulse response h of basis filter, with determination of number of coefficients $L+1$.
- **Step 3**: The new impulse response h' is derived by rounding h according to the equation

$$h'_l = r \cdot round\left(\frac{h_l}{r}\right), \quad l = 0, 1, ..., L. \tag{3.1}$$

  where L is the order of basis filter, r is the rounding constant and function $round()$ rounds to the nearest integer.

The rounding operation reduces the number of step changes in the impulse response as demonstrated in example 3-1. Consequently, the number of additions and integer multiplications will decrease. If S represents the number of additions after rounding and R denotes

the number of integer multiplications, then both these terms varies with the change of rounding factor r and the shape of h. The principal objectives of designing an efficient filter through the method of rounding are

- To find the values of number of filter coefficients L, and r that results in a rounded response with minimum values of S and R, and
- The modified magnitude response H'(θ) should satisfies the given filter specifications.

Unfortunately, there are no straightforward solution to this problem, since there is no direct relationship between S and R, and L and r.

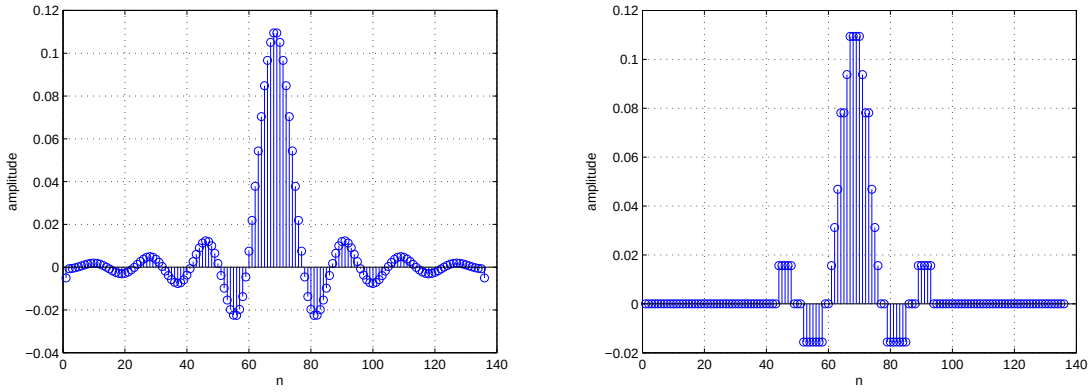
Example 3-1.

Design the basis filter with the normalized stopband frequency 0.125 and the corresponding passband frequency is $\frac{3}{4}$ of the stopband frequency. The maximum passband ripple is 0.09 dB and stopband ripple of 40 dB.

The corresponding equiripple filter has an order of 135 and requires 52 multipliers. Applying the rounding technique with the factor $r = 0.015625$ on the basis filter, gives the rounded filter with 5 integer multiplications and 37 additions. The basis filter impulse response and rounded filter impulse response are shown in Fig. 3.1 and consequent effect of rounding the basis filter on its magnitude response is compared in the Fig. 3.2. Through repetitive experimentation, it has been found that rounding technique offers worthwhile gains in computational efficiency for the following two conditions

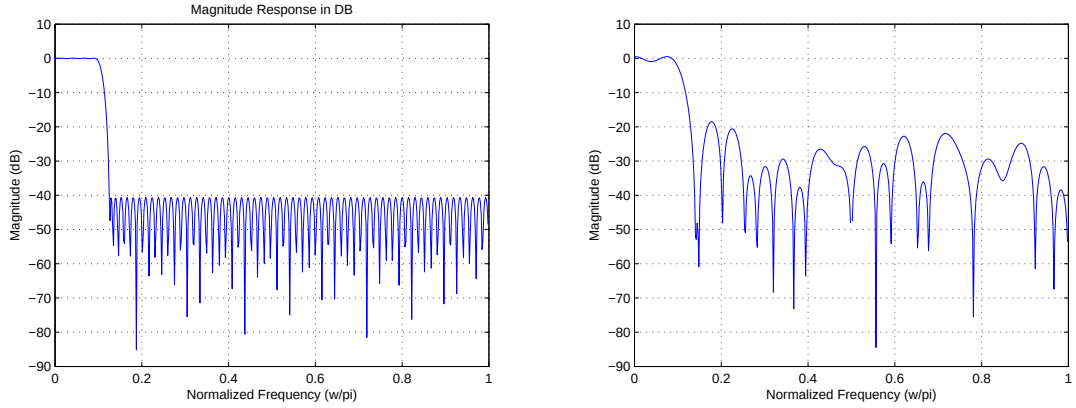
- The narrowband filters with cut-off frequencies below around 0.05 normalized frequency,and
- Transition bands of these filters should be in the region of about 0.01 or less, depending on the cut-off frequencies.

Thus, design of an optimum filter may therefore requires experimentation with a series of rounding constants and impulse response curves.



(a) (b)

Figure 3.1: Original and Rounded Impulse Response.



(a) (b)

Figure 3.2: Original and Rounded Magnitude Response.

3.2.2. Sharpening

The necessity to improve the performance of the filter leads to process the data by repeated passes through the same filter [16]. This process is called filter *sharpening*. The objective of sharpening is to achieve less passband error and more stopband attenuation. Since a long time various work have been performed by different investigators like [4, 14, 15, 26, 28], who have done sharpening of FIR filters as well as cascaded integrated comb (CIC) filters to obtain the desired response. Filter sharpening method depends on amplitude change function (ACF) idea presented by Hamming and Kaiser [16]. The ACF

describes the change of input filter amplitude with respect to output filter amplitude irrespective of frequency. This method is restrictive to symmetric non-recursive (finite impulse response) filters with piecewise constant passband and stopband. The desired response of ACF is to have a tangent change at both passband and stopband. A useful generalization is to construct a polynomial having an $nth-order$ tangency at zero, an $mth-order$ tangency at unity, and passing through the points (0,0) and (1,1)as shown in Fig. 3.3.

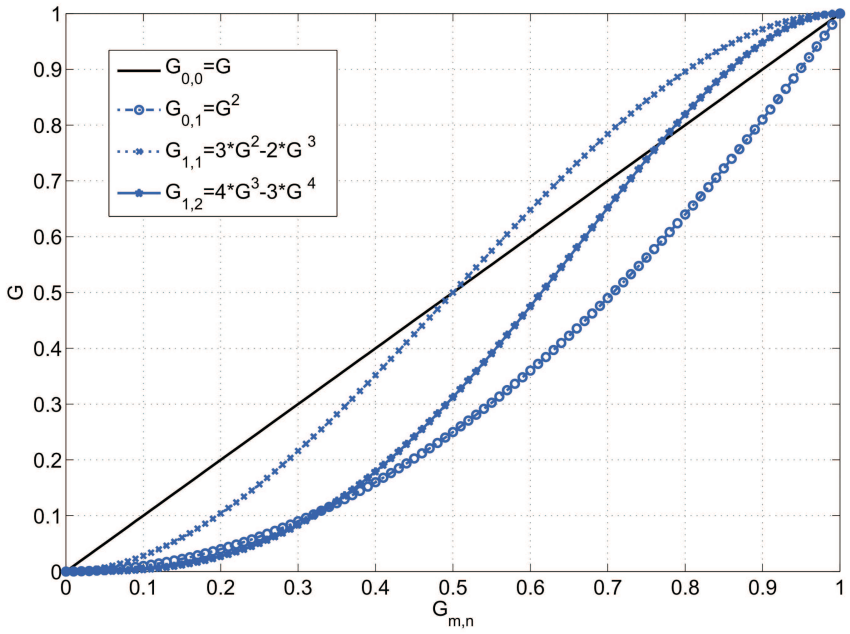


Figure 3.3: Amplitude Change function.

The desired polynomial can be shown as

$$H_{out}(f) = H^{n+1}(f)\sum_{\tau=0}^{m}\frac{(n+1)!}{n!k!}[1-H(f)]^k = H^{n+1}(f)\sum_{\tau=0}^{m}C(n+k,k)\overline{H}^k(f). \qquad (3.2)$$

where $C(n+k,k)$ is the well known binomial coefficient.

A list of few sharpening polynomials for m=n is given in the table below

Example 3-2.

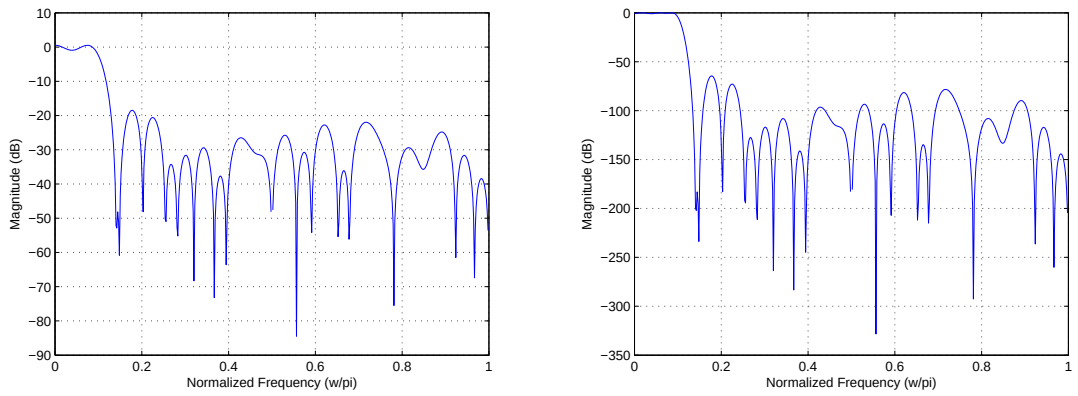
Continuing with the Example 3-1 of rounding, as the rounding of impulse response of basis filter has distorted the magnitude response at the passband and stopband,

Table 3.1: Sharpening Polynomials for m=n.

$n = m$	H_{out}	$AlternateForm$
0	H	H
1	$H^2(1 + 2\overline{H})$	$H^2(3 - 2H)$
2	$H^3(1 + 3\overline{H} + 6\overline{H}^2)$	$H^3(10 - 15H + 6H^2)$
3	$H^4(1 + 4\overline{H} + 10\overline{H}^2 + 20\overline{H}^3)$	$H^4(35 + 84H + 70H^2 + 20H^3)$

this distortion can be recovered by sharpening the same rounded basis filter until the desired specification is achieved.

The resulting improved basis filter along with the distorted one are shown in Fig. 3.4 However, filter sharpening led to increase in filter order, and computation amount.



(a) (b)

Figure 3.4: Rounded and Sharpened magnitude Response.

3.2.3. Interpolated Finite Impulse Response (IFIR) Filter

IFIR filter is a multistage filter proposed by Neuvo, Chang and Mitra in 1984 [22]. IFIR filter consists of cascade of expanded model filter $G(z^M)$ and interpolator filter I(z) as shown in Fig. 3.5. The IFIR filter has a relaxed design procedure replacing high order single stage filter H(z) with less order multistage filters.

The design procedure of IFIR filter involves the following steps.

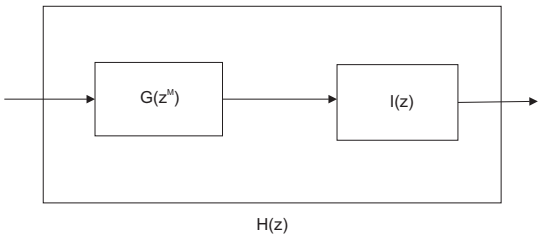


Figure 3.5: IFIR filter.

- Step 1: *Designing of Model Filter G(z).*

  The specification of the model filter are related to the specification of the filter to be design as well as the decimation factor. For Example, if the specification of desired filter H(z) are

$$\begin{aligned} \omega_p &\rightarrow passband \quad frequency; \\ \omega_s &\rightarrow stopband \quad frequency; \\ R_p &\rightarrow passband \quad ripples \quad in \quad dBs; \\ A_s &\rightarrow stopband \quad attenuations \quad in \quad dBs. \end{aligned} \tag{3.3}$$

  Then, the specification of model filter G(z) will be [22]

$$\begin{aligned} \omega_p^G &= \omega_p * M; \\ \omega_s^G &= \omega_s * M; \\ R_p^G &= R_p/2; \\ A_s^G &= A_s/2; \end{aligned} \tag{3.4}$$

- Step 2: *Designing of Expanded Model Filter* $G(z^M)$.

  Designing of an expanded model filter is achieved by upsampling the impulse response g(n) of model filter G(z) by M. This makes the insertion of M - 1 zeros between every samples of g(n). The consequent effect in the frequency response will be the formation of images at every $2\pi \cdot n/M$ for $n = 0, 1, ..., M-1$. This raises the demand of another filter which will eliminate the images and also provides the desired filter specification.

- Step 3: *Designing of Interpolation Filter I(z)*

  The specification of interpolator filter are such that the images of expanded model

filter will get eliminated and also the desired filter will be obtained on cascading both of them. To obtain this the passband of the interpolator filter is same as the passband of desired filter. While the stopband will be little more than the desired filter i.e $2\pi/M - \omega_s$, in order to eliminate all the images of extended model filter. Hence, wider transition band filter decreases the filter order and number of multipliers. The simulated figure of IFIR filter with the given specification along with the comparision table is described in example below.

Example 5-1.

Design an IFIR decimation filter with the decimation factor M as 16, normalized passband frequency ω_p as 3/4 of normalized stopband frequency ω_s. The passband ripples is 0.1 dB and the stopband attenuation is of 80 dB.

To design the IFIR filter we follow the step 1 to 3 demonstrated above. The specifications of model and interpolator filter for decimation factor M splitted to $M_1 = 8$ and $M_2 = 2$ and model filter is extended by M_1 are given in Table 5.4.

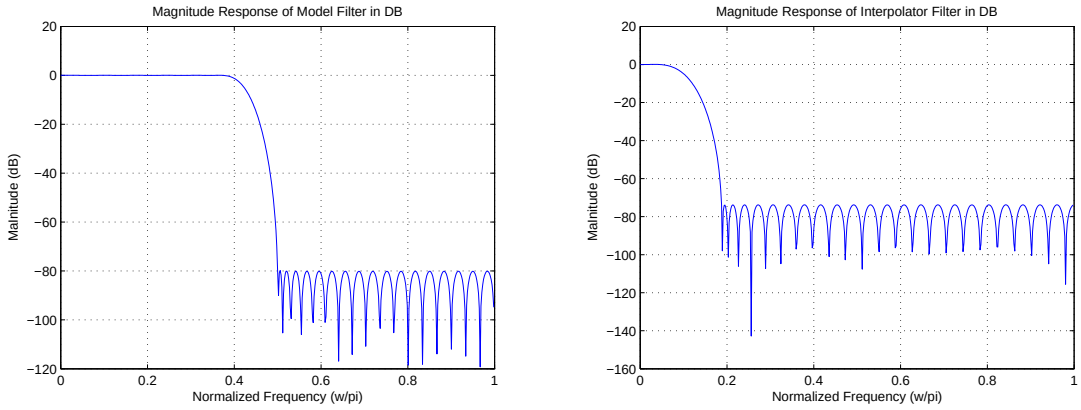
Table 3.2: Specifications of model and interpolator filters.

Specifications M	Model filter G(z)	Interpolator filter I(z)
Stopband Frequency, ω_s	$M_1 \times \omega_s$	$2\pi/M_1$ - ω_s
Passband Frequency, ω_p	$M_1 \times \omega_p$	ω_p
Passband Ripples, R_p	$R_p/2$ dB	$R_p/2$ dB
Stopband Attenuations, A_s	$A_s/2$ dB	$A_s/2$ dB

Designing model and interpolator filters with the above given specifications using Parks-McClellan algorithm will give us the magnitude responses shown in Fig. 3.6.

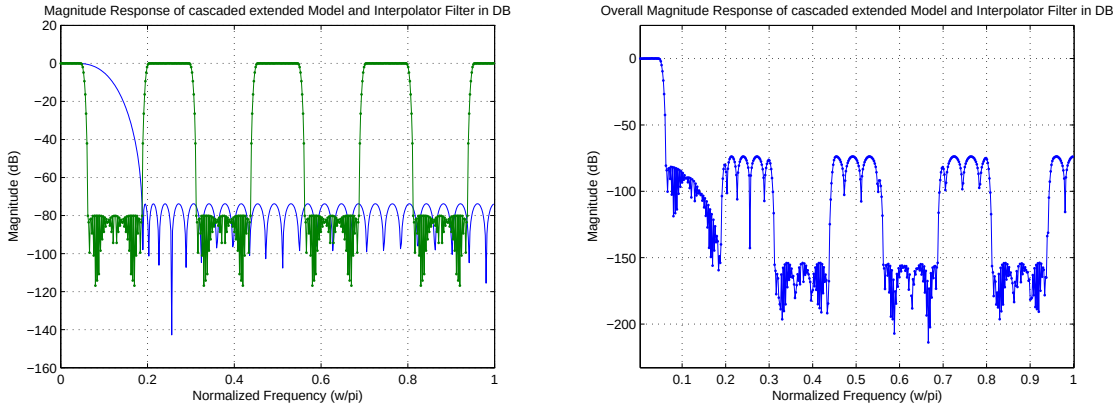
Now, cascade the expanded model filter and interpolator filter. The consequent magnitude responses are shown in Fig. 3.7. This complete procedure of designing IFIR can be drawn as given in Fig. 3.8 and Fig. 3.9.

The comparative study of resource requirements for an IFIR filter for fractional SRC are given in Table 3.3. Thus, from Table 3.3 it can be concluded that the IFIR filter reduces



(a) (b)

Figure 3.6: Model and Interpolator Filters Magnitude response.



(a) (b)

Figure 3.7: Cascaded Model and Interpolator Filters Magnitude response.

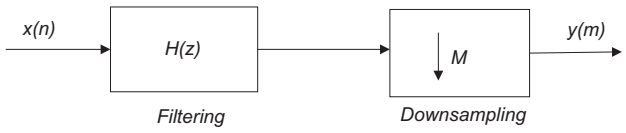


Figure 3.8: Decimation Filter.

the filter order and number of multipliers. Hence the complexity and power consumption also gets decreased.

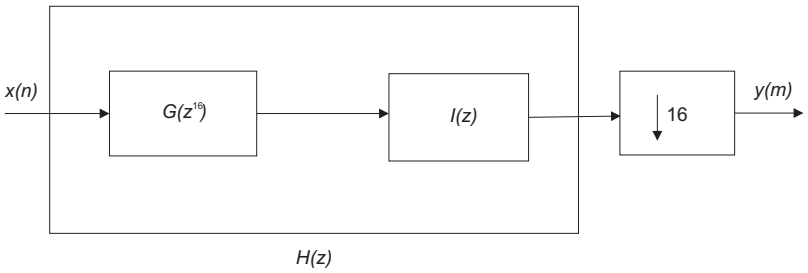


Figure 3.9: IFIR Decimation filter by factor $M = 16$.

Table 3.3: IFIR Filter required resources.

Resources	Original Filter H(z)	IFIR Filter
Adders	426	107
Multipliers	74	53
Filter order	426	108

3.3. Methods of analysing FIR filters

3.3.1. Raising Procedure

The raising procedure is widely applicable in the area of control system for the design and analysis of periodic controllers. Recently, it has been found that this procedure has great potential for the analysis and synthesis of DSPs [11].

Raising procedure is a technique of transforming a linear periodically time-varying system (LPTV) to a linear time-invariant (LTI) system. For a single input single output (SISO) periodic system, raising technique can be applied to convert it into multiple input multiple output (MIMO) system. In a multirate system, which is actually a combination of LPTV systems as well as LTI systems, raising technique is applied to each of them to obtain a complete LTI MIMO system. This system can now be synthesized easily.

Raising Procedure involves the analysis of system by raising their state-space equations. For example, a discrete-time periodically time-varying SISO system can be repre-

sented in state-space equation form as follows :

$$\begin{aligned} \underline{z}(n+1) &= A_n \cdot \underline{z}(n) + B_n \cdot x(nT) \\ y(nT) &= C_n \cdot \underline{z}(n) + D_n \cdot x(nT)\cdot \end{aligned} \tag{3.5}$$

Here, x(nT) and y(nT) are the input signal and output signal respectively. *$\underline{z}(n)$* is the state vector of the system and A_n,B_n,C_n, and D_n are N periodically time-varying system matrices represented as

$$\begin{aligned} A_{n+iN} &= A_n \\ B_{n+iN} &= B_n \\ C_{n+iN} &= C_n \\ D_{n+iN} &= D_n \\ for \quad i \quad \epsilon \quad Z\cdot & \end{aligned} \tag{3.6}$$

On raising the state-space form by N, each of these terms get raised. The raising of the input signal and output signal by N gives N- polyphase components of the respective signals which are represented as

$$\begin{aligned} \underline{\mathbf{x}}(n.NT) &= [x_0(n.NT)x_1(n.NT)...x_{N-1}(n.NT)]^T \\ \underline{\mathbf{y}}(n.NT) &= [y_0(n.NT)y_1(n.NT)...y_{N-1}(n.NT)]^T\cdot \end{aligned} \tag{3.7}$$

From z-transform we get

$$\begin{aligned} \underline{\mathbf{X}}(n.NT) &= [X_0(n.NT)X_1(n.NT)...X_{N-1}(n.NT)]^T \\ \underline{\mathbf{Y}}(n.NT) &= [Y_0(n.NT)Y_1(n.NT)...Y_{N-1}(n.NT)]^T\cdot \end{aligned} \tag{3.8}$$

Similarly, raised state vector and system matrices are given as

$$\underline{\mathbf{z}}(n) = \underline{z}(nN) \tag{3.9}$$

$$\mathbf{A} = A_{N-1}A_{N-2}...A_0 \tag{3.10}$$

$$\mathbf{B} = [A_{N-1}...A_1B_0, A_{N-1}...A_2B_1...A_{N-1}B_{N-2}, B_{N-1}] \tag{3.11}$$

$$\mathbf{C} = \begin{bmatrix} C_0 \\ C_1A_0 \\ C_2A_1A_0 \\ \vdots \\ C_{N-1}A_{N-2}...A_0 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} D_0 & 0 & \dots & 0 \\ C_1B_0 & D_1 & \dots & 0 \\ C_2A_1B_0 & C_2B_1 & \dots & 0 \\ \vdots & & & \\ C_{N-1}A_{N-2}...A_1B_0 & C_{N-1}A_{N-2}...A_2B_1... & & D_{N-1} \end{bmatrix}$$

Thus, raising procedure makes it easy to analyze LPTV systems by using the methods and tools of LTI systems. Raising procedure conserve the linearity of the system. However, while implementing a raised LPTV system intra-period oscillation and sensitivity to high frequency might be expected. To analyze these factors it is necessary to understand the frequency characteristics of the raising procedure.

The frequency behavior of a discrete time system can be analyzed by evaluating z-transform of raised transfer matrix of the system given as

$$\underline{\mathbf{Y}}(z) = \mathbf{H}(z) \cdot \underline{\mathbf{X}}(z) \cdot \tag{3.12}$$

Here, $\mathbf{H}$(z) is the raised transfer function given by

$$\mathbf{H}(z) = \mathbf{C}(z\mathbf{I} - A)^{-1}\mathbf{B} + \mathbf{D}. \tag{3.13}$$

Exploiting, the input-output relationship of the raised system with respect to the modulation components [9] can be expressed by means of $H(z^N)$ as

$$\underline{\mathbf{Y}}^{[p]}(z) = \frac{1}{M} \cdot W_M \cdot \underline{\mathbf{Y}}^{(m)}(z) = \frac{1}{M} \cdot W_M \cdot \underline{\mathbf{Y}}^{(m)}(z) \tag{3.14}$$

Therefore, substituting Eq.(3.14) in Eq.(3.12) will give

$$\underline{\mathbf{Y}}^{(m)}(z) = W_M^{-1} \cdot \mathbf{H}(z) \cdot W_M \cdot \underline{\mathbf{X}}^{(m)}(z) \cdot \tag{3.15}$$

Here, W_M is the DFT matrix (see Appendix), and $\underline{\mathbf{X}}^{(m)}(z)$ and $\underline{\mathbf{Y}}^{(m)}(z)$ are the modulation components of raised input signal and output signal. Replacing $W_M^{-1} \cdot \mathbf{H}(z) \cdot W_M$ by the modulation matrix $\mathbf{H}^{(m)}(z)$, Eq.(3.15) can be rewritten as

$$\underline{\mathbf{Y}}^{(m)}(z) = \mathbf{H}^{(m)}(z) \cdot \underline{\mathbf{X}}^{(m)}(z) \cdot \tag{3.16}$$

Its Fourier transform gives

$$\underline{\mathbf{Y}}^{(m)}(e^{j\omega}) = \mathbf{H}^{[m]}(e^{j\omega}) \cdot \underline{\mathbf{X}}^{[m]}(e^{j\omega}) \tag{3.17}$$

where, $\omega = 2\pi fT$.

Comparing Eq.(3.17) with the decimation equation in frequency domain reveals that the spectrum of the output signal of an N-periodic LPTV system equals a sum of frequency shifted spectra of the original signal, as in the case of decimation. However, the transfer function in the raised filters system are the components of alias component matrix. They determine the amount of aliasing due to the superposition of the different spectra.

3.3.2. Block Filtering

The block filtering is greatly related to the multirate systems and the raising procedure [11]. Fig. 3.10 shows a (L,M)-shift-invariant system.

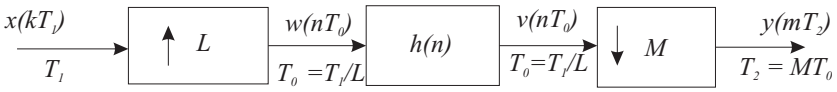


Figure 3.10: LM-shift invariant system.

Block filtering of (L,M)-shift-invariant system leads to the block version of input and output signals, and also the block version of transfer function is represented by transfer matrix $H^{[B]}(z)$.

Since the L,M-shift-invariant system is a LPTV, it means a shift of M in the input signal will be equal to L shifted output signal. This also signifies that the length of block versions for input and output signals are different. Also there are signals of different sample

rates, so it is necessary to relate the block version of them as

$$\underline{\mathbf{w}}(m.LMT) = \begin{bmatrix} w'(m \cdot LMT) \\ w'((m \cdot LM + 1)T) \\ \vdots \\ w'((m \cdot LM + LM - 1)T) \end{bmatrix}$$

$$\underline{\mathbf{v}}(m.LMT) = \begin{bmatrix} v'(m \cdot LMT) \\ v'((m \cdot LM + 1)T) \\ \vdots \\ v'((m \cdot LM + LM - 1)T) \end{bmatrix}$$

with the elements of $\underline{\mathbf{w}}(m.LMT)$ being defined by

$$w'((mLM + i)T_1) = \begin{cases} w'((mM + k)T), i = L_0 + kL, k = 0, 1, ..., M - 1 \\ 0, \quad \text{otherwise.} \end{cases} \tag{3.18}$$

and the output signal of the whole system is given by the components of the raised output signal of the filter

$$y((mL + k)T_2) = v'((mLM + M_0 + kM)T), k = 0, 1, ..., L - 1 \tag{3.19}$$

where $T = T_1/L = T_2/M$.

In Fig.3.10, an upsampler takes MJ samples and introduces L -1 zero samples between each of the input samples, thus delivering LMJ samples at its output. Downsampler will take LMJ samples and just pass every M^{th} sample, and letting LJ samples pass, where J is the periodicity factor. Hence, the respective matrices describing these processes must have the dimensions LJ x LMJ and LMJ x MJ, respectively. Downsampling is achieved by taking a vector of length LMJ and multiplying it from the left with the downsampling matrix, which yields a vector of length LJ. Upsampling works equivalently. As both upsamplers and downsamplers are memoryless, their transfer matrices describe a simple mapping of elements of the input vector to elements of the output vector. By inspection the upsampling and downsampling matrices can be expressed as

$$\mathbf{H}^{[up]}_{(LMJ,MJ)} = [l_{i,k}] \tag{3.20}$$

with

$$l_{i,k} = \begin{cases} 1, i = L_0 + kL; k = 0, 1, ..., MJ - 1; 0 \leq L_0 < L \\ 0, \quad \text{otherwise} \end{cases} \tag{3.21}$$

where L_0 is the upsampling offset, L is the upsampling factor and MJ is the size of input signal vector.

$$\mathbf{H}^{[down]}_{(LJ,LMJ)} = [m_{i,k}] \tag{3.22}$$

with

$$m_{i,k} = \begin{cases} 1, \quad k = M_0 + iM; i = 0, 1, ..., LJ - 1; 0 \leq M_0 < M \\ 0, \quad \text{otherwise} \end{cases} \tag{3.23}$$

here, M_0 is the downsampling offset, M is the downsampling factor and LJ is the size of output signal vector.

In order to describe the complete (L,M)-shift-invariant SISO system in its block form, the following MJ-raised and LJ-raised vectors are defined

$$\underline{\mathbf{x}}_{(MJ)}(m \cdot MJT_1) = \begin{bmatrix} x(mMJT_1) \\ x((mMJ+1)T_1) \\ \vdots \\ x((mMJ+MJ-1)T_1) \end{bmatrix}$$

$$\underline{\mathbf{y}}_{(LJ)}(m \cdot LJT_2) = \begin{bmatrix} y(mLJT_2) \\ y((mLJ+1)T_2) \\ \vdots \\ y((mLJ+LJ-1)T_2) \end{bmatrix}$$

which are the block input and output signals, respectively, of the MJ-input LJ-output block system. It should be noted that $MT_1 = LT_2$. Eventually, the (LJ,MJ)-shift-invariant SISO system can be described by the raised LMJ-input LMJ-output version of the system with the MJ-raised and LJ-raised vector as input and output signals, respectively

$$\begin{aligned} \underline{\mathbf{z}}(k+1) &= \mathbf{A} \cdot \underline{\mathbf{z}}(k) + \mathbf{B} \cdot \mathbf{H}^{[up]} \cdot \underline{\mathbf{x}}_{(MJ)}(k \cdot MJT_1) \\ \underline{\mathbf{y}}(k \cdot LJT_2) &= \mathbf{H}_{[down]} \cdot [\mathbf{C} \cdot \underline{\mathbf{z}}(k) + \mathbf{D} \cdot \mathbf{H}^{[up]} \cdot \mathbf{x}_{(MJ)}(k \cdot MJT_1)] \cdot \end{aligned} \tag{3.24}$$

This is equivalent to

$$\begin{aligned} \underline{\mathbf{z}}(k+1) &= \tilde{\mathbf{A}} \cdot \underline{\mathbf{z}}(k) + \tilde{\mathbf{B}} \cdot \mathbf{H}^{[up]} \cdot \underline{\mathbf{x}}_{(MJ)}(k \cdot MJT_1) \\ \underline{\mathbf{y}}(k \cdot LJT_2) &= \mathbf{H}_{[down]} \cdot [\tilde{\mathbf{C}} \cdot \underline{\mathbf{z}}(k) + \tilde{\mathbf{D}} \cdot \mathbf{H}^{[up]} \cdot \underline{\mathbf{x}}_{(MJ)}(k \cdot MJT_1)] \cdot \end{aligned} \tag{3.25}$$

Finally, the block-transfer matrix of the system is given as:

$$\mathbf{H}^{[B]}(z) = \mathbf{H}^{[down]} \cdot \mathbf{H}(z) \cdot \mathbf{H}^{[up]}. \tag{3.26}$$

3.3.3. Time Varying Polyphase Structures

The basic idea of polyphase structure is to simply use the well known polyphase filters for interpolation or decimation. A combination with an additional upsampler (together with a polyphase decimator) or a downsampler (together with a polyphase interpolator) reveals that the structure does not need to be changed when using them for fractional SRC. Just the controlling must be adapted to the application to fractional SRC [11].

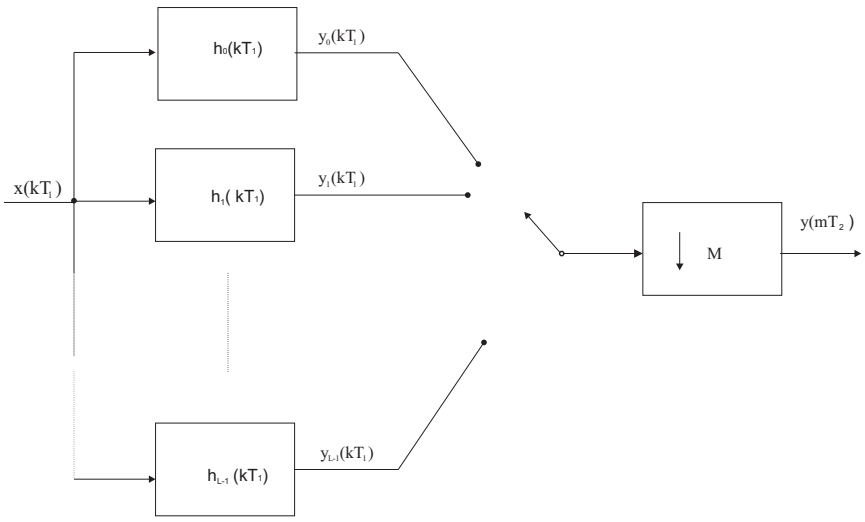


Figure 3.11: Polyphase interpolator with downsampler.

Fig. 3.11 illustrate the block diagram of time varying polyphase interpolator followed by downsampler. The sample rate converted signal at the output of Interpolator is obtain by modifying Eq.(2.71) and introducing an offset λ

$$y(m+\lambda) = \sum_{k=-\infty}^{\infty} h(nL)x(\lfloor \frac{m+\lambda}{L} \rfloor - n) \tag{3.27}$$

This response is implemented using polyphase representation and then the commutator rotating switch behavior is use to select the proper polyphase branch outputs of L polyphase branches and deliver the output sample. These output samples are then pass through the downsampler with factor M. The final output is given as

$$y(mM+\lambda) = \sum_{k=-\infty}^{\infty} h(nL)x(\lfloor \frac{mM+\lambda}{L} \rfloor - n) \tag{3.28}$$

The time varying polyphase structure is widely applicable for fractional SRC due to the following reasons

1. The resulting system is time varying.
2. The timing relation between input and output are mutually asynchronous.
3. The hardware effort for implementing this structure is of same order as conventional filter but it is clocked at the rate of output sample.

Similar, effort exist in case of upsampler followed by polyphase decimator. In this case the upsampled input signal is decimated by a polyphase decimator. This structure is shown in Fig. 3.12 below and is widely applicable for high ratio fractional sampling conversion.

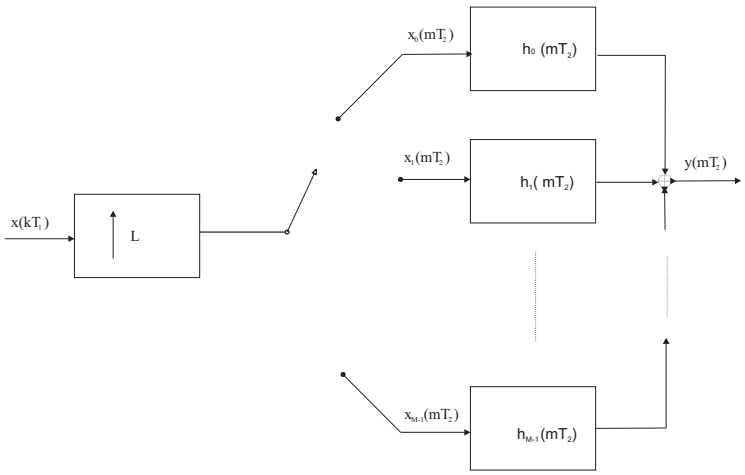


Figure 3.12: Upsampler followed by Polyphase Decimator.

4

Overview of some existing methods for designing SRC filters

This chapter presents the overview of some existing methods for designing SRC filters which are being proposed during the last few decades like cascaded-integrated-comb (CIC) filter, Farrow filter, and time varying CIC filter.

4.1. Introduction

Sampling rate conversion system for fractional decimation needs to design an efficient anti-aliasing. During downsampling the anti-aliasing filter prevents the overlapping of signal spectrum. Various work has been done since last few decades to design anti-aliasing filters. This chapter discusses some of the well known decimation filters. The next section describes about CIC filter. In the further sections Farrow filter and Time-Varying CIC filter has been briefly illustrated.

4.2. Cascaded Integrated Comb Filter based methods

A class of digital linear phase finite impulse response (FIR) filters for decimation (sample rate decrease) and interpolation (sample rate increase) has been presented by Eugene B. Hogenauer [12]. This filter requires no multipliers and use limited storage, making it an economical alternative to conventional implementation for certain applications. The

basic structure of CIC decimation filter consists of integrator section operating at high sampling rate. This section is also called as an accumulator or single pole IIR filter. This can be represented as:

$$y[n] = y[n-1] + x[n]. \tag{4.1}$$

$$H_I(z) = \frac{1}{1 - z^{-1}}. \tag{4.2}$$

$$|H_I(e^{j\omega})|^2 = \frac{1}{2(1 - cos\omega)}. \tag{4.3}$$

The structure of single integrator is drawn in Fig. 4.1

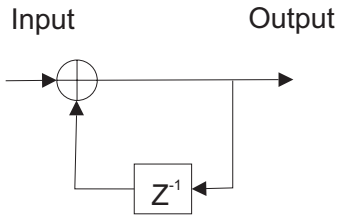


Figure 4.1: Integrator Section.

Following the integrator section, CIC filter has Comb section described by

$$y[n] = x[n] - x[n - RM]. \tag{4.4}$$

Here, R is the differential delay of samples per stage and M is the rate change factor. The differential delay is a filter design parameter to control the frequency response of filter. The corresponding transfer function of comb section is

$$H_c(z) = 1 - z^{-RM}. \tag{4.5}$$

and power transfer function

$$|H_c(e^{j\omega})|^2 = 2(1 - cosRM\omega). \tag{4.6}$$

The structure of single comb structure is drawn in Fig.4.2.

Hence, the complete CIC filter structure contains Integrator section followed by comb

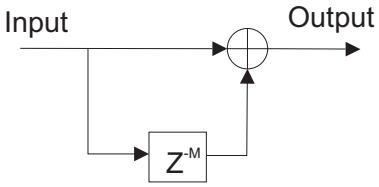


Figure 4.2: Comb Section.

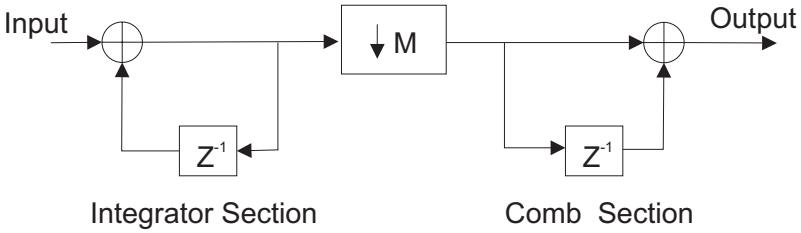


Figure 4.3: CIC Decimation filter.

section at higher sampling rates. This structure has been simplified by using multirate identity and shifting the comb section after rate change factor 'M'.

Hogenaur discussed three important aspects of CIC filter which includes the existence of nulls at multiples of $f = 1/R$. Here, R is the differential delay, which controls the placement of nulls at passband aliasing causing regions. From implementation point of view, register growth, and truncation or rounding have been explained with relation to filter design parameters and characteristics, respectively. After designing the CIC filter of desired characteristics based on rate change factor M, differential delay M and number of integrator and comb stages N, the most significant bit of the filter is determined as the function of overall register growth. The maximum register growth is defined as the maximum output magnitude resulting from the worst possible input signal relative to the maximum input magnitude. This growth is used in the CIC filter design process to insure that no data are lost due to register overflow. Using this definition, the maximum register growth from the first stage up to and including the last stage is

$$G_{max} = (RM)^N. \tag{4.7}$$

If the number of bits in the input data stream is B_{in}, then the register growth can be used

to calculate B_{max}, the most significant bit at the filter output as

$$B_{max} = \lceil Nlog_2RM + B_{in} - 1 \rceil. \tag{4.8}$$

where the least significant bit (LSB) of the input register is considered to be bit number zero and $\lceil x \rceil$is the smallest integer not less than x. B_{max} is large for many practical cases and can result in large register widths; however, truncation or rounding may be used at each filter stage reducing register widths significantly. It is assumed that the number of bits retained in the output register is B_{out}, so the number of LSB's discarded is

$$B_{2N+1} = B_{max} - B_{out} + 1. \tag{4.9}$$

In the next example a design procedure of CIC filter has been explained. The implementation uses two building blocks: a 16 bit integrator and a 16 bit comb sections. Each building block has a 16 bit input, 16 bit output, carry-in and carry out.

Example 4-1.

Design a CIC decimation filter to reduce the sampling rate from 6 MHz to 240 kHz with a passband of 30 kHz. The aliasing attenuation must be better than 60 dB with a falloff in the passband of less then 3 dB.The rate change factor is 25 and the number of integrator and comb section are 4 with differential delay 1 and cutoff frequency 1/8.

The frequency characteristics of CIC filter is lowpass and hence frequency response is evaluated at frequency $f = f_s/R$, where f_s is the sampling frequency. Hence, depending on the chosen parameters, it provide acceptable passband characteristics over the range zero to a predetermined cutoff frequency f_c.Thus, the power response is

$$P(f) = \left[\frac{sinRf}{sinM}\right]^{2N}. \tag{4.10}$$

This signifies that the nulls in the power response exist at multiples of $f = 1/R$, and thus differential delay R can be used as a design parameter to control the placement of nulls. For CIC decimation filters, the region around every R^{th} null is folded into the

passband causing aliasing errors specifically, this band ranges are

$$(i - f_c) \leq f \leq (i + f_c). \tag{4.11}$$

for $f \leq \frac{1}{2}$ and $i = 1, 2, ..., \lfloor \frac{M}{2} \rfloor$ where $\lfloor x \rfloor$ is the largest integer not greater than x. Based on this the following frequency response is obtained

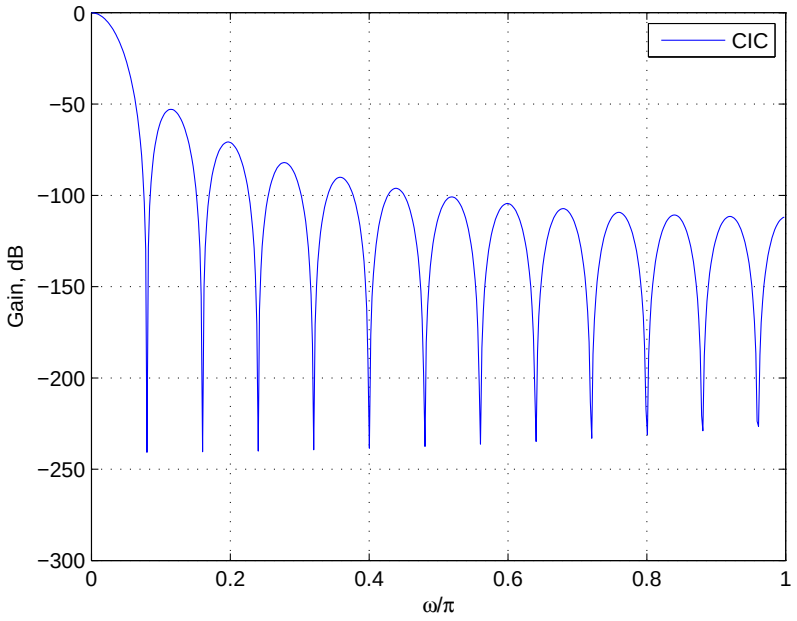


Figure 4.4: CIC filter Frequency response.

The SNR of the filter response is 15 dB, where the SNR is defined as the power ratio after lowpass filtering of the lowest power level in the desired signal to the highest power level in the images. From Eq.(4.8) and Eq.(4.9), the MSB bit of filter $B_{max} = 34$ and number of bits discarded $B_{2N+1} = 19$ have been calculated.

Thus, CIC filter has the good characteristics like no multipliers and limited storage requirement with simple design. However, it has some limitations like high passband droop, and low stopband attenuation. To improve the frequency response of CIC filter many researchers have proposed various methods. Some of the methods are Sharpening of CIC filter given by Kwentus [4], Compensator filters proposed by Chan and Yeung, Modified CIC filter of Abu-Al-Saud [3], and Stepped Triangular CIC filter given by Dolecek and Mitra [13].

4.3. Farrow Filter

The Farrow filter [8] offers two alternative approaches to the arbitrary resampling problem. In the first approach, coefficients of the polyphase filter stages are computed on the fly from the low order piecewise polynomials. These polynomials form approximations to segments of the impulse response of the original interpolating filter prior to the polyphase partition. In the second alternative, the coefficients of the approximating polynomials are rearranged and are applied directly to the input data to form data-dependent, locally valid, polynomial approximations to the input data as opposed to the filter. The data polynomial is in turn evaluated at the desired sample points.

The classical interpolator for arbitrary sampling are designed and implemented concurrently for up sampling and down sampling by using polyphase components. The number of polyphase components provides the delay for the output sample interval. Down sampling is implemented concurrently to realize rational ratio sample rate change. The next step involves the mapping of the prototype filter by doing partition of the polyphase filters for applying polynomial approximation. This mapping can be easily visualized by two dimensional array. The two-dimensional array maps the N-tap prototype filter to Pth component polyphase filter rows and N/P columns. Each of the rows of the partition corresponds to a sample point in the interpolated data stream. The next issue is the column coefficients, which hold the coefficients of each stage of polyphase filter. These coefficients are then modeled into P-sample wide section representing a smooth continuous function. Polynomial approximation of these function can be done by using the low-order polynomials in order to coefficients for a filter stage at any position.

The mth coefficient of the interpolation polynomial for the offset $\triangle$ is computed by evaluating the polynomial $P_m(x)$ at $x = \triangle$. The polynomial P_m is the approximation assigned to the mth column of the polynomial interpolator. The form of the polynomial is given in Eq.(4.12).

$$P_m(x) = \sum_{l=0}^{M} b(l, m)x^l. \tag{4.12}$$

where M is the order of the polynomial approximation. The output of the filter using the

coefficients from Eq.(4.12) is shown in Eq.(4.13)

$$y(n+\triangle) = \sum_{m=0}^{M} P_{\triangle} x(n-m). \tag{4.13}$$

By substituting Eq.(4.12) in Eq.(4.13), we get Eq.(4.14)

$$y(n+\triangle) = \sum_{m=0}^{M} \sum_{l=0}^{M} b(l,m)\triangle^{l} \cdot x(n-m). \tag{4.14}$$

representing the new set of data dependent coefficients c(n,l) as

$$c(n,l) = \sum_{m=0}^{M} b(l,m) \cdot x(n-m). \tag{4.15}$$

The consequent substituted equation is

$$y(n+\triangle) = \sum_{l=0}^{M} c(n,l)\triangle^{l}. \tag{4.16}$$

Examining Eq.(4.16), we recognize the form as the Taylor series representation of the output sequence. These represents output of the Farrow filter.

A processor flow diagram is given in Fig. 4.5

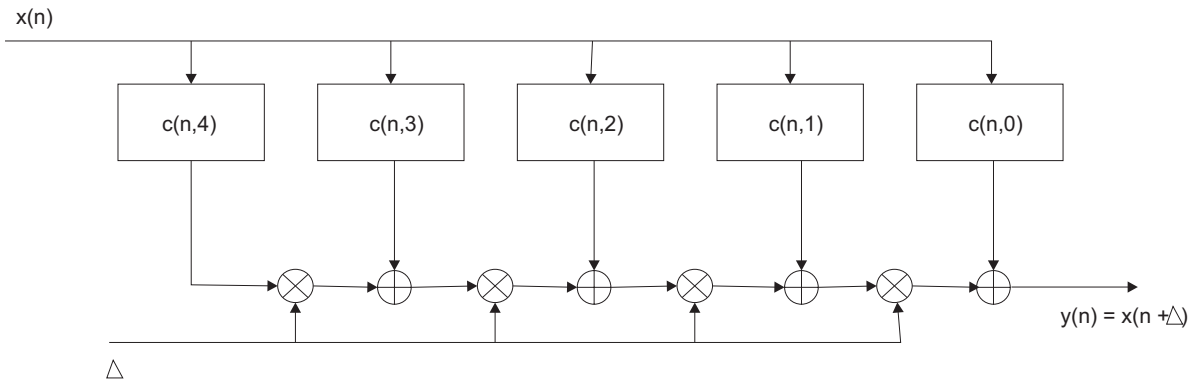


Figure 4.5: Farrow Structure.

The Farrow filter can be characterized by the following positive points:

- Easy design
- Applicable for arbitrary rational conversion factor
- Reconfigurable for different designs

The unique problem of Farrow filter based decimation method is the requirement of multipliers. To reduce the multipliers operation various methods have been proposed like Farrow structure based fractional decimation filter of Tim Hentschel [11] and the modified transposed Farrow structure based SRC filter for Software Radio proposed by Li and Tomisawa [19].

4.4. Time Varying Cascaded Integrated Comb Filters

Fractional SRC as discussed in chapter 2 consists of an expander followed by an anti-imaging-anti-aliasing filter $h(n)$ and then a compressor is again drawn below

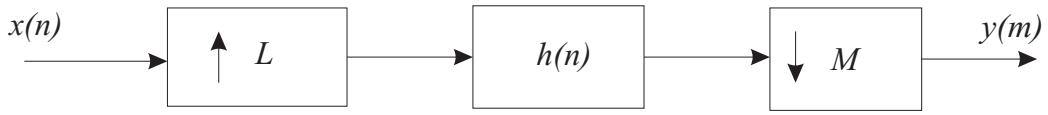


Figure 4.6: Fractional SRC system.

Fig. 4.6 can be obtain by using CIC interpolation filter in cascade with CIC decimation filter with the selection of order m depending on the requirement of suppressing images and aliasing effect as illustrated below

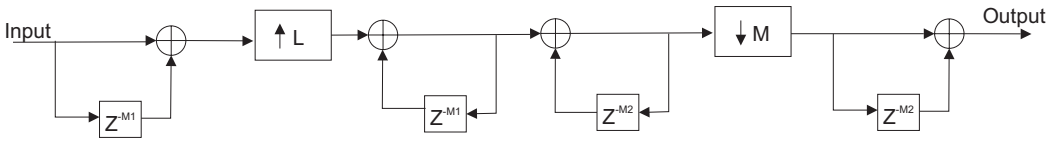


Figure 4.7: Fractional SRC system using CIC filter.

From Fig. 4.7, it can be concluded that the integrator section of CIC interpolator filter and CIC decimator filter can be group together. This will give new structure of fractional CIC SRC system with single integrator section of order m. Hence, Fig. 4.6 is redrawn as Fig. 4.9 implies that the integrator section has always to work at sample rate which is 'L' times of input sampling rate. This will be a problem, as high sample rate will makes the device to work at higher rate, consequently consumption of energy increases lead to the recharging of device frequently. To eliminate this problem, time-varying devices has

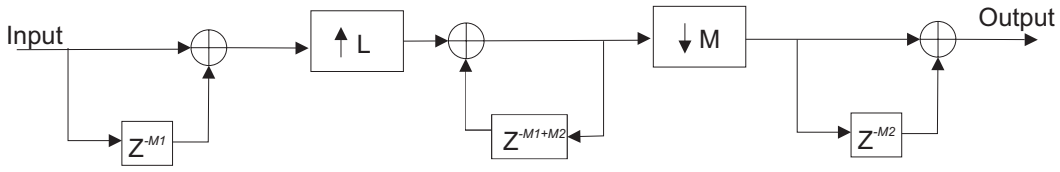


Figure 4.8: Integrated fractional SRC system using CIC filter.

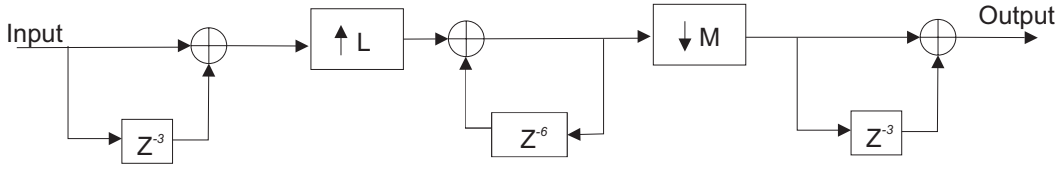


Figure 4.9: Third-order CIC filter.

been proposed [10]. The main objective of time-varying devices is to select the desired information during run time and at the input sampling rate.

Time-varying CIC (TVCIC) filters also based on the same concept. Since, in TVCIC the integrator section has to work at higher rate. So it can be replaced by time-varying structure. TVCIC exploits *raising procedure* and *block filtering* methods to generate a *time-varying based polyphase structures* as discussed in Chapter-3. The example below demonstrates the steps for designing a third order TVCIC filters.

Example 4-4.

Design a third order fractional decimation TVCIC filter from the CIC based fractional decimation filter given below. Take $L = 13$ and $M = 15$.

Considering the *expander* and *integrator section*, TVCIC filter is derived from above structure by the following steps

Step 1: Find the state space equation of *expander* and *integrator section*.

An expander is a LPTV system with single input and single output. The output $w(m)$ is related to input $x(n)$ as :

$$w(m) = \begin{cases} x(n/L) & \text{for} \quad \text{n} = \text{mL}, \quad \text{m} \quad \text{is} \quad \text{an} \quad \text{integer}. \\ 0 & \text{otherwise}. \end{cases} \tag{4.17}$$

The corresponding symbol is shown in Fig. 4.10 The state-space equation of expander is

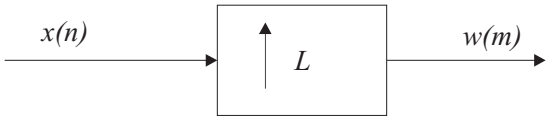


Figure 4.10: Expander.

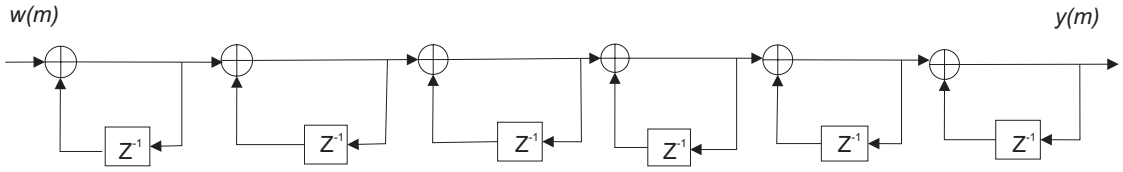


Figure 4.11: Integrator Section.

$$w(m) = x(n/L) \tag{4.18}$$

hence, system matrices are

$$A = 0; \tag{4.19}$$

$$B = 0; \tag{4.20}$$

$$C = 0; \tag{4.21}$$

$$D = 1 \tag{4.22}$$

Alternative representation of integrator section can be as State space equation obtain from Eq.(3.5) is rewritten as

$$\begin{aligned} \underline{z}(n+1) &= A \cdot \underline{z}(n) + B \cdot w(n) \\ y(n) &= C \cdot \underline{z}(n) + D \cdot w(n) \end{aligned} \tag{4.23}$$

thus, applying this equation at each node of Fig. 4.11, we get the system matrices as follows

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 1 \end{bmatrix}$$

Step 2: Apply raising procedure for expander and integrator section.

Since, expander is a LPTV system, so we can refer to section 3.3 to raise it. Here, raising is done by factor 'L' to have a less complex system. The raised expander in state space is given as:

$$\underline{\mathbf{x}}_{(MJ)}(m \cdot L = \begin{bmatrix} x(mLT_1) \\ x((mL+1)T_1) \\ \vdots \\ x((mL+L-1)T_1) \end{bmatrix}$$

$$\underline{\mathbf{w}}_{(LJ)}(m \cdot L) = \begin{bmatrix} w(mL) \\ w((mL+1)) \\ \vdots \\ w((mL+L-1)) \end{bmatrix}$$

thus, raising the input signal and output signal of LPTV system gives the polyphase components of the respective signals.

Applying raising technique to the integrator section shown in Fig. 4.11 for $L_0 = 0$, which is a LTI system. Hence following section 3.3 of raising procedure for LTI system we can get the state-space equation as:

$$\begin{aligned} \underline{z}(n+1) &= A^L \cdot \underline{z}(n) + A^{L-1} \cdot B \cdot \underline{w}(n) \\ \underline{y}(n) &= C^L \cdot \underline{z}(n) + D^L \cdot \underline{w}(n) \end{aligned} \tag{4.24}$$

where, the raised system matrices for $L = 13$ by refereing Eqs.(3.10),(3.11),(3.3.1)and (3.3.1) are expressed as

$$A^L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 13 & 1 & 0 & 0 & 0 & 0 \\ 91 & 13 & 1 & 0 & 0 & 0 \\ 455 & 91 & 13 & 1 & 0 & 0 \\ 1820 & 455 & 91 & 13 & 1 & 0 \\ 6188 & 1820 & 455 & 91 & 13 & 1 \end{bmatrix}$$

$$B^L = \begin{bmatrix} 1 \\ 13 \\ 91 \\ 455 \\ 1820 \\ 6188 \end{bmatrix}$$

$$C^L = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$D^L = \begin{bmatrix} 1 \end{bmatrix}$$

It should be noted here that y(n) is the block output signal of the integrator section before the downsampler.

Step 3: Apply block filtering to the raised expander and integrator section. Utilizing the block filtering method, the upsampling and downsampling matrices of the 'L' raised

system can be expressed as :

$$H_{(L,1)}^{[up]} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

$$H_{(L,1)}^{[up]} = I(L) \tag{4.25}$$

This will lead to change 4.24 as:

$$\underline{z}(n+1) = A^L \cdot \underline{z}(n) + B \cdot \underline{w}(n)$$

$$\underline{y}(n) = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{L-1} \end{bmatrix} \cdot \underline{z}(n) + \begin{bmatrix} 0 \\ vdots \\ 0 \\ D \end{bmatrix} \cdot \underline{w}(n) \tag{4.26}$$

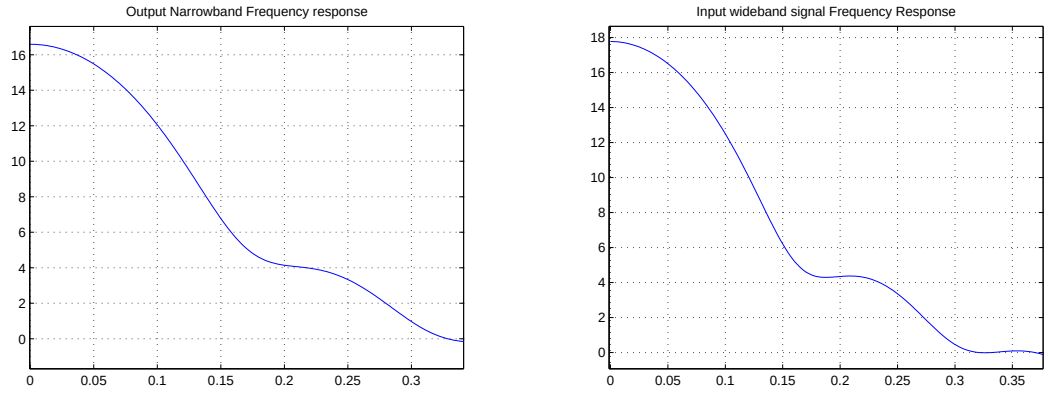
Step 4: Downsample the output of the L-raised integrator section. With

$$A^L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ c_1 & 1 & 0 & 0 & 0 \\ c_2 & c_1 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_{R-1} & c_{R-2} & c_{R-3} & \dots & 1 \end{bmatrix}$$

where,

$$c_1 = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{L-1} \end{bmatrix} \tag{4.27}$$

The respective MATLAB simulated filter response are shown below The structure of Time-Varying CIC filter is given in Fig. 4.13



(a) (b)

Figure 4.12: Third order Time Varying CIC filter response.

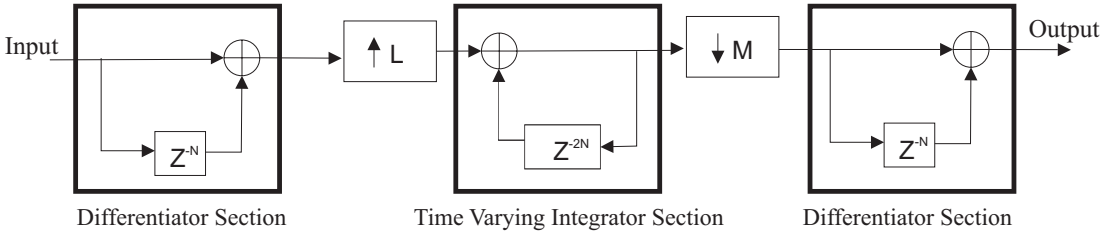


Figure 4.13: Third-order time-varying CIC filter.

5

Proposed Structure for Fractional SRC Systems

This chapter presents an efficient method of design filters for fractional sampling rate conversion (SRC) system based on IFIR filter along with rounding and sharpening techniques. This method designs a less complex multiplierless fractional decimation filter and can be applicable both at high sampling rate and at low sampling rate.

5.1. Introduction

A fractional SRC system is a combination of interpolation and decimation systems. It has an upsampling factor L, a FIR filter $H(z)$ which can do anti-aliasing as well as anti-imaging, and a downsampling factor M. For decimation, the downsampling factor M is greater than the upsampling factor L. A general block representation of fractional SRC system is drawn in Fig. 5.1.

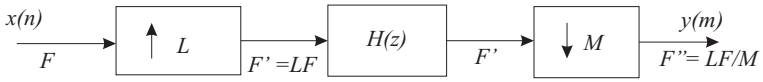


Figure 5.1: Fractional Factor Sampling Rate Conversion.

For high downsampling factor M, for example in case of GSM communication standard, the upsampling factor $L = 13$ and downsampling factor $M = 3840$, it is necessary to split the downsampling factor M into smaller factors

$$M = M_0 \cdot M_1 \cdot M_2 \tag{5.1}$$

where,

$$\begin{aligned} M_0 &= 256 \\ M_1 &= 5\cdot \\ M_2 &= 3 \end{aligned} \tag{5.2}$$

Thus, the conversion factor can be an integer as well as fractional

$$\begin{aligned} Integer \quad Factor &= \frac{1}{256} \\ . \quad Fractional \quad Factor &= \frac{13}{5\cdot 3}. \end{aligned} \tag{5.3}$$

These conversion factors can be arrange in two ways in a SRC system. The first way is to place the integer factor before the fractional factor. And the second way is to place the fractional factor before the integer factor. Both these structures are shown in Fig. 5.2 and Fig. 5.3

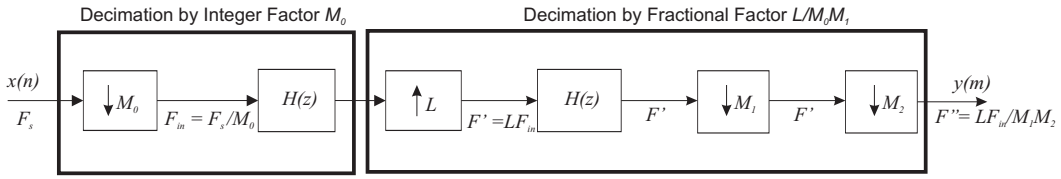


Figure 5.2: Low Sampling Rate Structure.

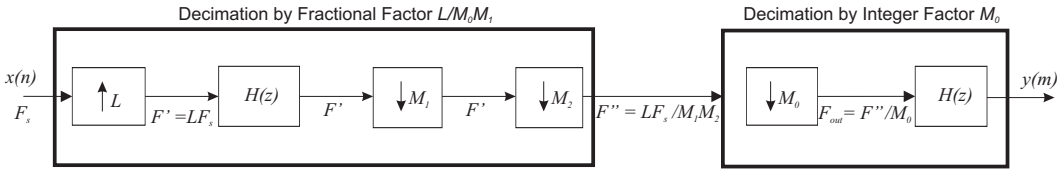


Figure 5.3: High Sampling Rate Structure.

In both these structures, the proposed method is applied only for fractional factor decimation. Integer factor decimation has not been considered.

5.2. Description of Proposed Algorithm

The advantages of IFIR filter as discussed in Chapter 3 are the present demands of communication applications. Considering this, the IFIR filter has been utilized for a

fractional sampling rate conversion and the results are satisfying the current requirements. The proposed algorithm is the implementation of all the techniques to make an efficient FIR filter discussed in Chapter 3 on the IFIR filter. In case of fractional sampling rate conversion as discussed before, the upsampling factor is followed by an anti-imaging-anti-aliasing filter and finally a downsampler factor is redrawn in Fig. 5.4

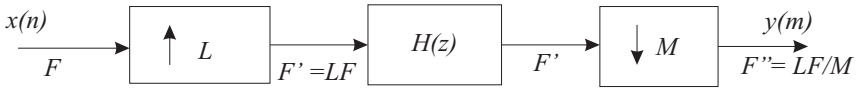


Figure 5.4: Fractional Sampling Rate Conversion.

In this figure, we can notice that the filter $H(z)$ works at high sampling rate which as a consequence increases the power consumption. Therefore, it is important to decrease the complexity of this filter. In the following, we propose a simple method to decrease the complexity of fractional SRC system. We replace the filter $H(z)$ by an IFIR filter and split the decimation factor $M = M_1.M_2$ as shown in Fig. 5.5.

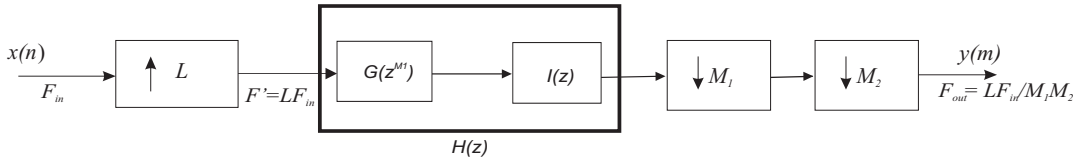


Figure 5.5: Fractional Factor Sampling Rate Conversion using IFIR filter.

Using multirate identity, we get the structure given in Fig. 5.6

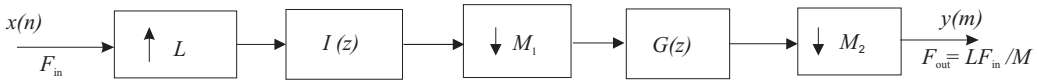


Figure 5.6: More efficient structure .

This structure can be further simplified by using the polyphase representation of the filter $I(z)$ leading to the structure shown in Fig. 5.7

Next issue is to eliminate the multipliers. To this end we apply the rounding technique to the interpolation and the model filter. The resulting filter is a multiplierfree filter because the integer multiplications can be realized by only additions and shifts. Unfortunately,

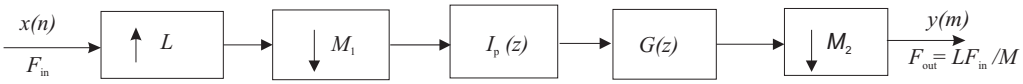


Figure 5.7: Polyphase Interpolator based structure.

this filter has an distorted magnitude response. In order to improve its magnitude response and finally satisfy the desired specification we apply the sharpening technique. The resulting structure is presented in Fig. 5.8, where index r means rounding and $Sh\{.\}$ means sharpening.

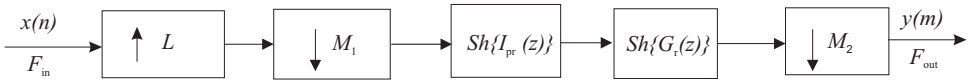


Figure 5.8: Polyphase Interpolator based structure.

The method is illustrated in the following example.

Example 5-2.

Design an IFIR fractional factor decimation filter with Interpolation factor L is 13 and decimation factor M is 15. The normalized passband frequency is 0.0025 with passband ripples 0.001. The stopband frequency is set to 0.07 with minimum stopband attenuation of 110 dB. The rounding factor r is 2^{-8}.

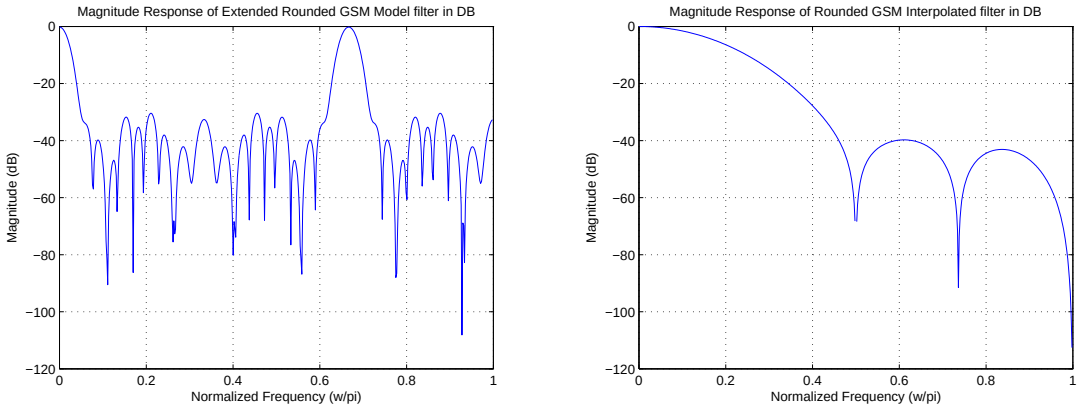
To design the IFIR filter we calculate the specifications of model and interpolator filter for decimation factor M splitted to $M_1 = 5$ and $M_2 = 3$ and the model filter is expanded by M_1 .

Table 5.1: Specifications of model and interpolator filters.

Specifications M	Model filter G(z)	Interpolator filter I(z)
Stopband Frequency, ω_s	$M_1 \times \omega_s$	$2\pi/M_1$ - ω_s
Passband Frequency, ω_p	$M_1 \times \omega_p$	ω_p
Passband Ripples, R_p	$R_p/2$ dBs	$R_p/2$ dBs
Stopband Attenuation, A_s	A_s dBs	A_s dBs

Using calculated specification based on Table 5.1, we design model and interpolator filters

by Parks-McClellan algorithm . Then, expand the model filter by 5. Further, on applying rounding to the expanded model and interpolator filters the magnitude responses shown in Fig. 5.9(a) and Fig. 5.9(b) are obtained



(a) (b)

Figure 5.9: Rounded Expanded Model and Interpolator Filters Magnitude response.

Now, cascade the rounded and sharpened extended model filter and interpolator filter according to the sharpening polynomial. The consequent magnitude responses is shown in Fig. 5.2

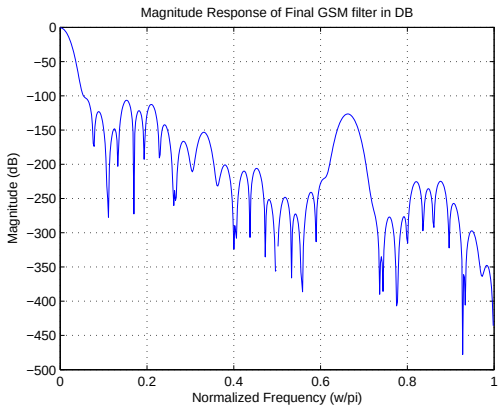


Figure 5.10: Final Magnitude Response.

The sharpening polynomial chosen for obtaining the desired filter response are

$$\begin{aligned} Sh\{G_r(z)\} &= G_r(z)^3, \\ Sh\{I_r(z)\} &= I_r(z)^3. \end{aligned} \tag{5.4}$$

Table 5.2 shows the resource required by the the proposed structure From Table 5.2, it

Table 5.2: IFIR Filter required resources.

Resources	Model Filter $G(z)$	Interpolator Filter $I(z)$	Total
Adders	26	10	36
Filter order	12	51	63

can be concluded that the IFIR filter reduces the filter order and number of adders with no multipliers as multipliers can be applied by add and shift method. Thus, the proposed structure has low complexity and hence less power consumption.

5.3. Application of proposed method at low sampling rate

Using the low sampling rate structure shown in Fig. 5.2, we can place the proposed structure for fractional decimation system. Fig. 5.3 shows the placement of proposed structure at low sampling rate.

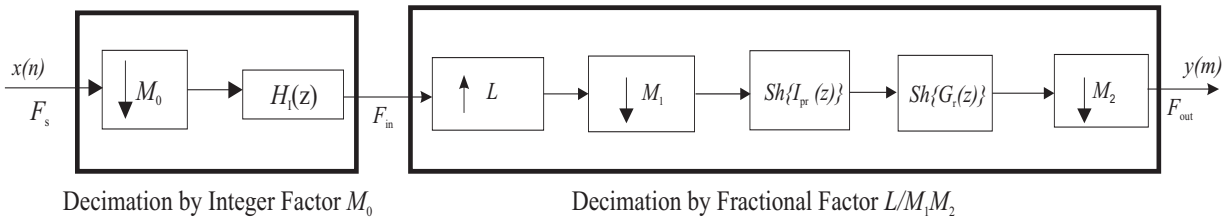


Figure 5.11: Proposed fractional decimation system at low sampling rate.

The next example will demonstrate the fractional decimation system after integer factor decimation for GSM communication standard.

Example 5-3.

This example designs a fractional decimation filter of factor 13/15 based on proposed algorithm at low sampling rate having passband ripple of 0.001 dB and stopband attenuation of 110 dBs. The selected rounding factor r is 2^{-12}.

Considering the specification of GSM standard, the integer factor decimation is 256, if the sampling frequency is 80 MHz, then the input frequency of the fractional decimation system will be 80/ 256 = 3125 KHz. Based on this input frequency, to design the filter decimation factor M is splitted to $M_1 = 5$ and $M_2 = 3$ and model filter is expanded by M_1.

Table 5.3: Desired Filter *H(z)* Specification.

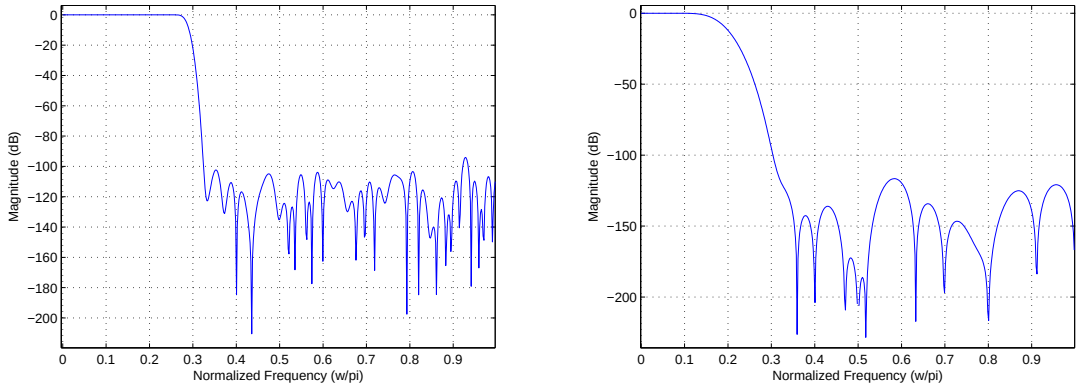
Normalized Passband Frequency, ω_p	0.0492
Normalized Stopband Frequency, ω_s	0.0666
Passband Ripples, R_p	0.001 dB
Stopband Attenuations, A_s	110 dBs

Replacing the above FIR filter by IFIR filter, the respective calculated specification of the model and interpolator filter is shown in Table 5.4 The corresponding simulation

Table 5.4: Specifications of $G(z)$ and I(z).

Specifications	$G(z)$	I(z)
Passband Frequency	0.2460	0.0492
Stopband Frequency	0.3333	0.3333
Passband Ripples	0.0005 dB	0.0005 dB
Stopband Attenuations	110 dBs	110 dBs
Rounding Factor	2^{-12}	2^{-12}
Sharpening Polynomial	$3 \cdot G(z)^2 - 2 \cdot G(z)^3$	$3 \cdot I(z)^2 - 2 \cdot I(z)^3$

in MATLAB with these specification and applying rounding and sharpening are shown in Fig. 5.12 and Fig. 5.13.



(a) (b)

Figure 5.12: Rounded and Sharpened Model and Interpolator Filters.

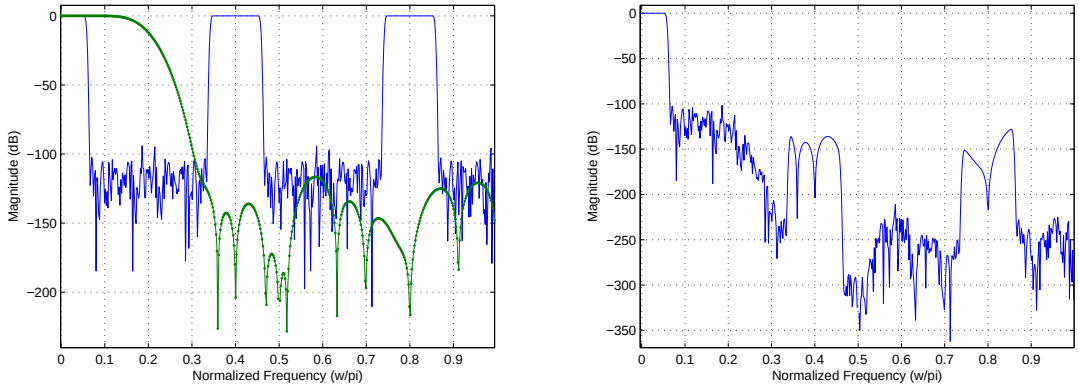


Figure 5.13: Cascaded and Final Magnitude response at Low sampling rate.

The sharpening polynomials for the model and interpolator filters are respectively

$$\begin{aligned} Sh\{G_r(z)\} &= 3 \cdot G(z)^2 - 2 \cdot G(z)^3, \\ Sh\{I_r(z)\} &= 3 \cdot I(z)^2 - 2 \cdot I(z)^3. \end{aligned} \tag{5.5}$$

5.4. Application of proposed method at high sampling rate

The proposed algorithm can also be placed at high sampling rate by replacing the fractional decimation system of Fig. 5.3 with the proposed efficient structure of Fig. 5.8. The new structure we get is shown as

The next example describes the proposed structure at high sampling rate for GSM com-

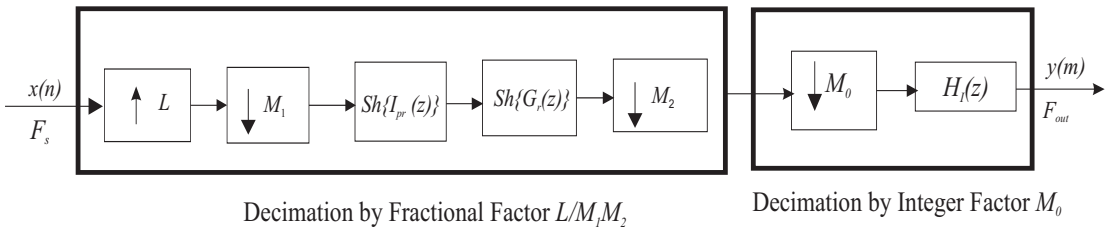


Figure 5.14: Fractional SRC System at high sampling rate.

munication standards.

Example 5-4.

This example designs a fractional decimation filter of factor 13/15 based on proposed algorithm at high sampling rate having passband ripple of 0.001 dB and stopband attenuation of 110 dB. The selected rounding factor r is 2^{-12}

In this case the fractional system has to work at high sampling rate of 13 times of sampling frequency 80 MHz. Then, the derived cutoff frequencies for the factorized value of decimation factor M to $M_1 = 5$ and $M_2 = 3$ and model filter is expanded by M_1. will be

Table 5.5: Desired Filter *H(z)* Specification.

Normalized Passband Frequency, ω_p	0.00019
Normalized Stopband Frequency, ω_s	0.0666
Passband Ripples, R_p	0.001 dB
Stopband Attenuations, A_s	100 dBs

On Replacing the filter H(z) with IFIR filter, the specification of the calculated model and interpolation filter are as follows.

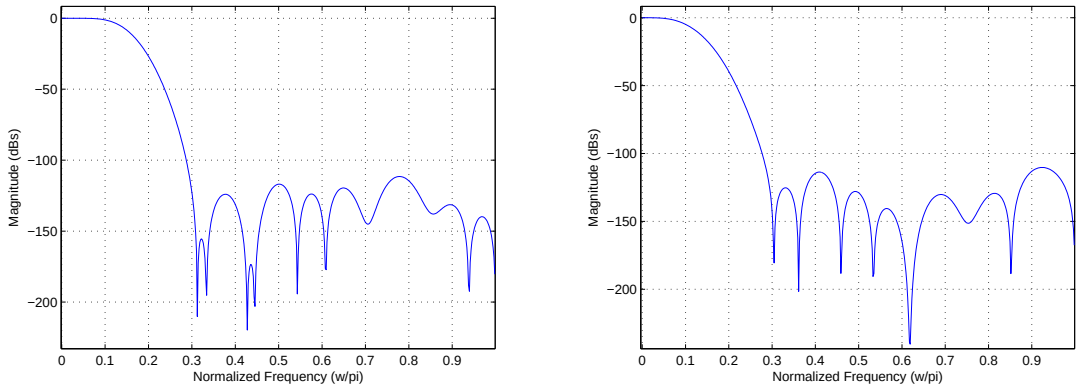
The sharpening polynomials for the model and interpolator filters are

$$\begin{aligned} Sh\{G_r(z)\} &= 3 \cdot G(z)^2 - 2 \cdot G(z)^3, \\ Sh\{I_r(z)\} &= 3 \cdot I(z)^2 - 2 \cdot I(z)^3. \end{aligned} \tag{5.6}$$

The corresponding simulation in MATLAB with these specification and applying rounding and sharpening are shown in Fig. 5.15 and Fig. 5.16.

Table 5.6: Specifications of $G(z)$ and I(z).

Specifications	Model Filter *G(z)*	Interpolator Filter *I(z)*
Passband Frequency, ω_p	0.00095	0.00019
Stopband Frequency, ω_s	0.3333	0.3333
Passband Ripples, *Rp*	0.0005	0.0005
Stopband Attenuations	100 dBs	100 dBs
Rounding Factor	2^{-12}	2^{-12}
Sharpening Polynomial	$3 \cdot G(z)^2 - 2 \cdot G(z)^3$	$3 \cdot I(z)^2 - 2 \cdot I(z)^3$
Decimation Factor	3	5
Polyphase Length	93	17
Filter Length	277	83

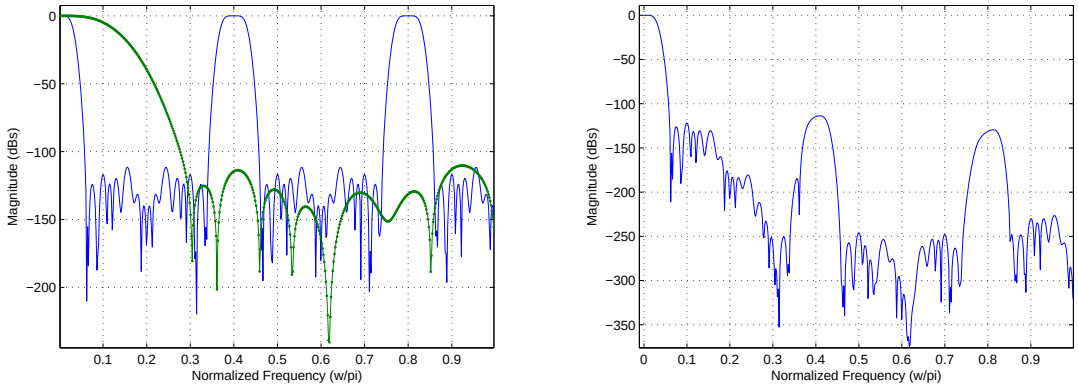


(a) (b)

Figure 5.15: Rounded and Sharpened Model and Interpolator Filter.

5.5. Discussion of the results

This section provides the analysis of proposed method applied at low sampling rate and at high sampling rate. The implementation cost of the proposed structure at low sampling rate of 312.5 kHz are shown in Table 5.7. The interpolator filter has length 83 with five polyphase component each of length 17. And the Model filter is of 277 length with three polyphase component each of length 93.



(a) (b)

Figure 5.16: Cascaded and Final Magnitude response at high sampling rate.

Table 5.7: Implementation cost of proposed structure at low sampling rate.

Specifications	*G(z)*	*I(z)*	Cascaded
Number of Adders	251	74	325
Number of states	92	16	108
Multiplication rate per Input Sample	84	15	31.8
Addition rate per Input Sample	83.66	14.8	31.533

Similarly, Table , Table **??**, and Table 5.8 shows the implementation cost of the proposed structure when it is at high input frequency of 1040 MHz. In this case the interpolator and model filter are of equal length 67 with five and three polyphase components, each of length 14 and 23, respectively. The comparison of proposed structure at low and high sampling rate with respect to above discussed parameters are presented in Table 5.9

From Table 5.9, it can be seen that the number of adders gets decreased from 325 to 122. From this, it can be concluded that the proposed structure gives less filter complexity , when applied at high sampling rate.

On carefully examining Table 5.9, it is found that the reason for decreasing the filter complexity at high sampling rate is the widening of transition band of model filter and interpolator filter. Hence, following the conclusion made in Chapter 2 regarding fractional

Table 5.8: Implementation cost of proposed structure at high sampling rate.

Specifications	*G(z)*	*I(z)*	Cascaded
Number of Adders	62	60	122
Number of states	22	13	35
Multiplication rate per Input Sample	21	12.2	16.4
Addition rate per Input Sample	20.667	12	16.133

Table 5.9: Comparison of Proposed Structure at low and high sampling rate.

Specifications	*H(z)* low F_s	*H(z)* at high F_s
Passband Frequency, ω_p	0.0492	0.00019
Stopband Frequency, ω_s	0.0666	0.0666
Transition Band, TB	0.0174	0.0664
Passband Ripples, *Rp*	0.001	0.001
Stopband Attenuations	110 dBs	100 dBs
Rounding Factor	2^{-12}	2^{-12}
Sharpening Polynomial	$3 \cdot G(z)^2 - 2 \cdot G(z)^3$	$3 \cdot I(z)^2 - 2 \cdot I(z)^3$
Number of Adders	325	122
Number of states	108	35
Multiplication rate per Input Sample	31.8	16.4
Addition rate per Input Sample	31.5333	16.1333

decimation filter placement before or after the integer factor decimation filter; it is justified, that the filter complexity decreases at high sampling rate , while the filter has to work at high multiplication rate. However, the utilization of the multirate identity and the polyphase representation help to move the filters at low sampling rate region. Thus, the complete structure works at intermediate sampling rate.

Hence, it can be said that both structures are efficient with respect to the desired requirement. If resource requirement is to be concern than it is recommendable to place the structure at high sampling rate. But in case of multiplication rate to be a matter of importance, then the proposed structure should be placed at the low sampling rate.

6

Implementation of the Proposed Structure

This chapter presents the implementation of the proposed algorithm in SPARTAN 3 family of Field Programmable Gate Array (FPGA). An introduction of the SPARTAN 3 is provided. Among the various computer arithmetic algorithms for Multiplication, Add and Shift algorithm have been selected to implement the rounded integer coefficients of the proposed algorithm. The implementation design of the proposed structure at low sampling rate and at high sampling rate is done in the later part of the chapter.

6.1. Introduction

The Spartan 3 family of FPGA is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. The eight member family offers densities ranging from 50,000 to five million system gates, as shown in Table 1. The Spartan 3 family builds on the success of the earlier Spartan IIE family by increasing the amount of logic resources, the capacity of internal RAM, the total number of I/Os, and the overall level of performance as well as by improving clock management functions. Numerous enhancements derive from the Virtex II platform technology. These Spartan 3 enhancements, combined with advanced process technology, deliver more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry. Because of their exceptionally low cost, Spartan 3 FPGAs are ideally suited to a wide range of consumer electronics applications, including broad-

band access, home networking, display/projection and digital television equipment. The Spartan 3 family is a superior alternative to mask programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs.

Table 1: **Summary of Spartan-3 FPGA Attributes**

Device	System Gates	Equivalent Logic Cells[1]	CLB Array (One CLB = Four Slices)			Distributed RAM Bits (K=1024)	Block RAM Bits (K=1024)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50[2]	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200[2]	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400[2]	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000[2]	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	712	312
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	784	344

Notes:
1. Logic Cell = 4-input Look-Up Table (LUT) plus a 'D' flip-flop. "Equivalent Logic Cells" equals "Total CLBs" x 8 Logic Cells/CLB x 1.125 effectiveness.
2. These devices are available in Xilinx Automotive versions as described in DS314: Spartan-3 Automotive XA FPGA Family.

6.2. Fixed Point Multiplication

A multiplication circuit is usually referred to as a multiplier [7]. A multiplier takes two binary operands to generate a product. A binary number can represent either a fixed point or a floating point unsigned number or a 2's complement number. To implement the multiplication of these numbers in hardware, fixed point multiplication algorithms has been described.

The multiplication algorithm for binary numbers is basically similar to the traditional pencil and paper method. The traditional pencil paper method for two digit binary numbers involves the formation of first partial product by multiplying least significant digit of the multiplier, with the multiplicand. The second partial product is obtained by multiplying the second least significant digit of the multiplier with the multiplicand. These two partial products are then added together to form the product. In digital designs, binary numbers are usually used. For a four bit unsigned binary representation of multiplicand

and multiplier, there are four partial products. the multiplication of two binary bits is simply a logical AND function. There is no carry necessary in getting the binary partial product. This simplifies the hardware. The partial products are either the same as the multiplicand or zero, and they are shifted left to the corresponding bit position of the multiplier. These four partial products are then added together to form the decimal equivalent product.

Various algorithms for replacing the multipliers by shift registers and adders has been proposed. In the next section two among the available fixed point multiplication algorithms has been described.

6.2.1. Unsigned Multiplication

Add and shift algorithm is the method of performing unsigned multiplication, which has unsigned input data. It involves the adders and shift registers to perform the multiplication operation. The data can be either fixed point or floating point. We are working on fixed point data.

Considering Partial products obtained by AND operation of multiplier and multiplicand digit, the next straightforward approach of adding these partial products is to use three adders. A different concept of using Carry Save Adders (CSA) to save carry for the next stage can also be applicable [17]. The CSA has two outputs, one is the sum bit and the other is the carry bit. In order to add these partial products,a simple shift and add algorithm is shown in Fig. 6.1.

In this algorithm, the multiplicand, ALU register and multiplier are all 32-bits wide, with only the product register left at 64-bits. Then the product is shifted right. The hardware interpretation of Add and Shift algorithm is shown in Fig. 6.2.

The explanation of the algorithm is given in the next example.

Example 6-1.

Multiply 0010 with 0011 using Add and Shift algorithm in Fig.6.1.

Four partial products are formed by multiplying the multiplicand with each multiplier bits. These four partial product is are then added by shifting each of them. The complete

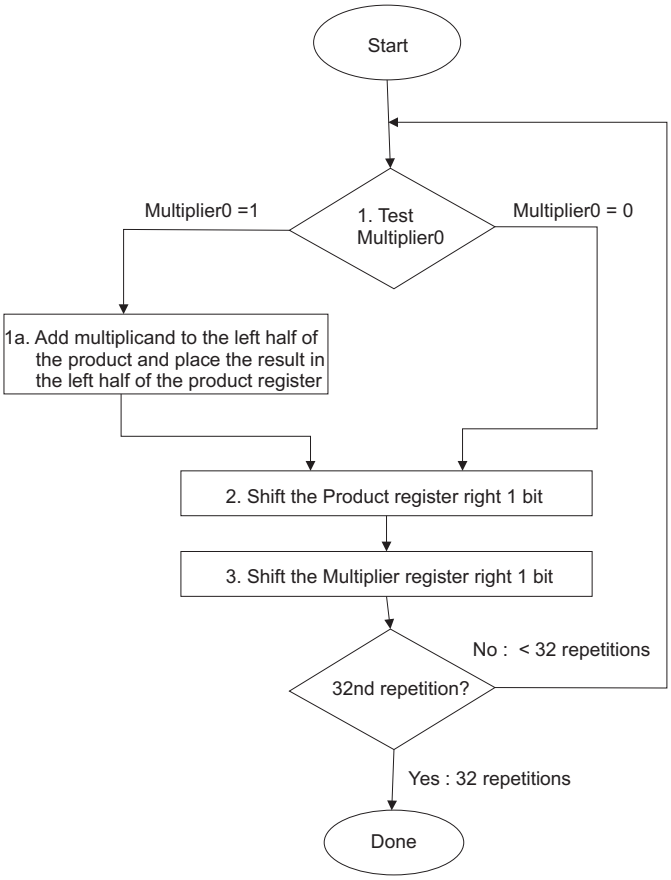


Figure 6.1: Shift and Add Algorithm for unsigned Multiplication.

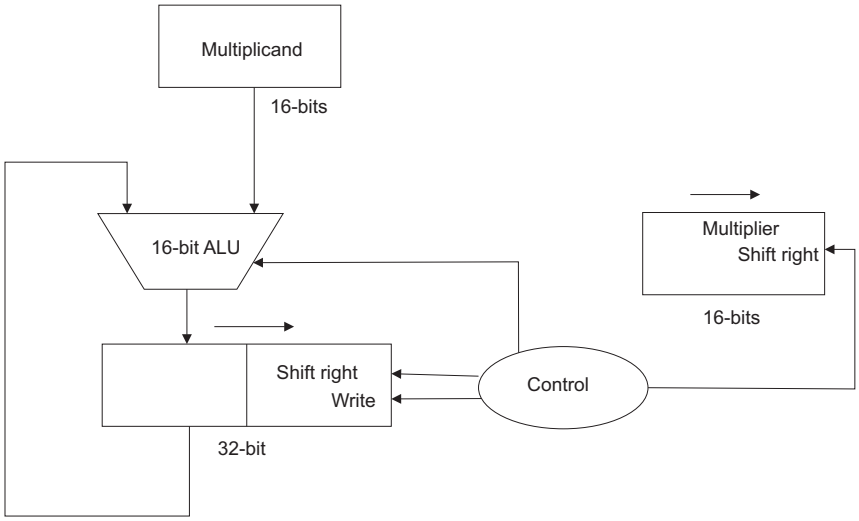


Figure 6.2: Shift and Add Multiplication Hardware for unsigned Multiplication.

process is shown in Table 6.1.

Table 6.1: Parameters for SRC.

Iteration	Step	Multiplicand	Product
0	Initial Values	0010	0000 001**1**
1	1a: 1 =>Prod=Prod+Mcand	0010	0010 0011
1	2: Shift right Product	0010	0001 000**1**
2	1a: 1 =>Prod=Prod+Mcand	0010	0001 000**1**
2	2: Shift right Product	0010	0011 100**0**
3	1: 0 =>no operation	0010	0001 **1**000
3	2: Shift right Product	0010	0000 110**0**
4	1: 0 =>no operation	0010	0000 110**0**
4	2: Shift right Product	0010	0000 0110

In this algorithm the right most half of the product register is utilized to assign multipliers placing zero at the upper half.

6.2.2. Signed Multiplication

The simplest way to convert the add and shift algorithm to signed numbers is to first convert the multiplier and multiplicand to positive numbers and then remember the original signs. Later, a more elegant approach to multiply signed numbers is introduced by Booth in 1952, and is called Booth's Algorithm [7]. Booth suggested a recoding scheme to reduce the number of partial products for the signed numbers. Booth invented this approach in a quest for speed, believing that shifting was faster than addition. Indeed, for some patterns his algorithm would be faster, and handle signed numbers. The key to Booth's algorithm is in his classifying groups of bits into the beginning, the middle, or the end of a run of 1s [23]. For the 2 bits recoding pattern, the partial products changes according to the Table 3 [23].

Booth's algorithm changes the first step of Add and Shift algorithm, where the multiplicand is added by shifting when the multiplier bit is one. In case of Booth's algorithm, two

Table 6.2: Parameters for SRC.

Current bit	Bit to the right	Explanation	Example
1	0	Beginning of a run of 1s	0000111**10**00
1	1	Middle of a run of 1s	00001**11**000
0	1	End of a run of 1s	000**01**111000
0	0	Middle of a run of 0s	0**00**01111000

or three bits of recoded bits are checked to decide whether the multiplicand has to added or subtracted. Booth's algorithm can be demonstrated in the following points [23]

- Depending on the current and previous bits, do one of the following steps

  00 : Middle of a string of 0s, so no arithmetic operation.

  01 : End of a string of 1s, so add the multiplicand to the left half of the product.

  10 : Beginning of a string of 1s, so subtract the multiplicand from the left half of the product.

  11 : Middle of a string of 1s, so no arithmetic operation.

- As the previous algorithm, shift the product register right by 1 bit.

The main requirement of Booth's algorithm is to preserve the sign of intermediate result, in case of dealing with signed numbers. A more clear explanation of Booth's algorithm can be obtain from the following example [23].

Example 6-2.

Multiply 2 and -3 to get -6, or 0010 by 1101 = 1111 1010.

Table 6.3 shows the steps of Booth's algorithm.

Booth's observation about replacing arithmetic by shifts can be applied when multiplying by constants. Some compilers replace multiplications by short constants with a series of shifts,adds, and subtracts. Because 1 bit to the left represents a number twice as large as in base 2, shifting the bits left has the same effect as multiplying by a power of 2, so almost every compiler will substitute a left shift for a multiplication by a constant that is a power of 2.

Table 6.3: Steps of Booth's Algorithm.

Iteration	Step	Multiplicand	Product
0	Initial Values	0010	0000 1101 **0**
1	1c: 10 =>Prod=Prod-Mcand	0010	1110 1101 0
1	2: Shift right Product	0010	1111 0110 **1**
2	1b: 01 =>Prod=Prod+Mcand	0010	0001 0110 **1**
2	2: Shift right Product	0010	1111 1011 **0**
3	1c: 10 =>Prod=Prod-Mcand	0010	1110 1011 0
3	2: Shift right Product	0010	1111 0101 **1**
4	1: 0 =>no operation	0010	1111 0101 1
4	2: Shift right Product	0010	1111 1010 1

6.3. Proposed Algorithm at low sampling rate

Proposed algorithm at low sampling rate as given in Chapter 5 is redrawn in Fig.6.3

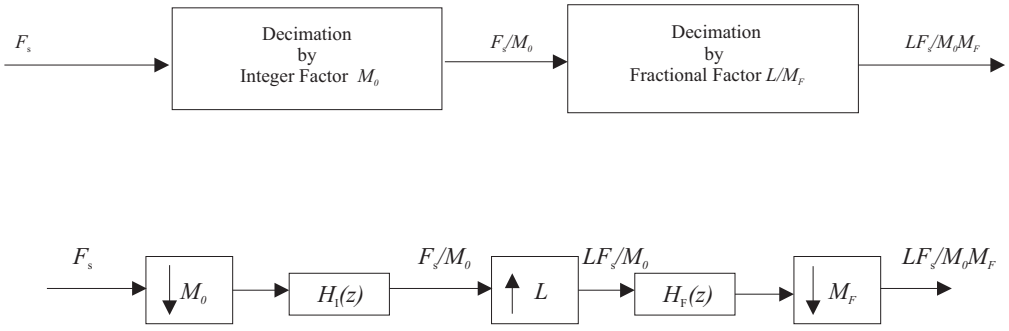


Figure 6.3: Fractional SRC System at low sampling rate.

The MATLAB has been used to generate the VHDL program of the proposed structure at low sampling rate. The specifications given in Table **??** of Chapter 5 has been used to generate the VHDL program. The VHDL program allocates 16bits for the input signal and the filters coefficients, while the output signal has been represented by 34 bits variables. The multiplication operation has been done by the available multipliers of the FPGA, as well as by the optimization of the filters.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	1,888	26,624	7%	
Number of 4 input LUTs	11,722	26,624	44%	
Logic Distribution				
Number of occupied Slices	6,536	13,312	49%	
Number of Slices containing only related logic	6,536	6,536	100%	
Number of Slices containing unrelated logic	0	6,536	0%	
Total Number 4 input LUTs	12,271	26,624	46%	
Number used as logic	11,722			
Number used as a route-thru	549			
Number of bonded IOBs	53	333	15%	
IOB Flip Flops	66			
Number of MULT18X18s	32	32	100%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	279,231			
Additional JTAG gate count for IOBs	2,544			

Figure 6.4: Device Utilization Summary for Proposed Structure at low sampling rate

6.3.1. Implementation

The design summary for the implemented proposed structure at low sampling rate is given in Fig. 6.4 as generated by Xilinx ISE 8.2i for xc3s1500-5fg456. The corresponding RTL diagram of the implemented structure is shown in Fig. 6.5

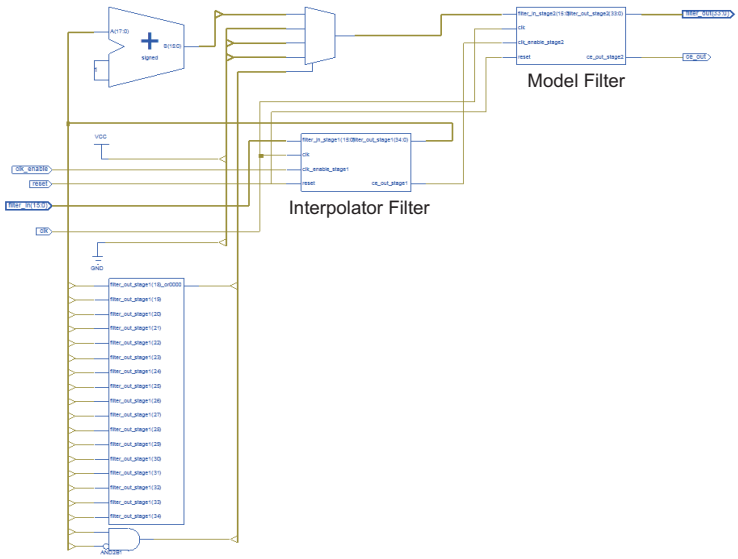


Figure 6.5: RTL diagram of Proposed Structure at low sampling rate

The more clear picture of Interpolator Filter RTL diagram is shown in Fig. 6.6

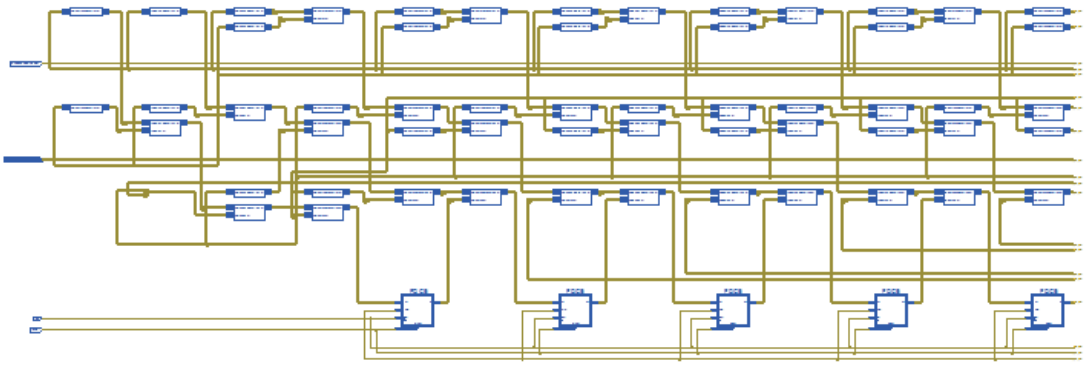


Figure 6.6: RTL diagram of Interpolator Filter

After replacing the multiplier instance created by Xilinx ISE 8.2i with add and shift registers, we get the new RTL diagram shown in Fig. 6.7

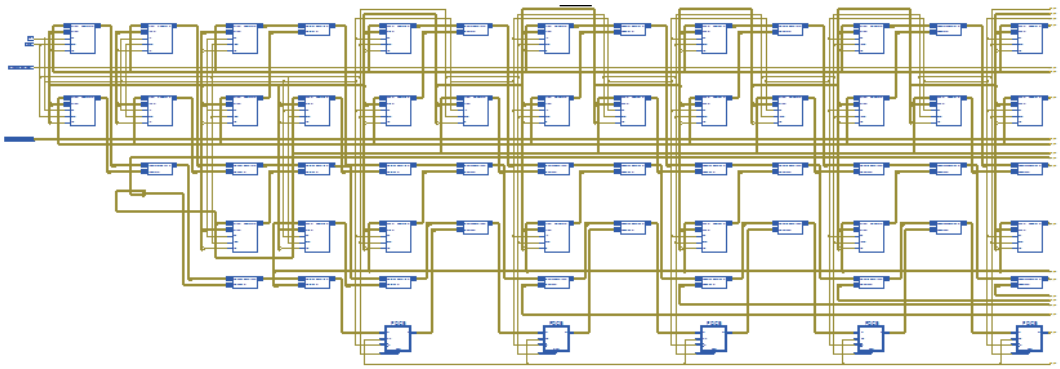


Figure 6.7: RTL diagram of modified Interpolator Filter

The number of equivalent gates used by the proposed structure is 279,231. This can be reduced by replacing the multipliers of the model and interpolator filter of proposed structure with Add and Shift algorithm of multiplication. Consequently, the device utilization gets decreased as given in Fig. 6.17. The new utilization of FPGA is 37,586 equivalent gates.

The waveforms for the input and output are obtained by simulating the proposed structure as shown in Fig. 6.9. For this simulation Symphony EDA Sonata 3.1 has been used. This waveform shows that the output starts changing from 2.7ns.

A difference in the input and output waveforms exist while replacing the multipliers by

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	3,297	26,624	12%	
Number of 4 input LUTs	818	26,624	3%	
Logic Distribution				
Number of occupied Slices	2,016	13,312	15%	
Number of Slices containing only related logic	2,016	2,016	100%	
Number of Slices containing unrelated logic	0	2,016	0%	
Total Number 4 input LUTs	1,141	26,624	4%	
Number used as logic	818			
Number used as a route-thru	323			
Number of bonded IOBs	55	333	16%	
IOB Flip Flops	52			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	37,586			
Additional JTAG gate count for IOBs	2,640			

Figure 6.8: Device Utilization Summary after replacing Multipliers by Add and Shift algorithm

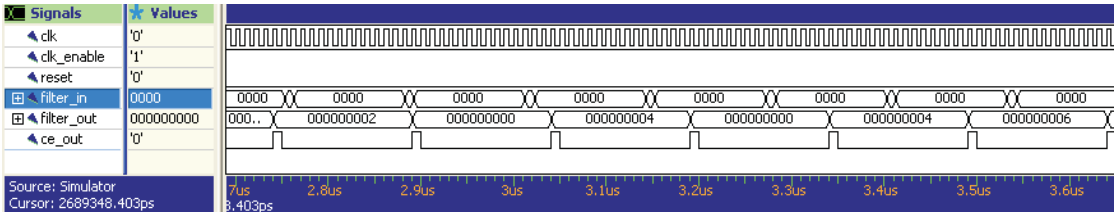


Figure 6.9: Input Output Waveform of Proposed Structure at low sampling rate

add and shift algorithm. Since, add and shift algorithm is an unsigned multiplication algorithm, for obtaining the multiplication of signed data, a recoding of the VHDL program has to be done in order to get the same output. The another solution to replace the multipliers is to perform multiplication using Booth's Wallace Tree algorithm of multiplication. The Unique advantage of Booth Wallace algorithm is that it is fast method of multiplying the signed data with no recoding of the main program. However, Booth Wallace Tree algorithm increases the device utilization and hence the chip area.

From Fig.6.4 and Fig. 6.17 it has been found that on replacing the multipliers of the model and interpolated filter by fixed point of add and shift algorithm of multiplication, the number of equivalent FPGA gates decreased by 7.43 times with no multipliers.

6.4. Proposed Algorithm at high sampling rate

Proposed algorithm at high sampling rate as given in Chapter 5 is redrawn in Fig.6.10

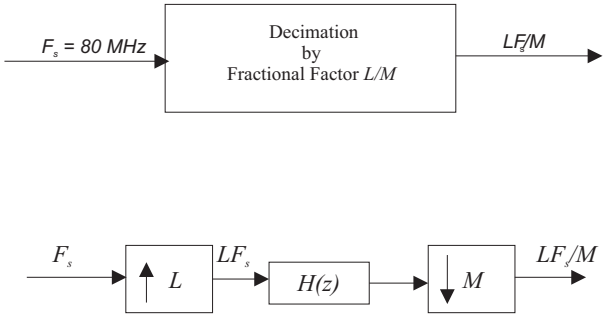


Figure 6.10: Fractional Sampling Rate Conversion system at high sampling rate

The VHDL code generation of the proposed algorithm has been done using MATLAB instructions like *generatehdl* and *generatetb* based on the filter design specifications given in Table **??**. The quantization of the input and filter coefficient are done by using *set* instruction which quantize the data for full precision fixed point arithmetic. The consequent input data and filter coefficient variables are of 16 bits in VHDL, where as the filter output is of 34 bits. The multiplication performed by MATLAB generate code is of fixed point and utilizes the multipliers available in SPARTAN 3 family.

6.4.1. Implementation

The above given structure is implemented in FPGA by using the MATLAB generated VHDL code based on the specification given in Table 5.6 of Chapter 5. The VHDL program involves the formation of Entity for model and interpolator filter. Testbench has been generated based on the sinusoidal input. The multiplication operation between the input and filter coefficients has been performed by the simple multiply instruction which has perform by the available multipliers in the FPGA. The rest of the multiplication has been managed by the FPGA utilizing the registers and adders.

The design summary for the implemented proposed structure at high sampling rate as

generated by Xilinx ISE 8.2i for xc3s1500-5fg456 SPARTAN 3 FPGA is given below in Fig.6.11.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	678	26,624	2%	
Number of 4 input LUTs	3,996	26,624	15%	
Logic Distribution				
Number of occupied Slices	2,327	13,312	17%	
Number of Slices containing only related logic	2,327	2,327	100%	
Number of Slices containing unrelated logic	0	2,327	0%	
Total Number of 4 input LUTs	4,296	26,624	16%	
Number used as logic	3,996			
Number used as a route-thru	300			
Number of bonded IOBs	54	221	24%	
IOB Flip Flops	51			
Number of MULT18X18s	32	32	100%	
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	182,950			
Additional JTAG gate count for IOBs	2,592			

Figure 6.11: Device Utilization Summary for Proposed Structure at high sampling rate

From this table it can be summarize that the proposed structure is utilizing all the multipliers which are available in the FPGA. Apart from this, other multiplication operation has been implemented in FPGA by the optimization of the filters done by Xilinx ISE 8.2i. The number of equivalent gates utilized is 182,950. The corresponding RTL diagram of the implemented structure is shown in Fig. 6.12. On replacing each of the multipliers in the model and interpolator filter of proposed structure by the add and shift registers, the device utilization gets decreased as given in Fig.6.15

The more clear picture of Model Filter RTL diagram is shown in Fig.6.13 After replacing the multiplier instance created by Xilinx ISE 8.2i with add and shift algorithm, we get the new RTL diagram shown in Fig. 6.14

The number of gates is now 9,186, which is due to the replacement of default multiplication operation of Xilinx ISE 8.2i by the Add and Shift algorithm of multiplication.

The input output waveforms obtain by simulating the proposed structure in Symphony EDA Sonata 3.1 is shown in Fig. 6.16 The waveform at the output represents filter output of 34 bits. It starts generating the initial values equals to zero, with the changes occur

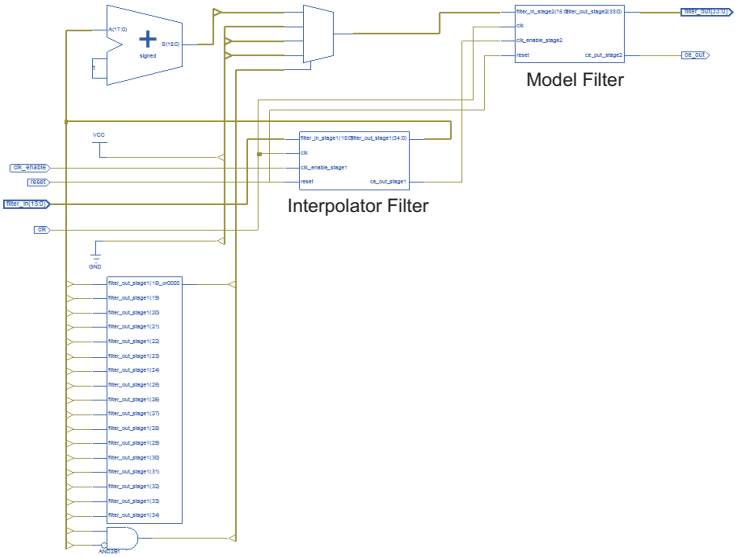


Figure 6.12: RTL diagram of Proposed Structure at high sampling rate

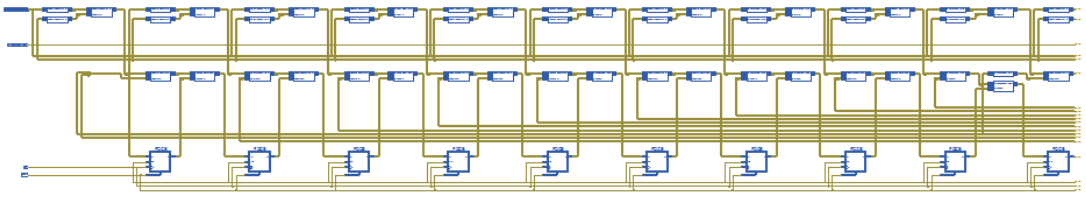


Figure 6.13: RTL diagram of Model Filter

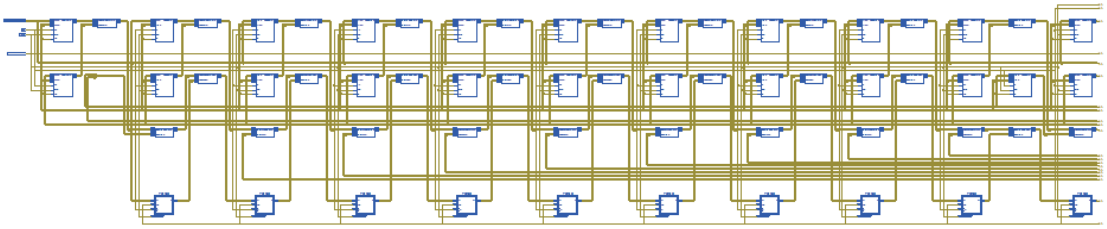


Figure 6.14: RTL diagram of modified Model Filter

from 29.3ns.

From Fig. 6.11 and Fig. 6.15 it has been found that on replacing the multipliers of the model and interpolated filter by fixed point of add and shift algorithm of multiplication, the number of equivalent FPGA gates decreased by 19.9 times with no multipliers.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	713	26,624	2%	
Number of 4 input LUTs	260	26,624	1%	
Logic Distribution				
Number of occupied Slices	484	13,312	3%	
Number of Slices containing only related logic	484	484	100%	
Number of Slices containing unrelated logic	0	484	0%	
Total Number of 4 input LUTs	336	26,624	1%	
Number used as logic	260			
Number used as a route-thru	76			
Number of bonded IOBs	55	221	24%	
IOB Flip Flops	52			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	9,186			
Additional JTAG gate count for IOBs	2,640			

Figure 6.15: Device Utilization Summary after replacing Multipliers by Add and Shift algorithm

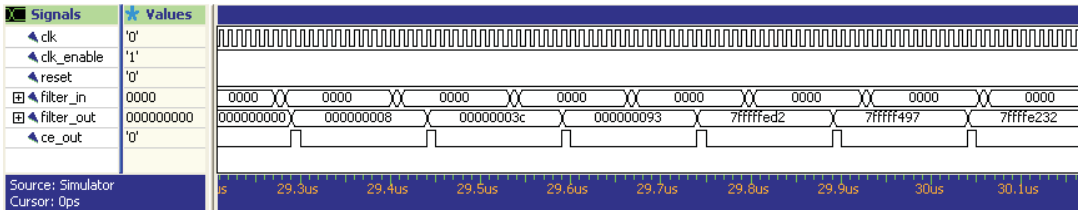


Figure 6.16: Input Output Waveform of Proposed Structure at high sampling rate

6.5. Discussion of the implemented structures

This section discuss the implementation results obtained by low sampling rate structure and high sampling rate structure, which are implemented by replacing filter multipliers with add and shift algorithm.

To analyze, Fig. 6.17 and Fig. 6.18 are again shown. From these, it can be seen that the number of equivalent FPGA gates utilization decreased in case of proposed structure at high sampling rate.

This justifies the conclusion made in Chapter 5 regarding the trade off between input sampling rate and filter complexity. The number of gates in high sampling structure is 3.82 times less than the low sampling rate structure. However, high sampling rate structure

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	3,297	26,624	12%	
Number of 4 input LUTs	818	26,624	3%	
Logic Distribution				
Number of occupied Slices	2,016	13,312	15%	
Number of Slices containing only related logic	2,016	2,016	100%	
Number of Slices containing unrelated logic	0	2,016	0%	
Total Number 4 input LUTs	1,141	26,624	4%	
Number used as logic	818			
Number used as a route-thru	323			
Number of bonded IOBs	55	333	16%	
IOB Flip Flops	52			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	37,586			
Additional JTAG gate count for IOBs	2,640			

Figure 6.17: Device Utilization Summary at low sampling rate structure

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	713	26,624	2%	
Number of 4 input LUTs	260	26,624	1%	
Logic Distribution				
Number of occupied Slices	484	13,312	3%	
Number of Slices containing only related logic	484	484	100%	
Number of Slices containing unrelated logic	0	484	0%	
Total Number of 4 input LUTs	336	26,624	1%	
Number used as logic	260			
Number used as a route-thru	76			
Number of bonded IOBs	55	221	24%	
IOB Flip Flops	52			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	9,186			
Additional JTAG gate count for IOBs	2,640			

Figure 6.18: Device Utilization Summary at high sampling rate structure

has to work at high sampling rate.

During the whole procedure, the low complexity has been achieved successfully, but the desired output is still a promising job. Due to unsigned nature of add and shift algorithm, the change in output occurs when the input changes its sign. One way to get the desired output is apply CORDIC algorithm. In this way a more reliable result can be obtained.

tructure

Conclusions and Future Work

Software Radio terminals process various communication standards simply by converting the sampling rate to desired sampling rate. During sampling rate conversion, the aliasing and imaging effects occur.To eliminate these effects, anti-aliasing and anti-imaging filters are required. Thus, Sampling rate conversion is a filter designing problem. Methods of sampling rate conversion are well known as *Multirate Techniques* which includes *Decimation* and *Interpolation*. The implementation of multirate algorithms based filters requires programmable devices like Programmable Array Logic, Programmable Logic Array, Complex Programmable Logic Devices, and Field Programmable Gate Array.

The process of decreasing the sampling rate is called decimation. Decimation processes are performed by integer factor as well as fractional factor. Fractional factor decimation system is a combination of the interpolation and the decimation process. Software Radio requires decimation filter for both integer factor sampling rate conversion system and fractional factor sampling rate conversion system.

Considering the decimation filter for fractional factor sampling rate conversion system, it has been found that on placing it at high sampling rate, the filter complexity decreases but the filter has to work at high rate, which will cause more power consumption. However, on placing the fractional decimation filter for fractional factor sampling rate conversion system at low sampling rate will increases the filter complexity but decreases the power consumption.

Among various efficient filter designing methods, rounding technique is a simple method to get integer filter coefficients. Then, the filter coefficients can be implemented with the adders and the shift registers. Through repetitive experimentation, rounding technique has been found worthwhile for increasing

computational efficiency; when the filter cutoff frequencies are equal or below than 0.05 normalized frequency and the transition bands are in the range of 0.01 or less. The consequent effect of rounding filter coefficients is the distortion of filter response. This can be recover by using sharpening technique. Sharpening Technique involves the cascading of filter with itself following certain polynomials, in order to get minimum passband ripples and maximum stopband attenuation.

The proposed structure has been presented for receiver, but it can also be applicable for transmitter. Using multirate identities and polyphase representation, the filters have been moved to lower sampling rate region. This led to the filters to operate at lower rate and hence decreases the power consumption. The proposed structure has been applied at high sampling rate and at lower sampling rate for frequency range of GSM communication standard. This shows the trade off between input sampling rate and filter complexity.

The proposed structure has been implemented in FPGA using VHDL program generated by MATLAB. The recoding of VHDL program has been done to replace multipliers by adders and shift registers. The recoding of VHDL for the dynamic change of input signal has been done manually, so the outputs are not reliable. To get the reliable outputs CORDIC algorithm is required. However, replacing of multipliers led to the decrease of FPGA equivalent gates utilization. In this way a less complex structure is obtained by using add and shift algorithm of fixed point multiplication.

Some of the topics to be done in future are

1. The proposed algorithm has been used for the integer conversion factors less than or equal to 15. The work to be done is to verify the proposed algorithm for higher conversion factors with respect to the filter characteristics. This can be done by modifying the proposed algorithm on the basis of time varying characteristics.
2. The recoding of the MATLAB generated program has been done manu-

ally in order to replace the fixed point multipliers by the add and shift algorithm and to get the desired output. This recoding can be done more efficiently by using CORDIC algorithm.

tructure

A

MATLAB Functions

function[*H*] = cic(*R*,*N*)

This function generates the frequency response of the CIC filter.

Input Variables are

$R \rightarrow$ Integer Rate Change Factor.

$N \rightarrow$ Number of Stages.

Output Variable is

H $\rightarrow$ Filter Magnitude response.

function[*H*] = mcic(*d1*,*d2*,*d3*,*d4*)

This function generates the frequency response of the Modified CIC filter.

Input Variables are

d1, *d2*, *d3*, *d4* $\rightarrow$ Set of comb delays.

Output Variable is

H $\rightarrow$ Filter Magnitude response.

function[*H*] = stcic(*K1*,*K2*,*K3*,*M*,*M1*,*M2*,*R*)

This function generates the frequency response of the Stepped Triangular CIC filter.

Input Variables are

M, *M1*, *M2* $\rightarrow$ Set of comb delays.

$R \rightarrow$ Delay of expanded cosine filter.

K1, *K2*, *K3* → Number of Stages.

Output Variable is

H → Filter Magnitude response.

function[$y_f in$] = **tvcic(L,M,T1)**

This function generates the frequency response of the Time Varying CIC filter.

Input Variables are

L → Upsampling factor.

M → Downsampling Factor.

T1 → Input sampling period.

Output Variable is

$y_f in$ → Output Impulse Response.

function[n,x] = Ingen(fs,f1,f2,f3)

This function generates the sinusoidal input signal.

Input Variables are

fs → Sampling frequency.

f1 → Sinusoidal first centering frequency.

f2 → Sinusoidal second centering frequency.

f3 → Sinusoidal third centering frequency.

Output Variables are

n → Number of output samples.

x → Output samples.

function[N,h]=regobarb(wp,ws,Rp,As)

This function generates the frequency response of the FIR filter.

Input Variables are

wp → Normalized passband frequency.

ws → Normalized stopband frequency.

$Rp \rightarrow$ Passband ripples (dB).

$As \rightarrow$ Stopband attenuations (dB).

Output Variables are

$N \rightarrow$ Filter order.

$h \rightarrow$ Output impulse response.

function[B1,I1]=roundsimifir(R,A,wp,M,r,p)

This function generates the rounded and sharpened impulse response of model and interpolator filter.

Input Variables are

$R \rightarrow$ Passband ripples (dB).

$A \rightarrow$ Stopband attenuations (dB).

$wp \rightarrow$ Normalized passband frequency.

$M \rightarrow$ Decimation Factor.

$r \rightarrow$ Rounding factor for model filter.

$p \rightarrow$ Rounding factor for interpolator filter.

Output Variables are

$B1 \rightarrow$ Rounded and Sharpened impulse response of model filter.

$I1 \rightarrow$ Rounded and Sharpened impulse response of interpolator filter.

function[Hdm,Hdi,Hd] = improunifirfrac(L,M,R,A,wp,fs,f1,f2,f3,r,p);

This function generates the frequency response and VHDL programs of the proposed structure.

Input Variables are

$L \rightarrow$ Interpolation Factor.

$M \rightarrow$ Decimation Factor.

$R \rightarrow$ Passband ripples (dB).

$A \rightarrow$ Stopband attenuations (dB).

$wp \rightarrow$ Normalized passband frequency.

$fs \rightarrow$ Sampling frequency.

f1 → Sinusoidal first centering frequency.

f2 → Sinusoidal second centering frequency.

f3 → Sinusoidal third centering frequency.

r → Rounding factor for model filter.

p → Rounding factor for interpolator filter.

Output Variables are

Hdm → Rounded and Sharpened magnitude response of model filter.

Hdi → Rounded and Sharpened magnitude response of interpolator filter.

Hd → Cascaded magnitude response of model filter and interpolator filter.

B

VHDL Programs

B.1. Project for the proposed structure at low sampling rate

This project is for implementing proposed structure at low sampling rate with multipliers. It has following four entities

- $Hd - low - tb$: This entity defines input and output port mapping with the component $Hdl - low$. It has the following input-output ports.
  Input-output ports are
  clk as input clock
  $clk - enable$ as input enable clock.
  reset is the input reset pulse.
  $filter - in$ is input signal of 16 bits.
  $filter - out$ is the filtered output signal of 34 bits.
  $ce - out$ is the output enable clock.

- $Hdl - low$: This entity defines input and output port mapping with the component $Hdl - low - stage1$ for interpolator filter and the component $Hdl - low - stage2$ for model filter. It has the following input-output ports.
  Input-output ports are
  clk as input clock
  clk_enable as input enable clock.
  reset is the input reset pulse.

$filter - in$ is input signal of 16 bits.

$filter - out$ is the filtered output signal of 34 bits.

$ce - out$ is the output enable clock.

- $Hdl - low - stage1$: This entity defines interpolator filter.

  Input-output ports are

  clk as input clock

  $clk - enable - stage1$ as input enable clock for interpolator filter.

  reset is the input reset pulse.

  $filter - in - stage1$ is input signal of 16 bits for interpolator filter.

  $filter - out - stage1$ is the filtered output signal of 34 bits for interpolator filter.

  $ce - out - stage1$ is the output enable clock of interpolator filter.

- $Hdl - low - stage2$: This entity defines model filter.

  Input-output ports are

  clk as input clock

  $clk - enable - stage2$ as input enable clock for interpolator filter.

  reset is the input reset pulse.

  $filter - in - stage2$ is input signal of 16 bits for interpolator filter.

  $filter - out - stage2$ is the filtered output signal of 34 bits for interpolator filter.

  $ce - out - stage2$ is the output enable clock of interpolator filter.

B.2. Project for the proposed structure at low sampling rate by replacing multipliers

This project is for implementing proposed structure at low sampling rate without multipliers. It has following four entities

- $Hd - low_m od_t b$: This entity defines input and output port mapping with the component $Hdl_l ow_m od$. It has the following input-output ports.

  Input-output ports are

  clk as input clock

$clk - enable$ as input enable clock.

reset is the input reset pulse.

$filter - in$ is input signal of 16 bits.

$filter - out$ is the filtered output signal of 34 bits.

$ce - out$ is the output enable clock.

- $Hdl - low - mod$: This entity defines input and output port mapping with the component $Hdl - low - mod - stage1$ for interpolator filter and the component $Hdl - low - mod - stage2$ for model filter. It has the following input-output ports.

  Input-output ports are

  clk as input clock

  $clk - enable$ as input enable clock.

  reset is the input reset pulse.

  $filter - in$ is input signal of 16 bits.

  $filter - out$ is the filtered output signal of 34 bits.

  $ce - out$ is the output enable clock.

- $Hdl - low - mod - stage1$: This entity defines interpolator filter.

  Input-output ports are

  clk as input clock

  $clk - enable - stage1$ as input enable clock for interpolator filter.

  reset is the input reset pulse.

  $filter - in - stage1$ is input signal of 16 bits for interpolator filter.

  $filter - out - stage1$ is the filtered output signal of 34 bits for interpolator filter.

  $ce - out - stage1$ is the output enable clock of interpolator filter.

- $Hdl - low - mod - stage2$: This entity defines model filter.

  Input-output ports are

  clk as input clock

  $clk - enable - stage2$ as input enable clock for interpolator filter.

  reset is the input reset pulse.

  $filter - in - stage2$ is input signal of 16 bits for interpolator filter.

$filter - out - stage2$ is the filtered output signal of 34 bits for interpolator filter.

$ce - out - stage2$ is the output enable clock of interpolator filter.

B.3. Project for the proposed structure at high sampling rate

This project is for implementing proposed structure at high sampling rate with multipliers. It has following four entities

- $Hd - high - tb$: This entity defines input and output port mapping with the component $Hdl - high$. It has the following input-output ports.

  Input-output ports are

  clk as input clock

  $clk - enable$ as input enable clock.

  reset is the input reset pulse.

  $filter - in$ is input signal of 16 bits.

  $filter - out$ is the filtered output signal of 34 bits.

  $ce - out$ is the output enable clock.

- $Hdl - high$: This entity defines input and output port mapping with the component $Hdl - high - stage1$ for interpolator filter and the component $Hdl - high - stage2$ for model filter. It has the following input-output ports.

  Input-output ports are

  clk as input clock

  $clk - enable$ as input enable clock.

  reset is the input reset pulse.

  $filter - in$ is input signal of 16 bits.

  $filter - out$ is the filtered output signal of 34 bits.

  $ce - out$ is the output enable clock.

- $Hdl - high - stage1$: This entity defines interpolator filter.

  Input-output ports are

  clk as input clock

$clk - enable - stage1$ as input enable clock for interpolator filter.

reset is the input reset pulse.

$filter - in - stage1$ is input signal of 16 bits for interpolator filter.

$filter - out - stage1$ is the filtered output signal of 34 bits for interpolator filter.

$ce - out - stage1$ is the output enable clock of interpolator filter.

- $Hdl - high - stage2$: This entity defines model filter.

  Input-output ports are

  clk as input clock

  $clk - enable - stage2$ as input enable clock for interpolator filter.

  reset is the input reset pulse.

  $filter - in - stage2$ is input signal of 16 bits for interpolator filter.

  $filter - out - stage2$ is the filtered output signal of 34 bits for interpolator filter.

  $ce - out - stage2$ is the output enable clock of interpolator filter.

B.4. Project for the proposed structure at low sampling rate by replacing multipliers

This project is for implementing proposed structure at high sampling rate without multipliers. It has following four entities

- $Hd - high - mod - tb$: This entity defines input and output port mapping with the component $Hdl - high - mod$. It has the following input-output ports.

  Input-output ports are

  clk as input clock

  $clk - enable$ as input enable clock.

  reset is the input reset pulse.

  $filter - in$ is input signal of 16 bits.

  $filter - out$ is the filtered output signal of 34 bits.

  $ce - out$ is the output enable clock.

- $Hdl - high - mod$: This entity defines input and output port mapping with the

component $Hdl-high-mod-stage1$ for interpolator filter and the component $Hdl-high-mod-stage2$ for model filter. It has the following input-output ports.

Input-output ports are

clk as input clock

$clk-enable$ as input enable clock.

reset is the input reset pulse.

$filter-in$ is input signal of 16 bits.

$filter-out$ is the filtered output signal of 34 bits.

$ce-out$ is the output enable clock.

- $Hdl-high-mod-stage1$: This entity defines interpolator filter.

  Input-output ports are

  clk as input clock

  $clk-enable-stage1$ as input enable clock for interpolator filter.

  reset is the input reset pulse.

  $filter-in-stage1$ is input signal of 16 bits for interpolator filter.

  $filter-out-stage1$ is the filtered output signal of 34 bits for interpolator filter.

  $ce-out-stage1$ is the output enable clock of interpolator filter.

- $Hdl-high-mod-stage2$: This entity defines model filter.

  Input-output ports are

  clk as input clock

  $clk-enable-stage2$ as input enable clock for interpolator filter.

  reset is the input reset pulse.

  $filter-in-stage2$ is input signal of 16 bits for interpolator filter.

  $filter-out-stage2$ is the filtered output signal of 34 bits for interpolator filter.

  $ce-out-stage2$ is the output enable clock of interpolator filter.

C

Published Articles

Published

- Jovanovic Dolecek Gordana, Nagrale Rao Naina, Trejo Javier Fernando "**One Method for Multiplierless FIR Decimation Filter Design**", XXVIII International Congress of Electronics engineering ELECTRO-2006, Chihuahua, Mexico, October -2006.
- Jovanovic Dolecek Gordana, Trejo Javier Fernando, Nagrale Rao Naina "**Design of Multiplierless Multiband Decimation Filter**", 2nd International Conference of Electronics Design ICED- 2006, Veracruz, Mexico, November -2006.
- N. R. Nagrale, J. F. Trejo and G. Jovanovic Dolecek "**IFIR Based FIR Decimation Filter Design**", Memorias del Septimo Encuentro of Investigation INAOE 2006, Puebla, Mexico, November -2006, pp. 127 - 130.

Accepted

- N. R. Nagrale, G. Jovanovic Dolecek "**An Efficient Method to Design Fractional Decimation system**", Electronics, Robotics, and Automotive Mechanics Conference Cerma 2007, Cuernavaca Morelos, Mexico, September -2007.

Articles

List of Figures

IGURES

List of Tables

TABLES

References

[1] http://www.rfdesignline.com/encyclopedia.

[2] http://www.radioreference.com.

[3] W. A. Abu-Al-Saud and G. L. Stuber, "Modified cic filter for sample rate conversion in software radio systems," *IEEE Signal Processing Letters*, vol. 10, no. 5, pp. 152–154, May 2003.

[4] Z. J. Alan Y. Kwentus and A.Ñ. Willson, "Application of filter sharpening to cascaded integrator-comb decimation filters," *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 457–467, February 1997.

[5] A. Bartolo, B. D. Clymer, R. C. Burgess, and J. P. Turnbull, "An efficient method of FIR filtering based on impulse reponse rounding," *IEEE Transactions on Signal Processing*, vol. 46, no. 8, pp. 2243–2248, August 1998.

[6] E. Burrachini., "The software radio concept," *IEEE Trans. Acoustics., Speech, Signal Proceesing*, pp. 138–143, 2000.

[7] K. Chang, *Digital Systems Design with VHDL and Synthesis: An Integrated Approach.* USA: IEEE Computer Society, 1999.

[8] C. Farrow, "A continously variable digital delay element," *ISCAS-1988*, pp. 2641–2646, 1988.

[9] N. J. Fliege, *Multirate Digital Signal Processing.* New York: John and Wiley Sons, 1995.

[10] M. Henker, T. Hentshel, and G. Fettweis, "Time-variant cic-filters for sample-rate conversion with arbitrary rational factors," *IEEE 6th International Conference on Electronic, Circuits and Systems*, pp. 67–70, September 1999.

[11] T. hentchel, *Sample Rate Conversion in Software configurable Radios.* London: Artech House, 2002.

[12] E. B. Hogenauer., "An economical class of digital filters for decimation and interpolation," *IEEE Trans. Acoustics., Speech, Signal Proceesing*, vol. 29, no. 2, pp. 155–162, April 1981.

[13] G. Jovanovic-Dolecek and S. K. Mitra, "Stepped triangular cic filter," *APPCAS-2006*, pp. 918–922, November 2006.

[14] ——, "A new multistage comb-modified rotated sinc (rs) decimator with sharpened magnitude response," *IEEE Transactions on Informations and Systems*, vol. E88, no. 7, pp. 1331–1339, July 2005.

[15] ——, "A new two-stage sharpened comb decimator," *IEEE Transactions on Circuits and System-I: Regular Papers*, vol. 52, no. 7, pp. 1414–1420, July 2005.

[16] J. Kaiser and R. Hamming, "Sharpening the response of a symmetric nonrecursive filter by multiple use of the same filter," *IEEE Transactions on Acoustic, Speech, Signal, Processing*, vol. 25, no. 5, pp. 415–422, October 1977.

[17] I. Koren, *Computer Arithmetic Algorithms*. Massachusets USA: A.K Peters, 2002.

[18] R. Kuc, *Introduction to Digital Signal Processing*. New York USA: McGraw-Hill Book Company, 1988.

[19] W. Li and M. Tomisawa, "Transposed-farrow-structure-based multirate filters for symbol timing synchronization in software defined radio (sdr)," *IEEE Transactions*, pp. 1668–1672, 2004.

[20] J. Mitola, "Software radios: Survey, critical evaluation and future directions," *Proceedings of National Telesystems Conference*, 1992.

[21] ——, "Software radios," *IEEE communications Magazine*, pp. 24–38, May 1995.

[22] Y.Ñeuvo, C. Dong, and S. Mitra, "Interpolated finite impulse response filters," *IEEE Trans. Acoustics., Speech, Signal Proceesing*, vol. 32, no. 6, pp. 563–570, June 1984.

[23] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, The Hardware/Software Interface*. San Mateo USA: Morgan Kaufmann, 1994.

[24] Proakis, *Digital Signal Processing in Telecommunications*. New Jersey: Prentice Hall, 1995.

[25] T. S. Rappaport, *Wireless Communications Principles and Practice*. New Jersey USA: Prentice-Hall, Inc., Englewood Cliffs, 1983.

[26] G. B.-B. Richard J. Hartnett, "Improved filter sharpening," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 2805–2810, December 1995.

[27] L. R. R. Ronald E. Crochiere, *Multirate Digital Signal Processing*. New Jersey USA: Prentice-Hall, Inc., Englewood Cliffs, 1983.

[28] S. Samadi, "Explicit formula for improved filter sharpening polynomial," *IEEE Transactions on Signal Processing*, vol. 9, no. 10, pp. 2957–2959, October 2000.

[29] W. Tuttlebee, *Software Defined Radio*. England: John Wiley and Sons, LTD, 2002.

[30] P. Vaidyanathan and S. K. Mitra, "Very low sensitivity fir filter implementation using."