LOW COMPUTATIONAL COMPLEXITY CHANNELISATION ALGORITHM FOR MULTI-STANDARD SOFTWARE DEFINED RADIO RECEIVER

by

Otunniyi Temidayo Oluwafunke

Submitted in fulfillment of the requirements for the degree
Philosophiae Doctor (Computer Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

February 2022

SUMMARY

LOW COMPUTATIONAL COMPLEXITY CHANNELISATION ALGORITHM FOR MULTI-STANDARD SOFTWARE DEFINED RADIO RECEIVER

by

Otunniyi Temidayo Oluwafunke

Supervisor:	Prof. H.C. Myburgh
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Philosophiae Doctor (Computer Engineering)
Keywords:	Adders, canonical signed digit, channelisation, common subexpression elimination method, computational complexity, lookup tables (LUT), multi-rate, multi-stage, parallel distributed algorithm, poly-phase, Software Defined Radio (SDR)

With ever increasing wireless network demands, designing low computational complexity channelisation algorithms for software defined radio receiver platforms will continue to require research attention. Extracting and reconfiguring channels of choice from multi-standard receivers is computationally intensive. Higher number of multipliers involved in the filter designs impacts speed, resources utilisation, power, and consequently, costs.

Analyses of uniform and non-uniform channelisation algorithms show that Generalised Discrete Fourier Transform (GDFT) and Farrow Per Channel Channelisation (FPCC) compromise computational load and reconfigurability for the upcoming multi-standard SDR receivers.

Complexity increment in GDFT filter is attributed to extra three phase modifications added to its time and frequency offset. The resultant effects are higher filter order and higher delay in software defined radio (SDR) mobile systems.
Also, in FPCC, complexity compromise occurs during the process of approximating higher frequency

band with high level of distortion at passband ripples and stop band attenuation. Higher filtering operation is needed to do this. This increases filter computational complexities and reduces the performances of multi-standard receivers in software defined radio.

This work focuses on achieving a low complexity channelisation algorithm for these two algorithms for any frequency band whether low or high exploring three approaches in developmental sequence based on different computational rationales.

The first approach involves the development of two algorithms described as Hybrid Generalised Discrete Fourier Transform (HGDFT) and Hybrid Farrow (HFarrow) Channelisation Algorithms. These developments involve hybridising frequency response masking techniques and coefficient decimating filter with either the modulated GDFT or modulated Farrow filter. These methods efficiently reduce the computational complexity to some extent.

The second approach focuses on improving the performance of HGDFT and HFarrow algorithms using different digital number systems, implemented on parallel distributed arithmetic architecture. The digital number representations used are: Parallel Distributed Arithmetic Based Residual Number System (PDA-RNS), Parallel Distributed Arithmetic Based Canonical Signed Residual Number System (PDA-CSRNS), and Parallel Distributed Arithmetic Based Common Sub-expression Elimination Method (PDA-CSE). These approaches resulted in significantly fewer filter coefficients as well as a reduction in the number of multipliers and adders used. Thus, the computational complexities of HGDFT and HFarrow improved remarkably when optimised with the parallel distributed arithmetic and number systems.

However, the filter's passband ripples and the stop band attenuation degrade in performances due to the rounding of the continuous filter coefficients. A Genetic Algorithm (GA) was introduced to optimise the performances of the hybrid filter. Results obtained showed better filter structures with lower computational complexities.

The third approach uses the multi-rate, multi-stage method to reduce computational capability of the HGDFT and HFarrow channelisation algorithms. The large number of filter coefficients in HGDFT and HFarrow were factored into different stages. These relaxed the filter specifications and resulted in more reduced computational complexities.

Great is Thy faithfulness,

Oh Lord My Father!

There is no shadow of turning with thee;

Thou changeth not, Thy compassions they fail not,

As Thou has been, Thou forever will be.

(KJV Bible, James Chapter 1 vs. 17)

Acknowledgement

I would like to thank my family, especially my husband, Prof. Tunji Otunniyi for his unwavering support during this journey. My heartfelt appreciation to my children for their innocent good wishes and prayers which motivated and encouraged me throughout this study.

I would like to thank my supervisor, Prof. H.C. Myburgh for his support, contributions, encouragement and patience. I would like to express special thanks to Prof. H.C. Myburgh, Prof. G.P. Hancke, and Dr A. de Freitas and my departmental research secretary, Mrs Ferreira. Also to the ASN group, I appreciate you all.

My sincere appreciation to the Council for Scientific and Industrial Research (CSIR) for providing funding assistance.

Last but not least, the prayers and concerns during this period of my spiritual father, Pastor Jesulowo, will always be cherished.

LIST OF ABBREVIATIONS

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
CIC	Cascaded Integrator Comb
CRT	Chinese Remainder Theorem
CSLA	Carry Select Logic Adder
CSA	Carry Save Adder
CSD	Canonical Signed Digit
DFT	Discrete Fourier Transform
DI	Diminished One System
DSP	Digital Signal Processing
FIR	Finite Impulse Response filter
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
GSM	Global System for Mobile Communication
IF	Intermediate Frequency
NCO	Numerical Controlled Oscillator
PGDFT	Parallel General Discrete Fourier Transform
QMF	Quadrature Mirror Filter
RGDFT	Recombined General Discrete Fourier Transform
RNS	Residual Number System
SDR	Software Defined Radio
SM	Signed Magnitude
SRC	Sample Rate Converter
TQM-FB	Tree Quadrature Mirror Filter Bank

LIST OF SYMBOLS

Σ	Sum
Φ	Phi
$\|\|$	Modulus
$*$	Convolution
Re	Real values
Im	Imaginary values
$\prod$	Product
$\lfloor .. \rfloor$	Floor
$\lceil .. \rceil$	Ceil
$[]$	Matrix
$\geq$	Greater than
$\leq$	Less than
Δ	Transition width
μ	number of multipliers per samples
α	number of adders per samples

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 CHALLENGES OF SOFTWARE DEFINED RADIO

Due to new and emerging services demand in mobile computing, development of multi-standard receivers with better data rates, high processing speed, low execution time, minimal power consumption and different operating modes on single platform will continue to be of interest in mobile communication development. Such receivers must be able to process multiple communication standards with different channels bandwidth on a single platform. This can illustrated using real world receivers scenario with different air standards such as frequency modulation (FM), Global system for mobile communication (GSM), WLAN, CDMA, WCDMA, etc, with wide range of frequency bands, different bandwidths, different sample rates, different channel spacings and different modulation schemes can be implemented on a single platform. Conventionally, multi-standard receivers technology implement distinct standard on distinct receiver, resulting in high utilisation of processors resources, increased power consumption, and reduced operational speed, all leading to high costs in mobile systems. In contrast, software defined radio (SDR) is the platform for seamless multi-standard receiver capabilities.

Dynamically extracting and reconfiguring required channels on demand is done on SDR receiver system; a process known as channelisation. Channelisation extracts required or choice signals from wideband multi-standard input signals. This occurs at the digital front end (DFE) of the receiver, and involves placing the wideband analog digital converter (ADC) or digital analog converter (DAC) close to the receiver or the transmitter while digitising the input signals. Thus, the digitizing process is highly computationally intensive, involving several filtering stages such as digital down conversion, channel filtering and sample rate conversion. This is done using either channelisation algorithm or what can also be referred to as filter bank (FB) algorithm in SDR. However, the typical values of sample rates and dynamic ranges of the output signals to the channelizer and output signals of the synthesizer may be in the range of $60-350$ Msamples/s and $90-100$dB respectively [6].

Different filter banks or channelisation algorithms exist in literatures, [7, 8, 9, 10, 11, 12, 13, 6, 14, 15, 16, 17, 18, 19, 20]. Extents of computational complexity varied for different channelisation algorithms. A review of these algorithms is summarised in Table 1.1. The major challenges of channelisation algorithms as shown in Table 1.1 are higher filter orders with attendant computational load / complexity and low reconfigurability. Computational load is usually scaled as Very High, High and Low. Very High scale indicates higher filter order and filter coefficients. The High computational load indicates averagely high filter order while the Low scale denotes low filter order and filter coefficient. Also, the reconfigurability performance from Table 1.1 is scaled as: Good and Poor. Good in the sense of close to excellent performance and with ease of adaptability to different standards on the same platform while Poor on the scale shows difficulty in updating parameters on the same device.

The computational load of a given algorithm is measured from the filter order and filter coefficient's point of view while reconfigurability is evaluated based on ease or difficulty in updating parameters of specific functions or adding a totally new one on the same device. From Table 1.1, the TQMF, GDFT, HTQMB, PGDFT and FRM channelizers provide close to the most efficient reconfigurable solution. Any update means extracting the independent channels from the correct outputs of the structures. Per Channel, Binary Approach, discrete Fourier Transform (DFT), DFT (polyphase), EMFB, Farrow per channel channelisation (FPCC) and quadrature mirror Fourier transform (QMFT) are grouped as Poor on the reconfigurability scale. It can be seen from the Table 1.1 that there is a computational trade off in computational complexity and reconfigurability for generalised discrete Fourier transform (GDFT) and Farrow per channel channelisation (FPCC) algorithms. In this thesis, FPCC filter will also be referred to as Farrow filter.

The complexity increment in GDFT filter is attributed to extra three phase modifications added to its time and frequency offset which produce multiplicative complexity in the filter. The resultant effect is higher filter order and higher delay in software defined radio (SDR) mobile systems.
Also, in FPCC, complexity compromise is reached during the process of approximating higher frequency band close to Nyquist frequency with high level of distortion at passband ripples and stop band attenuation. A higher filtering operation is required to approximate a high-pass filter. This increases filter computational complexities and reduces performances of multi-standard receivers in software defined radio.

However, with rising demands of new and emerging technologies for multi-standard SDR based

receiver, it is interesting to assess how these two algorithms can cope going forward. With increasing services demands and upcoming radio standards, even the good algorithms as in Table 1.1 now require improvement. It is of interests in this work, to explore this improvement specifically for the generalised discrete Fourier transform filter bank (GDFT-FB) and Farrow Per channel (FPCC) channelisation algorithm. The target is to improve these algorithms to suit multi-standards based SDR receiver that can handle upcoming radio standards.

Table 1.1. Comparison of the different channelisation algorithms.

Very High: higher filter order and filter coefficient;
High: averagely high filter order; Low: low filter order and filter coefficients;
Good: easy adaptability to different standards on the same platform;
Poor: not easy adaptability to different standards on the same device.

Channelisation technique	Computational load	Reconfigurability
Per Channel [8, 9, 10]	Very High	Poor
Binary Approach [21]	High	Poor
DFT [22]	Very High	Poor
DFT (poly-phase) [7, 11, 12, 13]	High	Poor
EMFB [6, 14].	Low	Poor
FPCC [23, 24, 25, 26]	Very High	Poor
QMFB [27, 28]	High	Poor
TQMB [29]	High	Good
GDFT [22]	High	Good
HTQMB [15, 16]	High	Good
PGDFT [1, 6]	High	Good
RGDFT [3, 6]	High	Good
FRM [3, 30]	High	Good
CDM1 and 11 [17, 18, 19, 20]	Low	Poor
MCD1 and 11 [17, 18, 19, 20]	Low	Poor
ICDM 1 and 11 [17, 18, 19, 20]	Low	Poor

Improving the performance of multi-standard SDR receiver channelisation algorithm requires addressing the main causes of high computational complexity. These are the large number of multipliers and adders involved during the filtering operations. Multipliers contribute remarkably to complexity of

digital filters and channelisation algorithms, and this can be seen in the high filter orders obtained during implementation. Multipliers slow down computational speed, reduces dynamic reconfigurability, increase resources utilisation, increases production costs and power consumption. Reduced filter orders, total number of multipliers, and total number of adders, as well as reducing filter coefficients, in any channelisation algorithm, are indicators of lowering computational complexities.

Continuous efforts in developing a channelisation algorithm with low computational complexity, high speed and minimum power usage in the cellular frequency range is hoped pivotal to evolution of SDR platform in coping with multi-standard services demands continuously on the increase.

1.2 PROBLEM STATEMENT

While the conventional GDFT and Farrow channelisation algorithms have proven efficient on a SDR radio platform, it is imperative that their enhancement to cater for the upcoming multi-standard developments be uninterrupted, as multi standard services' demands on the platform continue to increase. Moreso, the conventional GDFT or traditional FPCC channelisation algorithms have high computational complexity.

Development of algorithm with fewer multipliers and adders and give preference to adders in the receiver circuit should lead to reduced computation complexity. In this context, some modifications on existing algorithms and prospects of employing different number system architecture can all be explored. Also, the optimisation of the designed filter using any metaheuristic approach will be considered.

1.2.1 Content of Problem Statement

Three phase modifications added to time and frequency offset in generalized discrete Fourier transform filter bank (GDFT-FB) increase its computational capabilities. Also the approximation of higher frequency signals at or close to Nyquist frequency band using Farrow filter, resulted into higher filter complexities.

Complexity in channelisation algorithm is a function of the number of channels used, filter order, number of multiplications and also the number of adders. The filter order and transition width are inversely proportional to each other. That means, the higher the filter order, the higher the number

of multipliers used. Also, the filter order and the number of multipliers are somehow related. If the design complexity can be reduced and if the multipliers can be replaced by the adders and the filter implemented with an add and shift mechanism, the number of filtering operations can be reduced.

1.2.2 Research Gaps

The extra phase modifications added to time and frequency offset in GDFT and the approximation of higher frequency band at or close to Nyquist frequency band using Farrow filter, render them unfit for the upcoming multi-standard receiver system.

These two algorithms still show high computational load, which manifests in signal delays, high resource utilisation, slow processing speed and very high power consumption. With the increasing demand for multi-standard channelisation, development of improved algorithms must be continuous.

1.3 RESEARCH OBJECTIVES AND QUESTIONS

. The general objective is to explore combination of low computational complexity channelisation algorithms and number systems that should give a flexible multi-standard channelisation algorithm with low computational complexity, high speed, low power and low delay capacity. Optimisation of these combinations is employed for improved filter performances. The channelisation must be chosen in conformity with the requirement of each channel standard to ensure aliasing free signals with minimum delay time or delivery time. The channelisation in SDR receivers must be realised to meet the stringent specifications of low power consumption and high speed [31] to enhance its performance for optimisation purposes.

- How does the channelisation algorithm such as hybridized GDFT and hybridized FPCC compare in multipliers when handling different standards, and computation complexities based on resources utilisation, processing speed and power consumed?
- What are the possible computational substitutes of digital components that contribute most to computational complexities in filters?
- How will different number systems affect performance of hybridized GDFT and hybridized Farrow filters, among others, with respect to resource utilisation?
- Can optimisation algorithm such as Genetic Algorithm, affect the performance of the different number systems used on the filter?

- To what extent can cascading multi-rate, multi-stage filter improve performance in channelisation?

1.4 HYPOTHESIS AND APPROACHES

Existing GDFT and FPCC channelisation algorithms have higher filter order with unquantifiable and unscaled numbers. Should the filter coefficients of the conventional GDFT and FPCC be reduced, then the computational complexity could be reduced. Should the input and digital filter coefficients of the hybridized channelisation algorithms be quantised, scaled, and digitised, the processing should lead to a reduced number of multiplication and addition in a channelisation algorithm, with the effect of reduced computational complexities in terms of silicon usage, power utilisation, and delay with corresponding improvement in speed of channelisation events. Should the number of multipliers used be replaced by adders, and the implementation carried out using add and shift mechanisms, and if the adders used are further optimised, then the computational complexity could be reduced.

Exploring a combination of channelisation transforms and varieties of arithmetic numbering representatives with optimisation algorithms are prospectively encouraging. A simulation of such channelisation algorithms, should reflect low computational complexity in terms of the number of filter order, the number of multipliers and the number of adders. These will achieve reduced complexity of channelisation in the SDR system.

Different receiver standards are available, however, this work will focus on cellular Bluetooth, Zigbee and Wideband Code Division Multiple Accesses (WCDMA). The existing channelisation algorithms, such as generalised discrete Fourier transform (GDFT) and Farrow Per Channel Channelisation (Farrow) filter will be implemented on these three frequency bands. The simulation will be done using the MATLAB/ Simulink software, and the co-simulation will be performed on Altera FPGA Stratix IV.

1.5 RESEARCH GOALS

The aim of the research is to develop a low computational complexity channelisation algorithm for multi-standard receiver platforms with improved performance relative to existing algorithm with respect to multipliers utilisation, adders consumption, silicon area utilisation, processing speed and power consumption. In this context, specific investigation objectives are to:

- Improve the generalised discrete Fourier transform and Farrow per channel filter bank algorithm

by a kind of hybridisation for reduced computational complexity, and simulate the hybrid algorithms to assess the performance.

- Explore further improvement on developed hybrid algorithms using different number systems with genetic algorithm, towards reducing number of multipliers and adders in the filter bank for further reduction of computational complexity, and simulate the new algorithms to assess the performance relative to existing algorithms. The hybrid filter performance is further optimised using any of the known meta-heuristic algorithms and their performances compared.
- Develop multi-rate, multi-stage hybrid generalised discrete Fourier transform and hybrid Farrow filters in an attempt to reduce computational complexity of the filter to the barest minimum.

1.6 RESEARCH CONTRIBUTION

After pursuing the set goal of this project, the following specific contributions were made in development of channelisation algorithm with reduced complexity compared to existing algorithms in SDR communication:

- Development of a hybrid generalized discrete Fourier transform (HGDFT) filter bank and further improvement thereon.
- Development of a hybrid Farrow filter bank, and further improvement thereon.
- Development of a multi-rate, multi-stage hybrid generalised discrete Fourier transform and hybrid Farrow filter bank.

1.7 RESEARCH OUTPUTS

The publications resulting from this work are outlined below.

1. T.O. Otunniyi and H.C. Myburgh, "Low-Complexity Filter for Software-Defined Radio by Modulated Interpolated Coefficient Decimated Filter in a Hybrid Farrow", *Sensors*, vol. 22, number 3, p.1164, 2022.
2. Low Complexity Filter for Software Defined Radio by Modulation and Masking in a Hybrid GDFT (Under Review).
3. T.O. Otunniyi and H.C. Myburgh, " Improving Generalized Discrete Fourier Transform (GDFT) Filter Banks with Low-Complexity and Reconfigurable Hybrid Algorithm," *Digital*, vol.1, pp.1-17, 2021.

1.8 OVERVIEW OF STUDY

Chapter 1 presents background understanding and the motivation for the thesis.

Chapter 2 provides a broad review on software defined radio (SDR) architecture and channelisation algorithms for both uniform and non-uniform channel bandwidth. The theoretical background and mathematical knowledge as well as issues relating to the computational complexities of the filters are presented. It also focusses on fixed and floating-point number system and the different digital number system representations that are suitable for multi-standards receivers. Different optimisation algorithms are also discussed.

Chapter 3 discusses the development and improvement of a new channelisation algorithm described as Hybrid Generalised Discrete Fourier Transform (HGDFT) algorithm for efficient realization of low complexity multi-standards SDR receivers. An improvement was made on HGDFT channeliser using parallel distributed based residual number system (PDA-RNS), parallel distributed arithmetic based canonical signed digit (PDA-CSD) and parallel distributed arithmetic based common sub-expression residual number system (PDA CSE RNS). These improvements were further optimised using Genetic algorithm.

Chapter 4 discusses the development and improvement of another algorithm described as Hybrid Farrow (HFarrow) algorithm. An improvement was made on HFarrow channeliser using parallel distributed based residual number system (PDA-RNS), parallel distributed arithmetic based canonical signed digit (PDA-CSD) and parallel distributed arithmetic based common sub-expression residual number system (PDA CSE RNS). These improvements were additionally optimised using Genetic algorithm.

Chapter 5 introduces multi-rate, multi-stage HGDFT and HFarrow algorithm described as MHGDFT and MHFarrow channelisation algorithms, to further assist in lowering the complexity of HGDFT and HFarrow algorithms.

Chapter 6 presents the results of the three developed algorithms discussed in Chapter 3, Chapter 4 and Chapter 5. Chapter 6 has three sections. The first Section discusses the results obtained from the development and improvement of HGDFT channelisation algorithm and their performances in comparison with other existing channelisation algorithms from literature. The second Section also

discusses the results obtained from the development and improvement of HFarrow algorithm. Section 3 explains the results obtained from the development of MHGDFT and MHFarrow channelisation algorithms.

Chapter 7 summarises the major contributions of the thesis and also highlights the potential research work .

CHAPTER 2 LITERATURE STUDY

2.1 INTRODUCTION

Channelisation in a multi-standard receiver platform requires extracting channel of choices from wide-band input signals. This can be classified into two broad groups, namely: uniform and non-uniform channelisation. Uniform channeliser is efficient when single standard is considered. This can be seen when single channel is channelised from single standard such as frequency modulation (FM) signal; while non-uniform channeliser requires isolating wideband input signals with multiple standards and distinct access methods, modulation schemes, center bandwidth. SDR supports simultaneity of multiple communication standards by dynamically reconfiguring the same hardware platform. Different highly incompatible Standards such as Interim Standard- 95(IS-95), Global System for Mobile Communication (GSM) and Code-Division Multiple Access 2000 (CDMA 2000) are employed. The incompatibility can be expressed in terms of access methods: Frequency-Division Multiple Access (FDMA), Code-Division Multiple Access (CDMA), Time-Division Multiple Access (TDMA) and modulation schemes, namely: Frequency Modulation, Analog Modulation (AM), Quadrature-Phase Shift keying (QPSK), Minimum Shift Keying (MSK) and Gaussian Minimum Shift Keying (GMSK) and Radio Frequency (RF) operation bands such as VHF, UHF, SHF, or EHF [32, 33, 34].

Table 2.1 shows the different air interfaces' characteristics such as access method, modulation schemes, RF frequency band and channel spacing.

The classification is based on the bandwidth capacity, the channel capacity, as well as the communication standards.

Many factors influence the channelisation of multi-standard receivers in SDR. Among these factors are computational complexity and dynamic reconfigurability. The type of receiver architecture, the

Table 2.1. Characteristics of different air waveforms.

Standards	Access Techniques	Modulation Type	Frequency Band	Channel Spacing
GSM	FDMA/TDMA	GMSK	890-915/915-960	0.2
AMS	FDD	FM	824-845/ 869-894	0.03
DCS-1800	TDMA	GMSK	1710-1785/1805-1880	0.2
PCS-1900	TDMA	GMSK	1880-1910/1930-1955	0.2
IS-95	CDMA	OQPSK	824-849/869-894	0.2
EDGE	TDMA	FDD	1850-1910/1930-1990	0.2
ZIGBEE	CDMA/FH	CFSK	2402-2480	5
IEEE 802.1b	DSSS	D-BPSK/D-QPSK	2400	22

chosen algorithms and the digital filter designers' choice of adders or multipliers used will influence the computational complexity and reconfigurability of the multi-standard SDR radio.

This chapter will commence by reviewing the background knowledge of Software Defined Radio as seen in Section 2.2. Section 2.3 will present the typical software defined radio receiver architecture. Following is the presentation of different Channelisation algorithms for both uniform and non-uniform bandwidth channels, together with their computational complexity as well as reconfigurabilities as will be explained in Section 2.4. Comparison and benchmarking to existing literatures are given. Section 2.5 will provide a brief introduction as to what digital system representation entails, while Section 2.5.1 will provide a comprehensive understanding of different digital system representations. The different optimisation algorithms will be explained in Section 2.6 and the Chapter will close with Concluding Remarks in 2.7.

2.2 BACKGROUND KNOWLEDGE OF SOFTWARE DEFINED RADIO

The first wireless transmission came into existence in 1890 [35, 36, 37] and since then the radio transmission techniques have been undergoing remarkable development in order to provide seamless connectivity to end-users at high data rate [38]. Mid-1930 witnessed another revolution in analog voice transmission technology, albeit with limited bandwidth. In 1950, analog television broadcast became realisable to the end-users' satisfaction, with high cost implication in bandwidth consumption. The rapid technological revolution brought improvement in communication technology for long

distances communincation, both for wired and wireless connection via ARPANET, called Internet [39], and wireless satellite [40]. Expansion in technological development facilitates the emergence of mobile phones or cellular phones and high speed mobile applications such as wireless fidelity (WI-Fi), worldwide interoperability for microwave access (WIMAX), wireless local area network (WLAN), Bluetooth (BT), code division multiple access (CDMA), global system of mobile communications (GSM) and other wireless applications [41], providing users with ubiquitous wireless connectivities such as voice and data communication anywhere and anytime [42]. Despite the exponential growth in communication, the technology behind the development is hardware-based [35].

Traditional radio technologies, with its varied components like filters, analog converters, modulators and de-modulators, are analog in nature. These technologies are based on different architectures with various applications placed on different integrated circuits (IC). The approach had proven efficient in time past, but inflexibility in hardware configuration and architecture makes the design implementation difficult. Thus, design errors are difficult to eradicate and in most cases result in the complete redesign of the products [35]. Also, implementation cost, as well as time to market the product, constitutes a major factor to the production of the analog products.

Inherent inflexibility in the reuse of traditional radio design approach led to the gradual migration from traditional radio techniques to digital radio. The replacement of analog radio with digital ones was proposed by Mitola in the mid 80s and was known as software defined radio (SDR). In 1990, Mitola designed Speak-Easy trans-receiver platform that was based purely on SDR technology for Rome Air Force Base, New York. The design provided tactical military communications from 2 MHz to 2 GHz and allowed seamless interoperability between the different air interface standards of the different branches of the armed forces. Speakeasy II [43], however, achieved greater success due to advances in electronics, wireless communication, re-usability and modular programming technology.

The term ’Software Defined Radio’ (SDR) or ’Software Radio’ (SR) has different definitions and interpretations to different people, but according to the father of SDR, it is defined as "as the type of radio that can be implemented on software technology". It is a wireless technology in which the transmitter modulation is generated by computer, while the receiver uses the computer to recover the signal information and intelligence [44, 45]. Quite a large number of definitions exist for SDR but SDR Forum, along with IEEE P1900, defined it as a radio technology in which all the physical layers are software defined.

SDR is a radio platform that can transform programmable software to hardware reprogrammable logic. It is such a flexible device that allows different air interfaces to take different forms that can be altered or modified to suit the need at hand. The flexibility can be done with general purpose signal processors (DSPs) or field programmable gate arrays (FPGAs). In order to take advantage of such digital processing, traditional analog signals must be converted into the digital domain [46]. This is accomplished with the help of Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC). Thus, SDR signal operates in the digital domain, digitising and reconstructing the signal as close as possible to the antenna.

Reconfigurability and reprogrammability are important features of SDR. Different air standards such as frequency modulation (FM), global system for mobile system (GSM), wireless local area network (WLAN), code division multiple access (CDMA), wideband code division multiple access (WCDMA), and so forth, with wide range of frequency bands, different bandwidths, different sample rates, different channel spacings and different modulation schemes can be implemented on a single platform.

Another important feature of SDR communication systems that distinguishes it from conventional radio counterparts, is their ability to mutate, depending on the air interface(s) of the communication system(s) which they are attempting to communicate with. The mutation is performed by software-reprogramming of the signal processing components while avoiding any form of modification or upgrade to the radio hardware [43]. SDR is useful for real-time application such as satellite mobile communication, mobile military communication, air or sea traffic control and WLAN [47]; cellular communication systems find it highly attractive.

2.3 SOFTWARE DEFINED RADIO RECEIVER ARCHITECTURE

SDR receiver is composed into three groups, namely: analog front end, digital front end and digital back end. The analog front end performs the radio frequency (RF) functionalities; the digital front end performs the demodulation, digitising and sample rate conversion operation while the digital back-end performs the base-band operation such as channel coding, channelisation, equalisation, and timing recovery. Block diagram illustrating software defined radio receiver architecture is depicted in Figure 2.1.

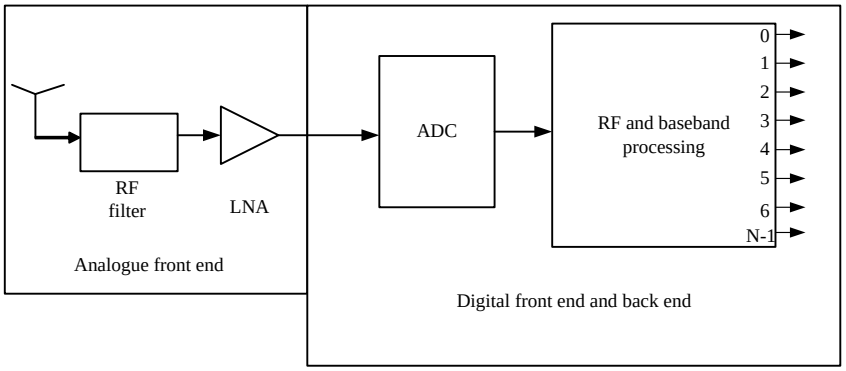

Figure 2.1. Block Diagram of Software Defined Radio Receiver (Modified from [1]).

Radio receivers have different architectures which varied depending on the user's choice, system applications, system complexities, power consumption, external component count and system cost. Amstrong in early 1918 proposed super-heterodyne receiver architecture in an attempt to down-convert the received RF signals to an intermediate frequency (IF) in two or more steps, while amplification is distributed across several frequency stages, which somehow relaxes the gain, stability requirement in each stage and increases the total possible gain [48]. In super-heterodyne architecture, the input RF signal is down-converted to a lower IF, which is digitised prior to digital filtering and demodulation. The architecture consists of bandpass filter (BPF), low noise amplifier (LNA), mixer, local oscillator (LO), IF amplifier and ADC anti-aliasing filter, as shown in Figure 2.2.

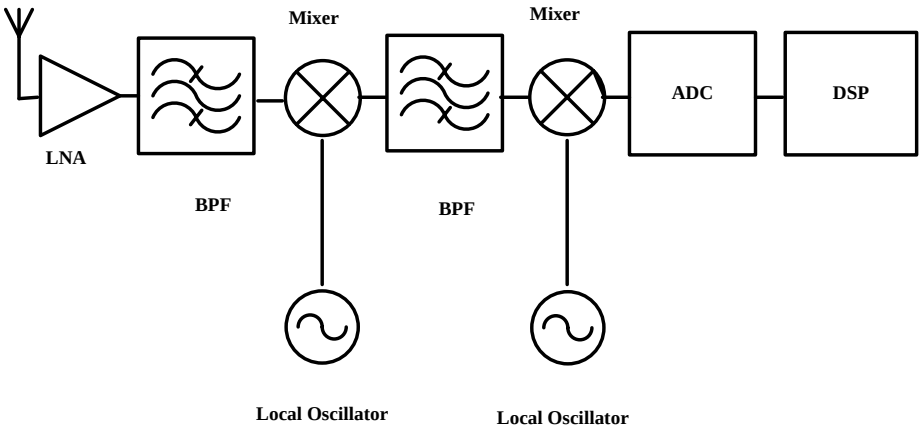

Figure 2.2. Super-heterodyne receiver architecture (Modified from [1]).

The signal band is translated into lower frequencies, using the mixer. In order to bring the centre frequency from ω_1 to ω_2 , the signal is first mixed with the sinusoidal wave, $A_0 \, cos \, \omega_0 t$, where ω_0=ω_1 -ω_2 , thereby yielding one band around ω_2 and another frequency band around $2(\omega_1 - \omega_2)$.

A low-pass filter is inserted to remove the latter in a process called down-conversion mixing or simply down-conversion. This operation generates high noise levels which can disrupt the original signal. Different RF channels are also selected by tuning the local oscillator frequency.

Generally, a super-heterodyne receiver offers good frequency selectivity and high sensitivity but it must be provided with an image rejection filter to suppress the image signal causing distortion to the original signal. However, its main drawback is that bulky filter is used which makes integration on chip difficult.

Most trans-receiver systems suitable for single chip integration use direct conversion, wide-band intermediate frequency (IF) or low-IF architecture. The direct-conversion architecture or zero-IF is a popular architecture in modern wireless receivers. Due to its high level of integration, it can process multi-standards wireless communications on a single platform. If the sampling frequency is carefully chosen, the multiple frequencies can be intentionally aliased to non-overlapping portions of

the Nyquist bandwidth and a single output stream, containing signals from all frequencies of interest, will be generated. In direct conversion architecture, the signal chain in the traditional super-heterodyne receiver architecture is replaced with the RF sampling ADC [49], as shown in the Figure 2.3.

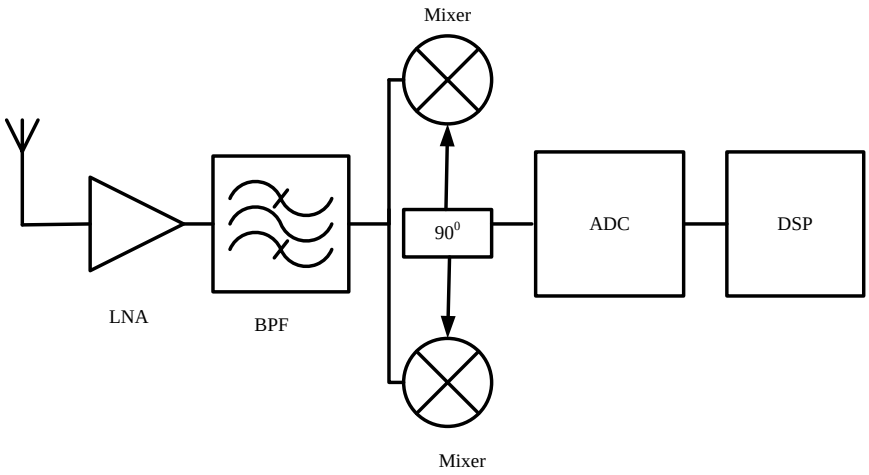

Figure 2.3. Direct conversion receiver architecture (Modified from [1]).

The data converter digitises the wide-band frequency spectrum directly at radio frequency (RF) and sends the signal to digital signal processors (DSP) at the digital front end for channelising. A shift from analog signal processor to digital domain is noticeable. The RF signal is sampled without mixing the signal down to IF. A new class of direct RF sampling ADC is being designed in advanced complementary metal oxide semi-conductor (CMOS) processes that allows higher conversion rates and reduced power than some previous generations. The receiver samples at a lower frequency than the original carrier, provided that the sampling frequency obeys the Nyquist theorem. With the help of an appropriate bandpass filter, the sampled signal gets aliased down to intermediate frequency (IF). that is within the Nyquist bandwidth [50, 51]. In this case, the received RF signal is down-converted to DC or low-IF. The image signal is at the desired signal frequency, and there is no need for image-rejection filters. Thus, direct-conversion receiver is suitable for on-chip integration since local oscillator (LO) frequency is equal to the input carrier frequency. The channel selection requires low-pass filter with sharp cut-off frequency. The use of a single RF chain for all signals simplifies the design time, resulting in cost reduction and elimination of potential sources of differential line bias. A differential line bias causes problems when sampling with multi-frequency carrier signals [52]. The major drawbacks are DC offset and second order distortion [53].

A wide-band IF trans-receiver, including sliding IF configuration, has a high IF ranging from 100 MHz to 200 MHz. Therefore, image signals using external bandpass filters and image rejection mixers using complex multiplication are rejected. This approach can accelerate image rejection mode, thus the wide-band IF trans-receivers can be applied to wireless communication.

Individual radio receiver architecture has its merits and drawbacks, but both super-heterodyne and direct-conversion offer good frequency selectivity and high sensitivity and can be used to demodulate any air wave-forms.

Ideal SDR receiver architecture is not practically realisable; therefore, the practical SDR receiver architecture can be used in order to make the implementation feasible. Figure 2.4 illustrates a practical SDR receiver model: the signal is digitally converted as close as possible to the antenna implementing all the radio functionalities by means of a digital signal processor (DSP), using only the minimal essential amount of analog hardware. The receiver allows the radio to digitise the entire RF band, while the DSP is used for receiver functionalities such as tuning, filtering and demodulation.

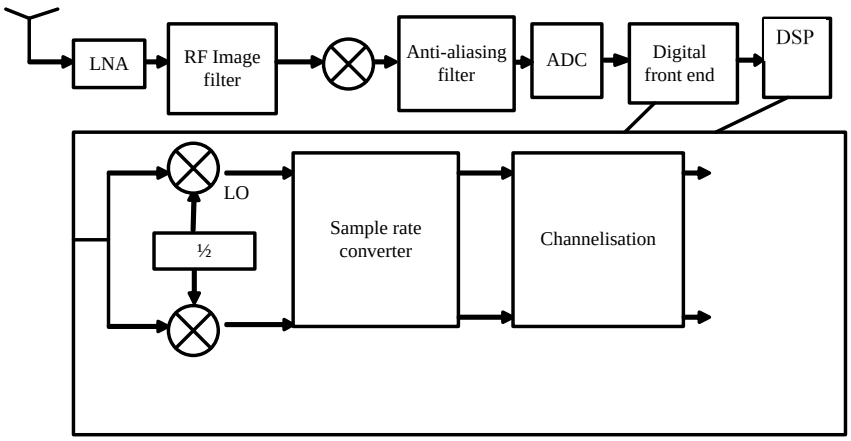

Figure 2.4. Practical SDR Implementation.

2.4 CHANNELISATION ALGORITHM

New generation mobile phones and other portable devices capable of transmitting and receiving new standards such as global system of mobile communications (GSM), wideband code division multiple access (WCDMA) and wireless local area network (WLAN) and so forth, are demanding single access

point technology that will allow ranges of signals to be transmitted from a single base station. This will improve spectrum efficiency in terms of signal-to-noise ratio and data throughput.

Channelisation extracts the channel of interest from the multi-standard receiver with different channel characteristics and different standards. However, the ADC that will digitise multi-gigahertz wide-band signals is not realisable using today's technology. The performance of such ADC is limited by a dynamic range of the input signal, the power interferer [54] and high energy dissipation [55] level. The sampling frequency is sampled at least twice the widest channel bandwidth of the supported standards to prevent aliasing, as well as loss of information signal. This increases the data rate of the signal, as well as the resolution required to capture the information. However, the high sample rates at ADC prevent realisation of ideal SDR. In order to reduce the sampling frequency, the sample output is fed into the digital down converter (DDC). DDC is a monolithic chip with three main parts: digital mixer, digital logic oscillator and finite impulse response filter (FIR). The digital mixer shifts the IF digital samples to base-band while the FIR low-pass filter limits the bandwidth of the final signal [56].

However, the increased flexibility of SDR systems comes at a cost. Performing many of the digital functionalities, such as channelisation at the digital front end, increases the computational task of the system. The computational requirement is further intensified by interoperability functionalities of SDR systems under different air waveforms. This incorporates the use of digital functions that are not required in conventional digital communication, such as SRC [57, 58].

Attempts to deploy these multi-standard airwaves onto a single mobile or wireless platform requires low complexity, reconfigurable channelisation techniques with minimum power dissipation, overhead and lower computational resources.

The computational complexity of a receiver is described in terms of resources required to carry out a particular task. This can be expressed in terms of filter order, the accumulator, the kind of arithmetic operation performed on each filter such as addition, multiplications and the memory allocation or space occupied during the digital implementation. Thus, computational complexity is a function of filter length and filter coefficient. The smaller the filter coefficient, the smaller the memory space occupied and the faster the speed of the channelisation. The higher the filter length, the more the power consumed and the slower the processor. The channelisation in SDR receivers must be realised to meet the stringent specifications of low power consumption and high speed [31] to enhance its

performance for optimisation purposes.

2.4.1 Per Channel Algorithm

Per Channel (PC) channelisation is a uniform channelisation algorithm, based on one-to-one parallel implementation of the digital processing chain [8, 9, 10]. Each digital chain processes its operations until base-band signal is obtained. The wide-band input signals are delivered to every digital chain for down-conversion and filtering processes. The digital wide-band signals at the digital front end contain all the communication channels at both sides of the PC. Each parallel signal chain demodulates its interested channel from the centre of frequency to DC before applying a low pass filter, $H(z)$. The filtered signals can be down-sampled in order to reduce the sample rate.

The order of operation is from down-sampling to filtering and sometimes this order can be reversed. It can be either digital down-conversion, followed by filtering and vice versa, as in Equation (2.1).

$$X_{BB,K}(n) = h(n) \times x(n) e^{-(j2\pi f_{CHK} n)} \tag{2.1}$$

where $h(n)$ denotes the impulse response; $x(n)$ represents the input signal and f_{CHK} is the center frequency for the given channel. This is followed by a down-sampling operation in Equation (2.2).

$$\begin{aligned} Y_{BB,K}(n) &= X_{BB,k}(nD) \\ \text{for K } &= 0, 1, ..., k-1 \end{aligned} \tag{2.2}$$

The frequency domain representation is indicated in Equation (2.3) and Equation (2.4) respectively.

$$X_{BB,K}(n) = H(z) \times X(z e^{j\omega_{CHK}}) \tag{2.3}$$

$$\begin{aligned} Y_{BB,K}(n) &= \frac{1}{D} \sum_{i=0}^{D-1} X_{BB,K}(z^{\frac{1}{D}} W_D^i) \\ W_D^k &= e^{j2\pi \frac{k}{D}} \end{aligned} \tag{2.4}$$

However, for uniform standard channels, the center frequencies must match multiple channel spacing defined for the particular system. Each channel is allocated a frequency that is an integer multiple of the channel spacing from DC frequency, as shown in Equation (2.5).

$$f_{CH} = \pm m f_{cs} \quad \text{for m= 1,2,..., } \frac{(k-1)}{2} \tag{2.5}$$

The computational load for the PC channelisation algorithm is high as a result of the operations carried out concurrently as indicated by the Equation (2.6), where N is the order of the real coefficients FIR low-pass filter.

$$\begin{aligned} \mu_{pc} &= N+5 \\ \alpha_{pc} &= 4(N+1) \end{aligned} \tag{2.6}$$

where μ_{pc} represents the addition per input sample and α_{pc} is the multiplication per unit sample.

PC is simple in approach and implementation. Its main drawback is that increase in the received channels as well as increase in the digital signal processing chains result in increases in the computational complexities of the system. Apart from these, the algorithm is not reconfigurable since each branch is used to channelise a single channel at a time.

Reducing the sampling rate up to the limit called the "Nyquist rate", which says that the sampling rate must be higher than the bandwidth of the signal to avoid aliasing [4]. For band pass signal, the frequency of interest must fall within the integer band, as shown in Equation (2.7).

$$\frac{kf_s}{2R} < f < (k+1)\frac{f_s}{2R} \tag{2.7}$$

If it does not fall within the band, there may be aliasing due to copies from negative frequency bands and this can be compensated by using a digital sinc compensation filter, $\left(\frac{1}{\sin(x)}\right)$.

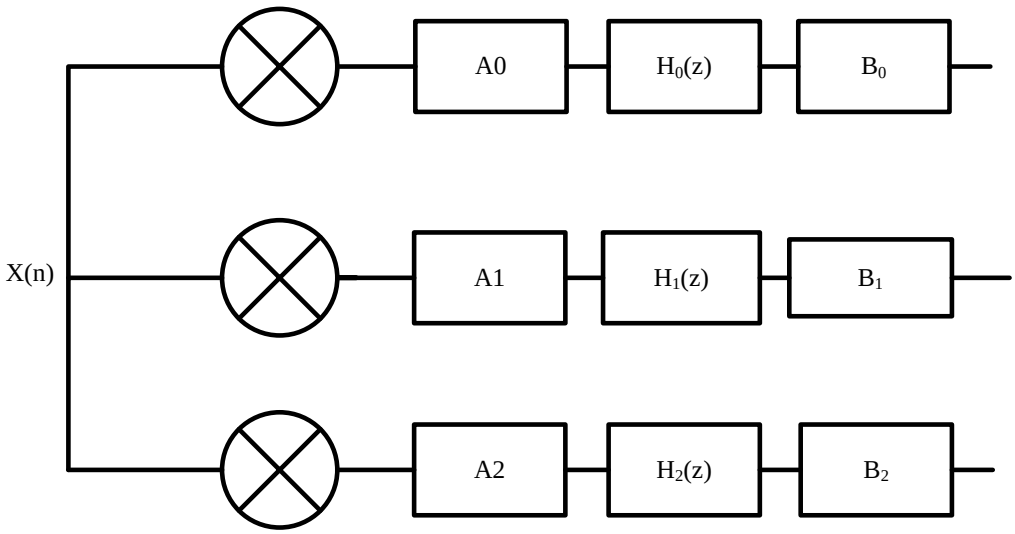

Figure 2.5. Per Channel Channelisation Algorithm.

2.4.2 Multi-rate Technique

A multi-rate system is a system which utilises different sampling rates at different stages along the digital chain of operation [61]. It reduces the sampling rate of discrete signals to a lower output rate and converts the given sampling rate to the desired sampling rate without causing distortion to the signal of interest. The sample rate alteration can be either up-sampling or down-sampling. Different approaches to reduce computational complexity entails a reduction of the number of multiplications and reduction of the sample rate.

A band-limited continuous signal, $x_c(t)$, can be converted into two discrete signals, $x[n]$ and $y[n]$ respectively, by sampling the original continuous signal, $x_c(t)$, at two different sampling frequencies rates, F and $F^{'}$ respectively, as in Equation (2.8).

$$\begin{aligned} x[n] &= x_c(T) \\ y[n] &= x_c(T^{'}) \\ F &= \frac{1}{T} \\ F^{'} &= \frac{1}{T^{'}} \end{aligned} \tag{2.8}$$

The sampling frequencies must follow the Nyquist criteria to prevent aliasing due to distortion. Hence, it is possible to reconstruct the original signal from the discrete signals and vice versa. It uses up-sampling and down-sampling to execute its implementation as follows.

2.4.2.1 UP-Sampling

The multi-rate technique uses up-sampling to increase the sampling rate of the signal without increasing the spectral content of the signals. It increases the signal separation between the spectral of image signal without necessarily adding any information to the existing signal [27]. Up-sampling increases the sampling rate, F, of the original discrete signal, $x[nT]$, to $F^{'}$. Up-sampling by integer factor L insert $L-1$ zeros between consecutive signals as shown in Figure 2.6.

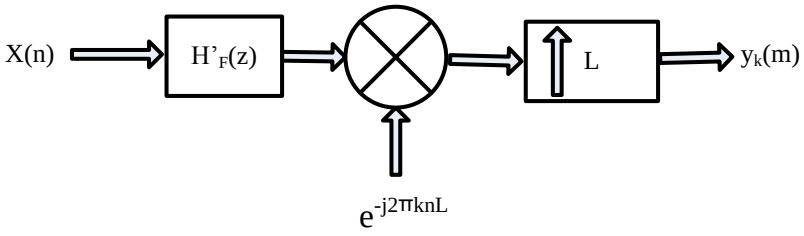

Figure 2.6. Basic Up-sampling method (Modified from [1] with permission).

The new discrete signal is $y[m]$ with sampling frequency, $F^{'}$, which is L times larger than the original discrete signal, as shown in Equation (2.9).

$$y[m] = \begin{cases} x[M/L], & \text{provided } m = 0, \pm L, \pm 2L... \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

Interpolation by factor, L, inserts zeros into the new rate at two consecutive signals according to Equation (2.9) and Equation (2.10). In z-domain, this can be represented as in Equation (2.10).

$$\begin{aligned} Y(z) &= \sum_{n=-\infty}^{n=\infty} y(n) z^{-n} \\ &= \sum_{n=-\infty}^{n\infty} y(n) z^{-n} \end{aligned} \tag{2.10}$$

The new sampling frequency, $F^{'}$, and the original signal, F, are related by $F^{'} = \frac{F}{L}$. This implies the new discrete signal $x'[n] = \frac{x[n]}{L}$, are constructed by interpolating the reconstructed samples at point $\frac{nF}{L}$, as indicated in Equation (2.11).

$$\begin{aligned} x_i[F^{'}] &= x\left[\frac{F}{L}\right] \\ &= \sum_{n=-\infty}^{n=\infty} x(kF)sinc\left(\frac{n}{L}-k\right)pl \end{aligned} \tag{2.11}$$

The instance of interpolator in Equation (2.11) is impractical because of an infinite nature of the sinc function. Truncating the sinc function results in a realisable and practical interpolator, as shown in the Equation (2.12).

$$\begin{aligned} x_i[F^{'}] &= x\left[\frac{F_T}{L}\right] \\ &= \sum_{n=-L}^{n=L} x(kF)sinc\left(\frac{n}{L}-k\right) \end{aligned} \tag{2.12}$$

Computational complexities of the interpolator is high because of high operations and some computational redundancy involved.

2.4.2.2 Decimation

Decimation or down-sampling is a multi-rate technique which decreases the sample rate of a given frequency by a certain factor, M. Implementing the new sequence, $y(M)$, as a function of the old sequence by discarding the, $M-1$, sample sequences, while retaining every M^{th} sample, is shown in Equation (2.13).

$$y[m] = x[mM] \tag{2.13}$$

Assuming the sampling function is defined as shown in Equation (2.14).

$$s_M[n] = \begin{cases} 1, & \text{provided } n = 0, \pm M, \pm 2M... \\ 0, & \text{otherwise} \end{cases} \tag{2.14}$$

Multiplying the discrete sequence with the sampling function yields an intermediate sampling rate, $y_n[s_M(n)]$, as shown in Equation (2.15).

$$y_n(s_M[n]) = \begin{cases} x[n], & \text{provided } n = 0, \pm M, \pm 2M \\ 0, & \text{otherwise} \end{cases} \tag{2.15}$$

Multi-rate techniques, cascaded interpolator comb (CIC) filter, polyphase FIR filter, half-band filter and frequency response masking (FRM) represent possible multi-standards reduction solutions approaches by factor up to, D. The ratio of an oversampled wideband signal's sample rate and the signal of interest bandwidth, determines the factor D, as indicated in Equation (2.16).

$$D \leqslant \frac{f_s}{Bw} \tag{2.16}$$

After A/D conversion, the signal of interest occupies small fragments of the signal bands, and therefore low-pass or bandpass filters can be used to filter out the choice signal at a reduced sampled rate. However, the sampling rate can only be reduced to a limit called the 'Nyquist rate'.

Polyphase decomposition can be used for channelisation of spectral bands with different signal characteristics such as phases and magnitude. The technique is useful for implementing decimation in the FIR filter and filter bank [62, 63]. In multi-rate design, the arrangement of the filter can influence the flow of the signal, as shown in Figure 2.7. The order can be from filtering to down-sampling or vice-versa.

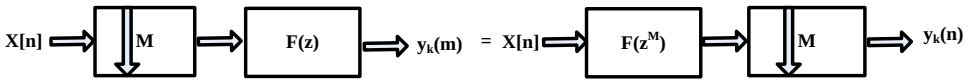

Figure 2.7. Noble identity.

These re-arrangements are called noble relations [64]. For decimation, it follows that Equation (2.17) applies.

$$(\downarrow M)F(z) = F(z^M)(\downarrow M) \tag{2.17}$$

That implies that, if the down-sampling is done first, filter length $F(z^M)$ is produced by a factor of M. In case of interpolation, the noble identity is defined as indicated in Equation (2.18).

$$F(z)(\downarrow M) = (\downarrow M)(F(z^M)) \tag{2.18}$$

In an interpolation, placing the filter before the expander, results in shorter filter of, M-times. Cascading down-sampler of factor M, to FIR Filter structure in polyphase decomposition of FIR filters yields, $y[n]$, output at time instant $y(0)$, $y[M]$, $y[2M]$ etc. This indicates that there is no need to compute the sum of the products of the convolution $x[n]f[n-k]$. All that is required is to multiply $x[0]$ with $f[0], f[M], f[2M]$ etc. The coefficients also need to be multiplied with $x[M], x[2M]$. Wide-band splitting of the input signal into M separate sequences is shown in Equation (2.19).

$$\begin{aligned} X[n] &= \sum_{M=0}^{n-1} X_M[n] \\ X_0[n] &= [X[0], X[M]] \\ X_1[n] &= [X[1], X[M+1]] \\ &\vdots \\ X_{M-1}[n] &= [X[M-1], X[2M-1]] \end{aligned} \tag{2.19}$$

The filter, $f[n]$, can be split into M sequences as shown in Equation (2.20).

$$\begin{aligned} f[n] &= \sum_{k=0}^{M-1} f_m[n] \\ f_0[n] &= f[0], f[M] \\ f_1[n] &= f[1], f[M+1] \end{aligned} \tag{2.20}$$

Given $x[n]$ and M, Equation (2.21) is obtained.

$$\begin{aligned} x[n] &= \sum_{k=0}^{M-1} x_k^{(p)}[n] \\ &= \sum_{k=0}^{M-1} x[n] s_m[n-k] \end{aligned} \tag{2.21}$$

Moreover, decomposing $x[n]$ into M polyphase components is shown in Equation (2.22).

$$\begin{aligned} x(z) &= \sum_{k=0}^{M-1} \sum_{m=-\infty}^{\infty} x[mM+K] z^{-(mM+k)} \\ &= \sum_{k=0}^{M-1} z^{-k} X_k^p(z^m) \\ X_k^p(z^m) &= \sum_{m=0}^{\infty} x[mM+K] z^{-(mM)} \end{aligned} \tag{2.22}$$

The z-transform of the polyphase component is shown in Equation (2.23).

$$\{X_k^p[n] \leftrightarrow z^{-k} X_k^{(p)}(z^m)\} \quad \text{k=0,1,2..., M-1} \tag{2.23}$$

Grouping filter coefficients of the polyphase decomposition with the same delay preserved time and

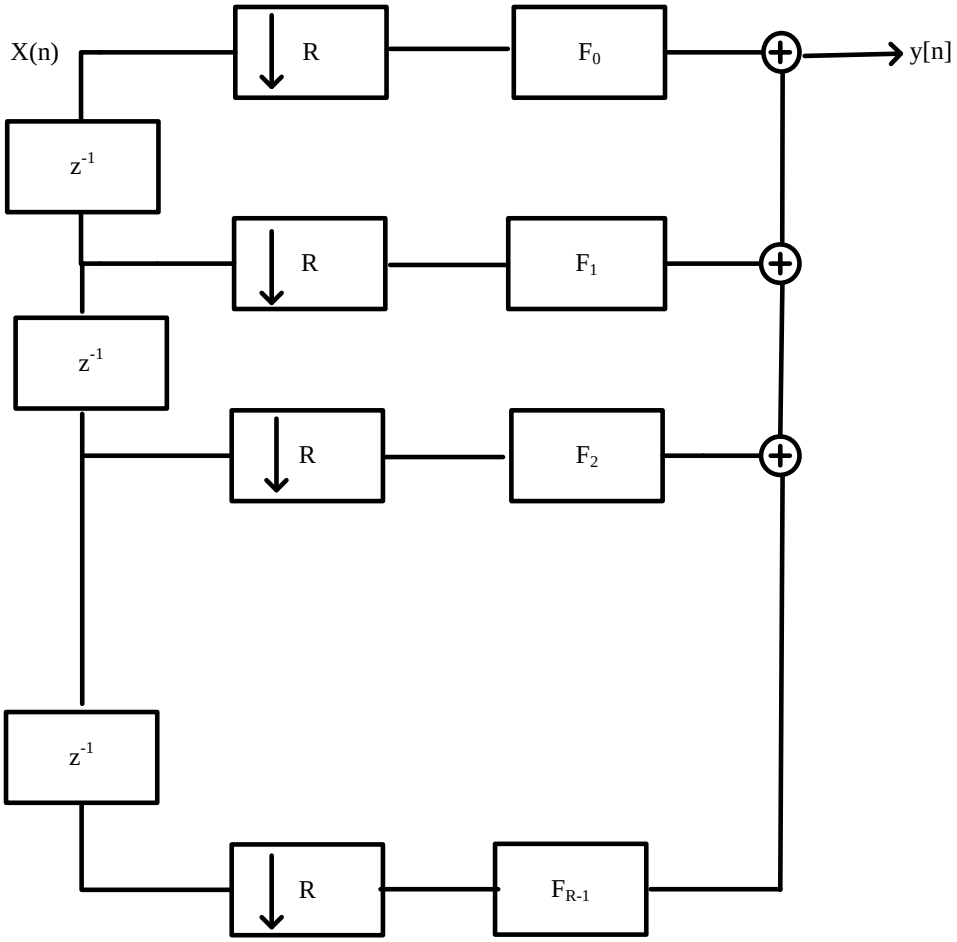

Figure 2.8. Polyphase Decomposition (Modified from [2] with permission).

operations to be carried out [4, 34, 65] and further, results in lower computational effort. For instance,

given two fold polyphase representations of the analysis filter, as indicated in Equation (2.24).

$$\begin{aligned} H(z) &= \sum_{n=-\infty}^{\infty} h(n)z^{-n} \\ &= \sum_{n=-\infty}^{\infty} h(2n)z^{-2n} + \sum_{n=-\infty}^{\infty} h(2n+1)z^{(-2n+1)} \\ &= \underbrace{\sum_{n=-\infty}^{\infty} h(2n)z^{-2n}}_{\text{Even}} + \underbrace{z^{-1} \sum_{n=-\infty}^{\infty} h(2n+1)z^{-2n}}_{\text{Odd}} \end{aligned} \tag{2.24}$$

The filter, $H(z)$, can be grouped or classified into two polyphase components, $E_0(z^2)$ and $E_1(z^2)$. With these grouping methods, the application of polyphase decomposition to PC is made possible. Polyphase representation of the FIR filter is represented in Figure 2.8.

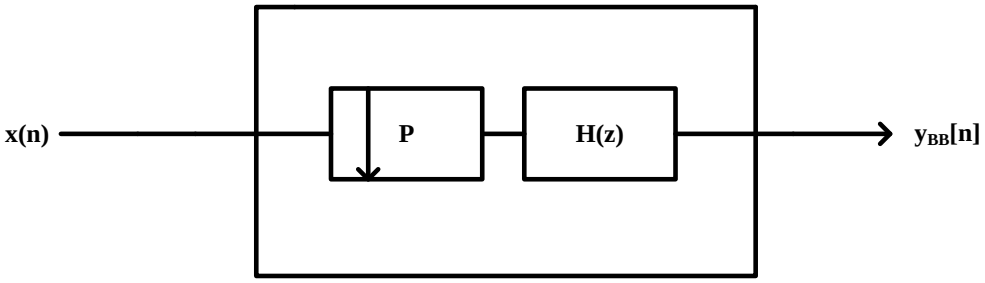

Figure 2.9. Noble identity polyphase decomposition of filter.

The filter coefficient can be categorised into different grouping factors. The even and odd filter coefficients are grouped separately instead of processing them concurrently. This increases the speed of operation, which further reduces the computational load as the number of real multiplication and additions per input for PC channelisation. The MPIS and APIS for PC approach is reduced to Equation (2.25).

$$\begin{aligned} \mu_{PC} &= \frac{N+1}{D} + 4 \\ \alpha_{PC} &= \frac{2N}{D} + 2 \end{aligned} \tag{2.25}$$

Equation (2.25) shows that the order of a filter is proportional to the number of operations and the filter coefficients.

2.4.3 Coefficient Decimation

Coefficient Decimation method (CD-1), was realised for SDR reconfigurable filters. Every M^{th} coefficient of N taps FIR modal filter is decimated [17, 18, 66]. This means that every M^{th} coefficient is retained whereas the remaining coefficients are replaced by zero, to obtain a FIR filter with multi-band frequency responses. The center frequency of the CD-1 is $\frac{2\pi k}{M}$, where k is an integer ranging from 0 to $(M-1)$. CD-1 offers flexible pass bandwidth and passband center frequency. The frequency response is obtained by scaling the coefficients by the variable value of M whose values are even numbers. The stop-band attenuation reduces as M increases whereas the transition bandwidth remains unaltered for any value of M. Modified form of CDM is known as modified coefficient decimation method (MCDM) was proposed to offer solution to variable digital filter [67]. MCDM technique provides higher frequency response flexibility and offers twice the center frequency resolution than the classical coefficient decimation method (CDM). Two coefficient decimation operations can be performed on MCDM. One of the operations is for extracting different multi-band frequency responses (termed MCDM-1) and the other operation offers flexible passband width and passband center frequency of the prototype filter (termed MCDM-11). The combination of coefficient decimation method 1 (CDM-1) and modified coefficient decimation method (MCDM-1) is referred to as an improved coefficient decimation method (ICDM-1) [68] whereas the combination of coefficient decimation method 11 (CDM1-11) and modified coefficient decimation method 11 (MCDM-11) is termed an improved coefficient decimation method (ICDM-11). The main drawback of coefficient decimation is that the passband ripples and stop-band attenuation deteriorates in the frequency responses by factor of D. The number of multipliers consumed by CDM method [66] was 901.

Let $h(n)$ be the original set of coefficients. Replacing all the coefficients, other than every M^{th} coefficient by zero, the newly generated coefficients will be as shown in Equation (2.26).

$$\begin{aligned} &h'(n) = h(n)C_m(n) \\ &C_m(n) = 1 \\ &\text{for n=mM, and m=0,1,2,...m-1} \\ &0, \text{if otherwise} \end{aligned} \tag{2.26}$$

The Fourier series expansion is given as indicated in Equation (2.27), where C_k are complex-valued Fourier series coefficient represented as in Equation (2.28).

$$C_m(n) = \frac{1}{M}\sum_{i=0}^{M-1} C(k)e^{\frac{j2\pi kn}{M}} \tag{2.27}$$

$$C(k) = \sum_{n=0}^{M-1} C_m(n)e^{\frac{-j2\pi kn}{M}} \tag{2.28}$$

Assuming $C(k) = 1$, for all values of k, then Equation (2.29) holds.

$$C_m(n) = \frac{1}{M}\sum_{i=0}^{M-1} e^{\frac{j2\pi kn}{M}} \tag{2.29}$$

The Fourier transform of the modified coefficients, $h(n)$, can be represented as in Equation (2.30).

$$\begin{aligned} H'(e^{jw}) &= \sum_{n=-\infty}^{\infty} h'(n)e^{j\omega n} = \sum_{n=-\infty}^{\infty} h(n)C_m(n)e^{j\omega n} \\ &= \sum_{n=-\infty}^{\infty} h(n)\frac{1}{M}\sum_{i=0}^{M-1} e^{\frac{j2\pi kn}{M}}e^{j\omega n} \end{aligned} \tag{2.30}$$

Interchanging the summation in Equation (2.30), the expression can be re-written as shown in Equation (2.31).

$$\begin{aligned} H'(e^{jw}) &= \frac{1}{M}\sum_{k=0}^{M-1}\sum_{n=-\infty}^{\infty} h(n)e^{-jn(w-\frac{2\pi k}{M})} \\ &= \frac{1}{M}\sum_{k=0}^{M-1}\sum_{n=-\infty}^{\infty} H(e^{-jn(w-\frac{2\pi k}{M})}) \\ &= \frac{1}{M}\sum_{k=0}^{M-1}\sum_{n=-\infty}^{\infty} H(z^{\frac{1}{M}}w_M^{-K}) \end{aligned} \tag{2.31}$$

From Equation (2.31), the frequency response has to be decimated by a factor, M, and the replicas of the frequency spectrum are introduced at integer multiplies of $\frac{2\pi}{M}$. Reconfigurability is achieved by varying the values of M with a given set of filter coefficients such that different multi-band frequency responses are obtained.

2.4.4 Frequency Response Masking

This specific filter applies interpolation for the design of the wideband sharp cut off filters. It consists of two branches, namely: the base and the complementary branches with the base, $H_{af}(z)$, and

complementary frequency responses, $H_{cf}(z)$ [3]. When both are interpolated by a factor, L, the passband and transition bandwidth of their frequency responses get reduced by the same factor. The bandpass and the highpass replicas of the interpolated frequency responses appear at frequencies which are multiples of $\frac{2\pi}{L}$ rads. The function of the masking filter, $H_{maf}(z)$, and the complementary masking filter, $H_{mcf}(z)$, is to filter the interpolated frequency responses of the base and complementary filters so that only few of the replicas remains. Further complexity reduction were seen when frequency-response masking (FRM) and non-maximally decimated filter bank (NMDUFB) were used [17].

The transfer function for the FRM filter is represented as shown in Equation (2.32), where $H_{af}(z^L)$ and $H_{cf}(z^L)$ are the base and complementary filters, whereas $H_{maf}(z)$ and $H_{mcf}(z)$ are the positive and complementary masking filters respectively.

$$H(z) = H_{af}(z^L)H_{maf}(z) + H_{cf}(z^L)H_{mcf}(z) \tag{2.32}$$

There is a complementary relationship between the base filter, $H_{af}(z^L)$ and complementary filter, $H_{cf}(z^L)$. This can be expressed as shown in the Equation (2.33), where N_A is the base filter order.

$$H_{cf}(z^L) = z^{-\frac{N_A}{2}} - H_{af}(z^L) \tag{2.33}$$

This implies that the complementary filter can be obtained by subtracting the $H_{af}(z)$ from the delayed version of the input signal. The block diagram of frequency masking filter is shown in Figure 2.10.

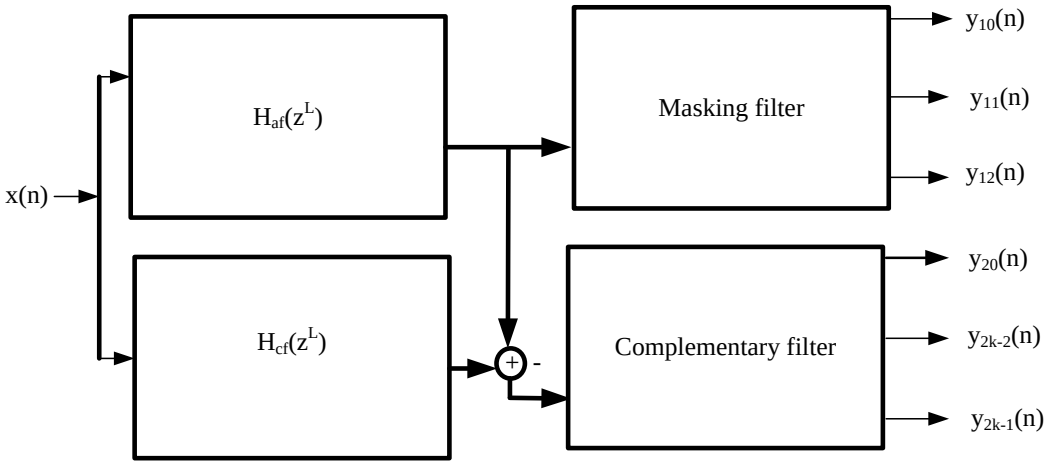

Figure 2.10. Block diagram of Frequency response masking filter (Modified from [3] with permission).

When narrowband signal is desired, the interpolation and masking filters of the positive FRM are sufficient. The transition band is purely given by the first image of the interpolated base filter frequency response. The transfer function for Equation (2.32) will then be reduced to Equation (2.34).

$$H(z) = H_{af}(z^L)H_{maf}(z) \tag{2.34}$$

The different passband and stopband of the base and masking filter can be determined once the desired passband, (ω_p), and stopband, (ω_s), cut-off frequencies are given using Table 2.2.

Table 2.2. The cut-off frequency of prototype, masking and complementary filter.

Parameter	Case 1	Case2	narrowband FRM
$H_a(z)$	$\theta_a = 2m\pi - \omega_s L$	$\theta_a = \omega_s L - 2\pi m$	$\theta = \omega_p L$
	$\phi_a = 2m\pi - \omega_p L$	$\phi_a = \omega_p L - 2m\pi$	$\theta = \omega_p L$
$H_{ma(z)}$	$\omega_{mp}=\frac{2\pi(m+1)-\phi_a}{L}$	$\omega_{mp}=\frac{2\pi m-\phi_a}{L}$	
	$\omega_{ms}=\frac{2\pi m+\phi_a}{L}$	$\omega_{ms}=\frac{2\pi m-\phi_a}{L}$	
$H_{mc}(z)$	$\omega_{mcp}=\frac{2\pi)(m+1)+\phi_a}{L}$	$\omega_{mcp}=\frac{2\pi m-\phi_a}{L}$	
	$\omega_{mcs}=\frac{2\pi m-\phi_a}{L}$	$\omega_{mcs}=\frac{2\pi m-\phi_a}{L}$	

The full FRM design in Figure 2.11 utilises two cases. In case 1, the final filter transition band is given by the base filter interpolated replicas, whereas in the case 2, the final filter is given by the complementary filter replicas. For narrowband FRM, only the positive branch is used and only one design method is used.

For case 1 and case 2, the values of, m, is given by Equation (2.35).

$$\begin{aligned} case1 : m &= \lfloor \tfrac{\omega_{si}}{2\pi} \rfloor \\ case2 : m &= \lceil \tfrac{\omega_{si}}{2\pi} \rceil \end{aligned} \tag{2.35}$$

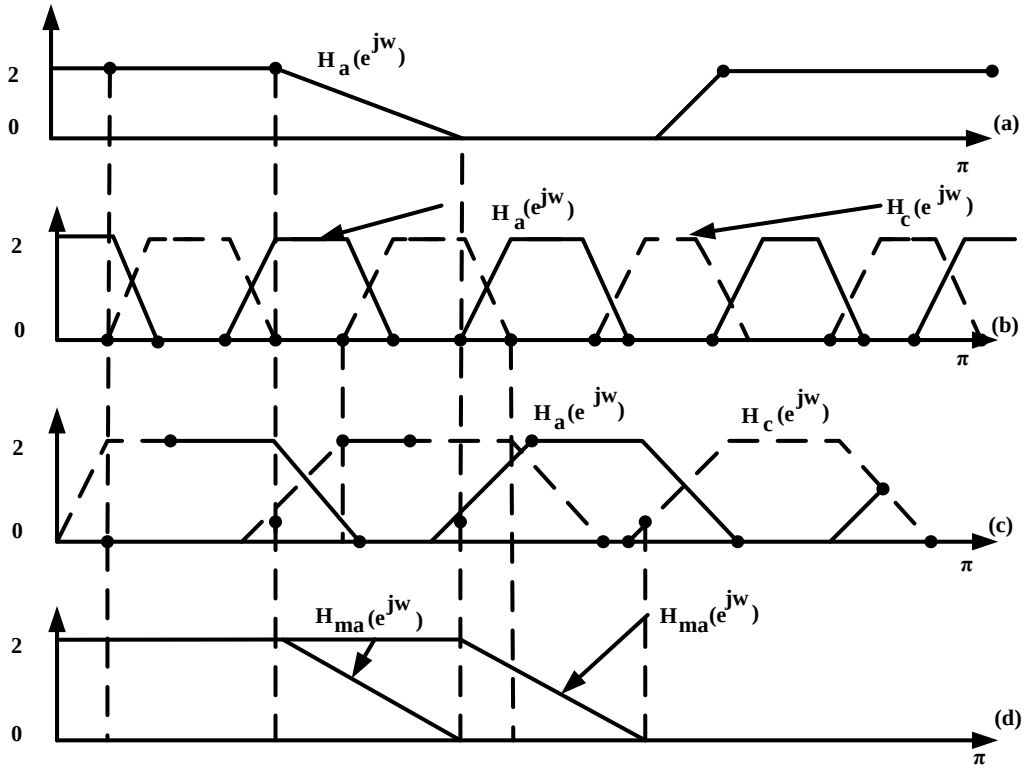

Figure 2.11. Frequency response masking filter.

2.4.5 Farrow Channelisation

Farrow filter is a variable digital filter with adjustable control parameters. This control parameter may be adjustable delay that changes arbitrarily. Farrow filter is used in multi-standard receiver channelisation algorithms and for sample rate conversion. It performs best at low frequencies but with significant performance degradation at higher frequency bands. The magnitude response of the Farrow structure is flat at low frequencies only which limits its adaptability to other frequency bands. However, introducing adaptive rational sample rate converter (SRC) and filtering at every branch of the channelizer allow different sub bands signals to be isolated, but with signals distortion. Approximating such signals increase the computational capability of the filter. Different attempts to reduce complexity and improve performances of Farrow filter have been proposed. A modified Farrow structure [69] has proven efficient for implementing rational sampling rate conversion by reducing number of operators in the Farrow structure. A transposed modified Farrow structure [70, 71] was proposed for reducing complexity in Farrow filter by transposing the modified filter and thereby reduces the number of operators. Using transposed modified Farrow structure, major advantages were lower

sample conversion and reduced computational complexity [70, 71]. A modulated generalised discrete Fourier transform (GDFT) filter bank was proposed to reduce the complexity of Farrow filter [72]. The filter operates on series of frequency shifted oversampled sub band signals. The approach attained -55 dB in reconstruction error.

In another work, a low complexity structure with sample rate converter (SRC) using Farrow sub-filters together with frequency response masking filter was developed to reduce complexity of Farrow filter. The number of multipliers used was 291 which implies ten times lower in complexity when compared with the original Farrow filter. Fractional sample delayer [73] was developed to improve the performance of Farrow filter by offering high precision beam steering at the baseband sampling frequency. The filter has very flat magnitude response greater than 20-bit resolution for a 35-tap finite impulse response filter (FIR). Farrow interpolation [74] was designed for converting sample rate by a fractional or rational factor. The number of coefficients utilised were moderate. A two dimensional prototype filter [75] was designed to improve the complexity of Farrow filter with its coefficients being stored in memory. From these stored coefficients , a filter coefficient for a $1-D$ filter is calculated on-the-fly for desired cut-off frequency. This leads to precise frequency response although the computational complexity is too high with filter order of 890. Variable cut-off frequency (VCF) was used to control the cut off frequency of Farrow filter in order to reduce the computational load [76]. VCF filter with discrete control over frequency uses variable coefficient filter, coefficient decimation based filter (CDM) and frequency response masking based filter approaches. The main drawback of coefficient decimation is that the passband ripples and stop-band attenuation deteriorate in the frequency responses by factor of D. The number of multipliers consumed by CDM method [66] was 901. Lagrange Interpolation method [77] was used for fractional delay approximation in Farrow filter. The method increases the sampling rate of signals while the computational complexities are reduced to minimal. Also, low complexity 3-level filter bank [78] was developed for generating array of sub-filters from single finite impulse response filter using fractional interpolation technique. The method generates arrays of sub-bands from prototype filter with the number of sub filter multipliers used being 28.

2.4.6 Farrow Per Channel Channelisation (FPCC)

Conventionally, channelisation operates on a per channel architecture using a dedicated circuit. That means, many branches are employed in parallel for channelising many channels. Each branch frequency shifted the required channel to DC and then filtered it before down converting to baseband. This process is cumbersome and highly inflexible. Therefore, the Farrow structure is introduced to allow different

channels to be processed on the same processing blocks using rational sample rate converter (SRC) and filtering. Each branch is designed to extract multi-channels input signals by down conversion using a variable frequency mixer and filtering as indicated in Figure 2.12. However, the total number of channels processed in parallel is too high, resulting in very high computational complexities and very low reconfigurabilities.

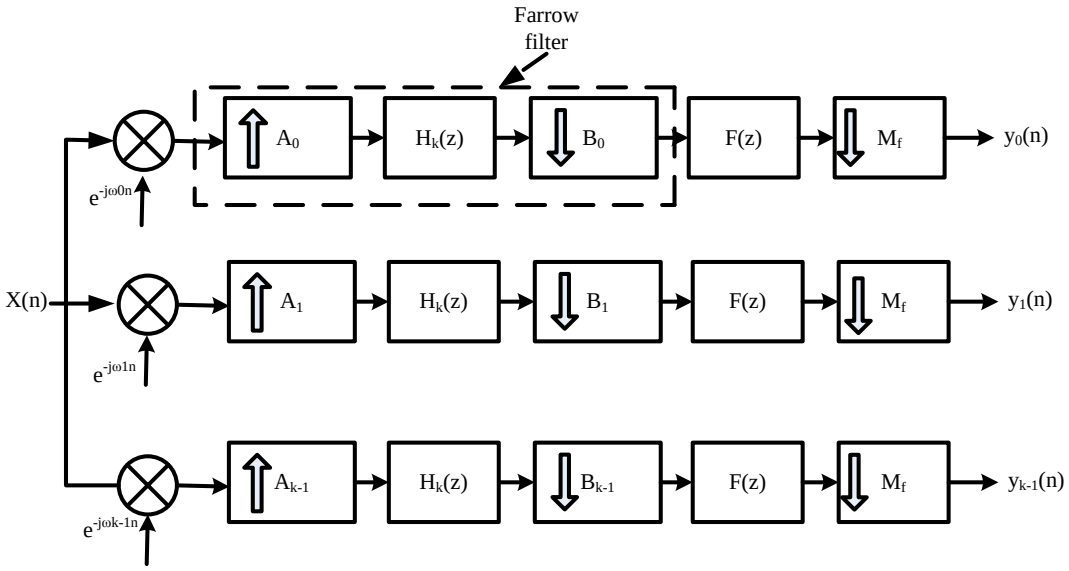

Figure 2.12. Farrow per channel channelisation Algorithm (Modified from [1] with permission).

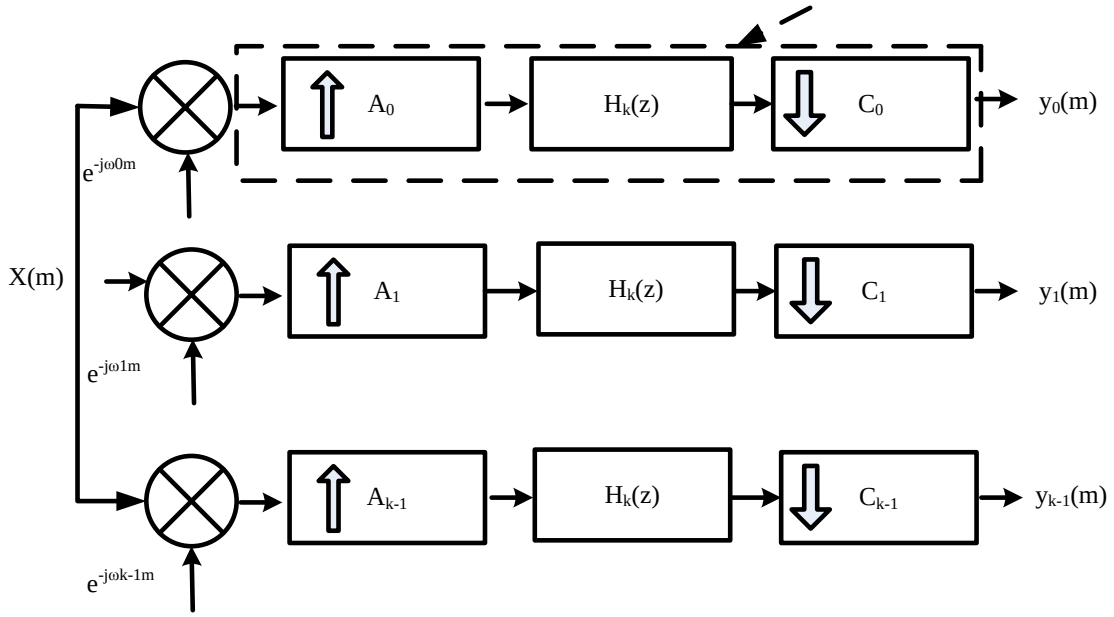

Figure 2.13. Efficient realisation of Farrow per channel channelisation Algorithm (Modified from [1] with permission).

Any upgrade to new communication standard requires update in the coefficient of Farrow filter. Each k branch of the channeliser is designed to extract each of the input signal change by down conversion using variable mixer and filtering of the Farrow filter. The Farrow filter is formed from the filter, $H_k(z)$, and sample rate converter (SRC). This indicated that any branch can extract any of the j types of communication from multiple channels input signals. The Farrow filter is followed by fixed integer decimation factor, M_f, as shown in Figure 2.12. By replacing the filter, $F(z)$, and M_f, the Farrow is left alone to complete the SRC in each branch as seen in Figure 2.13. The computational load is constrained by the large number of channels that should be handled. Since it does not share any combinational load among the branches, its computational load grows linearly with the number of channels required.

2.4.7 Discrete Fourier Transform Algorithm

The DFT filter bank (DFT-FB) is a uniform modulated filter bank in replacement of the PC algorithm, for extracting large number of channels with uniform bandwidth. Implementation with polyphase decomposition enhances its performance. Low pass filter prototype and complex modulation of the bandpass filters produced DFT FB [22, 62, 79], as shown in Figure 2.14.

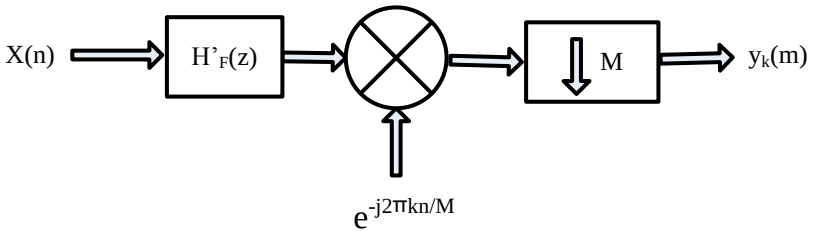

Figure 2.14. Discrete Fourier Transform (Modified from [1] with permission).

The design of low-pass filter is done using the filter specifications while the remaining bandpass filters automatically inherit the same properties after their modulations from the low-pass prototype. This type of filter requires the implementation of one filter and one DFT matrix [1, 4, 21, 27, 80].

$$y_k(m) = h(n)^{'} e^{-j2k\pi\frac{n}{m}} \tag{2.36}$$

By critically sampling the bandpass signal, polyphase decomposition of the filter, $h^{'}[n]$, and the input signal, $x[n]$, with R polyphase signal is obtained, according to Equation (2.36).

$$h^{'}[n] = \sum_{k=0}^{R-1} h_k^{'}[n] \leftrightarrow h_k[m] = h^{'}[mR-k] \tag{2.37}$$

$$x[n] = \sum_{k=0}^{R-1} x_k[n] \leftrightarrow x_k[m] = x[mR-k] \tag{2.38}$$

Substituting Equation (2.37) into Equation (2.38), the bandpass filters, $h^{r}[m]$, share the same polyphase filter, $h_k^{'}[n]$, although with different twiddle factors for each filter. The twiddle multiplication for $h^{'r}[n]$ corresponds to the r^{th} DFT component with input vector $x_0[n]$, $x_1[n]$, ..., $x_{r-1}[n]$ are represented in Figure 2.15.

The DFT component can be R polyphase filters, followed by the critically or oversampled DFT or FFT components. It was proven that DFT modulated filter-bank with down-sampling can be realised

as a tapped delay line of size N, followed by $M \times N$ polynomial matrix $B(z)$ containing polyphase components of the prototype h_0' and finally an $M \times N$ DFT matrix.

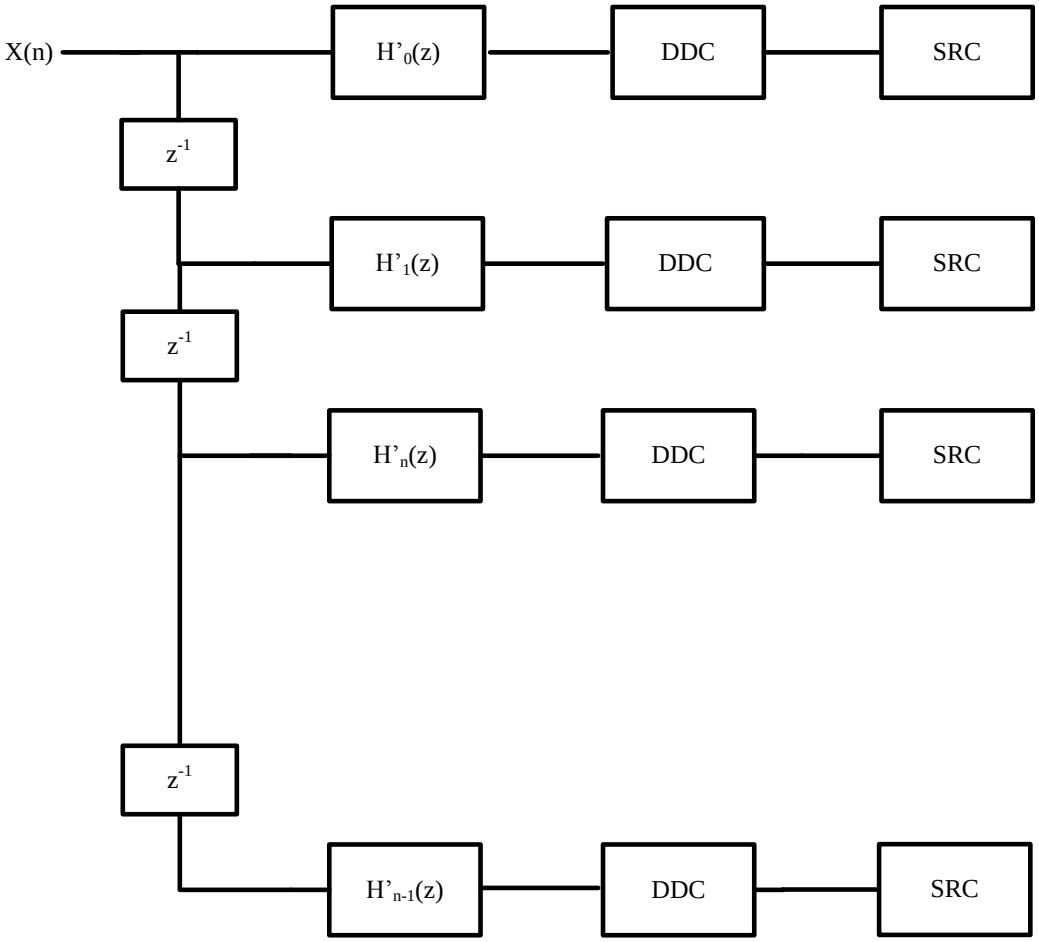

Figure 2.15. DFT channelisation Algorithm (Modified from [4] with permission).

DFT modulates the analysis filter bank with, M, sub-bands, followed by N-fold down-samples. By employing the FFT techniques and polyphase decomposition, $B(z)$, an efficient implementation is possible. The frequency domain representation is shown in Equation (2.39), where W_k^{-kp} is the DFT matrix, whose elements are the polyphase branch number, p, and channel, k.

$$H_k^{'}(z) = \sum_{k=0}^{R-1} h_k^{'}(z^R) z^{-k} W_k^{-kp} \qquad (2.39)$$
$$p = 0,1,..., k\text{-}1$$

Polyphase implementation of DFT has proven efficient in computational complexity reduction at the receiver stage because of its ability to computationally share resources among all the receivers/analysis filters. This results in decrements in computation load, by factor of K. The implementation of polyphase decomposition brought remarkable conservation of computation resources such as the filter and memory, by the down-sampling operation before filtering.

The modulation operation using the DFT algorithm requires, k^2, complex multiplication since both DFT coefficients and input signals sample are complex numbers [1, 81]. Therefore, the computational load of the DFT algorithm increases as the number of channels increases. The choice of implementing DFT begins with the selection of a short DFT algorithm. The short DFT is used to develop long DFTs using schemes provided by the Cooley-Turkey algorithm, Good Thomas or Winograd algorithm [4, 21, 81]. From these algorithms, the most important criteria for choosing the DFT implementation algorithm is minimum multiplication complexity. Table 2.3 shows the different multiplication complexities of different DFT algorithms.

Table 2.3. Multiplication complexities of DFT Algorithms.

DFT method	Good-Thomas	Cooley-Turkey	Winograd
Direct	$X.12^2 = X.144$	$X(43+6)$	0
RPFA	$X(3(x-1)^2)+X(3-1)^2$	$2.3.X$	0
WFTA	$3.0.2+X.2.2$	$16+X.6$	0

It was shown that the Winograd FFT is the most attractive algorithm, based purely on multiplication complexity criteria. A reduction in the number of DFT operations can be achieved by choosing, D, and, K, as the power of two, using the index or split radix [12]. Important properties of FFT algorithms of length $N = \Pi N_k$ are shown in Table 2.4.

Table 2.4. Comparison of different DFT Algorithms.

Property	Cooley-Turkey	Good-Thomas	Winograd
Any Transform Length	Yes	No	No
Maximum order of W	No	Max(N_k)	Max(N_k)
Twiddle factors tested	Yes	No	No
Multiplication	Bad	Fair	Best
Addition Number	Fair	Fair	Fair
Index Computation Effort	Best	Fair	Poor
Data in Place	Yes	Yes	Yes

2.4.8 Quadrature Mirror Filter Algorithm

Quadrature Mirror Filter has been of interest to researchers since its introduction by Crosier, Esteband and Galand. The algorithm involves splitting the consecutive frequency spectral bands into sub-bands such that each sub-band can be processed independently. However, at some points, the sub- band signals must be recombined to achieve for perfect reconstruction [28].

Spectral band is split into two sub-bands: one of the sub-bands represents the low-pass filter and the other sub-band is the high-pass filter [27, 28]. Each filter is proceeded by a half-band decimating filter. High-pass signal, $x_2(n)$, and low-pass signal, $x_1(n)$, are decimated by a factor of 2. At the receiver side, the decoded signals passed through an interpolator. The decimator-interpolator cascade causes aliasing and imaging.

The overlapping analysis filter at the output of decimator permits and reduces the effect of aliasing and chooses the synthesis filter such that the imaging produced by the interpolator cancels the aliasing [1]. The high-pass filter can be derived from the low-pass filter, using Equation (2.40) and Equation (2.41).

$$H_H(z) = H_L(-z) \tag{2.40}$$

$$H_H[n] = (-1)^n H_L(-z)[n] \tag{2.41}$$

Equation (2.41) implies that the sign magnitude differentiates polyphase low-pass filters from the high-pass filters. However, both the low-pass and high-pass filters share the same poly phase components.

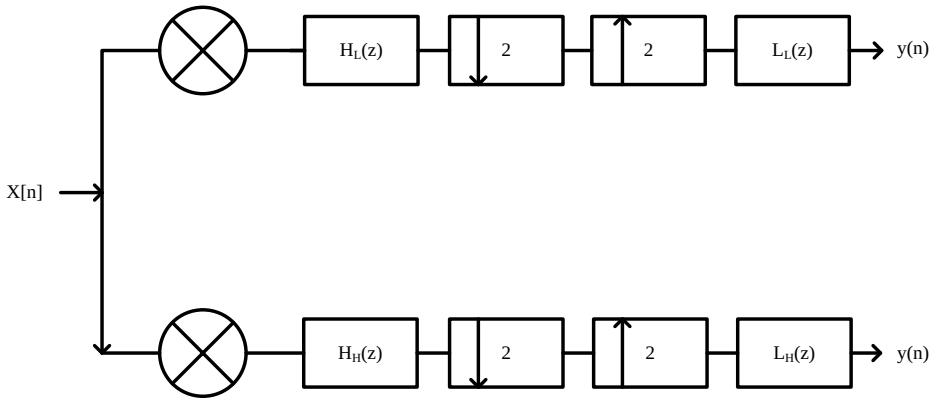

Figure 2.16. Block diagram of QMF.

$$H_L(z) = E_0(z^2) + z^{-1}E(z^2)$$
$$H_H(z) = E_0(z^2) - z^{-1}E(z^2) \tag{2.42}$$

It is a common practice to define the low pass filter, $X_L(z)$, and to use its definition to specify the high-pass filter as indicated in Equation (2.41). Figure 2.16 shows the block diagram of filters that are composed of complementary low-pass and high-pass half filters [21]. They have symmetrical impulse responses and odd coefficients, with the exception of the one placed at DC on, n, that is equal to zero. In the frequency domain, this translates to Equation (2.43).

$$|H(e^{j\omega})| = |H(e^{j(\omega-\pi)})| \tag{2.43}$$

This means that the low-pass and high-pass filter are magnitude complements of each other because the two filters have mirror symmetry, centred on $\frac{\pi}{2}$. The H_L , H_H, L_H and L_L represent the low-pass and high-pass filters for two input bands. Given $H_L(e^{j\omega})$, $H_H(e^{j\omega})$, $L_H(e^{j\omega})$ and $L_L(e^{j\omega})$ as the Fourier transform of H_L , H_H, G_H and L_L, Equation (2.44) is derived.

$$L_L(e^{j\omega}) = \frac{1}{2}\left[\frac{X(e^{j\omega})}{2}\frac{H_L(e^{j\omega})}{2} + \frac{X(e^{j\omega+2\pi})}{2}\frac{H_L(e^{j\omega+2\pi})}{2}\right]$$
$$L_H(e^{j\omega}) = \frac{1}{2}\left[\frac{X(e^{j\omega})}{2}\frac{H_H(e^{j\omega})}{2} + \frac{X(e^{j\omega+2\pi})}{2}\frac{H_H(e^{j\omega+2\pi})}{2}\right] \tag{2.44}$$

The first requirement is that the low-pass and high-pass filters, H_L and H_H, are frequency translation of the common low-pass filter denoted as $h(n)$, as shown in Equation (2.45).

$$\begin{aligned} &H_L(n) = h(n) \\ &H_H(n) = (-1)^n h(n) \\ &\text{for all n} \end{aligned} \tag{2.45}$$

The frequency samples are such that $H_L(e^{j\omega})$= $H(e^{j\omega})$ and $H_H(e^{j\omega})$ = $H(e^{j\omega})$. This condition implies that the filters $H_L(e^{j\omega})$ and $H_H(e^{j\omega})$ are mirror image symmetries about $\omega = \frac{\pi}{2}$.

An efficient realisation of QMF is by using polyphase decomposition where the low-pass or high-pass filters are expressed in terms of two polyphase components, as given in Equation (2.46).

$$G_L(z) = \sum_{n=-\infty}^{\infty} h_k(n)x(2m-n)$$
$$\text{k=0, 1 ,..., N} \tag{2.46}$$

By substituting Equation (2.46) into Equation (2.45), Equation (2.47), is derived for low-pass filter.

$$L_L(z) = \sum_{n=-\infty}^{\infty} (-1)^{kn} h(n)x(2m-n)$$
$$\text{k=0, 1... , N} \tag{2.47}$$

Changing the variable parameters, Equation (2.48) is obtained, where $p = 0, 1$.

$$\begin{aligned} &L_L(m) = \sum_{p=0}^{n=2r+p}\sum_{r=0}^{\infty}(-1)^{k(2r+p)}h(2r+p)x(2(m-r)-p) \\ &P_p(m) = h(2m+p) \\ &x_p(m) = x(2m-p) \end{aligned} \tag{2.48}$$

When $p = 0$, it means it is associated with even samples of $h(n)$ and $x(n)$ and when $p = 1$, it means it is associated with odd samples. For N (even) tap filter with coefficients in the range $0 \leqslant n \leqslant N-1$, the polyphase filter $P_p(m)$ each has $\frac{N}{2}$ coefficients in the range $0 \leqslant n \leqslant \frac{N}{2} - 1$.

Applying these polyphase definition to Equation (2.48), Equation (2.49) is derived, where $*$ denotes discrete convolution.

$$\begin{aligned} L_L(m) &= \sum_{p=0}^{1} \sum_{r=0}^{(\frac{N}{2}-1)} (-1)^{kp} P_p(r) x_p(m-r) \\ &= P_0(m) * x_0(m) + (-1)^k P_1(m) * x_1(m) \\ \mathrm{x} &= 0, 1, ..., N \end{aligned} \tag{2.49}$$

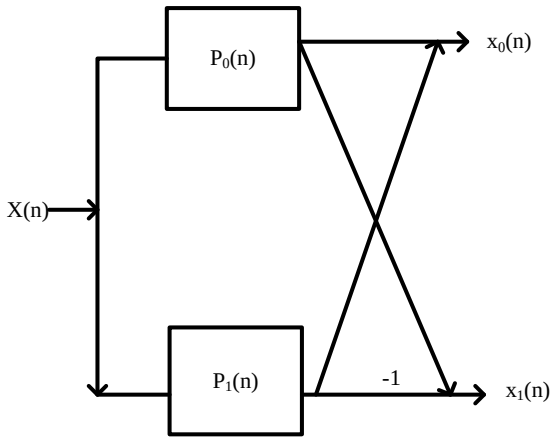

Figure 2.17. Block diagram of two-channel QMF filter bank.

In order for the filter bank to have the flat response, its polyphase filter must satisfy the following conditions, as shown in Equation (2.50).

$$P_0(m) * p_1(m) = U_0(m) = \begin{cases} 1, & \text{provided n =0} \\ 0, & \text{otherwise} \end{cases} \tag{2.50}$$

For even-length symmetrical filters, $P_0(m)$ and $p_1(m)$ are time-reversed versions of each other that is, $P_0(m) = p_1\left(\frac{N}{2} - 1\right)$; and $P_0(m) = p_1\left(\frac{N}{2} - 3\right)$. Therefore, they have exactly inverse phase relations

and equal magnitude responses. Thus, for an effective impulse response, all pass filters must be used. To achieve the filter bank with the flat response, these polyphase filters must approximate all-pass functions. Also, the even input samples occur at an odd sample input time, as a result of inherent delay of one sample, in addition to the delay of the filter, whereas, DFT polyphase structure defined for N odd has no such inherent delay.

For perfect reconstruction of QMF filter banks, phase and amplitude distortion must be zero, since it does not exist. The filter bank can be made aliases-free by choosing the synthesis filters, as given in Equation (2.51).

$$\begin{aligned} L_H(z) &= -2H_L(-z) \\ L_L(z) &= -2H_H(-z) \end{aligned} \tag{2.51}$$

The transfer function of the aliases-free QMF is given in Equation (2.52).

$$\begin{aligned} T(z) &= \frac{1}{2}\left[H_L^2(z) - H_H^2(z)\right] \\ &= \frac{1}{2}\left[H_L^2(z)] - H_L^2(z)\right] \end{aligned} \tag{2.52}$$

A low-pass filter, $H_L(z)$, with linear phase must be designed in such a way that the overall transfer function, $T(z)$, will also have a linear phase, which results in eliminating the phase distortion. By carefully selecting, $H_L(z)$, with good pass-band and stop-band responses, the amplitude distortion can be eliminated resulting in near-perfect reconstruction [82]. The large number of channels involved increase the computational complexity of the filter and any update to new filter channel is considered bad.

2.4.9 Tree QMF

Two QMFs cascaded together result in a tree-structured QMF filter banks. The input signal is split into two-channels sub-bands at the initial stage and decimated by the half-band filter, using two channel analysis banks. Each sub-band is again sub-divided into two other sub-bands and further decimated by the half-band multi-stage decimator filter [15, 16, 29, 83]. This process continues until the required number of channels are obtained.

The M-channels tree structure filter bank with decimator factor, M_1, M_2, ..., M_{m-1}, is said to be maximally decimated if the condition in Equation (2.53) is satisfied [82].

$$\sum_{k=0}^{M} \frac{1}{M_k} = 1 \tag{2.53}$$

The computation in this filter bank structure can be accomplished in an eight-cycle process, since the maximum decimation ratio in the structure is eight. In each cycle, for one input sample, $x(n)$, obtained, one branch of the tree is computed and one output-band is computed. Paths through the tree with higher sampling rates are computed more often than paths with lower sampling rates. The output of the filters are in this order: $(0,3,2,4,1,3,2,4)$ for cycle 0 to 7. Bands 0 and 1 are computed only once in this process, whereas bands 2, 3 and 4 are computed twice in each eight cycle iteration. In this way, approximately one eighth cycle filter is computed in each cycle and the computing is distributed in time. This approach is effective for achieving both uniform and non-uniform channelisation algorithms. The performance of TQMF depends on the initial performance of the two-channel QMF. If the two-channel QMF banks achieved maximal PR, the TQMF structure will have linear phase and zero aliasing [27, 79].

Perfect reconstruction is practically feasible in a two channels QMF with sharp cut-off frequency and good stop-band characteristics [27] but Near Perfect Reconstruction (NPR) QMF banks [84] can be achieved. These filter banks have small aliasing and can therefore be made aliased-free by inserting appropriate transfer functions instead of pure delays. The block diagram of the five-channels tree structure filter bank with decimator factors [28, 85, 84] and its parallel structural equivalent, are shown in Figure 2.18.

To obtain PR or NPR filter banks, sufficient delay must be inserted in the channels, such as z^{-k}, z^{-k2}, z^{-k3}. The equivalent filters are obtained by applying noble identities [27]. For five-channels filters, the equivalent channel tree structure filter banks are obtained as follows, in Equation (2.54).

$$
\begin{aligned}
H_0(z) &= H_L(z)H_L(z^2)H_L(z^4)H_L(z^8) \\
H_1(z) &= H_L(z)H_L(z^2)H_L(z^4)H_H(z^8) \\
H_2(z) &= H_L(z)H_L(z^2)H_H(z^4) \\
H_3(z) &= H_L(z)H_H(z^2) \\
H_4(z) &= H_H(z) \\
G_0(z) &= G_L(z)G_L(z^2)G_L(z^4)G_L(z^8) \\
G_1(z) &= G_L(z)G_L(z^2)G_L(z^4)G_H(z^8) \\
G_2(z) &= G_L(z)G_L(z^2)G_H(z^4) \\
G_3(z) &= G_L(z)G_H(z^2) \\
G_4(z) &= G_H(z)
\end{aligned}
\tag{2.54}
$$

For tree structure, the amplitude distortion can theoretically be made zero, using Equation (2.55), where $H_k(\varepsilon^{jw})$ is the frequency response of the K^{th} analysis filter of the equivalent structure.

$$
\sum_{K=0}^{M-1} |H_k(\varepsilon^{jw})|^2 = 1 \tag{2.55}
$$

For five channels tree structure filter banks, Equation (2.54) can be expanded in z-domain.

$$
\begin{aligned}
&|H_L(z)H_L(z^2)H_L(z^4)H_L(z^8)|^2 \\
&+|H_L(z)H_L(z^2)H_H(z^4)|^2 \\
&+|H_L(z)H_H(z^2)|^2 \\
&+|H_H(z)|^2 = 1
\end{aligned}
\tag{2.56}
$$

From Equation (2.55), the high-pass filter can be obtained from the low-pass filter using Equation (2.58).

$$
\begin{aligned}
|H_L(z^8)|^2 = |H_L(z^4)|^2 = |H_L(z^2)|^2 = |H_L(z)|^2 \\
|H_H(z^8)|^2 = |H_H(z^4)|^2 = |H_H(z^2)|^2 = |H_H(z)|^2
\end{aligned}
\tag{2.57}
$$

$$
H_H(z) = H_L(-z) \tag{2.58}
$$

The tree QMF Equation is derived as shown in Equation (2.59).

$$\begin{aligned}&|H_L(z)|^8+|H_L(z)|^6.|H_L(-z)|^2\\&+|H_L(z)|^4.|H_L(-z)|^2\\&+|H_L(z)|^2.|H_L(-z)|^2\\&+|H_L(-z)|^2=1\end{aligned} \tag{2.59}$$

The frequency domain can be represented as seen in Equation (2.60).

$$\begin{aligned}&|H_L(e^{j\omega})|^8\\&+|H_L(e^{j\omega})|^6.|H_L(-e^{j\omega})|^2\\&+|H_L(e^{j\omega})|^4.|H_L(-e^{j\omega})|^2\\&+|H_L(e^{j\omega})|^2.|H_L(-e^{j\omega})|^2\\&+|H_L(-e^{j\omega})|^2=1\end{aligned} \tag{2.60}$$

QMF is a parallel combination of high-pass filter, (HPF) and low-pass filter, (LPF), which performs the action of frequency sub-division by splitting the signal spectrum into two spectrum bands. The analysis wide-band frequency is sub-divided into main sub-bands, using FIR filters. The analysis bank is composed of the collection of M-band filters ("analysis filter", "decimator filter"), with the common input signal. A block diagram of the five-channels tree structure filter bank, with decimation factors (16, 8, 4, 2) is shown in Figure 2.18. All branches of the tree in the same stage will have the same bandwidth and the same sample rate. Despite its simplicity and scalability, TQMF has its dependency between the channel spacing obtained at every tree stage and sample rate of the input signal. This increases the computational complexity, but the reconfigurability is straightforward. The channel spacing is equal to half the sampling rate divided by the power of two for each stage used [1]. Equation (2.61) shows the representation, where t is the stage of each filter.

$$\begin{aligned}&f_{ch,n}=\tfrac{f_s}{2^t}\\&\text{t= 1,2,3,..., n}\end{aligned} \tag{2.61}$$

The sampling frequency of the channel signals are multiple of channel spacing, which is the power of two stages. The computational load of TQMF can be reduced by implementing parallel structures of TQMF. This design can be implemented in terms of parallel integer band structure, especially for low order designs, due to its structural simplicity.

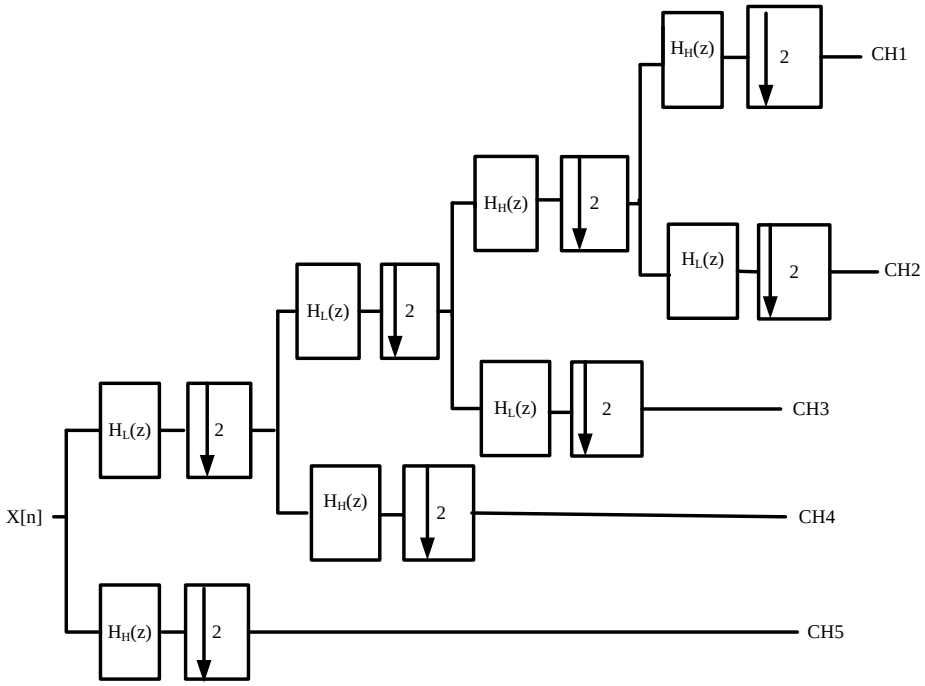

Figure 2.18. Tree structure filter bank (Modified from [1] with permission).

2.4.10 Generalised Discrete Fourier Transform Algorithm

The DFT filter bank is efficient for even uniform channel stacking. The need arises for modifications or upgrading to any frequency, which may be either odd or even stacking. GDFT is essential for both the uniform as well as non-uniform channels bandwidth and it is formed from complex modulation of different filter banks with a different phase and frequency offset [7, 6, 22]. The generalized discrete Fourier transform filter bank (GDFT- FB) is a variant of discrete Fourier transform (DFT), and it is useful in signal processing [86] and image compression [87]. It utilises the properties of the prototype filter poly-phase decomposition and noble identities [22] for efficient implementation. GDFT-filter bank adds extra phase modifications to the time and frequency offsets, resulting in a complex filter. This has been known to lead to computational load due to higher sampling rate utilised [6]. Towards reducing complexities in the mother GDFT filter, different algorithms have been developed. Some of these algorithms were built upon the classical GDFT while some were combined with other algorithms in order to improve the performances of GDFT filter bank and to reduce its complexities.

Parallel GDFT-FB (P-GDFT) processes the wide-band input signal by critically decimating odd-stacked GDFT-FBs, operating in parallel. The recombined oversampled GDFT-filter bank (R-GDFT) splits

the signal into uniformly spaced sub-bands and then recombine certain required groups to form wider sub-bands, which are integer multiples of the uniform channel spacing. Every recombined signal is formed by r-continuous sub-bands allocated from output of the GDFT-FB [6, 83]. Experiment carried out by [6] showed that P-GDFT filter reduces complexities by 85% multiplication per input samples while R-GDFT filter bank reduces the complexity by 67% multiplication per input samples [3].

Split radix GDFT filter [88] provides computational savings in terms of arithmetic operators to compute the length of N-GDFT filter. Oversampled polyphase GDFT filter bank [89] was proposed as fast Fourier algorithm for implementing complex valued signals. GDFT with Morlet wavelet and continuous wavelet transform [90] were proposed to improve the running speed of the classical GDFT filter.

Generalized orthogonal discrete wavelet transform (GODWT) filter [91] was designed as substitute to GDFT filter. GODWT filter bank inherits the features of discrete hatley transform (DHT) and discrete wavelet transform (DWT). The algorithm is fast because it requires real sequence unlike GDFT filter which requries complex signal transformation. Fast Fourier transform (FFT) [92] was used to implement faster computation on GDFT by converting the linear convolution to circular convolution. Exponential improvement of GDFT results were obtained.

Real generalized discrete Fourier transform algorithm [93] requires less memory space for its computation than the imaginary counterparts. One dimensional generalized discrete Fourier transform [94] was developed by factoring a vector A into two dimensional matrix. The two dimensional GDFT on matrix B require the same number of operators as one dimensional matrix. Also, weighted approximation method for generalized discrete Fourier Jacobi transform [95] was developed for reducing the complexity of GDFT filter. Two dimensional generalized discrete Fourier transform and related quasi-cyclic reed Solomon code were also developed to optimize GDFT filter. A generalized sliding discrete Fourier transform algorithm [96] for reducing the peak to average power ratio of filter bank multicarrier was developed also. The GDFT sliding was able to reduce the complexity of the signals by 40%. Elliptically generalized DFT [97] uses Jacobi elliptical functions as basis function instead of sine and cosine functions for its implementation.

From all these computational improvement to GDFT filter bank, the algorithm is still complex for SDR mobile systems. Best performing algorithm achieved about 33% reduction in the number of filter

multiplications per input sample [91]. Hence, research efforts are continuing towards improving on these indicators.

Equation (2.62) and Equation (2.63) show the Fourier transform and its inverse, where n_0 is the new reference from time origin, k_0, corresponding to the new reference for the discrete frequency, $W_k = e^{j(2\pi f/k)}$.

$$Y_k^{GDFT} = \sum_{n=o}^{k-1} y(n) W_k^{-(k+k_0)(n+n_0)} \quad \text{k= 0,1,..., k-1} \tag{2.62}$$

$$y(n) = \frac{1}{k} \sum_{n=o}^{k-1} Y_k^{GDFT} W_k^{-(k+k_0)(n+n_0)} \tag{2.63}$$

Equation (2.62) and Equation (2.63), represent the transformation and inverse transform pair, and n_0 and k_0 are rational fractions that are less than 1. If $n_0 = 0$ and $k_0 = \frac{1}{2}$, GDFT is referred to as odd-stacked DFT channel and when $n_0 = \frac{1}{2}$ and $k_0 = \frac{1}{2}$, it is referred to as odd-squared DFT. Substituting n_0 and k_0 into Equation (2.62) and Equation (2.63), GDFT can be related to DFT as follows in Equation (2.64), where k is the number of frequency channels, M is the decimation factor of the channel signals, $X_k^{GDFT}(m)$ and $h(n)$ are the filters.

$$Y_k^{GDFT} = W_k^{-(k+k_0)n_0} \sum_{n=o}^{k-1} [y(n).W_k^{-k_0 n}] W_k^{-kn} \quad \text{k=0,1,..., k-1} \tag{2.64}$$

$$y(n) = W_k^{k_0 n} \frac{1}{k} \sum_{n=o}^{k-1} [Y_k^{GDFT}.W_k^{-(k+k_0)n_0}] W_k^{kn} \quad \text{n=0,1,..., k-1} \tag{2.65}$$

$$Y_k^{GDFT}(m) = \sum_{n=-\infty}^{\infty} h(mM-n) y(n) W_k^{-(k+k_0)(n+n_0)} \quad \text{k=0,1,..., k-1} \tag{2.66}$$

This representation is illustrated in Figure 2.20. The center frequency of the channels are located at frequencies, $\omega_k = \frac{2\pi}{k}(k+k_0)$.

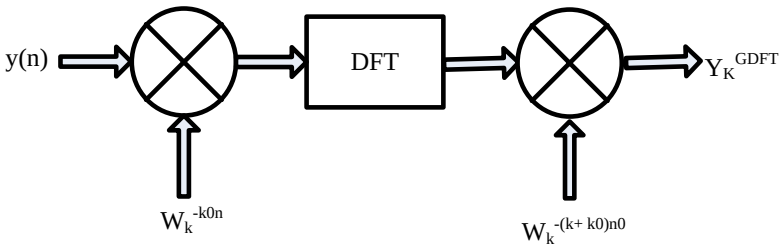

Figure 2.19. Diagram showing how GDFT can be derived from DFT.

Complex modulation of the filter bank can also be used for channelisation of single channel, k, as shown in Figure 2.20 and Figure 2.21, respectively. For the complex bandpass filter, Equation (2.67) holds.

$$\begin{aligned} X_k^{GDFT}(m) &= W_k^{-(k+k_0)(mM+n_0)} \sum_{n=-\infty}^{\infty} h_k(n)x(mM-n) \\ h_k(n) &= h(n)W_k^{(k+k_0)n} \\ \mathrm{k} &= 0,1,\ldots,k-1 \end{aligned} \tag{2.67}$$

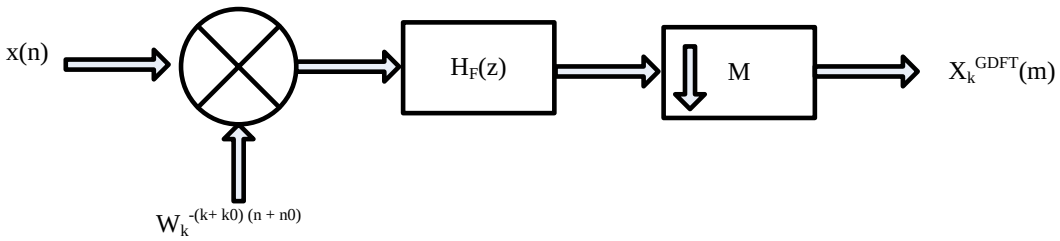

Figure 2.20. Complex Modulator model for GDFT FB for Channel K.

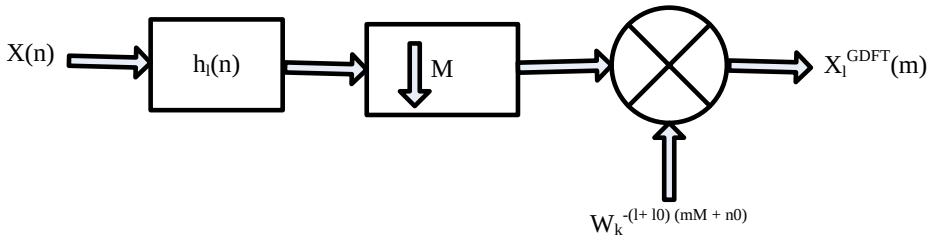

Figure 2.21. Complex bandpass model for k-channel GDFT.

Realising the polyphase implementation of GDFT filters, as shown in Figure 2.22, makes it more efficient than the other channelisation algorithms considered so far, since it allows processing load to be shared among the channels of the filter bank [1, 79]. Although the phase and frequency offsets introduce additional modulation to the structure, it is still considered the best approach to reduce the computational overhead in channelisation algorithm.

The additional modulation increases the computational overhead and renders it unfit for multi-standard SDR receiver, despite its easy reconfigurability. In a polyphase structure, Equation (2.68) applies, where clockwise commutator is used for the analysis signal.

$$\overline{p}_p(m) = h(mM - p) \quad \text{p=0, 1, ..., M-1} \tag{2.68}$$

According to polyphase branch signal, for inputs and outputs of the branches in the receiver side, Equation (2.69) is applicable.

$$x_p(m) = x(mM + p) \quad \text{p=0,1,..., M-1} \tag{2.69}$$

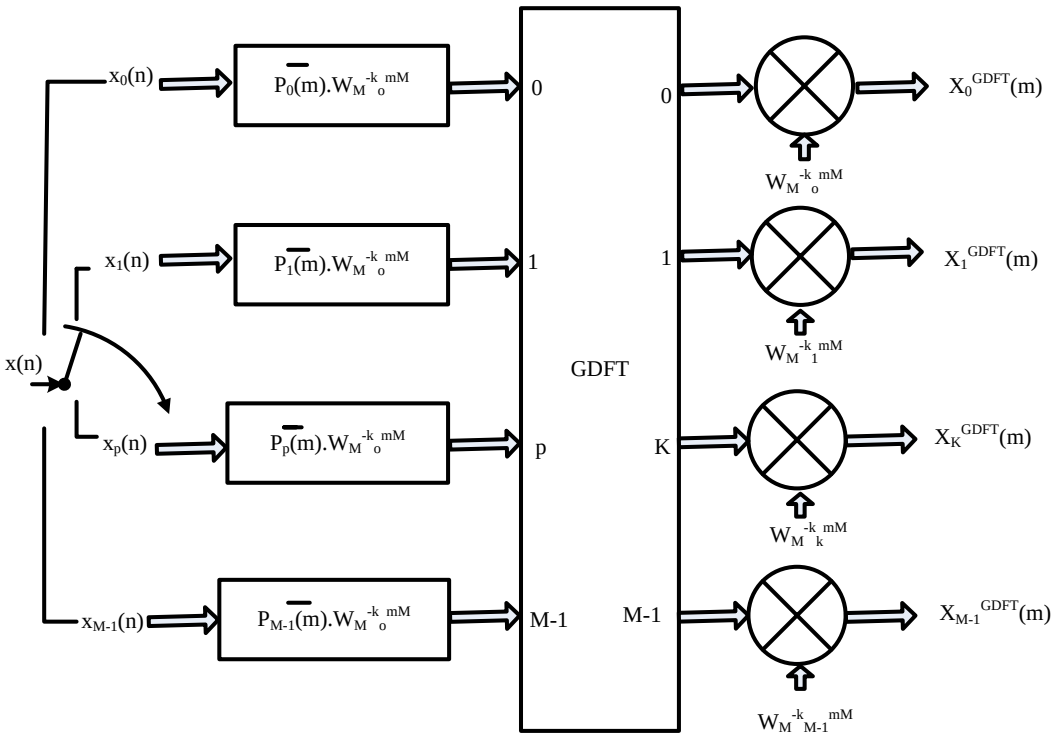

Figure 2.22. Polyphase Realisation of GDFT Filter Bank (Modified from [3] with permission).

Polyphase filter is more conveniently applied to critically sampled filter banks so that $k = M$. By applying the change of variables $n = rM + p$, the polyphase filter representation is as shown in Equation (2.70).

$$\begin{aligned} X_k^{GDFT} &= \sum_{p=0}^{M-1} W_M^{-(k+k_0)(p+n_0)} \sum_{n=-\infty}^{\infty} h(m-r)(M-p)x(rM+p)W_M^{-(k+k_0)rM} \\ &= W_M^{-k_0 mM} \sum_{p=0}^{M-1} W_M^{-(k+k_0)(p+n_0)} \sum_{n=-\infty}^{\infty} \overline{P}_p(m-r)W_M^{k_0(m-r)M} x_p(r) \\ &= W_M^{-k_0 mM} \sum_{p=0}^{M-1} W_M^{-(k+k_0)(p+n_0)} [x_p(m) * (\overline{P}_p(m)W_M^{k_0 mM})] \end{aligned} \tag{2.70}$$

2.4.11 Parallel GDFT-FB

Parallel GDFT [22] channelises the wide-band signal through number of different critically decimated odd-stacked GDFT-FB's, operating in parallel. The digitised wide-band input signals with common

sample rate, f_s, is fed into j-parallel filter banks. Depending on communication specifications and configurations of each filter bank, each filter bank implements its own frequency band based on its channel spacing in order to keep distortion to the minimum.

For even and odd-stacked cases, the channel spacing is given as in Equation (2.71).

$$\omega_{cs} = \frac{2\pi}{K} rad \tag{2.71}$$

In compliance with Equation (2.71), the common input signal sample rate must meet the condition specified in Equation (2.72), where j is the number of sub-bands supported and K_j and f_{csj} are the number of sub-bands and channel spacing in each GDFT-FB respectively.

$$f_s = K_j f_{csj} \quad \text{j=1,2,..., J} \tag{2.72}$$

If Equation (2.72) cannot be met by all j standards, an independent SRC could be applied before the required GDFT-FBs. Narrow band channels are extracted by selecting appropriate outputs from each of the filter banks.

P-GDFT is simple to implement and quite flexible but the ratio of unused sub-bands can be high. However, the computational complexity remains the same as GDFT-FB since all the sub-bands share the overall input signals.

2.4.12 Recombined GDFT-FB (R-GDFT FB)

R-GDFT FB is an over-sampled filter bank which analyses signal into uniformly spaced narrow sub-bands and then recombine groups of adjacent sub-bands to form wider sub-bands as seen in Figure 2.23. The granularity of each sub-band is determined by the bandwidth occupied by the minimum guard band required [6, 83]. Thus, the granularity and the total bandwidth occupied by the input signal determine the number of sub-bands required in R-GDFT, which also determine its computational complexity. Also, the size of the granularity with respect to the total signal bandwidth determines the center frequency of the channel. The larger the number of granularity bands, the greater the flexibility in choosing the center frequencies for the channel filters. The channel center frequencies are given as follows in Equation (2.73), where k represents the number of sub-bands, while R_F is the number of

sub-bands necessary to recombine specific types of narrow-band channels.

$$\begin{aligned} \omega_{ch} &= \frac{2\pi R_F}{k}\left(n+\frac{1}{2}\right) \\ &\text{for odd } R_F \\ \omega_{ch} &= \frac{2\pi R_F}{k}n \\ &\text{for even } R_F \end{aligned} \tag{2.73}$$

Every recombined signal denoted by $y_{k,R_F}(n)$ is formed by R_F- adjacent sub-bands allocated from the k^{th} output of GDFT-FB, as shown in Equation (2.74).

$$Y_{k,R_F}(z) = \sum_{r=0}^{R_F-1} {}_F e^{j\phi_r} Y_{k+r_f}(z^M) H_M(ze^{-j\beta_{r_f}}) \tag{2.74}$$

Recombination is achieved by first interpolating each of the R-sub-bands, by a factor M, frequency shifting by β_{r_f} to the correct center frequency and phase offset ϕ_{r_f} in order to finally add these shifted in-phase channels together.

The interpolating factor for R-GDFT FB is represented using Equation (2.75), where L_{DFT} is the over-sampled factor of GDFT.

$$M \leqslant \frac{R_F}{L_{DFT}} \tag{2.75}$$

When $M = \frac{R_F}{L_{DFT}}$, the frequency and phase shapes are represented as follows in Equation (2.76).

$$\begin{aligned} \beta_{r_f} &= \pi + \frac{\pi}{R_F}(2r_f+1) \\ \phi_{r_f} &= -\left(\frac{MN}{2D}+\frac{NM}{2}\right)\beta_{r_f} \\ &\text{for } r_f = 0,1,...,R_F-1 \end{aligned} \tag{2.76}$$

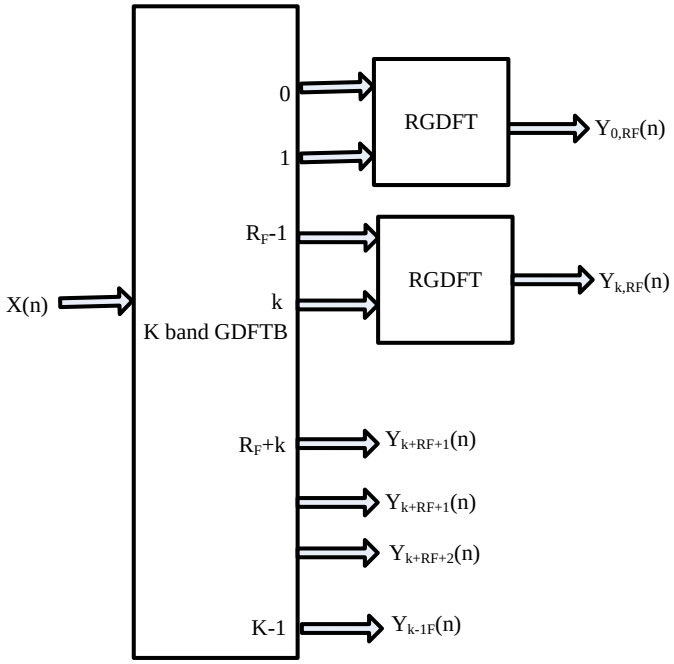

Figure 2.23. Recombined GDFT (Modified from [1] with permission).

2.4.13 Cosine Modulated Filter Bank

This is an attractive choice for the design and implementation of filter banks with a large number of sub-bands [1, 85, 98, 99, 100]. The main features of the cosine modulated filter bank are:

- Simple design approach for generating the low-pass prototype when perfect reconstructions must be achieved.
- Low implementation cost.

The linear phase prototype low-pass filter, $H(z)$, of order, N, has pass-band edge of $\omega_p = \left[\frac{\pi}{2M} - \rho\right]$ and stop-band edge of $\omega_s = \left[\frac{\pi}{2M} + \rho\right]$, and 2ρ is the width of the transition band. For even multiples of number M of sub-bands, the length is $N+1$, that is, $N = (2LM - 1)$.

Given the prototype filter, the cosine modulated version of the analysis filter bank is given as in Equation (2.77).

$$h_m(n) = 2h(n)cos\left[(2m+1)\frac{\pi}{2M}(n-\frac{N}{2})+(-1)^m\frac{\pi}{4}\right] \quad (2.77)$$
for n=0,1..., N and m= 0,1,..., (M -1) with N= (2LM -1)

The prototype filter can be decomposed into $2M$ polyphase components, as follows in Equation (2.78), where $S_j(z) = \sum_{i=0}^{L-1} h(2LM+j)z^{(-1)}$ are the components of the filter $H(z)$.

$$\begin{aligned} H(z) &= \sum_{i=0}^{L-1}\sum_{j=0}^{2M-1} h(2LM+j)z^{-(2LM+j)} \\ &= \sum_{j=0}^{2M-1} z^{-j}S_j(z^{2M}) \end{aligned} \quad (2.78)$$

The analysis filter bank can be described as shown in Equation (2.79).

$$\begin{aligned} H_m(z) &= \sum_{n=0}^{N} h(m)z^{(-n)} \\ &= \sum_{n=0}^{2LM-1} D_{m,n}h(n)z^{(-n)} \\ &= \sum_{l=0}^{L-1}\sum_{j=0}^{2LM-1} D_{m,2LM+j}h(2LM+j)z^{(-2LM+j)} \end{aligned} \quad (2.79)$$

Applying the cosine modulated algorithm as seen in Equation (2.80).

$$\begin{aligned} &cos\left\{(2M+1)\frac{\pi}{2M}[(N+2kM)-\frac{N}{2}+\Phi]\right\} \\ &= (-1)^k cos\left[(2M+1)\frac{\pi}{2M}(n-\frac{N}{2}+\Phi)\right] \end{aligned} \quad (2.80)$$

$$\begin{aligned} H_m &= D_{m,n+2kM} \\ &= (-1)^k D_{m,n} \end{aligned} \quad (2.81)$$

Substituting j for n, and l for K, Equation (2.81) becomes $D_{m,j+2LM} = (-1)^l D_{m,j}$. Rewriting Equation (2.79), the following Equation (2.82) is derived.

$$\begin{aligned} H_m(z) &= \sum_{j=0}^{2M-1} D_{m,j}z^{-j}\sum_{l=0}^{L-1}(-1)^L h(2LM+j)z^{(-2LM)} \\ &= \sum_{j=0}^{2M-1} D_{m,j}z^{-j}S_j(-z^{2M}) \end{aligned} \quad (2.82)$$

Equation (2.82) are represented as given in Equation (2.83), where D_1 and D_2 are $M \times N$ matrices, whose (m, j) elements are $D_{m,j}$ and $D_{m,j+m}$, respectively for $m, j = 0, 1, ...(m-1)$.

$$S(z) = \begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ \vdots \\ H_{m-1}(z) \end{bmatrix} = \begin{bmatrix} D_1 & D_2 \end{bmatrix} \begin{bmatrix} S_0(-z^{2m}) \\ z^{-1}S_1(-z^{2m}) \\ \vdots \\ \vdots \\ z^{(-2m-1)}S_{2m-1}(-z^{2m}) \end{bmatrix} \tag{2.83}$$

Representing $\delta(z)$ as shown in Equation (2.84).

$$\delta(z) = \begin{bmatrix} 1 & z^{-1} & \cdots & z^{-M+1} \end{bmatrix}^T \tag{2.84}$$

The polyphase representation of the matrix can be represented as in Equation (2.86), where $S(z)$ is the polyphase matrix.

$$e(z) = \begin{bmatrix} D_1 & D_2 \end{bmatrix} \begin{bmatrix} S_0(-z^{2m}) & & & 0 \\ & S_1(-z^{2m}) & & \\ & & \ddots & \\ & & \ddots & \\ 0 & & & S_{2m-1}(-z^{2m}) \end{bmatrix} \begin{bmatrix} \delta(z) \\ z^{-m}\delta(z) \end{bmatrix} \tag{2.85}$$

$$S(z^m)\delta(z) = \left\{ D_1 \begin{bmatrix} S_0(-z^{2m}) & & & 0 \\ & S_1(-z^{2m}) & & \\ & & \ddots & \\ & & \ddots & \\ 0 & & & S_{2m-1}(-z^{2m}) \end{bmatrix} + z^{-m}D_2 \begin{bmatrix} S_0(-z^{2m}) & & & 0 \\ & S_{M+1}(-z^{2m}) & & \\ & & \ddots & \\ & & \ddots & \\ 0 & & & S_{2m-1}(-z^{2m}) \end{bmatrix} \right\} \delta(z) \tag{2.86}$$

Figure 2.24 illustrates the design of cosine modulated filter bank.

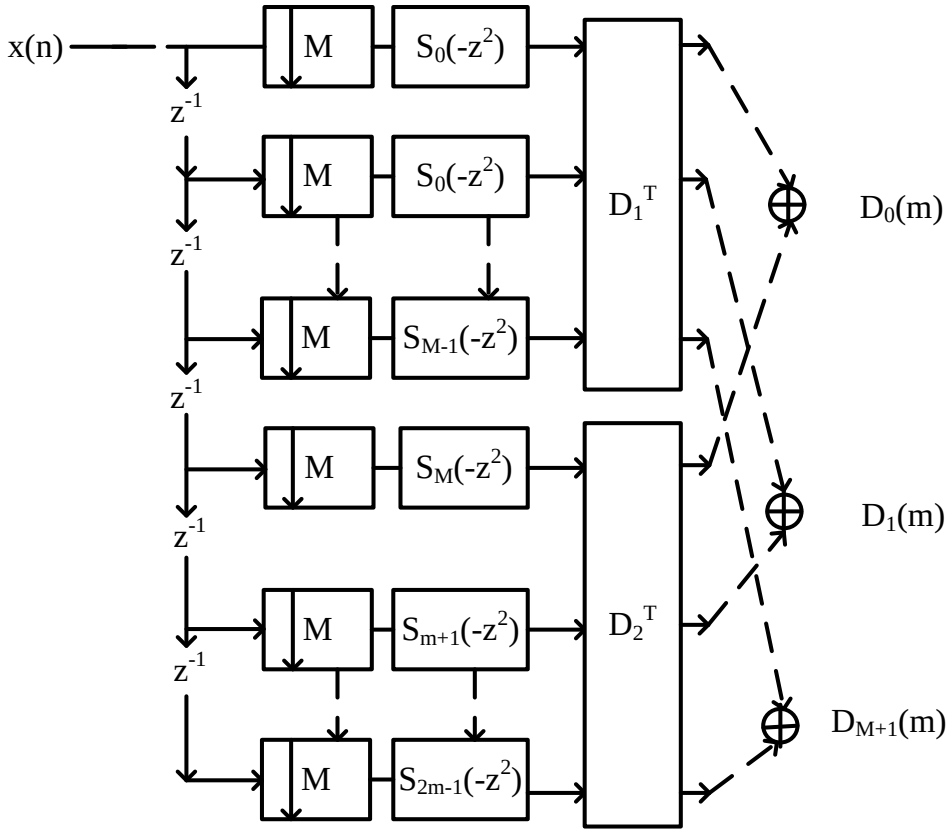

Figure 2.24. Cosine modulated filter bank (Modified from [1] with permission).

2.4.14 Comparison and Benchmarking of the Existing Literature

Table 2.5 shows comparison and benchmarking of the existing literature. From the literature [1], MPIS of FPCC is relatively higher while P-CDFB and HTQMFB are lower.

Table 2.5. Different Channelization algorithms [1]

Channelization Algorithm	FPCC	TQMFB	P-CDFB	HTQMFB	P-FRM	PGDFT	R-GDFT
f_s MHz	5	5	5	5	5	5	5
Passband Edge (kHz)	11.5	11.5	11.5	11.5	11.5	11.5	11.5
Stopband Edge	13.5	13.5	13.5	13.5	13.5	13.5	13.5
Passband Ripple (dB)	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Stopband Attenuation()	55	55	55	55	55	55	55
Multiplication Per sample	5.9e5	4.85e6	1.2E6	7E3	78E3	375	315

2.5 DIGITAL FILTER REPRESENTATION

Digital filter design that can meet the demands of integrating multiple standards on a single platform in Software Defined Radio (SDR) is quite challenging. This is as a result of high computational complexity involved in the processing of multiple standards concurrently on the same platform. Thus, there is a need to design digital filter that can conserve some of these resources to reducing the cost implication and increasing the operational speed. Efficient implementation of the digital filter components such as adder, multiplier and filter coefficients are necessary in order to reduce the computational complexity of the system. This reduces the hardware required to implement the process.

This Section considers binary number representations for fixed and floating-point number system as an approach to reduce the computational complexity of filter. The different fixed points representations and also the floating-point number systems were reviewed. Also, distributed number system was considered as an alternative to reduce the complexity of FIR filter. Evaluation of fixed and floating point residual number system (RNS) were investigated and analysed for small and large moduli set. Lastly, different optimisation algorithms were discussed.

2.5.1 Digital Filter Number System Representation

Number systems have been used from time past to implement filter coefficients efficiently in order to conserve hardware resources. This could be either fixed point or floating point number representation. The option of using either fixed or floating points depends on the designer's choice. However, the fixed point methods incur higher speed and low cost, while floating point systems have a higher dynamic range and no scaling.

2.5.2 Fixed Point Number System

Fixed point number systems are represented in most digital filters and it can be classified as either conventional or unconventional number systems [101, 102, 103, 104]. The fixed point number system can be either unsigned integers, signed magnitude number, two's complement, one's complement and diminished number system, canonical number system and residual number system and others. These number representations are explained in the subsection that follows.

2.5.2.1 Canonical Signed Digit (CSD)

Canonical signed digit (CSD) can be applied to reduce the area, word length and critical path of digital filter by reducing the generated partial products for the inner convolution products and guaranteeing that minimum number of non-zero bits are used [111, 112, 113, 114, 115, 116]. This is done by scanning the bits from the least significant bit to the most significant bit and replacing the continuous non-zero bits sequences by pair of non-zero positive and negative bits. This representation is useful to ensure low complexity algorithms design of filters and are also important for efficient implementation of multipliers.

By using CSD, there is reduction in the number of consecutive ones which leads to reduction in the number of computations and additions required for implementation. An n-bit Canonical Signed Digit (CSD) is a radix 2 signed digit encoding. An n-bit constant C can be represented as shown in Equation (2.87).

$$C = \sum_{I=0}^{n-1} S_i 2^i \quad \text{for } \mathrm{S}_i \in -1, 0, 1 \tag{2.87}$$

It encodes a constant multiplicand with signed digits 1, 0 and-1 while each of the digit S_i contributes a weight of 2^i to the constant.

The CSD representation has the following properties:

- No two consecutive bits in CSD representation are non-zero.
- The CSD representation of a number uses a minimum number of non-zero digits.
- The CSD representation of a number is unique.

The CSD representation of number can be recursively computed using strings properties. The strings of 1's are scanned from the least significant bit to the most significant bit. Any encounter with string $\bar{1}$ represents -1. The following examples illustrate CSD representations.

Example 1: Convert $16'b00110111100$ into CSD representation
In order to do this, scan the string from LSB to MSB and replace it with equivalent CSD representation as follows.

$$16'b00110111100 = 00111000\bar{1}00$$

This process is repeated until all the strings of $1's$ are replaced by their equivalent CSD representation. The final string values will be represented as follows.

$$0100\bar{1}0000\bar{1}00$$

It can be seen from above example, that the CSD representation restricts the repetition of two concurrent redundant one's in the number. Representing the multiplicand by a constant and multiplying it with the CSD multiplier can result in a reduced number of partial products.

Example 2: $y[n] = 0.961y(n-1) + s(n)$.
In Q1.15 format, this can be represented as $16'b0111010100111111$. Transforming the constants to

CSD representation by recursively applying the string property to the binary representation of the number as follows.

$$\begin{aligned} y[n] &= 16'b0111010100111111 \\ &= 0111010101000001\bar{1} \\ &= 100\bar{1}010101000001\bar{1} \\ &= 2^0 - 2^{-3} + 2^5 + 2^7 + 2^9 - 2^{15} \end{aligned}$$

The CSD representation of the constant has reduced the number of the non-zero digits from eleven to six. The implementation of multiplication by constant requires only six partial products. These partial products are generated by appropriately shifting $y[n-1]$ by weight of the bits depending on the Q format representation of the constants. The partial products generated are shown in Equation (2.88).

$$y[n-1] - y[n-1]2^3 + y[n-1]2^5 + y[n-1]2^7 + y[n-1]2^9 \tag{2.88}$$

The arithmetic can be optimised by incorporating the compression tree. The correction vector (CV) is calculated by adding the correction for multiplicand by 1 and $\bar{1}$ in the CSD representation of the number. CV requires multiplication by 1 to be computed by flipping the sign bit of the partial products and adding 1 to the location of the sign bit and extending the number of all 1.

Example 3:

$$\begin{aligned} 23 &= 2^4 + 2^2 + 2^1 + 2^0 \\ &= 10111 \end{aligned}$$

This number can also be represented as follows.

$$\begin{aligned} 23 &= 2^5 - 2^3 - 2^0 \\ &= 10 - 100 - 1 \end{aligned}$$

The first representation uses three additions, while the second representation requires two subtractions. CSD representations are used to implement FIR filters to obtain multiplier less FIR filters [117].

2.5.2.2 Residual Number System Representations (RNS)

Residual Number System (RNS) filter realisation has been investigated in [118, 119]. The prominent feature of RNS, which distinguishes it from the other number systems is its inherent parallelism, modularity and local carry propagate. These are special features in DSP, when large word length and high throughput are required for addition and multiplication operations. Residual Number system (RNS) was proposed to offer solution by providing high operating speed at reduced the word length, area utilization and power consumption.

The Residue Number System (RNS) is a non-weighted number system that can accelerate arithmetic operations up due to its peculiar features of carry-free propagation and parallelism. This results in carry-free addition, multiplication, and subtraction [120, 121, 122]. The most important factors to consider when choosing RNS for FIR filter is the moduli set. The choice of moduli set greatly influences the area utilisation, speed, cost and power consumption of the hardware design. Different research efforts on the influence of the moduli set on the hardware complexities are available in the literature. Sweidan and Hiasat proposed an algorithm that requires four binary adders, out of which two are operating in parallel mode, which resulted in higher speed and smaller silicon area [123]. Also, Amir Sabbagh and Keivan [124], presented two residue to binary conversion, using 2^n, $2^{n+1}+1$, $2^{n+1}-1$ moduli set. The moduli set consists of well-formed moduli and a balanced set, which resulted in better and faster RNS implementation. PremKumar in [125] described a residue number to binary converter, that converts number in the modulo set, $2n+2$, $2n+1$, $2n$, with 2 as a common factor. This algorithm achieved a faster conversion ratio in terms of speed. An algorithm in [126] discussed a high-speed realization of residue to binary converter for the 2^{n+1}, 2^n, 2^{n+1} moduli set, which doubly improved the leading implementation in terms of the overall delay time. The algorithm employed certain symmetrical properties in its implementation to reduce the hardware specification by $n-1$ full adders. It also reduces the redundancy in its implementation. Another approach was proposed to perform inner product computation based on distributed algorithm principles [127]. The input data were represented in the residue domain and encoded with a thermometer code when the output data are encoded with one of the hot code formats. The operating speed of one hot code modular adder was superior to the conventional binary code. A non-recursive digital filter was presented based on moduli set 2^{n-1}, 2^n, 2^{n+1} using diminished 1 representation [128]. The method investigated the usage of $n+1$ bits circuit in 2^n+1 bits channel.

Forward converter for RNS with diminished-1 encoded channels was proposed by [129]. Also,

multiplication was eliminated in the design of RNS converter [126]. Thus, fewer multipliers and adders were used in the design. This invariably reduced the hardware complexity and increased the speed. Dual sum carry look-ahead adder [130] which consists of a circular carry generator and a multiplexer was designed with reduced complexity. Jemmy, Yung Shem eliminated the bottleneck encountered in the carry propagation additions and modular adder free of the existing designs. This method resulted in both reduced power factor and leakage power.

Vinnakota and Rao discussed RNS to the binary converter [131] and illustrated it to be a simple modification of the well-known Mixed Radix Conversion techniques. Evaluation of this algorithm and comparison with the existing algorithms showed improvements in terms of speed and cost, however, not in terms of delay and area. A conjugate moduli set was presented in a hardware efficient two-level implementation of the weighted to RNS and RNS to weighted conversion [132]. The design offered 25 to 40% hardware savings, reduction of 80% complexity of the CRT and achieved a higher dynamic range. Kotha et al. [120] proposed new modular multiplication for 2^k-1, 2^k, $2^{k+1}-1$ for fixed-point coefficient FIR filter. This algorithm improved the clock rates and reduced the area and power consumption compared to the conventional modular multiplication. Ahmad Hiasat [123] designed a converter which consists of three $4n$ bits Carry Save Adder (CSA) together with an additional modulo $2^{4n}-1$ adder. This leads to a reduction in hardware requirements concerning area, delay, power, and energy efficiency. Richard Conway and John Nelson algorithm in [133] used moduli set of the form 2^n-1, 2^n, 2^n+1, which was primarily based on CSA and one Carry Propagation Adder (CPA) without the necessity for a Look-Up table (LUT). The design occupied lesser silicon space and was therefore very swift. The author proposed new CRT property, to reduce the total dynamic range and the overall result was faster and more efficient, with improved delay and area cost.

Kazeem Alagbe Gbolade et.al [134] used the CRT to obtain a reverse converter that uses mod $(2n-1)$ operations instead of mod $(2n+1)(2n-1)$ and $(2n)(2n-1)$. The approach is traditional in nature but the results yielded improved performance, in terms of conversion time, area, cost, and power consumption. Mohan [131] compared the design of Vinnokota and Raos and Piestrak, to the Andraros and Ahmad design. In comparison to Vinnokota and Rao's design it became evident that the Andraros and Ahmad design was cost-effective in terms of delay and speed. The design used the moduli set 2^n-1, 2^n, 2^n+1, which was a variation of the mixed radix conversion technique. Ahmad Hiasat [135] used the Chinese Remainder Theorem (CRT) approach to produce a simpler converter structure for four moduli set 2^n-1, 2^n, $2^{2n}+1$, 2^{2n+p}, using common factors. There was considerable reduction in

area, delay, time, energy and power utilization when compared with other published works.

RNS is defined as a set of relatively prime integer called modulo. The modulo operation is the remainder after the division operation by the chosen modulo. The remainder is called residues and can be repetitive, which means a mod b =c, where b is the modulo chosen and c is the residue or the remainder. RNS breaks large integer into smaller residues, denoted as r_1, r_2, r_n, where r_n is the n^{th} residue.

The RNS independently process each residue in parallel form. It operates with little or no carry propagation from each other. The addition, subtraction and multiplication are carry- free, which means that each digit of the arithmetic operation is independent of the neighbouring digits and thus allows parallelism between the operations without having to wait for the results from the adjacent bit. RNS resolves most of the carry bit propagation issues often experienced in adders and multipliers, as the carry propagation is limited to within a single residue or a few bits. The most important property of RNS is the significant speed for carrying out arithmetic intensive algorithms in signal processing algorithms.

In RNS, a number, x, is represented by the list of its residues with respect to k pairwise relatively prime moduli $M_{k-1} \gg M_1 > M_0$ [136, 137]. The residue x_i of x, with respect to the i^{th} modulus M_i is akin to a digit and the entire k-residue representation of x can be viewed as a k-digit number, where the digit set for the i^{th} position is $[0, M_{i-1}]$. Notationally, we write x_i= x mod M_i= $< x >_{mi}$ and specify RNS representation of x, by enclosing the list of residues in parenthesis.

The product M of the k-pairwise relatively prime modulo is the number of different representable values in RNS, known as dynamic range. That is, the dynamic range can be represented as in Equation (2.89).

$$M = M_{k-1} \times .... \times M_1 \times M_0 \tag{2.89}$$

For instance, RNS $< 8 \mid 7 \mid 5 \mid 3 >$ has dynamic values $M = 840$. There are two approaches for the choice of modulo for a given dynamic range: The small modulo set varies from the range of 2^n to $2^n \pm 1$ while the large modulo set is made up of a small set of prime numbers. The proper selection of

a modulo set depends on the given dynamic range, which determines the efficient residual to binary converter for a given digital filter [138].

Modulus set {2, 5, 7} has a dynamic range of M=2.5.7=70. The integer 7 has an RNS representation of: $7 = \{1,2,0\}_{rns}$ and also $4 = \{0,4,4\}_{rns}$. For a small word length, the RNS provides a speed of $2^4 \times 2$ bits while large modulus has a speed of $2^8 \times 2$ bits .

The RNS and the Chinese Remainder Theorem, (CRT) are based on [138, 139]. In order to implement RNS on the digital FIR filter, the modulus operation converts the binary values to RNS, while CRT converts from RNS to a binary system. The modulus operation is the remainder after the division operation by the chosen modulo. The remainder are called the residues and it can be repetitive, which means $a \ mod \ b = c$, where b is the modulus chosen, c is the residue or the remainder.

The representation of RNS in a digital filter is achieved by computing the dynamic range in terms of the number of bits. The dynamic range is the total production of the residues available, as indicated in Equation (2.90), where m_1, m_2, m_3 are the individual moduli and M is the dynamic range.

$$M = \prod m_1 m_2 m_3 \tag{2.90}$$

Example 4: If $m_1 = 3, m_2 = 5, m_3 = 7$, then M is 105 as follows.

$$M = \prod (3.5.7) = 105$$

Representing the dynamic range of Example 4, using Equation (2.91).

$$\text{Bits}= \frac{log_{10}(\prod m_1 m_2 m_3)}{log_{10}2} \tag{2.91}$$

From the Equation (2.91), the dynamic range is approximately equals to seven as follows.

$$\begin{aligned} \text{The number of bits } &= \frac{log_{10}(\prod m_1 m_2 m_3)}{log_{10}2} \\ &= \frac{log_{10}(105)}{log_{10}(2)} \\ &= 6.7142 \end{aligned}$$

2.5.2.3 Chinese Remainder Theorem (CRT)

The Chinese Remainder Theorem states that for a given residue number system representation, $[r_1, r_2, r_3]$, the integer 'x' is determined by using CRT if the moduli m_1, m_2, m_3 are mutually exclusive. That is, if the greatest common divisor is '1'. Mathematically, CRT as shown in Equation (2.92).

$$\left|X\right|_M = \left|\sum_{j=1}^{n} \widehat{m}_j \left|\frac{r_j}{\widehat{m_j}}\right|_{m_j}\right|_m \qquad \widehat{m}_j = \frac{M}{m_j} \tag{2.92}$$

$$M = \prod_{j=1}^{n} m_j \qquad m_j m_k = 1 \qquad \text{for } j \neq k \tag{2.93}$$

In order to illustrate this theorem, consider integer $x = 21$ and using the Example 4 with the following modulus: moduli $m_1 = 3$, $m_2 = 5$, $m_3 = 7$. The residue number representation is $0, 1, 0$. The first procedure is to convert the integer back to RNS as follows. Calculate the product of the modulo M as follows.

$$\mathrm{M} = 3 \times 5 \times 7$$

$$= 105$$

The individual weight is calculated as follows.

$$\begin{aligned}
\widehat{m_1} &= \frac{M}{m_1} \\
\frac{105}{3} &= 35 \\
\widehat{m_2} &= \frac{M}{m_2} \\
\frac{105}{5} &= 21 \\
\widehat{m_3} &= \frac{M}{m_3} \\
&= \frac{105}{7} \\
&= 15
\end{aligned}$$

Thus, the inverse of $\widehat{m_1}$, $\widehat{m_2}$, $\widehat{m_3}$ are calculated as follows.

$$\begin{aligned}
\left|\frac{1}{\widehat{M_1}}\right|_{m_1} &= \left|\frac{1}{35}\right|_3 \\
&= |35|_3 \times |X_1|_3 = |1|_3 \\
&= |2|_3 \times |X_1|_3 = |1|_3 \\
&= |X_1|_3 = 2 \\
\left|\frac{1}{\widehat{M_2}}\right|_{m_2} &= \left|\frac{1}{21}\right|_5 \\
&= |21|_5 \times |X_2|_5 = |1|_5 \\
&= |1|_5 \times |X_2|_5 = |1|_5 \\
&= |X_2|_5 = 6 \\
\left|\frac{1}{\widehat{M_3}}\right|_{m_3} &= \left|\frac{1}{15}\right|_7 \\
&= |15|_7 \times |X_3|_7 = |1|_7 \\
&= |1|_7 \times |X_3|_7 = |1|_7 \\
&= |X_3|_7 = 14
\end{aligned}$$

The overall weighted number is computed as follows.

$$\begin{aligned}|X_m| &= |35 \times |2 \times 2|_3 \\ &\quad + 21 \times |6 \times 1|_5 + 15 \times |1 \times 8|_7|_{105} \\ &= |35 \times |1|_3 + 21 \times |1|_5 + 15 \times |1|_7|_{105} \\ &= |35 + 21 + 15|_{105} \\ &= |71|_{105}\end{aligned}$$

Finally, the integer is calculated as follows.

$$\begin{aligned}|X_m| &= |35 \times |0 \times 2|_3 + 21 \times |6 \times 1|_5 + 15 \times |0 \times 8|_7|_{105} \\ &= |35 \times |0|_3 + 21 \times |6|_5 + 15 \times |0|_7|_{105} \\ &= |0 + 21 \times |6|_5 + |0|_{105} \\ &= |21|_{105} \\ &= 21\end{aligned}$$

Thus, the integer number $X = 21$ is obtained, uniquely from its residue, using CRT.

2.5.2.4 Common Sub-expression Techniques (CSE)

Common Sub-Expression techniques (CSE) can minimize the logic operators such as adders and logic depth of FIR filter. Two classical forms of CSE are horizontal and vertical sub-expression elimination techniques. The major complexity determinants in FIR filters are the number of adders used during the computation of the sum of the partial products and the critical paths which is referred to as the number of adder steps. Hence, the need to find an algorithm that can minimise the number of logic elements and the logic depth (LD). CSE eliminates the redundancy in the filter coefficient by using the most common factored bit pattern known as common sub-expressions (CS) that exist in the canonical signed digit representation.

The author in [140] used the coefficient sub-expression graph to reduce redundancy of two non-zero bits sub-expressions. [141] proposed using the most commonly occuring two bits CS. In [142], a non-recursive signed CSE algorithm was proposed to minimise the logic depth of the digital filter. [124] focussed on reordering computations sharing between different multipliers. There was 11% reduction

in adders used compared to the existing methods but there was compromise in delay performances. [143] designed differential coefficients where the differences betweeen the absolute values of filter coefficients were employed to reduce the dynamic range of computations. In [144], the algorithm proposed uses the redundancy among the CSE coefficients and the logical depth (LD) of the multipliers in order to minimise the redundancy in the adders. There was a drastic reduction in both the logical elements and the logical depth in the multiplier blocks. A contention resolution algorithm in [145] was developed for weighted two horizontal sub-expressions based on ingenious graph synthesis. The approach saved $1-3\%$ logic operators than NR-SCSE. The Bull- Horrocks algorithm was presented in [146] based on graphical representation of the multiplier block (MB) for reducing the number of LOs. [147] designed modified Bull Horrocks algorithm and n-dimensional adder graph (RAGn). This algorithm was able to reduce the number of LO's, however, there was an increase in the propagation delay. Vinod et al [148] uses the combination of horizontal and vertical sub-expression methods to reduce the logic depth as well as the logical operators in FIR filter. The design achieved 13% reduction rate when compared with contention resolution [145] However, when the hybrid approach proposed by Vinod was compared with multiple adder graph, the average reduction rate of the logical operator was 5% while the logical depth was reduced by 25%. Hatai et al [149] proposed using canonical signed digits based vertical and horizontal common sub-expression elimination algorithm. The method uses 4-bit common sub-expression in the vertical direction with either a 4- bits and 8-bits CSs in the vertical side, in order to reduce the logical operators and the logical depth in FIR filter. The method decreases the area consumption by 59% more than that of binary VHCSE technique. Marimuthul et al. [150] uses vertical horizontal binary common sub-expression elimination based constant multiplier design to reduce the adder steps and logical depth of the FIR filter.

Having seen the contributions of different authors to CSE, it is prudent taking that common sub-expressions elimination algorithm exploits the repetition seen in the binary digits of canonical signed digits representation. CSE is ideal for binary representation of the coefficients but in order to reduce the complexity of the digital filter, it is vital that the coefficients be converted to the CSD and then factoring out the common sub-expression techniques.

Example 5: Consider a two tap FIR filter with the following coefficients using *Q4* format as follows.

$$h_0 = 6'b011010$$

$$h_1 = 6'b010011$$

In order to implement the filter, the product of the two filter variables: h_0x_0 and h_1x_1 respectively must be calculated. These multiplications require shift and add operations as seen in Equation (2.94).

$$\begin{aligned} h_0x_n &= (x_n \gg 1) + (x_n \gg 2) + (x_n \gg 4) \\ h_1x_n &= (x_n \gg 1) + (x_n \gg 4) + (x_n \gg 5) \end{aligned} \tag{2.94}$$

The common factors between the two expressions are represented as indicated in Equation (2.95).

$$C_0 = (x_n \gg 1) + (x_n \gg 4) \tag{2.95}$$

This value is reused in the Equation (2.94) as depicted in Equation (2.96).

$$\begin{aligned} h_0x_n &= C_0 + (x_n \gg 2) \\ h_1x_n &= C_0 + (x_n \gg 5) \end{aligned} \tag{2.96}$$

Conventional CSE uses two different approaches to reduce the computational complexities. These are the horizontal sub-expression elimination technique and vertical sub-expression elimination technique.

2.5.2.5 Horizontal Sub-Expression Elimination Techniques (HSE)

The method represents a pattern that exists within each coefficient in a CSD representation and searches for a common bits or digit patterns that may appear in these representations. The expression is computed only once and then shifted to be used in other products that contain the pattern.

Example 6: Given the binary digit of 01001100111 as follows. This example is converted from binary pattern to CSD format as follows.

$$0101001\bar{1}0100\bar{1}$$

Assuming that the filter coefficients are represented as follows.

$$h_0 = 00\bar{1}0\bar{1}0100\bar{1}00$$

$$h_1 = 10100\bar{1}00\bar{1}001$$

$$h_2 = 100100\bar{1}0\bar{1}01$$

$$h_3 = 0100\bar{1}0\bar{1}00100$$

From the given filter coefficients, the binary patterns generated or seen are: $10\bar{1}$, $100\bar{1}$ and their negatives. In order to optimise the hardware, the pattern is implemented and the value shifted to cater for the second appearance as follows.

$$h_0 = 00\boxed{\bar{1}0\bar{1}}0\boxed{100\bar{1}}00$$

$$h_1 = \boxed{101}00\boxed{\bar{1}00\bar{1}}001$$

$$h_2 = \boxed{1001}00\boxed{\bar{1}0\bar{1}}01$$

$$h_3 = 0100\boxed{\bar{1}0\bar{1}}00100$$

2.5.2.6 Vertical Common Sub-Expression Elimination (VCSE) Technique

The VCSE method utilises vertical common sub-expression methods (VCS), that occur across the adjacent coefficients to handle multiple constant multiplications. The VCS such as $[11]$ and $[1\bar{1}]$ that exists across the coefficients shown in the bold rectangles in Table 2.5.2.5 are indicated as X_4 and X_5 respectively in Equation (2.97), where $x_1[-k]$ represents x_1 delayed by k units.

$$\begin{aligned} x_4 &= x_1 + x_1[-1] \\ x_5 &= x_1 - x_1[-1] \end{aligned} \tag{2.97}$$

With these VCSs, the output will be as indicated in Equation (2.98).

$$\begin{aligned} h_n x_n = {} & 2^{-2}x_4 + 2^{-6}x_1 - 2^{-8}x_5 + 2^{-10}x_4 + 2^{-}12x_4 \\ & + 2^{-14}x_5 - 2^{-16}x_4 - 2^{-2}x_1[-1] + 2^{-2}x_4[-2] + 2^{-5}x_4[-2] \\ & + 2^{-9}x_4[-2] - 2^{-15}x_4[-2] + 2^{-2}x_4[-4] - 2^{-4}x_1[-4] + 2^{-8}x_5[-4] \\ & + 2^{-10}x_4[-4] + 2^{-2}x_4[-4] - 2^{-4}x_1[-4] + 2^{-8}x_5[-4] + 2^{-10}x_4[-4] \\ & + 2^{-12}x_4[-4] - 2^{-15}x_5[-4] - 2^{-16}x_4[-4] + 2^{-6}x_1[-5] \end{aligned} \tag{2.98}$$

Since the bits that forms the VCSs occur across the coefficient, the symmetry of VCSs cannot be utilized when the bits are of opposite signs. Therefore, an additional multiplier block, MBAs are required to obtain the symmetrical parts of the coefficients when more than one VCSs with opposite bits occur. Consider the VCSs across the coefficient $h(0)$ and $h(1)$ in Table 2.6.

$$\begin{aligned} h_n x_n = {} & 2^{-2}x_4 + 2^{-6}x_1 - 2^{-8}x_5 + 2^{-10}x_4 \\ & 2^{-14}x_5 + 2^{-12}x_4 - 2^{-16}x_4 - 2^{-4}x_1[-1] \end{aligned} \tag{2.99}$$

It's symmetry VCSs part across the coefficient $h(4)$ and $h(5)$ is given in Equation (2.100).

$$\begin{aligned} h_n x_n = {} & 2^{-2}x_4[-4] - 2^{-4}x_1[-4] + 2^{-8}x_5[-4] + 2^{-10}x_4[-4] \\ & 2^{-12}x_4[-4] + 2^{-14}x_5[-4] - 2^{-16}x_4[-4] - 2^{-6}x_1[-5] \end{aligned} \tag{2.100}$$

The delays of certain terms in Equation (2.100) are different from the Equation (2.99), therefore Equation (2.101) can be obtained from Equation (2.99).

$$\begin{aligned} & h_n x_n = 2^{-2}x_4[-4] - 2^{-10}x_4 + 2^{-12}x_4 + 2^{16}x_4 \rightarrow \\ & 2^{-2}x_4[-4] + 2^{-10}x_4[-4] + 2^{-12}x_4[-4] - 2^{-16}x_4[-4] \\ & -2^{-8}x_5 - 2^{-14}x_5 \rightarrow 2^{8}x_5[-4] - 2^{-14}x_5 \end{aligned} \tag{2.101}$$

2.5.2.7 Horizontal Common Sub-Expression Elimination Techniques

This approach uses the CSs pattern $[101]$, $[10\bar{1}]$, $[1001]$ and $[100\bar{1}]$ with the corresponding negated versions to reduce the redundancy. Hartley showed that the use of two commonly appearing patterns in HCs [1 0 1] and [1 0] and the VCs [1 1] would further reduce the complexity. This can be illustrated in Table 2.6.

Table 2.6. Table showing the horizontal common sub-expression technique (HCSE) and vertical common sub-expression technique (VCSE).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x0		1				1	0	n		1	0	1		1		n
x1		1	0	n				1		1	0	1		n		n
x2		1			n				1						n	
x3		1			n				1						n	
x4		1	0	n				1	0	1		1	0	n		n
x5		1				1		n		1		1	0	1		n

2.5.3 Floating Point Binary Representation

Floating point binary representation provides high dynamic range with wider precision. It is an alternative choice when fixed point number fails due to its limited precision and low dynamic range. However, the floating point compromises its standard in terms of speed and cost. Most floating point systems [101, 151, 152, 153, 154, 155], have single or double precision IEEE floating point standards. A standard IEEE floating point format consists of sign bit S, exponent c, and an unsigned or fractional normalized mantissa, m. Floating point word can be represented algebraically as in Equation (2.102), where x_m is the mantissa and c is the number exponent with $\frac{1}{2} \leq |x_| < 1$.

$$x = x_m 2^c \tag{2.102}$$

2.5.4 Distributed Arithmetic

Distributed arithmetics (DA) is an alternative approach for implementing digital filter when hardware overhead is a major concern. It is a multiplierless memory based architecture proposed to replace the multiplications in signal processing with the combinational look-up table (LUT) [5, 156, 157, 158, 159]. It is a bit level restructuring of the multiply accumulate operation (MAC), into set of basic addition and shifting operation. It is simply implementation of multiply by constant. In FIR filter, the coefficient contributes an array of constants in some signed Q format where the tapped delay line forms the variables which changes at every sample clock. The DA replaces the MAC operation of convolution operation by using a bit serial look up table read and write operation.

DA was first introduced by Crosier and further development was carried out by Peled for efficient implementation of digital filter in serial form. Although DA implementations have numerous advantages, it is still facing the serious limitations of expontial growth of memory with higher filter order. These problems were addressed by many researchers [160, 161]. DA provides bit serial operation that implements a series of fixed point MAC operation in a known number of steps. Partial or full parallel structure overcame the speed limitation of bit serial DA, but at the cost of exponential increase in the memory requirements.

Yoo and Anderson also proposed the LUT-less architecture which comprises of multiplexers and adder pairs. However,the gain in area reduction was at the cost of increased critical path. LUT decomposition or slicing of LUT was proposed in [162]. Indexed LUT DA FIR filter was proposed [156]. It consists of indexed LUT pages, each of size 2^n and m bits multiplexer unit as a page selection module. Indexing of LUT controls the exponential DA growth and eliminates the need for adder. LUT partitioning was proposed by [163] to reduce the memory usage of the LUT for higher order FIR filter. The design provides less latency, less memory usage and high throughput when compared with the conventional DA.

The author in [164] proposed a memoryless distributed arithmetic based adaptive filter for low power and area efficiency. In this case, the conventional DA is replaced by $2:1$ multiplexers to reduce area. By replacing the algorithm with $4:2$ compressor adder instead of normal adder, there was area complexity enhancement. The author in [158] proposed using modified DA to compute the sum of product and saving considerable number of multiply and accumulation blocks and the circuit sizes reduces considerably. There was 40% less LUT flip-flop pairs used at the expense of speed. DA based

least mean square (LMS) adaptive filter using offset binary coding without LUT was presented to improve the performance of bit-serial operation [165]. Also, DA-RNS based filter implementation was used for effective calculation of modular inner products in FIR filter [157]. The RNS enhances high speed processing because of the absence of carry propagation and it thus proffers solution to the conventional DA approach.

In order to reduce the exponential size increment as well as the critical parts of the LUT in DA, the hybrid of canonical signed digit and residual number systems is used. Canonical signed digit can be used to reduce the area, word length and critical parts of DA by reducing the generated partial products for the inner convolution products by guaranteeing that minimum number of non-zero bits is used [111, 112, 113, 114, 115, 116]. This is done by scanning the bits from the least significant bits to the most significant bits and replacing the continuous non-zero bits sequences by pairs of non-zero positive and negative bits. This can be used to reduce the partial products during multiplication and addition of the binary number systems. This representation is useful to ensure low complexity algorithms design of filters and are also important for efficient implementation of multipliers. By using CSD, there is reduction in the number of consecutive ones which leads to reduction in the number of computations and additions required for implementation. An attempt to reduce the memory requirements of DA by reducing the area utilisation and the delay is very important for the implementation of digital FIR filter. Residual Number system (RNS) was proposed to offer solution by providing high operating speed at reduced word length, area utilisation and power consumption.
DA can be represented by Equation (2.103).

$$y = \sum_{k=0}^{K-1} B_k x_k \tag{2.103}$$

The ROM is P bits wide and 2^k wide to implement a look up table. The value of P is given using Equation (2.104).

$$P = \left\lfloor Log_2 \sum_{k=0}^{K-1} |B_k| \right\rfloor + 1 \tag{2.104}$$

Table 2.7 shows the ROM utilisation of distributed arithmetics.

Table 2.7. ROM utilization for Distributed Arithmetic.

x_{2b}	x_{1b}	x_{0b}	Contents of ROM
0	0	0	0
0	0	1	B_0
0	1	0	B_1
0	1	1	$B_1 + B_0$
1	0	0	B_2
1	0	1	$B_2 + B_0$
1	1	0	$B_2 + B_1$
1	1	1	$B_2 + B_1 + B_0$

The $\lfloor \cdot \rfloor$ is the floor operator that rounds a fractional value to its lowest integers. The contents of the lookup table are given in Table 2.7. The elements of the vectors $x = [x_0, x_1, x_2, ..., x_{N-1}]$ are pre stored in the shift registers. The architecture considers b^{th} elements in each cycles and concatenates them to form the address of the ROM. For most significant bits (MSB), the values in the ROM is subtracted from the running accumulator, and the rest of the bit location values from ROM are added in the accumulator. To cater for weights of the different bit locations in each cycle, the accumulator is set to $P + N$, where P bits adder add the current output of the ROM in the accumulator and the N bits of the accumulator are kept to the right side to cater for the shift operations. The data is input to the shift registers from the least significant bits (LSB). The DA algorithm takes N cycles to compute the summation. The architecture implementing the dot product for $k = 3$ and $P = 5$ and $N = 4$ is shown in Figure 2.25.

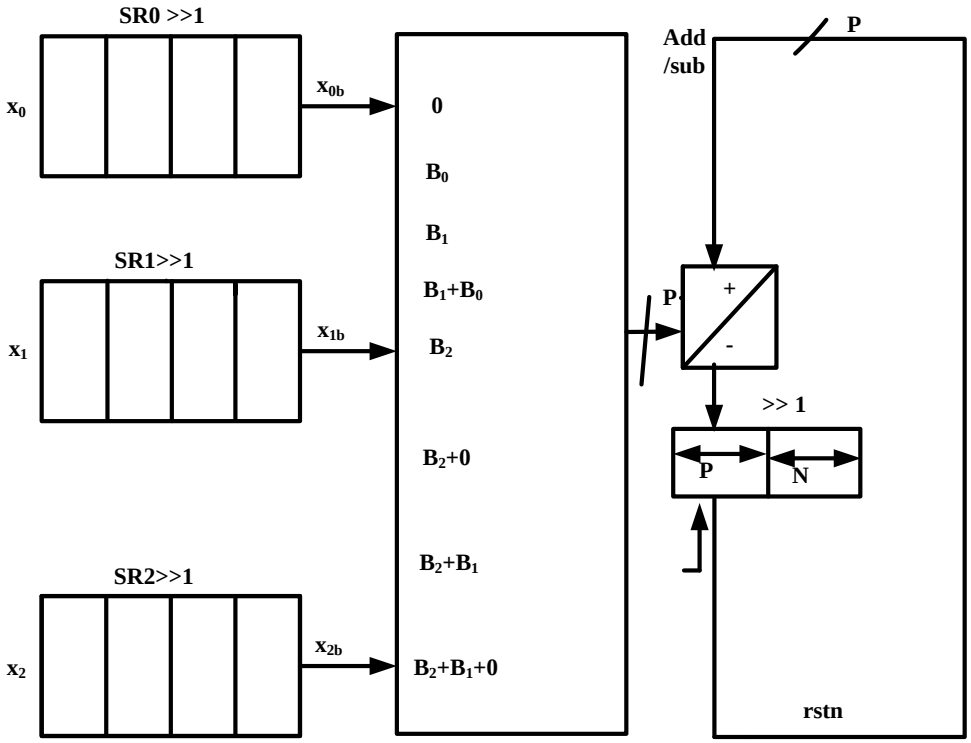

Figure 2.25. Architecture for DA (Modified from [5] with permission).

Example 7: Consider ROM to compute the dot products of 3 element vectors with the following vector elements: $B_0 = 3$, $B_1 = -1$ and $B_2 = 5$. Table 2.8 shows the implementation of the example being considered with lookup table.

Table 2.8. ROM utilization for Distributed Arithmetic.

x_{2b}	x_{1b}	x_{0b}	Contents of ROM	0
0	0	0	0	0
0	0	1	B_0	3
0	1	0	B_1	1
0	1	1	$B_1 + B_0$	3
1	0	0	B_2	5
1	0	1	$B_2 + B_0$	8
1	1	0	$B_2 + B_1$	4
1	1	1	$B_2 + B_1 + B_0$	7

Following is the given values of x_0, x_1 and x_2.

$$x_0 = -6 = 4'b1010$$

$$x_1 = 6 = 4'b0110$$

$$x_2 = -5 = 4'b1010$$

The contents of a 5 bits wide and 8 bits deep ROM is shown in Table 2.8. The shift registers are assumed to contain elements of vector x with the LSB in the right most bit location as indicated in Figure 2.26. After four cycles, the accumulator contains the value of the dot products:

$$9'b111001111 = -49_{10}$$

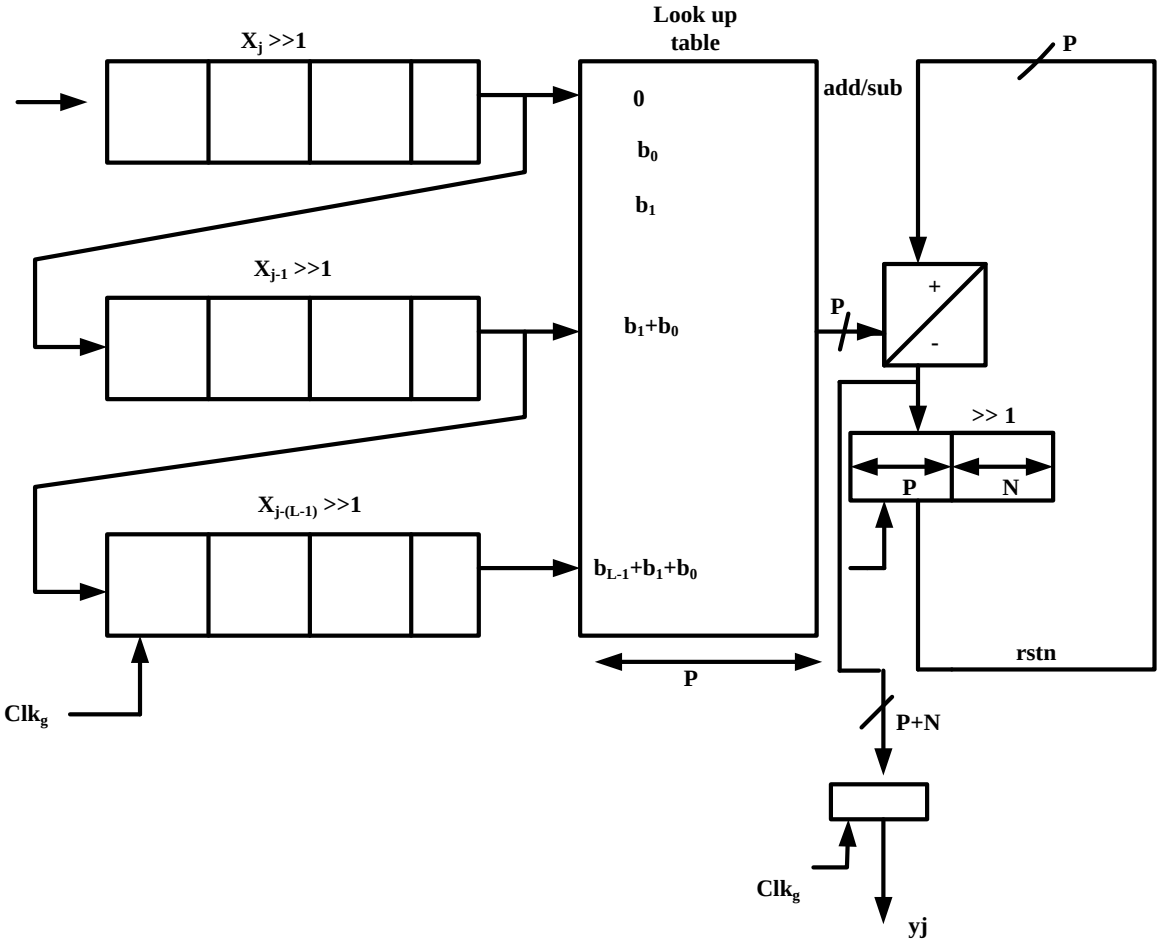

Figure 2.26. Content of Shift registers in DA architecture (Modified from [5] with permission).

2.6 OPTIMISATION ALGORITHM

2.6.1 Genetic Algorithm

Genetic Algorithm (GA) is a metaheuristic approach based on evolutionary processes. It was proposed by Holland in the early seventies as computer programs that mimic the natural evolutionary process. De Jong extended the GAs to functional optimisation and a detailed mathematical model of it was presented by Goldberg in I975 [166]. GA manipulates the initial population of individual in each generation, which consists of collection of chromosomes [167, 168] and it represents a set of solutions to the problem.

Within the population, the best survived chromosomes with minimum error functions are chosen for reproduction and then sent for the cross over operations. The selection criteria provide the necessary driving mechanism for better solutions to survive. Each solution proposed indicates how good it is, in comparison with other solutions in the population [169]. The cross over procedure exchanges a portion of data strings between the chromosomes, resulting in better and new chromosomes being produced. Following is the mutation procedure. Generally over a period of several generations, the genes tend to become homogeneous. Therefore, many chromosomes cannot continue to evolve before they reach the optimal state and some of the bits of the chromosome mutate randomly. Each GA iterations involve the following steps:

1. Initialisation: The algorithm begins by randomly creating set of initial population. For example, if the initial population consists of 50 individuals, which by default is the value of the population size in the population option.
2. Evaluation: The fitness value of each particle is calculated at the onset of each generation.
3. Selection: The algorithm involves chosen individuals for reproduction. This is probabilistic in nature depending on the relative fitness of the individual parent.
4. Mating: Mating is the resultant effect produced from parent's chromosome during the pairing process. New offspring is produced.
5. Mutation: This involves applying random changes to a single individual in the current generation in order to create a child.
6. Replacement: This is the last phase which destroys the old population and replaces it with new ones.

2.6.2 Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation (PSO) is a population based optimisation algorithm discovered by Kennnedy and Eberhart [170]. It is developed from swarm intelligence and inspired by social behaviour of birds flocking or fish schooling. The procedure starts with random population in which each particle is assigned initial velocity and initial position. Each particle represents solution [171, 172, 173]. The position and velocity of each particle is updated and optimal solution is obtained. The best value for each particle is called individual best (vbest) and the best value among all the individuals is called global best (wbest). The velocity and the position of particles are updated using Equation (2.105) and Equation (2.106) respectively, where v_1 represents the initial velocity of the i^{th} particles, w is the weighing functions, c_1 and c_2 are positive constraints, vbest(n) represents position

for i^{th} best particle, wbest(n) represents position for global best particle; $rand_1$ and $rand_2$ are the random number uniformly distributed in [0,1].

$$v_1(n+1) = w * v_i(n) + c_i * rand_1 * (vbest_1(n) - \theta_1(n)) + c_2 * rand_2 * (wbest_2(n) - \theta_i(n)) \tag{2.105}$$

$$\theta_i(n+1) = \theta_1(n) + v_1(n+1) \tag{2.106}$$

The basic step for particle swarming optimisation algorithms are outlined as follows:

1. Set the coefficient in the range to [-1, 1].
2. Set the initial population size to 100 and maximum iterations to 1000. Initialize c_1=2, c_2=2, v_1^{min}=0.01, v_2^{max}=1.0.
3. Evaluate the error fitness function values. Calculate the position and velocity for particle vbest and wbest.
4. Retain the updated values of vbest and wbest and discard the previous values.
5. Continue updating the iterating cycles.
6. Stop iteration when maximum iteration is reached.
7. The fitness values with least error are selected and used for filter design.

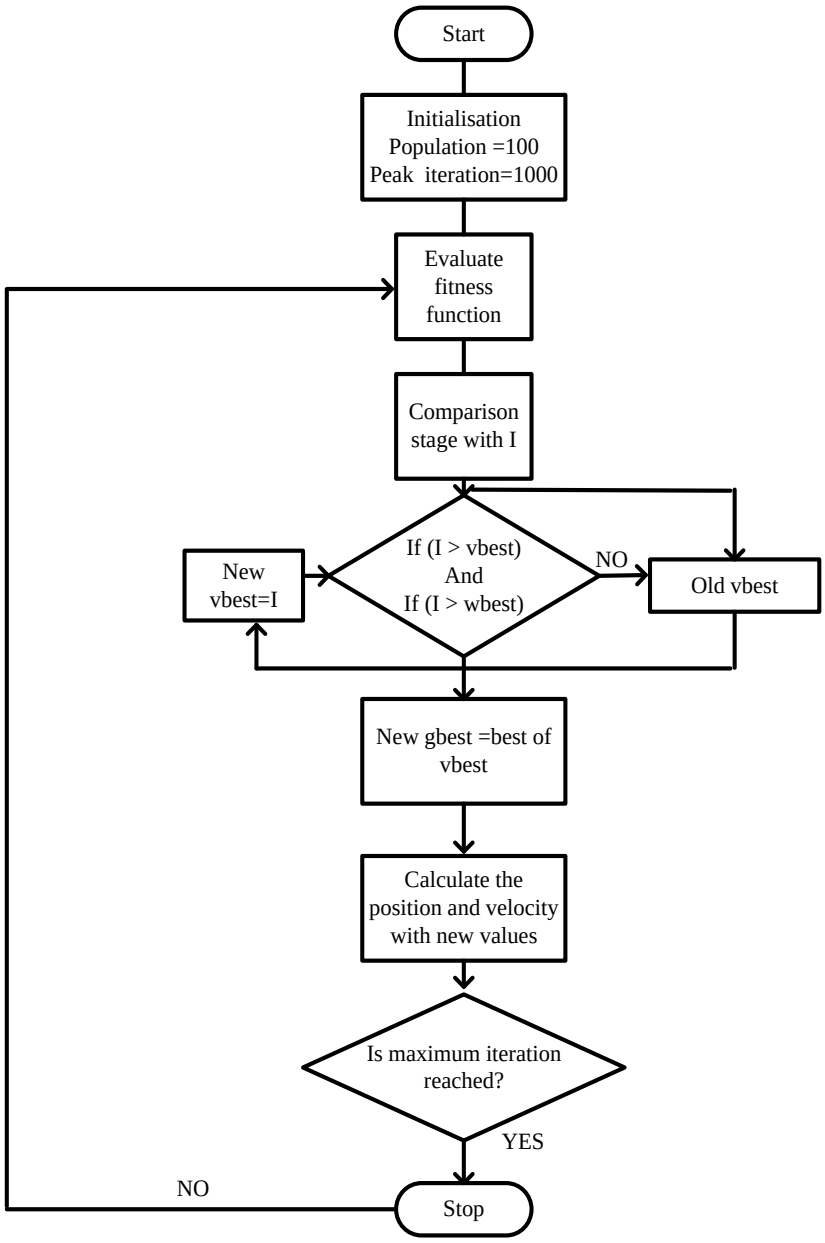

Figure 2.27. Flow Chart of Swarm Particle Optimisation Algorithm.

2.6.3 Cuckoo Search Algorithm

This metaheuristic algorithm was developed by Yand and Debb in 2009 [174]. It is a population based meta-heuristic algorithm based on parasitic behaviour of cuckoos combined with levy flights in order

to improve its performance [175, 176, 177, 178]. Cuckoo rules are:

1. Each cuckoo lays one egg at a time and dumps it into a random nest.
2. The best nest will carry the next generation.
3. If the eggs are discovered by a host bird, it then either kills it or leaves the nest.

The cuckoo performance is enhanced using levy flights formula as seen in Equation (2.107), where X_n is the new nest generated; X_1 is the randomly chosen nest; a_0 is the step size and $levy(\lambda)$ represents levy distribution.

$$X_n = X_1 + a_0 \oplus levy(\lambda) \tag{2.107}$$

The algorithm for cuckoo search algorithm is as follows:

1. Compute the initial population size to 100 and maximum iterations to 1000.
2. Use Equation (2.107) to generate the new nest and evaluate the error fitness function value.
3. Generate k_1 for X_1 and compare it with k_n.
4. If $k_1 > k_n$, then X_n replaces X_1.
5. Keep updating the iteration cycle.
6. The iteration steps end when the highest iterations are reached. The fitness value with least error is selected and used for filter design.

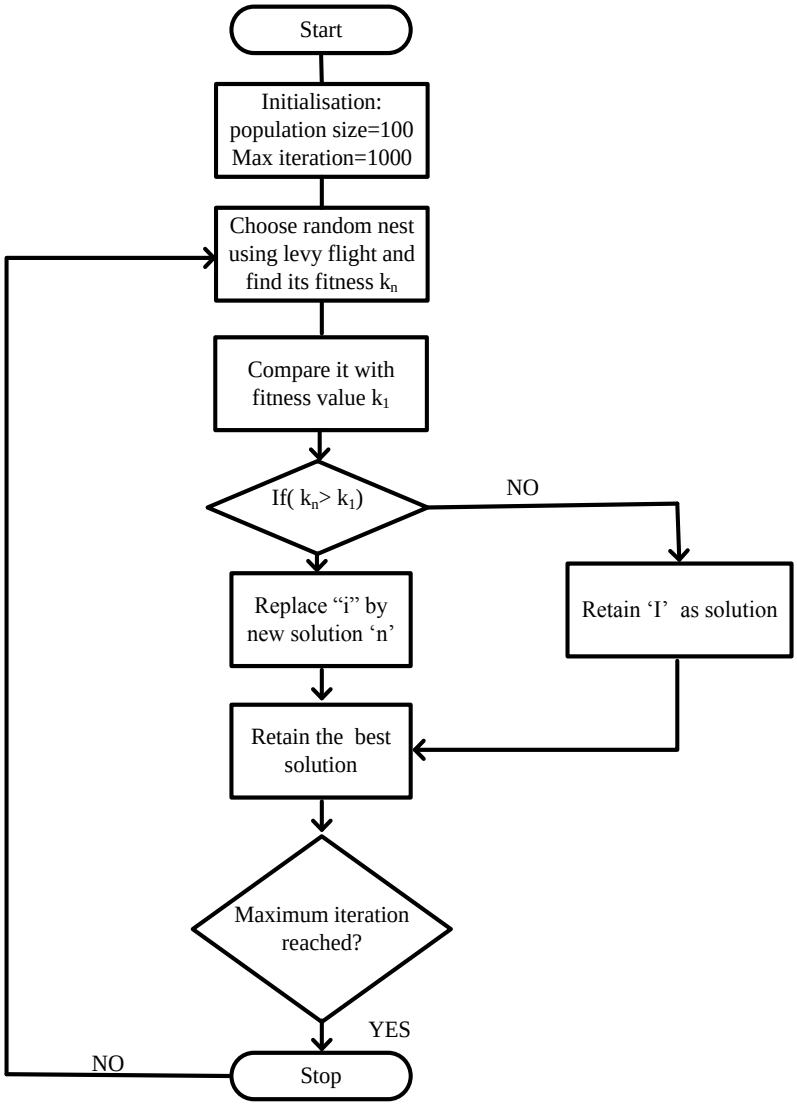

Figure 2.28. Flow Chart of Cuckoo Optimisation Algorithm.

2.7 CONCLUDING REMARKS

Inflexible constraints exhibited by analog technologies led to migration from analog design to digital radio. SDR is a radio technology in which the transmitter and receiver design can be implemented on programmable software before being transformed into hardware reprogrammable logic. It is a flexible technology that allows different air interfaces to take different forms for necessary modifications or alterations, as seen fit by the designer.

Flexibility, reconfiguration and re-programmability distinguish SDR from analog design. However, these features allow single mode single carrier, or multiple modes single carrier or multiple modes multiple carriers to be easily implemented on SDR. Different radio receiver architectures were studied in this Chapter together with their major drawbacks but super-heterodyne and direct conversion architectures are proven to be good candidates for SDR receiver design because they offer good selectivity and high sensitivity.

Also, different existing channelisation algorithms were reviewed with their computational complexity and reconfigurability. The uniform channelisation algorithms such as PC, pipelined/binary algorithm and DFT were studied in detail, whereas the non-uniform channelisation algorithms such as GDFT, PGDFT and RGDFT were exhaustively investigated. The polyphase implementation of each algorithm showed that it can be used to drastically reduce the computational efforts of the channelisation algorithms. Multi-rate filters such as decimator, interpolator, CIC and inverse sinc are among other alternatives to further lower the processing requirements of the channelisation algorithm. The choice of appropriate number system representation and optimisation algorithm can effectively reduce the computational complexities of a filter. Number system such as residue number system (RNS), Canonical signed digit (CSD), common sub- expression metbod (CSE) can reduce the word length and filter coefficient during digital signal operation. Implementing the number system with channelisation algorithm can reduce the filter characteristics such as stop band attenuation and pass band ripples. In order to maximize or minimize the filter characteristics, optimisation approaches are necessary. Such optimisation algorithms such as genetic algorithm, particle swarm and cuckoo search algorithm.

CHAPTER 3 HYBRID GDFT FILTER BANK

3.1 INTRODUCTION

Higher filter complexity experienced by GDFT filter bank is due to three phase modifications added to its time and frequency offset. This manifest as higher filter order and higher delay, which reduces its efficiencies in software defined radio (SDR) mobile systems.

High computational complexity experienced in generalised discrete Fourier transform filter bank (GDFT-FBs), as discussed in Chapter 2, Section 2.4.10 renders it unfit to handle the upcoming radio standards in software Defined Radio (SDR) mobile receiver.

The determining factors for the higher filter order is the vast number of multipliers consumed during the channelisation operations. Multipliers contribute remarkably to the complexity of digital filters and channelisation algorithms, and consequently slow down the computational speed, limit filter bank re-configurability, increase resources utilisation, increase production costs and power consumption.

Therefore, the goal of this work is to improve the performance of the generalised discrete Fourier transform, in terms of multipliers, adders, speed, area utilisation, and delay time. The chapter consists of two distinct sections. The first Section describes the development of hybrid generalised discrete Fourier Transform algorithm as an improvement over generalised discrete Fourier Transform channelisation algorithm.

The Section is outlined as follows: Section 3.2 describes Hybrid generalized discrete Fourier transform channelisation algorithm (HGDFT). The Section details the design steps for HGDFT channelisation algorithm.

In Section 3.2.4, case applications study for channelisation of three non-uniform input channels are investigated. The second Section mentions the various methods employed to improve the performance of the developed HGDFT channelisation algorithms with or without genetic algorithm. Section 3.3.1 discusses improvement methods using parallel distributed based residual number system (PDA-RNS). Section 3.3.3 exploits parallel distributed based canonical residual number system (PDA-CRNS as an alternative improvement methods for HGDFT filter, Section 3.3.4 explains in detail the parallel distributed based diminished one canonical residual number system (PDA-DCSRN), while Section 3.3.5 presents parallel distributed based hybrid common sub-expression residual number system (PDA-CSERNS). These improvement methods are further optimised using genetic algorithm and their performances compared with each other. Other similar channelisation algorithm in literature such as coefficient decimated filter bank (CDFB), improved coefficient decimated filter bank (ICDM FB) and non-maximally decimated filter bank (NMDUFB) are also used for benchmarking. Lastly, the chapter concludes with the concluding remarks.

3.2 HYBRID GENERALISED DISCRETE FOURIER TRANSFORM CHANNELISATION (HGDFT) ALGORITHM

Two distinct processes are involved in the development of hybrid generalised discrete Fourier transform (HGDFT) channelisation algorithm. The first one is the modulation of GDFT algorithms and the second method is the cascading of modulated GDFT with Frequency response masking filter (FRM), interpolation coefficient decimation filter.

3.2.1 Modulation of GDFT

This approach is a modification to the existing GDFT filter bank. Here, the input signal, $x_k(m)$ is obtained by modulating the existing GDFT filter bank by varying frequency offset, such that the in-phase and the quadrature phase are offset from each other by 90^0. The input signal is centred at frequencies, $\omega = \pm\frac{\omega_\Delta}{2}$, where ω_Δ is the width of the band. The input signal is related to existing GDFT filter bank as follows.

$$X_k'^{GDFT}(m) = Re\left[X_k^{GDFT}(m)cos\left(\frac{\omega_\Delta mM}{2}\right)\right] + Im\left[X_k^{GDFT}(m)sin\left(\frac{\omega_\Delta mM}{2}\right)\right] \tag{3.1}$$

where Re[.] denotes the real part of the quantity in the brackets and Im[.] denotes imaginary part of the quantity in the brackets, $X_k^{GDFT}(m)$ is the classical or base GDFT filter, $X_k'^{GDFT}(m)$ is the modulated GDFT filter, m is the time shift of mM samples, n_0 is the new reference from origin, M is

the decimation factor, k_0 is the new reference for discrete frequency channels and K is the transition size or band width. Thus, Equation 3.1 becomes Equation 3.2.

$$\begin{aligned} X_k^{\prime GDFT}(m) &= X_k^{GDFT}(m)e^{\frac{j\omega_\Delta mM}{2}} \\ X_k^{GDFT}(m) &= x_(n).\sum_{n=-M-1}^{N} h(mK-1).W_K^{(k+k_0)}(n+n_0) \\ X_k^{\prime GDFT}(m) &= x_(n).\sum_{n=-M-1}^{N} h(mK-1)W_K^{(k+k_0)(n+n_0)}e^{\frac{j\omega_\Delta mM}{2}} \end{aligned} \tag{3.2}$$

3.2.1.1 Variation of frequency offset, $k_0 = \frac{1}{4}$ for full modulation

Assuming the channel signals are critically and uniformly spaced in the frequency range of $0 \leq \omega \leq \pi$. This type of filter can be developed with $k_0 = \frac{1}{4}$ and $n_0 = 0$, so that the number of channels is equal to the transition size, K. In such situation, the decimation ratio M is analogous to the transition size K. This means that $M = K$ and the width of the channel ω_Δ and the channel center frequency ω_k are represented as shown in Equation 3.3.

$$\begin{aligned} \omega_\Delta &= \frac{\pi}{M} = \frac{\pi}{K} \\ \omega_k &= \frac{2\pi(k+k_0)}{K} \end{aligned} \tag{3.3}$$

Where ω_k is the center frequency of the channels and ω_Δ is the width of the frequency channels. By substituting Equation 3.3 into Equation 3.2, the newly generated GDFT filter will be as follows in Equation 3.4.

$$X_k^{\prime GDFT}(m) = \sum_{n=-M-1}^{N} h(mK-1).x(n).e^{-j(\frac{2\pi}{K})(K+\frac{1}{4})(n+n_0)}e^{j\frac{\pi m}{2}} \tag{3.4}$$

Further manipulation of these terms can be expressed in the form of Equation 3.5, where $h_k(n)$ is the filter impulse coefficient.

$$X_k^{\prime GDFT}(m) = \sum_{n=-M-1}^{N} h_k(mK-1)x(n).e^{-j(\frac{2\pi}{K})(K+\frac{1}{4})(mK-n-n_0)} \tag{3.5}$$

where

$$h_k(n) = 2h(n)cos\left[\frac{2\pi}{K}\left(k+\frac{1}{4}\right)(n-n_0)\right] \tag{3.6}$$

By evaluating Equation 3.5 the following terms are collated as shown in Equation 3.7.

$$\begin{aligned} X_k'^{GDFT}(m) &= \sum_{n=-M-1}^{N} h_k(mK-1)x(n).e^{-j(\frac{2\pi}{K})(kmK-kn-\frac{1}{4}mK-\frac{1}{4}n)} \\ X_k'^{GDFT}(m) &= \sum_{n=-M-1}^{N} h_k(mK-1)x(n)e^{-j2\pi Km}e^{j\frac{2\pi}{K}kn}.e^{j\frac{2\pi}{4}m}.e^{j\frac{2\pi}{4K}n} \end{aligned} \tag{3.7}$$

From Equation 3.7, the following can be deduced.

$$\begin{aligned} e^{j\frac{2\pi}{K}kn} &= 1 \\ e^{j\frac{2\pi}{4}m} &= j^m \\ W_K &= e^{j2\pi Kn} \end{aligned} \tag{3.8}$$

Thus, the classical generalized discrete Fourier transform filter bank can be reduced to the following filter as seen in Equation 3.9.

$$X_k'^{GDFT}(m) = \sum_{n=-M-1}^{N} h_k(mK-1)x(n).W_M^{\frac{1n}{4}}DFT \tag{3.9}$$

3.2.1.2 Variation of frequency offset, $k_0 = \frac{1}{2}$ for full modulation

The choice of the GDFT filter bank influences the channel staking arrangements and the number of channels in the filter. If $k = 0$, it means there will be $\frac{K}{2}$ uniform channels spanning the region of $0 \leq \omega \leq 0$. This implies that the full band is not utilised as there are $\frac{K}{2}$ independent channels. If $n_0 = 0$ and $k_0 = \frac{1}{2}$, GDFT is referred to as odd-stacked DFT channel and when $n_0 = \frac{1}{2}$ and $k_0 = \frac{1}{2}$, it is referred to as odd-squared DFT. In this situation, there are $\frac{K}{2}$ independent filter bank channels with center frequencies as indicated in Equation 3.10.

$$\begin{aligned} \omega_k &= \frac{2\pi}{K}\left(k+\frac{1}{2}\right) \\ \omega_\Delta &= \frac{2\pi}{K} \\ k &= 0,1,....,\frac{K}{2}-1 \end{aligned} \tag{3.10}$$

If the channel signal is critically sampled, the decimation ratio and the bandwidth are related as follows in Equation 3.11.

$$M = \frac{K}{2} \tag{3.11}$$

This is because there are only $\frac{K}{2}$ unique bands from the bandwidth in Equation 3.10. Using Equation 3.2, and replacing K_0 with $\frac{1}{2}$, the following Equation 3.12 is obtained.

$$X_k'^{GDFT}(m) = X_k^{GDFT}(m)e^{\frac{j\omega_\Delta mM}{2}} \tag{3.12}$$

$$X_k'^{GDFT}(m) = \sum_{n=-M-1}^{N} h_k(mK-1)x(n)e^{-j(\frac{2\pi}{K})(K+\frac{1}{2})(n+n_0)}e^{j\frac{\pi m}{2}} \tag{3.13}$$

where

$$\begin{aligned} h_k(n) = h(n)\Bigg(2cos\Bigg((2M+1)\frac{\pi}{2M}\Bigg[(N+2kM)-\frac{N}{2}+\Phi\Bigg]\Bigg) \\ +2jsin\Bigg((2M+1)\frac{\pi}{2M}\Bigg[(N+2kM)-\frac{N}{2}+\Phi\Bigg]\Bigg)\Bigg) \end{aligned} \tag{3.14}$$

By evaluating Equation 3.13, the following terms are collated.

$$\begin{aligned} X_k'^{GDFT}(m) &= \sum_{n=-M-1}^{N} h_k(mK-1)x(n).e^{-j(\frac{2\pi}{K})(kmK-kn-\frac{1}{2}mK-\frac{1}{2}n)} \\ X_k'^{GDFT}(m) &= \sum_{n=-M-1}^{N} h_k(mK-1)x(n)e^{-j2\pi Km}e^{j\frac{2\pi}{K}kn}.e^{j\frac{2\pi}{2}m}.e^{j\frac{2\pi}{2k}n} \end{aligned} \tag{3.15}$$

From Equation 3.15, it was observed that Equation 3.16 was obtained.

$$\begin{aligned} e^{j\frac{2\pi}{K}kn} &= 1 \\ e^{j\frac{2\pi}{4}m} &= (-1)^m \\ W_K = e^{j2\pi Kn} &= 1 \end{aligned} \tag{3.16}$$

Thus, the generalized discrete Fourier transform filter bank can be reduced to the following filter bank in Equation 3.17.

$$X_k'^{GDFT}(m) = (-1)^{km}\sum_{n=-M-1}^{N} h_k(mK-1)x(n) \tag{3.17}$$

If m is replaced by L and K is replaced by $2M$, and if linear phase prototype low-pass filter $H(z)$ of order N has a pass-band edge of $\theta_a = \frac{2m\pi-\omega_{\omega_\Delta}}{M}$ and stop-band edge of $\phi_a = \frac{2m\pi-\omega_{\omega_\Delta}}{M}$ with ω_Δ as the width of the transition band. For even multiples of number M of sub-bands, the length is $N+1$, that is, $N = (2LM-1)$.

Here, the impulse response, $h_k(mK-1)$, will be reduced as follows. It can be seen from Equation 3.14

that h_k contains terms that multiply h_n and this can be expressed as variable D as seen in Equation 3.18.

$$\begin{aligned} h_k(n) &= h(n)\left(2cos\left((2M+1)\tfrac{\pi}{2M}\left[(N+2kM)-\tfrac{N}{2}+\Phi\right]\right)\right. \\ &\left.+2jsin\left((2M+1)\tfrac{\pi}{2M}\left[(N+2kM)-\tfrac{N}{2}+\Phi\right]\right)\right) \end{aligned} \tag{3.18}$$

By employing symmetrical impulse response, where

$$\begin{aligned} &2cos\left((2M+1)\tfrac{\pi}{2M}[(N+2kM)-\tfrac{N}{2}+\Phi]\right) \\ &=\left(-1^k\right)2cos\left((2M+1)\tfrac{\pi}{2M}[(N+2kM)-\tfrac{N}{2}+\Phi]\right) \end{aligned} \tag{3.19}$$

Thus, Equation 3.18 becomes Equation 3.19.

The newly generated impulse response will now be represented as shown in Equation 3.20.

$$\begin{aligned} h_k(n) &= h(n)\left(2cos\left((2M+1)\tfrac{\pi}{2M}\left[(N+2kM)-\tfrac{N}{2}+\Phi\right]\right)\right. \\ &\left.+2jsin\left((2M+1)\tfrac{\pi}{2M}\left[(N+2kM)-\tfrac{N}{2}+\Phi\right]\right)\right) \\ h_k(n) &= h(n)\left(D_{m,n}+(-1^k)D(m,n)\right) \end{aligned} \tag{3.20}$$

The polyphase version of the analysis filter bank can be described as shown in Equation 3.21.

$$\begin{aligned} H_k(z) &= \sum_{n=0}^{N} h_k(n)z^{(2M)} \\ H_k(z) &= \sum_{l=0}^{L-1}\sum_{j=0}^{2M-1}\left[D_{m,n}+(-1^k)D(m,n)\right]h(2LM+j)z^{-(2LM+j)} \end{aligned} \tag{3.21}$$

Substituting j for n, and L for $K=2M$, rewriting Equation (3.21), the following Equation (3.22) is derived.

The prototype filter can be decomposed into $2M$ poly-phase components, as follows in Equation (3.22), where $S_j(z)=\sum_{i=0}^{L-1} h(2LM+j)z^{(-L)}$ are the poly-phase components of the filter $H(z)$.

$$\begin{aligned} H(z) &= \sum_{l=0}^{L-1} z^{-j}\sum_{j=0}^{2M-1}(D_{m,n}+(-1^k)D(m,n))h(2LM+j)z^{-(2LM+j)} \\ &= \sum_{l=0}^{L-1} z^{-j}\sum_{j=0}^{2M-1}(D_{m,n}+(-1^k)D(m,n))h(2LM+j)z^{(-2M-j)} \\ &= \sum_{j=0}^{2M-1}(D_1+D_2)z^{-j}S_j(z^{-2M}) \end{aligned} \tag{3.22}$$

From Equation (3.23), it can be shown that D_1 and D_2 are $M \times N$ matrices, whose (m, j) elements are $D_{m,j}$ and $D_{m,j+m}$, respectively for $m, j = 0, 1, ...(m-1)$.

$$H(z) = \begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ \vdots \\ H_{m-1}(z) \end{bmatrix} = \begin{bmatrix} D_1 & D_2 \end{bmatrix} \begin{bmatrix} S_0(z^{2M}) \\ z^{-1}S_1(z^{2M}) \\ \vdots \\ \vdots \\ z^{(-2m-1)}S_{2m-1}(z^{2M}) \end{bmatrix} \tag{3.23}$$

Representing $\delta(z)$ as shown in Equation (3.24).

$$\delta(z) = \begin{bmatrix} 1 & z^{-1} & \dots & z^{-M+1} \end{bmatrix}^T \tag{3.24}$$

The poly-phase representation of the matrix can be represented as in Equation (3.26), where $S(z)$ is the poly-phase matrix.

$$H(z) = \begin{bmatrix} D_1 & D_2 \end{bmatrix} \begin{bmatrix} S_0(z^{2M}) & & & 0 \\ & S_1(z^{2M}) & & \\ & & \ddots & \\ & & \ddots & \\ 0 & & & S_{2m-1}(z^{2M}) \end{bmatrix} \begin{bmatrix} \delta(z) \\ z^{-m}\delta(z) \end{bmatrix} \tag{3.25}$$

$$H(z) = \left\{ \begin{array}{l} D_1 \begin{bmatrix} S_0(z^{2M}) & & & 0 \\ & S_1(z^{2M}) & & \\ & & \ddots & \\ & & \ddots & \\ 0 & & & S_{2m-1}(z^{2M}) \end{bmatrix} \\ \qquad + \\ z^{-m}D_2 \begin{bmatrix} S_0(z^{2M}) & & & 0 \\ & S_{M+1}(z^{2M}) & & \\ & & \ddots & \\ & & \ddots & \\ 0 & & & S_{2m-1}(z^{2M}) \end{bmatrix} \end{array} \right\} \delta(z) \tag{3.26}$$

3.2.2 Hybrid Generalised Discrete Fourier Transform Channelisation (HGDFT) Algorithm

Although two methods were discussed to reduce the complexities of GDFT filter, we resort to use the second method explained in section 3.2.1.2 when $k_0 = \frac{1}{2}$. After improvements obtained from the modulated algorithm in Section 3.2.1, further work was done to achieve some improvements by hybridizing the modulated algorithm with the frequency response masking algorithm. This involves cascading the developed GDFT filter with frequency response masking interpolated coefficient decimated filter, in a process described as hybrid GDFT (HGDFT) filter. The HGDFT based channelisation algorithm consists of two branches: the upper and the lower branches. The upper branch is made up of frequency response masking (FRM) interpolated coefficient decimated filter and the masking filter, while the lower branch consists of complementary FRM interpolated coefficient decimated filter and the complementary masking filter.

A low-pass interpolated coefficient decimated linear phase FIR filter, $H_a(z^{\frac{L}{M}})$, is formed from the cascade of base interpolating filter, $H_a(z^L)$, and the coefficient decimating filter, $H_c(z^{1/M})$, to extract the sharp narrow-band channel of choice.

Also, a bandpass edge complementary interpolating coefficient decimated base filter, $H_c(z^{\frac{L}{M}})$, is formed from the cascade of complementary base interpolating filter, $H'_a(z^L)$, and the complementary coefficient decimating filter, $H'_c(z^M)$, to isolate multi-bands frequency responses. The low-pass interpolated coefficient base filter, $H_a(z^{L/M})$, cascades with the masking filter, $A_k(z)$, in the upper branch and the bandpass complementary interpolated coefficient base filter, $H_a(z^{L/M})$ cascades with the complementary masking filter, $B_k(z)$, in the lower branch to produce a reconfigurable low computational multi-narrow frequency bands. The desired passband, (ω_p), and (ω_s), cut off frequency of the base filter response, $H_a(z)$, is calculated as indicated in Table 3.1.

Table 3.1. The cut-off frequency of prototype, masking and complementary filter.

Parameter	Case 1	Case2
$H_a(z)$	$\theta_a = 2m\pi - \omega_s L/M$	$\theta_a = \omega_s L/M - 2\pi m$
	$\phi_a = 2m\pi - \omega_p L/M$	$\phi_a = \omega_p L/M - 2m\pi$
$H_{ma(z)}$	$\omega_{mp}=\frac{2\pi(m+1)-\phi_a M}{L}$	$\omega_{mp}=\frac{2\pi m-\phi_a M}{L}$
	$\omega_{ms}=\frac{2\pi m+\phi_a M}{L}$	$\omega_{ms}=\frac{2\pi m-\phi_a M}{L}$
	$\omega_{mcs}=\frac{2\pi m-\phi_a}{L}$	$\omega_{mcs}=\frac{2\pi m-\phi_a}{L}$

The transfer function of the FRM interpolated coefficient decimated filter is given by Equation (3.27).

$$H(z) = \frac{L}{M} H_a(z^{\frac{L}{M}}) H_{Ma}(z) + \frac{L}{M} H_c(z^{\frac{L}{M}}) H_{Mc}(z) \tag{3.27}$$

The interpolated coefficient decimated base and complementary filters are symmetrical and asymmetrical linear phase FIR filter and can be expressed as $H_a(z^{\frac{L}{M}}) = H_c(-z^{\frac{L}{M}})$. A half band filter is introduced into the FRM Interpolated coefficient decimated filter to further reduce its computation complexity. This is possible as a result of symmetrical properties possessed by half-band filter. The time-domain impulse response of the CD-1 technique has every other component to be zero except the components at the center. That indicates that it is symmetrical around the center. This translates to reduced complexity in terms of the number of the multiplies required by the filter.

The transfer function of the half band FRM interpolated coefficient decimated filter can be expressed in terms of two polyphase components as in Equation (3.28).

$$\begin{aligned} H_a(z^{\frac{L}{M}}) &= \tfrac{L}{M} H_{a0}(z^{\frac{2L}{M}}) + z^{-\frac{L}{M}} \tfrac{1}{M} H_{a1}(z^{\frac{2L}{M}}) \\ H_c(z^{\frac{L}{M}}) &= \tfrac{1}{M} H_{a0}(z^{\frac{2L}{M}}) - z^{-\frac{L}{M}} \tfrac{L}{M} H_{a1}(z^{\frac{2L}{M}}) \end{aligned} \tag{3.28}$$

The masking filters are replaced with two GDFT-FBs as shown in Figure 3.1.

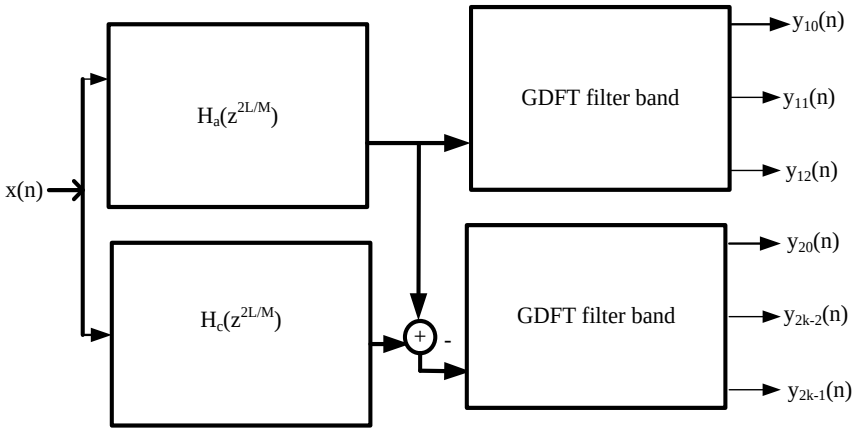

Figure 3.1. Block Diagram of FRM Interpolated Coefficient Decimated FIR Filter.

Also, the modulated GDFT poly phase filter is represented as shown in Equation (3.29).

$$\begin{aligned} H_{Ma}(z) &= \textstyle\sum_{n=0}^{2M-1} z^{-n}(D_1 + D_2) S_n(z^{2M}) \\ H_{Mc}(z) &= \textstyle\sum_{i=0}^{2M-1} z^{-n}(D_1 - D_2) S_n(z^{2M}) \end{aligned} \tag{3.29}$$

where $S_{AI}(z^k)$ and $S_{Bi}(z^k)$ are the k poly phase components of $H_{Ma}(z)$ and $H_{Mc}(z)$, respectively. The cascading of the modulation GDFT filter to the masking and complementary filter is shown in Equation (3.30).

$$H_k(z)=\frac{L}{M}\left[\left(H_{a0}(z^{\frac{2L}{M}})+z^{\frac{-2L}{M}}H_{a1}(z^{\frac{2L}{M}})\right)H_{Ma}(z)+\left(H_{a0}(z^{\frac{2L}{M}})-z^{\frac{-2L}{M}}H_{a1}(z^{\frac{2L}{M}})\right)H_{Mc}(z)\right] \tag{3.30}$$

But $H_{Ma}(z^{\frac{L}{M}})=H_{Mc}(-z^{\frac{L}{M}})$

The developed modulation and masking algorithm can be represented as shown in Equation (3.31) and the block diagram for hybrid GDFT masking filter is depicted in Figure 3.2.

$$H_k(z^{\frac{L}{M}})=\frac{2L}{M}\left[\left(H_{a0}(z^{\frac{2L}{M}})+z^{-\frac{2L}{M}}H_{a1}(z^{\frac{2L}{M}})\right)\sum_{i=0}^{2M-1}z^{-n}(D_1+D_2)S_n(z^{\frac{2L}{M}})\right] \tag{3.31}$$

Transition band of the HGDFT FB is centred at $\frac{\pi}{2}$ rad whereas the complementary filter bank is centred at $\frac{2\pi K}{M}$, where K is an integer ranging from 0 to $(M-1)$. The design used Parks-McClellan algorithm and the filter is realized using the direct transposed FIR in its implementation.

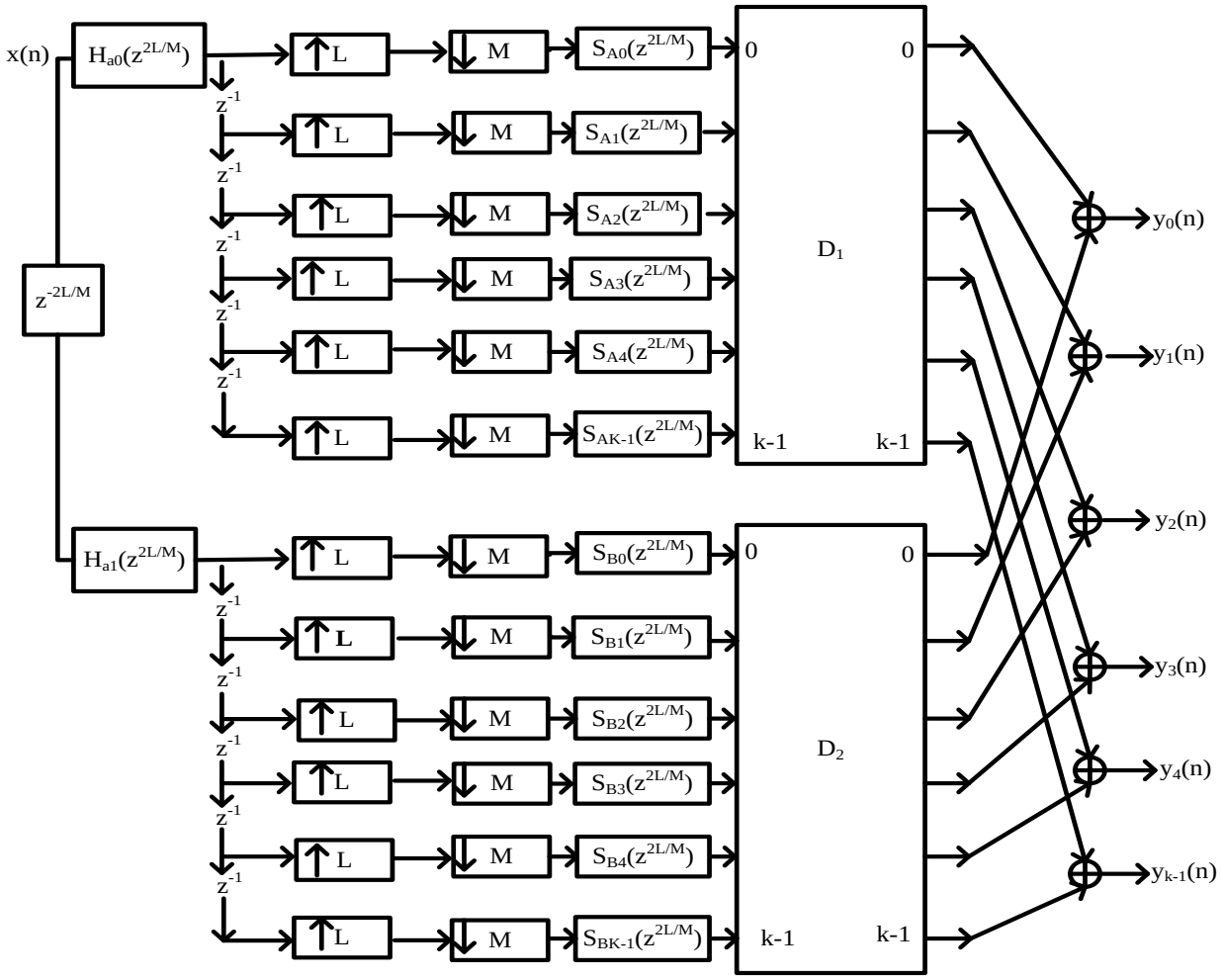

Figure 3.2. Diagram depicting HGDFT masking channelisation Algorithm.

3.2.3 HGDFT Channelisation Design Steps

The design steps for the proposed filter bank are outlined below:

1. Normalize all the channel bandwidths, (BWs), such that BW_i and transition bandwidth Δ_i specifications range from 0 to 1, and 1 corresponds to $\frac{f_s}{2}$, where f_s is the sampling frequency.
2. Vary the parameters m, n, N, L and M where N=$2LM-1$.
3. Express D_1 and D_2 as functions of Equation 3.19.
4. Determine poly phase components, $S_j(z)$ using impulse response in Equation 3.14.
5. Apply rectangular window to the ideal filter.
6. Determine the $H_m(z)$.
7. Calculate each channel stopband frequency such that $\omega_{s_i} = \frac{BW_i}{2}$.
8. Calculate the modal bandwidth such that BW_{Modal}=$\frac{GCD(BW_1',BW_2',BW_3')}{2}$. This corresponds to the modal stop band frequency.
9. Calculate the decimation factor,M, of the masking filter using this formula M=$\frac{\pi}{\omega_{s_i}}$. The interpolated factor is a factor calculated using the formula L=$\left\lfloor \frac{\pi}{\omega_{si}} \right\rfloor$, where ω_{si} is the stopband frequency for each channels. Thus the fractional factor for masking filter can be calculated here as $\frac{L_i}{M_i}$.
10. Calculate the decimator factor of the complementary filter using the formula M=$\frac{\pi}{\pi+\omega_{sk}}$. The interpolated factor is a factor calculated using the formula L=$\left\lfloor \frac{\pi}{\pi+\omega_{sk}} \right\rfloor$, where ω_{sk} is the stopband frequency for each complementary masking filter channels. Thus, the fractional rate for complementary filter can be calculated thus: $\frac{L_k}{M_k}$.
11. Determine transition bandwidth for masking and complementary filter, Δ_k such that:Δ_k'=$\Delta_k \times e^{-\frac{L_k}{M_k}}$ where $\frac{L_k}{M_k}$ is the fractional rate for masking or complementary filter.
12. Determine the base modal or complementary modal TBW as:
    Δ_{modal}=$\min(\Delta_1', \Delta_2', ..., \Delta_n')$. This corresponds to the modal transition width.
13. Calculate the modal, masking and complementary passband width using ω_p= $\omega_s - \Delta_{modal}$.
14. Determine the passband edge and stop band edge of the modal or prototype filter, masking and complementary filter using Table 3.1, where m= $\left\lfloor \frac{\omega_p \frac{L}{M}}{2pi} \right\rfloor$ for masking filter and m= $\left\lceil \frac{\omega_s \frac{L}{M}}{2pi} \right\rceil$ for complementary filter.
15. Find the stopband ripple using δ_{s1}'=$\delta_{s1}\frac{L_i}{M_i}$.
16. The modal passband peak ripple is calculated as: δ_{pmodal}=rounded($\min(\delta_{p1}', \delta_{p2}', ..., \delta_{pn}')$) .
17. The modal stopband peak ripple is calculated as: δ_{pmodal}=rounded($\min(\delta_{s1}', \delta_{s2}', ..., \delta_{sn}')$).
18. The cut off frequency of the prototype and masking filter are calculated using Table 3.1.

19. Determine the prototype filter order and the individual channels filter order using Bellanger formula as: $N = \frac{-2log_{10}(\delta_p' \delta_s')}{3\Delta_{TBW}}$ -1 [179].

3.2.4 Case Applications: Channelisation of Non-uniform input channels

Using Matlab 2020 as simulation tool, the developed hybrid GDFT algorithm was applied to Zigbee, Bluetooth (BT) and wideband code division multiplexers (WCDMA). Channel band width parameters used for BT, zigbee, and WCDMA were 1 MHz, 4 MHz, and 5 MHz, respectively. Also, BT, zigbee, and WCDMA transition width were specified as 50 kHz, 200 kHz, and 500 kHz, respectively. The pass band ripples and stop band attenuation for BT and Zigbee were specified as 0.1 and -40 dB, while WCDMA channels pass band ripples and stop band attenuation were specified as s0.1 and -55 dB, respectively. The algorithm procedure in Section 3.2.3 were implemented and the filter results were recorded.

The metrics used for the computational complexity are filter order, number of multipliers, stop band attenuation and pass band ripples. The results obtained were compared with designs that used coefficient decimation filter bank (CDFB) and improved coefficient decimation method (ICDM) for such channels [30, 66].

The following filter specifications are chosen for the input signals as seen in Table 3.2.

Table 3.2. The filter specification for case study under consideration.

Filter specifications	Sampling frequency $f_s(MHz)$	Channel Bandwidth (MHz)	Transition Bandwidth Δ_{TBW}(kHz)	Passband ripples δ_p(dB)	Stopband ripples δ_s(dB)
Bluetooth	40	1	50	0.1	-55
Zigbee	40	4	200	0.1	-40
WCDMA	40	5	500	0.1	-40

3.3 IMPROVEMENT OF HGDFT CHANNELISATION ALGORITHM

Moreover, in order for the SDR mobile receiver to accommodate the existing and upcoming technologies with its small size and power, there is a need to further improve the performances of HGDFT algorithm by reducing or totally eliminate the multipliers used. Addition and shift mechanisms are alternative measures to implement multipliers in digital filter design. Different number systems representation methods can be used to realize the addition and shift mechanism as indicated in Chapter 2.

This chapter focuses on improving the HGDFT channelisation algorithm using different number system representations method as illustrated in Section 2.5.1. The HGDFT algorithm is optimized using four of these

number systems. The improvements will be made on parallel distributed arithmetic based residue number systems (PDA-RNS), parallel distributed arithmetic based canonical residue number system (PDA-CRNS), parallel arithmetic based diminished canonical signed residue number system (PDA-DCRNS) and parallel distributed arithmetic based common sub-expression elimination methods (PDA-CSE).

3.3.1 Improvement of HGDFT with Parallel Distributed Arithmetic Based Residual Number System (HGDFT PDA-RNS FB)

Residual number system is a good option for realizing further improvement to HGDT channelisation algorithm because of its modular and carry free addition peculiarities as seen in [120, 121, 122].

In an attempt to lower filter complexity of the HGDFT channelisation algorithm, we proposed the design of the second approach known as the parallel distributive arithmetic based residual number systems (PDA-RNS).

3.3.2 Design and implementation of HGDFT PDARNS FB

Consider the input signal sampling rate of 40MHz of the filter design example in Section 3.2.4, and the filter specifications given in Table 3.2. Moduli set of 2^n-1, 2^n, 2^n+1 are selected from the literature because of its high speed and reduced hardware complexity. If the value of $n=5$ bits, the relevant moduli set based on the moduli format will be $31, 32, 33$. The filter coefficients of the filter generated in Table 3.2 are converted into floating point integer or decimal values. The integer residual values are transformed from floating-point into fixed point values in these two steps. These involve quantizing and discretising the floating points values, b_i using MATLAB functions known as quantizer and b_{binary}=$num2bin(Q,1,b)$. The values of the parameter format create a parameter of binary numbers: word length, fractional length for signed fixed-point mode. The input signal and the filter coefficients used 16 bits precision format, with the parameter word length of 16 and fractional length=15.
Distributed arithmetic can be expressed as shown in Equation (3.32).

$$y_i = \sum_{i=1}^{D-1} H_i X_{i,j}[j] \tag{3.32}$$

The fixed-point binary values of the input signals, $X_{i,j}$, and the filter coefficients, H_i, are converted into the residue form using the arbitrary forward converter mechanism outlined below. Forward converter considers input signal whose period is t, with its equivalent binary numbers whose residues are sought for, and with its residues partitioned into t-bit blocks.
When DA inputs are converted to RNS, then Equation (3.32) becomes Equation (3.33).

$$y_i = \left| \sum_{j=0}^{D-1} \sum_{i=1}^{K} H_i \cdot X_{i,j}[j] \right|_p 2^j \tag{3.33}$$

In order to determine the residues, r_i, relative to modulus 2^n-1, 2^n, 2^n+1 respectively, the total residue, X, is calculated as total sum of the partitioned bits blocks with respect to the chosen modulus as depicted in Equation (3.34).

$$\begin{aligned} r_1 &= |X|_{2^n} \\ r_2 &= |X|_{2^n-1} \\ r_3 &= |X|_{2^n+1} \end{aligned} \tag{3.34}$$

The reverse converter converts the residual values to binary number and this operation takes place at the back end of the architecture using the shift addition method, while the principle of the Chinese remainder theorem (CRT) is used for its implementation. Assuming that, there are three residues, m_1, m_2 and m_3 with three moduli sets of 2^n-1, 2^n and 2^n+1 respectively as shown in Equation (3.35).

$$\begin{aligned} m_1 &= 2^n \\ m_2 &= 2^n-1 \\ m_3 &= 2^n+1 \end{aligned} \tag{3.35}$$

The reverse converter method is based on the projection of one modulus on the remaining moduli set as illustrated. Projection of m_1 on m_2 and m_3 as given in Equation (3.36) and Equation (3.37).

$$\begin{aligned} \widehat{m_2} &= \frac{m_2}{|m_2|_{m1}} \\ &= 2^n-1.\frac{1}{|2^n-1|_{2^n}} \\ &= 2^n-1.\frac{1}{|2^n.2^n-1+1|_{2^n}} \\ &= 2^n-1.\frac{1}{2^n-1+1} \\ &= 2^n-1 \end{aligned} \tag{3.36}$$

$$\begin{aligned} \widehat{m_3} &= \frac{m_3}{|m_3|_{m1}} \\ &= 2^n+1.\frac{1}{|2^n+1|_{2^n}} \\ &= 2^n+1.\frac{1}{|2^n+1-1|_{2^n}} \\ &= 2^n+1.-1 \\ &= -2^n+1 \end{aligned} \tag{3.37}$$

Projection of m_1 on m_2 and m_3 is given in Equation (3.38).

$$\begin{aligned} P_1 &= |m_1.\widehat{m_2}.\widehat{m_3}|_{m1} \\ &= |2^n.2^n - 1. - (2^n + 1)|_{2^n} \\ &= 2^n(2^n - 1) \\ &= -(2^{n-1} + 2) \end{aligned} \tag{3.38}$$

Projection of m_2 on m_1 and m_3 is given as in Equation (3.39), Equation (3.40) and Equation (3.41) respectively.

$$\begin{aligned} \widehat{m_1} &= \frac{m_1}{|m_1|_{m2}} \\ &= 2^n.\frac{1}{|2^n|_{2^n-1}} \\ &= 2^n.\frac{1}{|2^n|_{(2^n-1)}} \\ &= 2^n.\frac{1}{|(2^n+1-1|_{(2^n-1)}} \\ &= 2^n \end{aligned} \tag{3.39}$$

$$\begin{aligned} \widehat{m_3} &= \frac{m_3}{|m_3|_{m2}} \\ &= 2^n + 1.\frac{1}{|2^n+1|_{2^n-1}} \\ &= 2^n + 1.\frac{1}{|2^n-1+2|_{2^n-1}} \\ &= 2^n + 1.\frac{1}{2} \\ &= \frac{2^n+1}{2} \\ &= \frac{1}{2}(2^n + 1) \end{aligned} \tag{3.40}$$

$$\begin{aligned} P_2 &= |m_2.\widehat{m_1}.\widehat{m_3}|_{m2} \\ &= \frac{1}{2}|2^n - 1.2^n.2^n + 1|_{2^n-1} \\ &= \frac{1}{2}(2^n(2^n - 1)) \\ &= \frac{1}{2}(-2^{2n} + 2) - (2^{2n} - 2) \end{aligned} \tag{3.41}$$

Projection of m_3 on m_1 and m_2 is given as shown in Equation (3.42), Equation (3.43) and Equation (3.44).

$$\begin{aligned}\widehat{m_1} &= \frac{m_1}{|m_1|_{m3}} \\ &= 2^n.\frac{1}{|(2^n+1-1)|_{2^n+1}} \\ &= -2^n\end{aligned} \tag{3.42}$$

$$\begin{aligned}\widehat{m_2} &= \frac{m_2}{|m_2|_{m3}} \\ &= 2^n-1.\frac{1}{|2^n-1|_{2^n+1}} \\ &= \frac{2^n-1}{-2} \\ &= \frac{-1}{2}(2^n-1)\end{aligned} \tag{3.43}$$

$$\begin{aligned}P_3 &= |m_3\widehat{m_1}\widehat{m_2}|_{m3} \\ &= |(2^n+1)(-2^n)(\frac{-1}{2}(2^n-1))|_{2^n+1} \\ &= \frac{-1}{2}(2^n(2^n-1)) \\ &= -2^{2n}-2\end{aligned} \tag{3.44}$$

The final projection is calculated as shown in Equation (3.45).

$$\begin{aligned}X &= p_1r_1+P_2r_2+P_3r_3 \\ &= 2^n(2^n-1)r_1+\frac{1}{2}(2^n(2^n-1))r_2 \\ &-\frac{1}{2}(2^n(2^n-1))r_3\end{aligned} \tag{3.45}$$

By scaling down X by 2^n, Equation (3.46) is obtained.

$$\begin{aligned}\frac{X}{2^n} &= (2^n-1)r_1+\frac{1}{2}(2^n-1)r_2 \\ &-\frac{1}{2}(2^n-1)r_3\end{aligned} \tag{3.46}$$

Let $r_1 = 0...0b_{1,k-1}b_{1,k-2}...b_{1,2}b_{1,1}b_{q1,0}$.

$2^n-1\times r_1$ can be carried out by one right shift of r_1 as seen in Equation (3.47).

$$2^n-1\times r_1 = b_{1,0}0...00b_{1,k-1}b_{1,k-2}...b_{1,2}b_{1,1} \tag{3.47}$$

Let $r_2 = 0...0b_{2,k-1}b_{2,k-2}...b_{2,2}b_{2,1}b_{2,0}$.

$2^n - 1 \times r_2$ can be carried out by one right shift of r_2 shown in Equation (3.48).

$$2^n - 1 \times r_2 = b_{2,0}0...00b_{2,k-1}b_{2,k-2}...b_{2,2}b_{2,1} \tag{3.48}$$

Also, $\frac{1}{2}(2^n - 1)r_2$ is performed by one right shift as given in Equation (3.49).

$$\frac{1}{2}(2^n - 1)r_2 = b_{2,1}b_{2,0}\underbrace{0...00}_{\text{2n-1}}b_{2,k-1}b_{2,k-2}...b_{2,3}b_{2,2} \tag{3.49}$$

Let $r_3 = 0...00b_{3,k-1}b_{3,k-2}...b_{3,2}b_{3,1}b_{3,0}$

$2^n \times r_3$ can be done by left shift of r_3 $'n'$ times as shown in Equation (3.50).

$$2^n \times r_3 = b_{3,k-1}b_{3,k-2}...b_{3,2}b_{3,1}b_{3,0}00...0 \tag{3.50}$$

Also, $2^n \times r_3$ can be carried out by left circular shift of r_3 as shown in Equation (3.51).

$$2^n \times r_3 = 0...00b_{3,k-1}b_{3,k-2}...b_{3,2}b_{3,1}b_{3,0} \tag{3.51}$$

$2^n - 1 \times r_3$ can be carried out by one right shift of r_3 as given in Equation (3.52).

$$2^n - 1 \times r_3 = b_{3,0}0...00b_{3,k-1}b_{3,k-2}...b_{3,2}b_{3,1} \tag{3.52}$$

Also, $\frac{-1}{2}(2^n - 1)r_3$ is performed by one right shift as given in Equation (3.53).

$$\frac{-1}{2}(2^n - 1)r_3 = -(b_{3,1}b_{3,0}\underbrace{0...00}_{\text{2n-1}}b_{3,k-1}b_{3,k-2}...b_{3,3}b_{3,2}) \tag{3.53}$$

$$-r_3 = \tilde{b_{3,1}}\tilde{b_{3,0}}1...1\tilde{b_{3,k-1}}\tilde{b_{3,k-2}}...\tilde{b_{3,3}}\tilde{b_{3,2}} \tag{3.54}$$

The 2^k possible values of the r_1, r_2, and r_3 are precomputed and stored in a $2^k \times D$-bit LUT. After the residual values are computed, Equation (3.33) becomes Equation (3.55).

$$\begin{aligned} y_i &= \sum_{k=0}^{K-1} H_i X_{i,j}[j] \\ &= \sum_{j=0}^{J-1}\sum_{k=0}^{K-1} |2^{jp} B_j|_m H_i \\ &= \sum_{j=0}^{J-1}\sum_{k=0}^{K-1} |r_{i,j}|_m H_i \end{aligned} \tag{3.55}$$

Without loss of generality, the residual number r_k can be represented as indicated in Equation (3.56).

$$\begin{aligned} r_k &= -r_{k0}2^0 + \sum_{b=1}^{N-1} r_{kb}2^b \\ &= -r_{k0}2^0 + r_{k1}2^1 + \ldots + r_{k(N-1)}2^{N-1} \end{aligned} \tag{3.56}$$

The sum of the three residues are computed as follows in Equation (3.57).

$$\begin{aligned} r_k &= \sum_{k=0}^{K-1} (|2^{jp} B_j|_m) \\ &= \sum_{k=0}^{K-1} (-r_{k0}2^0 + \sum_{b=1}^{N-1} r_{kb}2^b) H_k \\ &= \sum_{k=0}^{K-1} (-r_{k0}2^0 + r_{k1}2^1 + \ldots + r_{k(N-1)}2^{N-1}) H_k \end{aligned} \tag{3.57}$$

Rearranging the terms in Equation (3.65) yields Equation (3.58).

$$y = -\sum_{k=0}^{K-1} H_k 2^0 + \sum_{b-1}^{N-1} 2^b \sum_{k=0}^{K-1} r_{kb} H_k \tag{3.58}$$

For K=2 and N=3, the rearrangement forms the following entries in the ROM as shown in Equation (3.59).

$$\begin{aligned} y = &(-r_{00}H_0 + r_{10}H_1 + r_{20}H_2)2^0 \\ &+(-r_{01}H_0 + r_{11}H_1 + r_{21}H_2)2^1 \\ &+(-r_{02}H_0 + r_{12}H_1 + r_{22}H_2)2^2 \end{aligned} \tag{3.59}$$

The input values and filter coefficients pre stored in the LUT tables are partitioned into different LUT tables and modulo accumulator (ACC) performs the modulo shift accumulate operation to generate y_i in D cycles as shown in Figure 3.3.

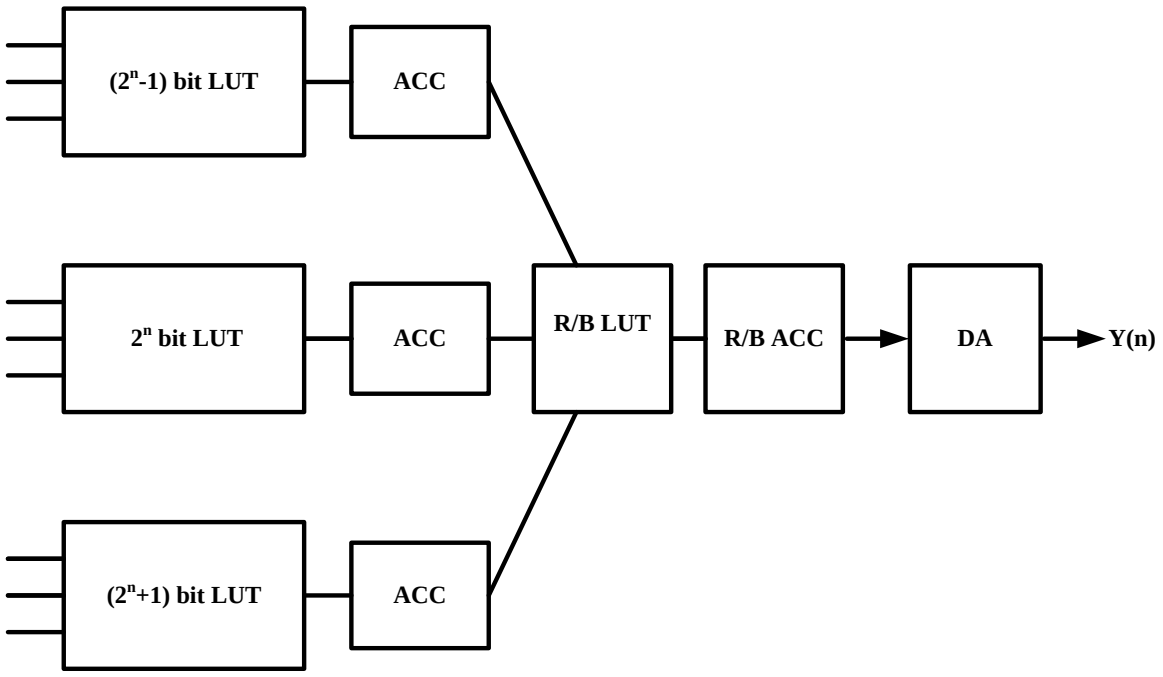

Figure 3.3. Distributed Arithmetic of the three moduli RNS filters.

3.3.2.1 OPTIMISATION OF IMPROVED HGDFT WITH GENETIC ALGORITHM

The results obtained from HGDF PDARNS FB are optimised further using genetic algorithm. In this way, the passband and the stopband characteristics are taken care of by minimizing the passband ripples and maximizing the stopband attenuation. The objective function is given as in Equation (3.60), where ω_s and ω_p are the passband ripples and stopband attenuation respectively.

$$\begin{aligned} &|max|H(e^{jw})| - 1| \\ &\text{where } \omega < \omega_p \\ &|H(e^{jw})| \\ &\text{where } \omega > \omega_p \end{aligned} \tag{3.60}$$

The objective function is to optimise the values above the restricted constraints. Two phases involved in the optimisation procedures are considered. The first phase is the optimisation of the prototype filter and the second phase involves the optimisation of the masking filter as discussed in the next two Subsections.

3.3.2.2 Phase 1: Optimisation of the prototype filter

The following procedures are required to optimise the prototype filter.

1. Initialisation: The initial chromosomes are produced by joining the first half of the continuous filter coefficients of the prototype filter. This is rounded to the nearest RNS representation with non-zero bit set to five. Random sampling of the initial chromosome resulted in a population pool of $N-1$ chromosomes, where N is the population size. The coefficients of these $N-1$ filters are converted into RNS representatives with restricted number of non-zero bits.
2. Fitness Evaluation of the existing chromosomes: Objective function generated in Equation (3.60) is used for evaluating each chromosome after which it is then ranked.
3. Selection Criteria: Most ranked chromosomes are selected from the population and used to form the mating pool. Roulette wheel rank selection is used for selecting the mates for cross-selection and mating.
4. Cross-over: Two points cross over are used in which the genes are hybridised between the parents.
5. Mutation: The best solution from the current population is propagated to the next population while the remaining chromosomes are mutated with the new information and propagated to the next generations.
6. Fitness Evaluation of the new population: Each chromosome in the new population is evaluated using the objective function in Equation (3.60) and then ranked. Looping of these steps continues until the maximum number of iterations are reached and GA is terminated. The mostly ranked chromosome is chosen from the population and decoded as the optimum RNS representation.

3.3.2.3 Phase 2: Optimisation of the masking filter

The masking filter of each channel is jointly optimised, in which the initial chromosome is generated by concatenating the first half of the continuous filter coefficients of all the masking filters of the channels. The steps in Section 3.3.2.2 are repeated until the maximum number of iteration is reached. Table 3.3 and Table 3.4 are the simulation parameters used for the prototype filter and the masking filters during the genetic algorithm optimisation.

Table 3.3. The Parameters used for Optimisation of the Prototype filters.

Number of iteration	500
Population size	100
Number of population members that survive each generation	1
Mutation rate	0.02
Number of the best population which is kept without change during mutation	10

Table 3.4. The Parameters used for Optimisation of the Masking filters.

Number of iteration	500
Population size	50
Number of population members that survive each generation	5
Mutation rate	0.2
Number of the best population which is kept without change during mutation	10

3.3.3 Improvement of HGDFT With Parallel Distributed Arithmetic Based Canonical Signed Residual Number System (HGDFT PDA-CSRNS FB)

The second method for improving HGDFT channelisation algorithm is using parallel distributed arithmetic based canonical signed residual number system (PDA-CSRNS), which is the hybrid of canonical signed digit (CSD) and RNS. Chapter 3 comprehensively explains these two theories as number system representations.

Canonical signed digit (CSD) has the tendency to reduce the number of consecutives non-zero bits in a binary number system. In doing so, the number of redundant number of one's generated during partial products are reduced to the minimum. Application of HGDFT FB with parallel distributed arithmetic based canonical residue number systems (PDA-CRNS) to the case study under consideration in Section 3.2.4, can be proven to further reduce the computational complexity of the system.

3.3.3.1 Design and Implementation of HGDFT PDA-CSRNS

Consider the input signal sampling rate of 40MHz applied to the filter design example in Section 3.2.4, and the filter specifications given in Table 3.2. The design is an enhancement of the design in Section 3.3.1. After the residual values are computed, Equation (3.32) becomes Equation (3.61).

$$\begin{aligned} y_i &= \sum_{k=0}^{K-1} H_i \dot{X}_{i,j}[j] \\ &= \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} |2^{jp} B_j|_m H_i \\ &= \sum_{j=0}^{J-1} \sum_{k=0}^{K-1} |r_{i,j}|_m H_i \end{aligned} \tag{3.61}$$

Without loss of generality, let us assume that r_k is a N-bit residual of Q1, (N-1) format number, such that Equation 3.62 holds.

$$\begin{aligned} r_k &= -r_{k0}2^0 + \sum_{b=1}^{N-1} r_{kb}2^b \\ &= -r_{k0}2^0 + r_{k1}2^1 + \ldots + r_{k(N-1)}2^{N-1} \end{aligned} \tag{3.62}$$

The bits generated from the residue's values are scanned from the least significant bits to the most significant bits. Any occurrence of continuous non-zero bits sequences are replaced by a pair of non-zero positive and negative bits. In this way, there is reduction in the number of consecutive one's which leads to reduction in the number of computations and additions required for implementation.

Thus, CSD can be represented as given in Equation (3.63).

$$\begin{aligned} C_i &= \sum_{i=0}^{N-1} r_i 2^i \\ &\text{for } \mathrm{r}_i \in -1, 0, 1 \end{aligned} \tag{3.63}$$

The total sum of the residues in Equation (3.62) using DA algorithm can be written as in Equation (3.64).

$$\begin{aligned} y &= \sum_{k=0}^{K-1} (|2^{jp} B_j|_m) \\ &= \sum_{k=0}^{K-1} (-C_{k0}2^0 + \Sigma_{b=1}^{N-1} C_{kb}2^b) H_k \\ &= \sum_{k=0}^{K-1} (-C_{k0}2^0 + C_{k1}2^1 + \ldots + C_{k(N-1)}2^{N-1}) H_k \end{aligned} \tag{3.64}$$

Rearranging the terms yields Equation (3.65).

$$y = -\sum_{k=0}^{K-1} C_{k0}H_k 2^0 + \sum_{b-1}^{N-1} 2^b \sum_{k=0}^{K-1} C_{kb}H_k \tag{3.65}$$

For K=2 and N=3, the rearrangement forms the following entries in the LUT as given in Equation (3.66).

$$\begin{aligned} y = &(-C_{00}H_0 + C_{10}H_1 + C_{20}H_2)2^0 \\ &+(-C_{01}H_0 + C_{11}H_1 + C_{21}H_2)2^1 \\ &+(-C_{02}H_0 + C_{12}H_1 + C_{22}H_2)2^2 \end{aligned} \tag{3.66}$$

The canonical values for the input values and filter coefficients pre-stored in the LUT tables are partitioned into different LUT tables and modulo accumulator (ACC) performs the modulo shift accumulate operation to generate y_i in D cycles as shown in Figure 3.3. Performance comparison is evaluated in terms of the resource's utilisation, power consumption, speed and delay.

The results obtained from HGDF PDACSRNS FB are further optimised using genetic algorithm. In this way, the same objective function, as well as the prototype and masking parameters as in Section 3.3.2.1 were used. Exception occurs when the initial coefficients of $N-1$ filters are converted into CSD representatives with restricted number of non-zero bits. The best chromosome is taken from the population and decoded to get the optimum CSRNS representation.

3.3.4 Improvement of HGDFT Channelisation Algorithm with Parallel Distributed Arithmetic Based Diminished One Canonical Signed Residual Number System (HGDFT PDA-DCRNS FB)

The performance of RNS filter can be optimized by using diminished one number system. Diminished one (DI) number system was firstly proposed by Leibowitz in [180]. Each number, X, is decremented by 1 and it is therefore effective for implementing modulo 2^n+1 binary addition.

The modulo 2^n+1 takes in $(n+1)$-bits operands unlike other remaining modulo such as 2^n and 2^n-1 which allows n-bit operands. Diminished 1 number system alleviates the problem of additional bits required in modulo 2^n+1 by allowing the additional bits to be one when the number to be represented is zero, which is achieved by subtracting one from the normal binary system. If X' represents the diminished one representation of the normal binary number as given in Equation (3.67), such that $X \in 0, 2^n$, where $X' \neq 0$, and $X' \in [0, 2^n-1]$ is an $n-bit$ numbers, therefore, $n+1$-bit circuits can be avoided.

$$X' =< X-1 >_{2^n+1} \tag{3.67}$$

The representation in Equation 3.67 can further lower the complexity of HGDFT filter.

3.3.4.1 Design and Implementation of HGDFT-DCRNS Channelisation Algorithm

The HGDFT PDA-DCRNS filter utilised three moduli set. Consider the 3 moduli set represented as in Equation (3.68), with dynamic range of $3n$ bits.

$$\begin{aligned} m_1 &= 2^n \\ m_2 &= 2^n-1 \\ m_3 &= 2^n+1 \end{aligned} \tag{3.68}$$

The first two moduli set remain intact while the third modulo set will be replaced with diminished-1 modulo. The decoded binary number is represented as in Equation (3.69).

$$\begin{aligned} X &= Y.2^n + X_1 \\ Y &= <(-2^{2N-1} + 2^{n-1})_{X3} \\ &\quad + (2^{2N-1} + 2^{n-1})_{X2} - 2^n X_1 >_{2^{2n-1}} \end{aligned} \tag{3.69}$$

The procedure below shows the realization of PDA-DCRNS filter. The residual number representation of the three numbers is given in Equation (3.70).

$$\begin{aligned} &\text{Let A} = < -2^n X_1 >_{2^{2n}} \\ &\text{Let B} = <(2^{2N-1} + 2^{n-1})_{X2} >_{2^{2n}} \\ &\text{Let C} = <(-2^{2N-1} + 2^{n-1})_{X3} >_{2^{2n}} \end{aligned} \tag{3.70}$$

Assuming that, X_1, is expressed in 2n bits, where the n-most significant bits are zero, then Equation (3.71) is given as, where the negative value is the second complement of the number.

$$\begin{aligned} X_1 &= \underbrace{00..0}_{\text{n}} \underbrace{X_{1,n-1}X_{1,n-2}...X_{1,0}}_{\text{n}} \\ A &= \bar{X}_{1,n-1} + \bar{X}_{1,n-2}... + \bar{X}_{1,0\text{n}}11...1_{\text{n}} \end{aligned} \tag{3.71}$$

The 2^n bit expression of X_2 is shown in Equation (3.72).

$$\begin{aligned} X_2 &= \underbrace{00..0}_{\text{n}} \underbrace{X_{1,n-1}X_{1,n-2}...X_{1,0}}_{\text{n}} \\ B &= X_{2,0}\underbrace{0...0}_{\text{n}}\underbrace{X_{2,n-1}...X_{2,1}}_{\text{n-1}} + \underbrace{X_{2,n-1}...X_{2,0}}_{\text{2n}}\underbrace{0...0}_{\text{n-1}} \\ &= \underbrace{X_{2,0}X_{2,n-1}...X_{2,0}}_{\text{n}}\underbrace{2_{n-1}...X_{2,1}}_{\text{n-1}} \end{aligned} \tag{3.72}$$

If residual number, X_3, is encoded in diminished-1 form as in Equation (3.73).

$$X_3 = X_3 + 1 \tag{3.73}$$

Then values of C will be given as in Equation (3.74).

$$\begin{aligned} C &= <(-2^{2N-1} + 2^{n-1})_{X'3} >_{2^{2n-1}-k} \\ K &= 2^{2N-1} - 2^{n-1} = \underbrace{01...1_{\text{n}}}\underbrace{0...0}_{\text{n-1}} \end{aligned} \tag{3.74}$$

Assuming that the $(n+1)$ bit expression of X'_3 is given in Equation (3.75).

$$x'_3 = \underbrace{x'_3 X'_{3,n-1} X'_{3,n-2} \ldots\ldots X'_{3,0}}_{\text{n}} \tag{3.75}$$

Therefore, parameter C can be evaluated as given in Equation (3.76).

$$\begin{aligned}
C &= < (-2^{2n-1} + 2^{n-1})_{X'3} >_{2^{2n-1}} \\
&= < -(X'_{3,0} \underbrace{0 \ldots 0}_{\text{n}} \underbrace{X'_{3,n-1} \ldots X'_{3,1}}_{\text{n-1}}) + 0 \underbrace{X'_{3,n-1} \ldots X'_{3,0}}_{\text{n}} \underbrace{0 \ldots 0}_{\text{n-1}} >_{2^{2n-1}} \\
&= < \underbrace{\bar{X}'_{3,0} X'_{3,n-1} \ldots X'_{3,0}}_{\text{n}} \underbrace{\bar{X}'_{3,n-1} \ldots \bar{X}'_{3,2}}_{\text{n-1}} + \underbrace{01 \ldots 1}_{\text{n}} \underbrace{0 \ldots 00}_{\text{n-1}} >_{2^{2n-1}} \\
&= < \underbrace{\bar{X}'_{3,0} X'_{3,n-1} \ldots X'_{3,0}}_{\text{n}} \underbrace{\bar{X}'_{3,n-1} \ldots \bar{X}'_{3,2}}_{\text{n-1}} + K >_{2^{2n-1}} \\
&\text{and } < (-2^{2N-1} + 2^{n-1})_{2^n X'3} >_{2^{2n-1}} = 0 \underbrace{X'_{3,n-1} \ldots X'_{3,0}}_{\text{n}} \underbrace{0 \ldots 0}_{\text{n-1}} >_{2^{2n-1}} \\
&\text{Finally}, C = \underbrace{\bar{X}'_{3,0} X'_{3,n-1} \ldots X'_{3,0}}_{\text{n}} \underbrace{\bar{X}'_{3,n-1} \ldots \bar{X}'_{3,2}}_{\text{n-1}}
\end{aligned} \tag{3.76}$$

To implement diminished-1 CSD on the HGDFT filter, the filter coefficient is quantized to binary number. The quantised binary number is partitioned into three sub-filters. The partitioned sub-filter contains the three RNS modules, 2^n, 2^n-1 and 2^n+1. The last sub-filter which contain the values of RNS module 2^n+1 are converted into diminished 1 number system. Finally, the partial products generated from the three moduli set inside the LUT are reduced using CSD.

The results obtained from HGDF PDA-DCRNS FB are optimised further using genetic algorithm. In this way, the same objective function, as well as the prototype and masking parameters as in Section 3.3.2.1 were used. Exception occurs when the initial coefficients of $N-1$ filters are converted into DCRNS representatives with restricted number of non-zero bits. The best chromosome is selected from the population and decoded to get the optimum DCRNS representation.

3.3.5 Improvement of HGDT With Parallel Distributed Arithmetic Based Common Sub-Expression Elimination RNS (HGDFT PDA-CSERNS) Techniques

We have seen that the performance of HGDFT filters are affected by multipliers and adders. Adders and shifters can be used to replace the multipliers. The only way to reduce or eliminate redundancies in multiplier is total migration to adders. However, the exponential growth rate of adders complicate the computation of the filter. It is therefore necessary to reduce the logic depth of adders, to improve the performance of HGDFT filter.

Common Sub-expression elimination methods are used to reduce the logical depth of adders and its critical paths by using a particular set of bit patterns. In order to reduce redundancy, the bit wise patterns are thus configured that the most common factored of the binary pattern of the canonical signed digit patterns are used in the filter

design. This Section focused on using the hybrid of common sub-expression methods on the bits obtained from the canonical residual number system as explained in Section 3.3.3.

The design in this thesis uses a hybrid of horizontal and vertical sub-expression elimination techniques to reduce the complexity of the filter multiplier. From the hybrid parallel canonical residual number method in Section 3.3.3, the following binary bit patterns were obtained. The hybrid common sub- expressions elimination method (HCSE) uses both the horizontal common sub-expressions (HCSs) and vertical common sub-expressions (VCSs) pattern bits. The common factored bits pattern used in HCSs are:[01], [101], [1001], [1101], [1100], [10001] while the common factored bits pattern in VCSs are [11], [111] and [1111]. These hybrid pattern together with their negated versions are used for the filter realization as indicated.

$$X_2 = 101 = 2$$

$$X_3 = [1001] = 3$$

$$-X_3 = [100\bar{1}] = -3$$

$$X_4 = [10001] = 4$$

$$-X_4 = [1000\bar{1}] = -4$$

$$X_5 = [10] = 5$$

$$X_6 = [01] = 6$$

$$X_7 = [11] = 7$$

$$X_8 = [111] = 8$$

$$X_6 = [1111] = 9$$

The reuse of the CS's obtained from Table 3.5 is depicted in Table 3.6.

Table 3.5. Bit pattern for the partial products of the HGDFT FB using PDA-CSE.

Filter Coeff.	R31	R32	R33
b0=b29	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1	0 0 0 0 0 0 0 0 0 0 0 1 0 1 1
b1=b28	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
b2=b27	0 0 0 0 0 0 0 0 0 0 1 0 0 $\bar{1}$ 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
b3=b26	0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1	0 0 0 0 0 0 0 0 0 0 0 1 0 1 0	0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
b4=b25	1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 0 0 0 0 1
b5=b24	1 1 1 1 1 1 1 1 1 1 1 0 1 1 0	1 1 1 1 1 1 1 1 1 1 0 0 0 1 0	1 1 1 1 1 1 1 1 1 1 0 1 0 0 1
b6=b23	1 1 1 1 1 1 1 1 1 1 1 1 1 0 0	1 1 1 1 1 1 1 1 1 1 1 0 0 1	1 1 1 1 1 1 1 1 1 1 0 0 0 1 0
b7=b22	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0	1 1 1 1 1 1 1 1 1 1 1 1 0 1
b8=b21	0 0 0 0 0 0 0 0 0 0 0 1 0 0 $\bar{1}$ 0 0	0 0 0 0 0 0 0 0 0 0 0 1 1 0 1	0 0 0 0 0 0 0 0 0 0 1 0 0 $\bar{1}$ 0
b9=b20	0 0 0 0 0 0 0 0 0 0 1 0 0 $\bar{1}$ 0 1	0 0 0 0 0 0 0 0 0 0 0 1 0 1 0	0 0 0 0 0 0 0 0 0 1 0 0 0 $\bar{1}$ 0
b10=b19	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0	0 0 0 0 0 0 0 0 0 0 1 0 0 $\bar{1}$ 0	0 0 0 0 0 0 0 0 0 1 0 0 0 $\bar{1}$
b11=b18	0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1	0 0 0 0 0 0 0 0 0 0 1 0 1 0 1	0 0 0 0 0 0 0 0 0 0 0 1 0
b12=b17	1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1	1 1 1 1 1 1 1 1 1 1 1 0 0 0 1	1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
b13=b16	1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0	1 1 1 1 1 1 1 1 1 1 0 0 1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
b14=b15	1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0	1 1 1 1 1 1 1 1 1 1 0 0 1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 0 1 1

Table 3.6. Table showing the partial products of the HGDFT FB using PDA-CSERNS.

R_{31}	R_{32}	R_{33}
0000000000060000	0000000000003000	00000000000005070
0000000000003000	0000000000050000	0000000000603000
0000000000-300000	0000000000000000	0000000000003000
0000000000000200	00000000000000070	000000000000-3000
9999999999987000	9999999999967606	9999999999900006
0000000000000070	0000000000000070	0000000000003000
0000000000006700	0000000000030000	0000000000000050
0000000000003000	0000000000050000	0000000000080020
00000000000-30000	00000000000005707	00000000000-30000
0000000000-300060	00000000000070050	0000000000-300000
0000000000000050	0000000000-400000	00000000000-40000
0000000000060700	0000000000000050	0000000000050200
9999999999900070	9999999999940000	9999999999970070
0000000000000700	0000000000060000	0000000000300000
0000000000007050	0000000000000760	0000000000870070

3.4 CONCLUDING REMARKS

Three phase modulations added to time and frequency offset increases the computational complexities of generalised discrete Fourier transform (GDFT) filter bank. To reduce this computational complexity, a lower complexity filter bank eliminating phase multiplicative factors was designed. In this Chapter, an hybrid generalised discrete Fourier transform (HGDFT) filter was designed by modulating the classical GDFT filter bank and cascading the modulated filter with frequency response masking interpolated coefficient decimated filter while varying the frequency offset, k_0.

This is efficient realisation for both uniform and non-uniform channelisation. The performance of HGDFT channelizer was further optimized using three different digital filter number representations. These are the PDA-RNS, PDA-CRNS, PDA-DCRNS and PDA-CSERNS. A case application study was used to investigate the performances of the developed HGDFT and the improvements made thereon. However, there were noticeable rounding errors when the different number systems representations were used. Genetic algorithm method was

deployed into the proposed filter bank in order to minimize the passband ripples while maximizing the stopband attenuation using derived objective functions.

CHAPTER 4 HYBRID FARROW FILTER BANK

4.1 INTRODUCTION

Farrow filter operates best at low frequencies while its performance degraded towards the Nyquist region. Approximating higher frequency band close to Nyquist band increases filter order of Farrow algorithm. This is compounded by the usage of fractional delay in Farrow filter, which introduces undue distortions at higher frequencies for the passband ripples and stop band attenuation, thereby making the filter approximation cumbersome. A substantial higher filter order is required to approximate a given design specification with prescribed tolerance. This increases the computational complexity of Farrow filter and reduces the performances of multi-standard receivers in software defined radio.

This chapter focuses on the design of another low computational channelisation algorithm described as Hybrid Farrow Channelisation algorithm for multiband channels. It is an improvement over the conventional Farrow Per Channel Channelisation, (FPCC), as explained in section 2.4.6 of chapter 2, for multi-standard based SDR receiver.

The chapter is organised into two sections. Section 4.2 describes the development of hybrid Farrow channelisation algorithm. Under this section, the hybrid Farrow algorithm (HFarrow) was designed by exploiting the symmetrical basis functions of the frequency impulse responses and by modulating the conventional FPPC. The modulated Farrow filter was cascaded with interpolated coefficient decimated filter in order to extract the required channels with different multiband responses. This is preceeded with algorithm design steps and followed by the case application study to implement the developed algorithm.
Section 4.3.1 discusses improvement of HFarrow filter using parallel distributed arithmetics residual number system (PDA-RNS); section 4.3.2 presents the improvement of HFarrow using parallel distributed arithmetic canonical signed residual number system (PDA-CSRNS), while section 4.3.3 considers improvement approaches by employing common sub-expression residual number system (PDA-CSERNS) method in order to enhance the performance of the developed algorithms. Each of the improvement methods is optimised further using genetic algorithm approach. Other similar channelisation algorithm in literature such as coefficient decimated filter bank (CDFB), improved coefficient decimated filter bank (ICDM FB) and non-maximally decimated filter bank

(NMDUFB) are also used for benchmarking. The chapter is summarised in the concluding remarks section.

4.2 HYBRID FARROW INTERPOLATION FILTER

The algorithm development here involves modulation of Farrow filter and its optimisation by using hybrid of frequency response masking (FRM) based interpolated filter bank, coefficient decimating filter (CD-1). Two stages are involved in the algorithm development of hybrid Farrow filter. The first stage involves the development of low Farrow coefficient using first stage differential method; development of symmetrical frequency responses and development of modulated Farrow filter. The second stage combines modulating Farrow filter with frequency masking filter and interpolation coefficient decimation to produce the algorithm referred to as hydrid Farrow (HFarrow) filter algorithm in this context forthwith.

4.2.1 Farrow interpolation using first order differential approach

The Farrow structure was implemented using LaGrange polynomial. It is piecewise approximation of the filter into a polynomial form that shares a common set of coefficients, which results in the interpolation of input signals. Two important design parameters are polynomial order, k, and Farrow sub-filter, N [23, 24, 25, 47, 70, 71, 69]. It is implemented as a direct form of FIR filter structure and it is obtained as an approximation of continuous time function, $X_c(t)$, by fractional delay, d, as indicated in Equation (4.1).

$$\begin{aligned} y(n) &= h(d) * x(n) \\ y(n) &= h(n,d) * x(n) \\ &= \sum x(n) * C_k d^k \end{aligned} \tag{4.1}$$

The impulse response is computed using Lagrange method. From the impulse response $h(n,d)$, the fixed coefficients can be determined. The coefficients, C_k, from Equation (4.1) are derived from the set of $N+1$ linear equation. These coefficients are expressed in terms of fractional delay in such a way that $0 \leq d \leq 1$. The filter coefficient, $h(n)$, can be expressed in terms of C_k as $C_0 + C_1 + C_2 + ...+ C_n$. The Farrow filter relies on a filter bank structure whereby each filter coefficient is approximated as N^{th} order polynomial, d, as shown in Equation (4.2).

$$\begin{aligned} &h(n,d) = \sum c_k(n) d^k \\ &\text{n=0,1,..., N} \\ &0 \leq d \leq 1 \end{aligned} \tag{4.2}$$

Expressing Equation (4.2) in z- domain, the filter transfer function is represented as in Equation (4.3).

$$\begin{aligned} H_d(z) &= \sum_{n=0}^{N} h(n,d) z^{-n} \\ &= \sum_{n=0}^{N} \left| \sum_{k=0}^{p} C_k(n) z^{-n} \right| d^k \\ &= \left| \sum_{k=0}^{p} C_k(z) d^k \right| \end{aligned} \tag{4.3}$$

Where $C_k(z)$ represents the set of M+1 FIR sub-filters. From the relation in Equation (4.3), the filter structure is made up of a bank of fixed-weighted fractional delay, d, and summed u at the output of every tap.

$$\begin{aligned} h(n,d) &= \prod_{(k=0,k\neq 0)}^{n} \frac{d-k}{n-k} \\ &= (-1)^{(N-n)} \begin{pmatrix} d \\ n \end{pmatrix} \begin{pmatrix} d-n-1 \\ N-n \end{pmatrix} \\ &= \frac{d}{n} X \frac{d-1}{n-1} X \frac{d-n+1}{1} X \frac{d-n-1}{-1} X \frac{d-n}{n-N} \end{aligned} \tag{4.4}$$

for n $= 0, 1, 2, 3, ..., N$

When $N = 3$ and the fractional delay is d, the impulse response is shown in Equation (4.5) and Equation (4.7).

$$h(n,d) = \prod_{(k=0,k\neq 0)}^{3} \frac{D-k}{n-k} \tag{4.5}$$

for n =0, 1, 2,3

The coefficient for the fourth order poly-phase filter is calculated using Equation (4.6), Equation (4.7), Equation (4.11) and Equation (4.9) respectively.

$$
\begin{aligned}
h(0,d) &= \prod_{(k=0,k=1,k\neq 0)}^{3} \frac{d-k}{0-k} \\
&= \frac{d-1}{-1} \times \frac{d-2}{-2} \times \frac{d-3}{-3} \\
&= \frac{1}{6}(d^3 - 6d^2 - 8d - 6) \\
h(1,d) &= \prod_{(k=0,k=2,k\neq 1)}^{3} \times \frac{d-k}{1-k} \\
&= \frac{d}{1} \times \frac{d-2}{-1} \times \frac{d-3}{-2} \\
&= \frac{1}{2}(d^3 - 5d^2 + 6d) \\
h(2,d) &= \prod_{(k=0,k=1,k\neq 2)}^{2} \frac{d-k}{2-k} \\
&= \frac{d}{2} \times \frac{d-1}{1} \times \frac{d-3}{-1} \\
&= \frac{-1}{2}(d^3 - 4d^2 + 3d) \\
h(3,d) &= \prod_{(k=0,k=1,k\neq 3)}^{3} \frac{d-k}{3-k} \\
&= \frac{d}{3} \times \frac{d-1}{-2} \times \frac{d-2}{1} \\
&= \frac{1}{6}(d^3 - 3d^2 + 2d)
\end{aligned}
\tag{4.6}
$$

$$
\begin{aligned}
H_d(z) &= \sum_{n=0}^{N} (h,d)z^{-n} = h(0,d) + h(1,d)z^{-1} + h(2,d)z^{-2} \\
&= \frac{1}{6}(d^3 - 6d^2 - 8d - 6) + \frac{1}{2}(d^3 - 5d^2 + 6d) + \frac{-1}{2}(d^3 - 4d^2 + 3d) + \frac{1}{6}(d^3 - 3d^2 + 2d)
\end{aligned}
\tag{4.7}
$$

$$
\begin{aligned}
C_0(z) &= 1 \\
C_1(z) &= \frac{8}{6} + 3z^{-1} - \frac{3}{2}z^{-2} + \frac{2}{6}z^{-3} \\
C_2(z) &= \frac{-5}{6} - \frac{5}{2}z^{-1} + 2z^2 - \frac{1}{2}z^{-3} \\
C_3(z) &= \frac{1}{6} + \frac{1}{2}z^{-1} - \frac{1}{2}z^{-2} + \frac{1}{6}z^{-3} \\
\overline{C(z)} &= \Phi^T \overline{z} = \begin{bmatrix} C_0(z) \\ C_1(z) \\ C_2(z) \\ C_3(z) \end{bmatrix}
\end{aligned}
\tag{4.8}
$$

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{-8}{6} & 3 & \frac{-3}{2} & \frac{-1}{3} \\ \frac{6}{6} & \frac{-5}{2} & 2 & \frac{-1}{2} \\ \frac{-1}{6} & \frac{1}{2} & \frac{-1}{2} & \frac{1}{6} \end{bmatrix} \tag{4.9}$$

The number of operators are further reduced by finding the first derivatives of the each filter impulse, $h(n,d)$ as shown in Equation 4.10.

$$\begin{aligned} h'(0,d) &= \frac{1}{6}(3d^2 - 12d - 8) \\ h'(1,d) &= \frac{1}{2}(3d^2 - 10d + 6) \\ h'(2,d) &= \frac{-1}{2}(3d^2 - 8d + 3) \\ h'(3,d) &= \frac{1}{6}(3d^2 - 6d + 2) \end{aligned} \tag{4.10}$$

$$\begin{aligned} C_0(z) &= \frac{8}{6} + 3z^{-1} - \frac{3}{2}z^{-2} + \frac{1}{3}z^{-3} \\ C_1(z) &= -2 - 5z^{-1} + 4z^{-2} - z^{-3} \\ C_2(z) &= \frac{1}{2} - \frac{3}{2}z^{-2} + \frac{1}{2}z^{-3} \\ C_3(z) &= 0 + 0 - 0 + 0 \end{aligned} \tag{4.11}$$

$$\overline{C(z)} = \Phi^T \bar{z} = \begin{bmatrix} C_0(z) \\ C_1(z) \\ C_2(z) \\ C_3(z) \end{bmatrix}$$

$$\Phi = \begin{bmatrix} \frac{8}{6} & 3 & \frac{3}{2} & \frac{1}{3} \\ 2 & -5 & 4 & -1 \\ \frac{1}{2} & \frac{3}{2} & \frac{-3}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.12}$$

Figure 4.1 shows the Farrow sub-filters.

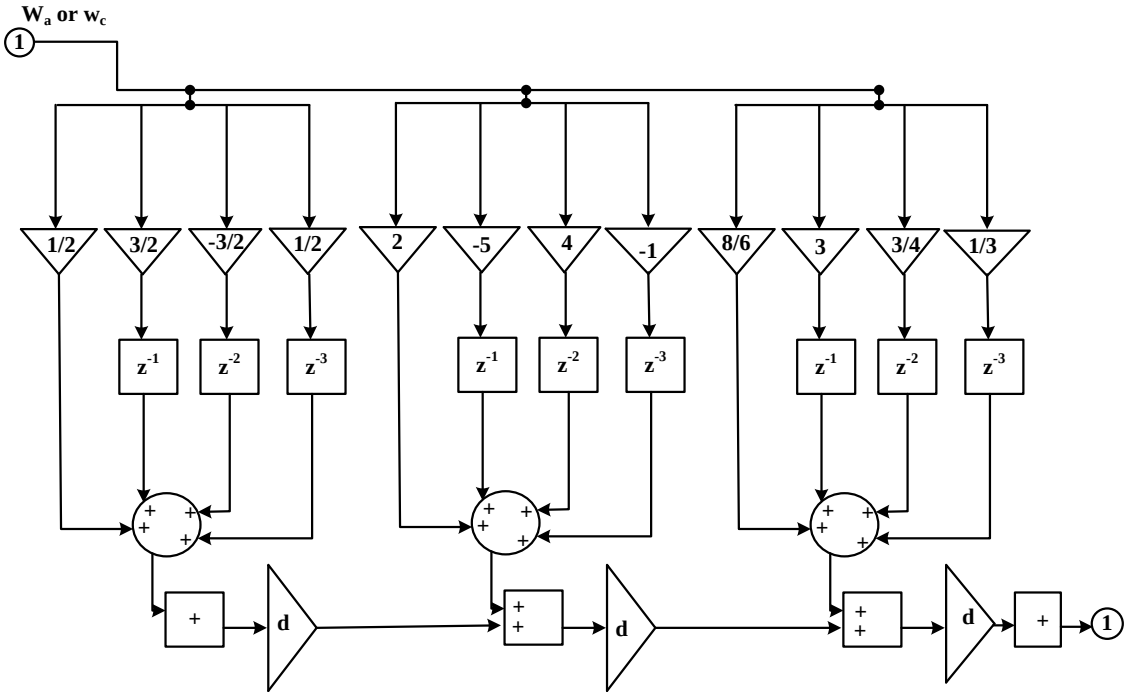

Figure 4.1. Farrow sub-filters.

4.2.2 Exploring Frequency responses of Symmetrical Farrow filter polynomial functions

The continuous impulse response, $h_c(t)$, of the Farrow structure is represented as linear combination of basis functions as can be seen in Equation 4.13.

$$h_c(t) = \begin{cases} \sum_{m=0}^{M} C_k(n)(\frac{t}{T_i} + t_n)^n, & (d_{min} - t_n)T_1 \leq t \leq (d_{max} - t_n)T_k \\ & \text{for n = 0,1,2, ..., N} \\ 0, & \text{otherwise} \end{cases} \tag{4.13}$$

Where t_n is the basepoint value, T_1 is the input sampling frequency, and C_k is the k^{th} sub-filter coefficient. The interval of the fractional delay d is represented as follows in Equation 4.14.

$$\begin{aligned} n &= \left\lfloor \frac{t}{T_1} \right\rfloor \\ d &= \frac{t_1}{T_1} - d \end{aligned} \tag{4.14}$$

The inter-sample interval is given as in Equation 4.15.

$$0 \leq d \leq 1 \tag{4.15}$$

By introducing the basis function for piecewise polynomial that are zero outside a given interval, Equation 4.16 was derived.

$$f(m,d) = \begin{cases} d^k, & d_{min} \leq d \leq d_{max} \\ 0, & otherwise \end{cases} \tag{4.16}$$

Therefore, the continuous impulse response, $h_c(t)$, is shown in Equation 4.17.

$$h_c(t) = \sum_{n=0}^{N} \sum_{m=0}^{M} C_k f(m,d) \tag{4.17}$$

The frequency domain representation of the impulse response is represented in Equation 4.18.

$$\begin{aligned} H(e^{jw},d) &= \sum_{n=0}^{M} C_k(\omega) d^k \\ H(\omega,d) &= \sum_{m=0}^{M} C_k(\omega) d^k \\ &= \sum_{m=0}^{M} C_k(\omega) G(\omega) \\ \text{where } G(\omega) &= d^k \end{aligned} \tag{4.18}$$

Then by applying Fourier transform, the fractional delay $G(\omega)$ can be expressed as indicated in Equation 4.19.

$$G(k,m,\omega) = F\{g(m,n,\omega)\} \tag{4.19}$$

The frequency variable ω is normalized to the input sampling period, that is, $\omega = \Omega T_1$ and delay d is replaced with μ. In order to obtain the continuous frequency responses, three cases are considered. The first to consider is when N is odd while the second one for consideration is when N is even and finally when $N > 0$.

$$G(m,n,\omega) = \int_{-\infty}^{\infty} g(m,n,\omega) e^{-j\omega\mu} d\mu$$

For odd N,

$$g(m,n,\omega) = \begin{cases} (\mu + n + \frac{1}{2})^k, & -n-1 \leq \mu \leq -n \\ (-1)^k (\mu - n - \frac{1}{2})^k, & n \leq \mu \leq n+1 \\ 0, & otherwise \end{cases} \tag{4.21}$$

$$\begin{aligned}
&\text{Where, } \mu_1 = \mu + (n+\tfrac{1}{2}) \\
&\mu = \mu_1 - (n-\tfrac{1}{2}) \\
&\dot{\mu}_1 = d\mu \\
&\text{Also,} \\
&\mu_2 = \mu - (n+\tfrac{1}{2}) \\
&\mu = \mu_2 + (n+\tfrac{1}{2}) \\
&\dot{\mu}_2 = d\mu
\end{aligned} \tag{4.22}$$

Therefore, the discrete version of fractional delay can be expressed as shown in Equation 4.23.

$$\begin{aligned}
G(m,n,\omega) &= \int_{-\infty}^{\infty} g(m,n,\omega)e^{-j\omega\mu}d\mu \\
&= \int_{-n-1}^{-n} (\mu+n+\frac{1}{2})^k \omega e^{-j\omega\mu}d\mu + \int_{n}^{n+1} (-1)^k(\mu-n-\frac{1}{2})^k \omega e^{-j\omega\mu}d\mu \\
&= \int_{-\frac{1}{2}}^{\frac{1}{2}} \mu_1 \omega e^{-j\omega\mu_1 - [n+\frac{1}{2}]}d\mu + \int_{-\frac{1}{2}}^{\frac{1}{2}} (-1)^m \mu_2 \omega e^{-j\omega\mu_2 - [n+\frac{1}{2}]}d\mu \\
&= e^{j(n+\frac{1}{2}\omega)} \int_{-\frac{1}{2}}^{\frac{1}{2}} \mu_1 \omega e^{-j\omega\mu_1} d_{\mu 1} + (-1)^m e^{j(n+\frac{1}{2})\omega} \int_{-\frac{1}{2}}^{\frac{1}{2}} (-1)^m \mu_2 \omega e^{-j\omega\mu_2} d_{\mu 2} \\
&= [e^{j(n+\frac{1}{2})\omega} + (-1)^m e^{j(n+\frac{1}{2})\omega}] \int_{-\frac{1}{2}}^{\frac{1}{2}} \mu_1 \omega e^{-j\omega d_\mu}
\end{aligned} \tag{4.23}$$

Fractional delay $G(m,n,\omega)$ can be expressed as function of two variables. That is, $\phi(m,n,\omega)$ and $\psi(m,\omega)$.

$$G(m,n,\omega) = \phi(m,n,\omega)\psi(m,\omega) \tag{4.24}$$

$$\text{Where } \phi(m,n,\omega) = \begin{cases} 2cos([n+\frac{1}{2}]\omega), & \text{for m=even }, N = odd \\ 2jsin([n+\frac{1}{2}]\omega), & \text{for m=even }, N = odd \end{cases} \tag{4.25}$$

In another instance, when $N = even$ and $n > 0$, then

$$\begin{aligned}
G(m,n,\omega) &= \int_{-n-\frac{1}{2}}^{-n+\frac{1}{2}} (\mu+n)^k \omega e^{-j\omega\mu}d\mu + \int_{n-\frac{1}{2}}^{n+\frac{1}{2}} (-1)^k(\mu-n)^k \omega e^{-j\omega\mu}d\mu \\
G(m,n,\omega) &= [e^{j\omega} + (-1)^m e^{j\omega}] \int_{-\frac{1}{2}}^{\frac{1}{2}} \mu^m e^{-j\omega}\mu_k \\
&= \phi(m,n,\omega)\psi(m,\omega) \\
\text{where } \psi(m,\omega) &= \int_{-\frac{1}{2}}^{\frac{1}{2}} \mu^m e^{-j\omega} d_\mu
\end{aligned} \tag{4.26}$$

$$G(m,n,\omega) = \phi(m,n,\omega) = \begin{cases} 2cos(n\omega), & m = even, N = odd \\ 2jsin(n,\omega), & m = odd, N > 0 \end{cases} \tag{4.27}$$

Lastly, for $n = even$, $n = 0$ *and* $m = even$

$$G(m,0,\omega) = \int_{-\frac{1}{2}}^{\frac{1}{2}} (\mu)^k \omega e^{-j\omega\mu} d\mu + \int_{n-\frac{1}{2}}^{n+\frac{1}{2}} (-1)^k (\mu - n)^k \omega e^{-j\omega\mu} d\mu \tag{4.28}$$

$$G(m,0,\omega) = \phi(m,n,\omega) \text{with} = \begin{cases} \phi(m,n,\omega) = 1, & N = even, m = even, n = 0 \\ \phi(m,n,\omega) = 0, & N = even, m = odd, n = 0 \end{cases} \tag{4.29}$$

The different variants of the basis functions $G(m,n,\omega)$ are represented as follows in Equation 4.30.

$$\phi(m,n,\omega) = \begin{cases} 1, & N = even, n = 0, m = even \\ 0, & N = even, n = 0, m = odd \\ 2cos(n,\omega), & N = even, n > 0, m = even \\ 2jsin(n,\omega), & N = even, n > 0, m = odd \\ 2cos([n+\frac{1}{2}],\omega), & N = odd, m = even \\ 2jsin([n+\frac{1}{2}],\omega), & N = odd, m = even \end{cases} \tag{4.30}$$

Considering the scaling function as shown in Equation 4.31.

$$\psi(m,\omega) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \mu^m e^{-j\omega\mu} d\mu \tag{4.31}$$

By multiplying the integral with unit rectangle $\prod(\mu)$, Equation 4.32 is obtained.

$$\psi(m,\omega) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \prod(\mu) \mu^m e^{-j\omega\mu} d\mu \tag{4.32}$$

Expressing the scaling function in terms of Fourier transform.

$$\begin{aligned} \psi(m,\omega) &= F\{\prod(\mu)\mu^m\} \\ &= j^m \frac{\mu^m}{\mu\omega^m} sinc(\tfrac{\omega}{2}) \end{aligned} \tag{4.33}$$

The basis function $G(m,n,\omega) = \psi(m,\omega)\phi(m,n,\omega)$ are both real and even. This implies that $H_c(j\omega)$ are both real and even functions and are derivatives of real valued and symmetrical nature of $h_c(t)$. The basis function $G(m,n,\omega)$ can be represented as real valued functions for $\phi(m,n,\omega)$ and $\psi(m,\omega)$ by transposing the imaginary unit j from $\phi(m,n,\omega)$ to $\psi(m,\omega)$ for odd m. This can be represented as shown in Equation 4.34.

$$G(m,n,\omega) = \psi(m,\omega)\phi(m,n,\omega) \text{ with } \psi(m,\omega) = (-1)^m \frac{\mu^m}{\mu\omega^m} sinc(\frac{\omega}{2}) \tag{4.34}$$

Expressing the scaling function as a real value as indicated by Equation 4.36.

$$\begin{aligned} \psi(m,\omega) &= sinc(\tfrac{\omega}{2}) \\ \psi(m,\omega) &= \textstyle\sum_{k=0}^{\infty} a_k \omega^k \\ \text{Where } \textstyle\sum_{k=0}^{\infty} a_k \omega^k \text{ with } a_k &= \begin{cases} \frac{(-1)^{\frac{k+m}{2}}(k+1)}{2^{k+m}(k+m+1)!}, & k+m \text{ is even} \\ 0, & k+m \text{ is odd} \end{cases} \end{aligned} \tag{4.35}$$

The values for $\psi(m,\omega)$ are identical for the real variant of $\psi(m,0$ when $\omega = 0$ as indicated in Equation 4.36.

$$\psi(m,0) = j^m a_0^m = \begin{cases} \frac{(-1)^{\frac{m}{2}}}{(m+1)!2^m}, & m \text{ is even} \\ 0, & m \text{ is odd} \end{cases} \tag{4.36}$$

The impulse response is scaled proportionally and this is able to reduce the distortion or numerical oscillation of the impulse response to minimal.

4.2.3 Modulation of Farrow filter

Having seen the symmetrically scaled impulse response of the Farrow filter, the Farrow filter can be modulated to reduce the computational complexity. The value of μ can be determined using Equation 4.37 with channel center frequency, ω_k as shown in Equation 4.38.

$$\frac{\pi}{\omega_s} \tag{4.37}$$

$$\omega_k = \frac{2\pi(k+k_0)}{K} \tag{4.38}$$

Where ω_k is the center frequency of the channels.

The modulated Farrow filter can be represented as shown in Equation 4.39.

$$\begin{aligned} H(e^{jw},\mu) &= \Sigma_{m=0}^{M} C_k(\omega) G(\omega) \\ &= \Sigma_{m=0}^{M} C_k(\omega)(2cos(n,\omega) + 2jsin(n,\omega)) \\ &= \Sigma_{m=0}^{M} C_k(\omega)\left[e^{j\omega n\mu} + (-1)^m e^{j\omega n\mu}\right] \end{aligned} \tag{4.39}$$

Thus, by modulating Equation 4.1 with $e^{-j\omega md}$, newly generated Farrow filter $y'(n)$ is obtained as shown in Equation 4.40.

$$\begin{aligned} y(n) &= h(n,\mu) * x(n) \\ y(n) &= h(n,\mu) * x(n) \\ y'(n) &= \sum x(n) * h(\omega,\mu) e^{-j\omega Mn\mu} \end{aligned} \tag{4.40}$$

If the channel signal is critically sampled, the decimation ratio and the bandwidth are related as follows in Equation 4.41.

$$M = \frac{K}{2} \tag{4.41}$$

Thus, the new Farrow filter bank can be reduced to the following filter bank in Equation 4.42.

$$y'_k(m) = (-1)^{km} \Sigma_{n=-M-1}^{N} C_k(n) h_k(mK-1) x_(n) \tag{4.42}$$

If m is replaced by L and K is replaced by $2M$, and if linear phase prototype low-pass filter $H(z)$ of order N has a pass-band edge of $\theta_a = \frac{2m\pi - \omega_{\omega_\Delta}}{M}$ and stop-band edge of $\phi_a = \frac{2m\pi - \omega_{\omega_\Delta}}{M}$, with ω_Δ as the width of the transition band. For even multiples of number M of sub-bands, the length is $N+1$, that is, $N = (2LM-1)$.
Here, the impulse response, $h_k(mK-1)$, will be reduced as follows. It can be seen that h_k contains terms that multiply h_n and this can be expressed as variable D as seen in Equation 4.43.

$$\begin{aligned} h_k(n) = C_k(n)h(n)\Bigg(2cos\Bigg((2M+1)\tfrac{\pi}{2M}\Big[(N+2kM) - \tfrac{N}{2} + \Phi\Big]\Bigg) \\ +2jsin\Bigg((2M+1)\tfrac{\pi}{2M}\Big[(N+2kM) - \tfrac{N}{2} + \Phi\Big]\Bigg)\Bigg) \end{aligned} \tag{4.43}$$

$$\begin{aligned} \text{Where } e^{j\omega n1d} &= 2cos\Bigg((2M+1)\tfrac{\pi}{2M}[(N+2kM) - \tfrac{N}{2} + \Phi]\Bigg) \\ (-1)^m e^{j\omega n2d} &= 2jsin\Bigg((2M+1)\tfrac{\pi}{2M}[(N+2kM) - \tfrac{N}{2} + \Phi]\Bigg) \end{aligned} \tag{4.44}$$

However, by exploiting symmetry in Equation 4.44, Equation 4.45 becomes

$$\begin{aligned} h_k(n) &= C_k(n)h(n)\Bigg(2cos\Bigg((2M+1)\tfrac{\pi}{2M}\Big[(N+2kM) - \tfrac{N}{2} + \Phi\Big]\Bigg)\Bigg) \\ h_k(n) &= C_n h(n)\Big[e^{j\omega n1\mu}\Big]\psi(m,\omega) \end{aligned} \tag{4.45}$$

The polyphase representation of the analysis filter bank can be described as shown in Equation 4.46.

$$\begin{aligned} H_k(z) &= \sum_{n=0}^{N} C_k(n)h_k(n)z^{(-n)} \\ H_k(z) &= \sum_{l=0}^{L-1}\sum_{j=0}^{2M-1}\Big[e^{j\omega n1\mu 1}\psi(m,\omega) + e^{j\omega n2\mu 2}\psi(m,\omega)\Big]C_k(2LM+j)h_k(2LM+j)z^{-(2LM+j)} \end{aligned} \tag{4.46}$$

The prototype filter can be decomposed into $2M$ poly-phase components, as follows in Equation (4.47), where $S_j(z) = \sum_{i=0}^{L-1} C_k(2LM+j)h_k(2LM+j)z^{(-L)}$ are the poly-phase components of the filter $H(z)$.

$$\begin{aligned} H(z) &= \sum_{j=0}^{2M-1} z^{-j} \sum_{l=0}^{L-1} \psi(m,\omega)\left[e^{j\omega n1\mu} + e^{j\omega n2\mu}\right] C_k(2LM+j)h_k(2LM+j)z^{(-2M-j)} \\ &= \sum_{j=0}^{2M-1} z^{-j} \sum_{l=0}^{L-1} \psi(m,\omega)\left[e^{j\omega n1\mu} + (-1)^m e^{j\omega n2\mu}\right] C_k(2LM+j)h_k(2LM+j)z^{(-2M-j)} \\ &= \sum_{j=0}^{2M-1} \psi(m,\omega)\left[e^{j\omega n1\mu} + (-1)^m e^{j\omega n2\mu}\right] z^{-j} S_j(z^{-2M}) \end{aligned} \tag{4.47}$$

Representing $D_1 = \psi(m,\omega)(e^{j\omega n1\mu 1})$ and $D_2 = \psi(m,\omega)(e^{j\omega n2\mu 1})$.

From Equation (4.48), it can be shown that D_1 and D_2 are $M \times N$ matrices, whose (m,j) elements are $D_{m,j}$ and $D_{m,j+m}$, respectively for $m, j = 0, 1, \ldots (m-1)$.

$$H(z) = \begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ \vdots \\ H_{m-1}(z) \end{bmatrix} = \begin{bmatrix} D_1 & D_2 \end{bmatrix} \begin{bmatrix} S_0(z^{2M}) \\ z^{-1}S_1(z^{2M}) \\ \vdots \\ \vdots \\ z^{(-2m-1)}S_{2m-1}(z^{2M}) \end{bmatrix} \tag{4.48}$$

Representing $\delta(z)$ as shown in Equation (4.49).

$$\delta(z) = \begin{bmatrix} 1 & z^{-1} & \cdots & z^{-M+1} \end{bmatrix}^T \tag{4.49}$$

The poly-phase representation of the matrix can be represented as in Equation (4.51), where $S(z)$ is the poly-phase matrix.

$$H(z) = \begin{bmatrix} \psi(m,\omega)(e^{j\omega n1\mu}) & \psi(m,\omega)((-1)^m e^{j\omega n2\mu}) \end{bmatrix} \begin{bmatrix} S_0(z^{2M}) & 0 & & \\ & S_1(z^{2M}) & & \\ & & \ddots & \\ & & \ddots & \\ & & 0 & S_{2m-1}(z^{2M}) \end{bmatrix} \begin{bmatrix} \delta(z) \\ z^{-m}\delta(z) \end{bmatrix} \tag{4.50}$$

$$H(z) = \left\{ D_1 \begin{bmatrix} S_0(z^{2M}) & 0 & & \\ & S_1(z^{2M}) & & \\ & \ddots & & \\ & \ddots & & \\ & 0 & & S_{2m-1}(z^{2M}) \end{bmatrix} + z^{-m} D_2 \begin{bmatrix} S_0(z^{2M}) & 0 & & \\ & S_{M+1}(z^{2M}) & & \\ & \ddots & & \\ & \ddots & & \\ & 0 & & S_{2m-1}(z^{2M}) \end{bmatrix} \right\} \delta(z) \tag{4.51}$$

4.2.4 Development of Hybrid Farrow Channelisation Algorithm

After improvements obtained from the modulation algorithm, further work was done to achieve some improvements by hybridizing the modulated algorithm with the frequency response masking algorithm [3].
The design involves hybrid of frequency response masking (FRM) based interpolated coefficient decimated filter and modulatedFarrow filter.

The hybrid Farrow (HFarrow) based filter bank consists of two branches namely: the upper and the lower branch. The upper branch is made up of FRM coefficient decimated filter and the masking filter, whereas the lower branch consists of complementary FRM coefficient decimated filter and the complementary masking filter. A low-pass coefficient base interpolated linear phase FIR filter, $H_a(z^{\frac{L}{M}})$, is formed from the cascade of base interpolating filter, $H_a(z^L)$, and the coefficient decimating filter, $H_{cd}(z^L)$, to extract the sharp narrow-band channel of choice.

Also, a bandpass edge complementary coefficient base interpolating filter, $H_c(z^{\frac{L}{M}})$, is formed from the cascade of complementary base interpolating filter, $H'_a(z^L)$, and the complementary coefficient decimating filter, $H'_{cd}(z^M)$, to isolate multi-bands frequency responses. The low-pass coefficient base interpolated filter, $H_a(z^{L/M})$, cascades with the farrow masking filter, $A_k(z)$, in the upper branch while the bandpass complementary coefficient base interpolating filter, $H_c(z^{L/M})$, cascades with the complementary masking filter, $B_k(z)$, in the lower branch to produce a low computational multi-narrow frequency bands.

The desired passband (ω_p) and (ω_s) cut off frequency of the base filter $H_a(z)$ response is calculated as indicated in Table 3.1 in the previous section.

The transfer function of the FRM coefficient decimation filter is given using Equation (4.52).

$$H(z) = \frac{L}{M}\left[H_a(z^{\frac{L}{M}})A(z) + H_c(z^{\frac{L}{M}})B(z)\right] \tag{4.52}$$

The coefficient decimated base and complementary filters are symmetrical and asymmetrical linear phase FIR filter which can be expressed as $H_a(z^{\frac{L}{M}}) = H_c(-z^{\frac{L}{M}})$. A half-band filter is introduced into the coefficient decimated based-FRM filter to further reduce its computation complexity. This is possible as a result of the symmetrical properties possessed by the half-band filter. The time-domain impulse response of the CD-1 technique requires every other component to be zero except the components at the centre. That indicates that it is symmetrical around the centre. This translates to reduced complexity in terms of the number of the multiplies required by the filter.

The transfer function of the half-band masking FRM coefficient decimated filter band can be expressed in terms of two polyphase components as shown in Equation (4.53).

$$\begin{aligned} H_a(z^{\frac{L}{M}}) &= \frac{L}{M}\left[H_{a0}(z^{\frac{2L}{M}}) + z^{-\frac{2L}{M}}H_{a1}(z^{\frac{2L}{M}})\right] \\ H_c(z^{\frac{L}{M}}) &= \frac{L}{M}\left[H_{a0}(z^{\frac{2L}{M}}) - z^{-\frac{2L}{M}}\frac{1}{M}H_{a1}(z^{\frac{2L}{M}})\right] \end{aligned} \tag{4.53}$$

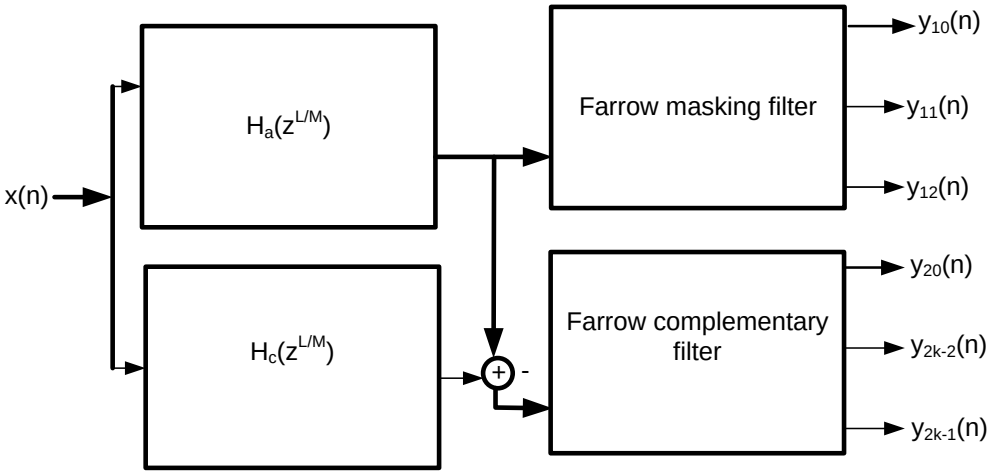

Figure 4.2. Block Diagram of Coefficient Decimated FRM Farrow based FIR Filter.

The masking filters are replaced with two farrow filters as shown in the Figure 4.2. The transfer function for HFarrow masking algorithm can be expressed as shown in Equation (4.54).

$$H_a(z^{\frac{2L}{M}}) = \frac{L}{M}\left[\left(H_{a0}(z^{\frac{2L}{M}}) + z^{-\frac{2L}{M}}H_{a1}(z^{\frac{2L}{M}})\right)A(z)\right] \tag{4.54}$$

The impulse response of the HFarrow channelisation algorithm is approximated with different fixed delay variables using the polynomial interpolation methods. A set of sub-filters with fixed delay μ_i, such that $i = 0, 1, ...i-1$ is designed. The impulse response is interpolated by the, L^{th}, order using, μ, as the variable. The resultant impulse response is the interpolated version of HFarrow parameterised by delay, μ. The parameter, μ, allows full unabridged control over the available bandwidth and the cut off frequencies of the multi-band channels.

The input band of masking FRM-CD signal is decomposed into multiple sub-bands, each with distinct bandwidth (BW). Each bandwidth is phase shift modulated by fractional delay, μ^k. Applying the poly-phase decomposition as seen in Equation 4.47, will result in Equation (4.55), where $S_{ki}(z^{-2M})$ are the K poly-phase components of $A_k(z)$ and $B_k(z)$.

$$\begin{aligned} A_k(z) &= \sum_{n=0}^{2M-1} \psi(m,\omega)\left(e^{j\omega n1\mu} + (-1)^m e^{j\omega n2\mu}\right) z^{-n} S_n(z^{-2M}) \\ B_k(z) &= \sum_{n=0}^{2M-1} \psi(m,\omega)\left(e^{j\omega n1\mu} - (-1)^m e^{j\omega n2\mu}\right) z^{-n} S_n(z^{-2M}) \end{aligned} \tag{4.55}$$

Finally, Equation (4.56), shows the representation of each of the modulated masking bandpass filter and the block diagram for hybrid GDFT masking filter is depicted in Figure 4.3.

.

$$\begin{aligned} H_a(z^{\frac{2L}{M}}) &= \frac{L}{M}\left[H_{a0}(z^{\frac{2L}{M}}) + z^{\frac{-2L}{M}} H_{a1}(z^{\frac{2L}{M}}) \sum_{n=0}^{2M-1} \psi(m,\omega)\left(e^{j\omega n1\mu} + (-1)^m e^{j\omega n2\mu}\right) z^{-n} S_n(z^{-\frac{2L}{M}})\right] \\ H_a(z^{\frac{2L}{M}}) &= \frac{L}{M}\left[H_{a0}(z^{\frac{2L}{M}}) + z^{-\frac{2L}{M}} H_{a1}(z^{\frac{2L}{M}}) \sum_{n=0}^{2M-1} z^{-n}(D_1 + D_2) S_n(z^{-\frac{L}{M}})\right] \end{aligned} \tag{4.56}$$

The transfer function of the FRM coefficient decimation filter is given using Equation (4.57).

$$H(z) = \frac{L}{M}\left[H_a(z^{\frac{L}{M}})A(z) + H_c(z^{\frac{L}{M}})B(z)\right]$$

But,

$$H_a(z^{\frac{L}{M}}) = H_c(-z^{\frac{L}{M}}) \tag{4.57}$$

Therefore,

$$H(z^{\frac{2L}{M}}) = \frac{2L}{M}\left[H_{a0}(z^{\frac{2L}{M}}) + z^{-\frac{2L}{M}} H_{a1}(z^{\frac{2L}{M}}) \sum_{n=0}^{2M-1} z^{-n}(D_1 + D_2) S_n(z^{-\frac{2L}{M}})\right]$$

The final output sequence, $y(n)$, can be expressed in terms of the convolution of $x(n)$ and the filter transfer function, $h(n)$.

$$\begin{aligned} y(n) &= x(n) * h(n) \\ Y(z) &= X(z)H(z) \end{aligned} \tag{4.58}$$

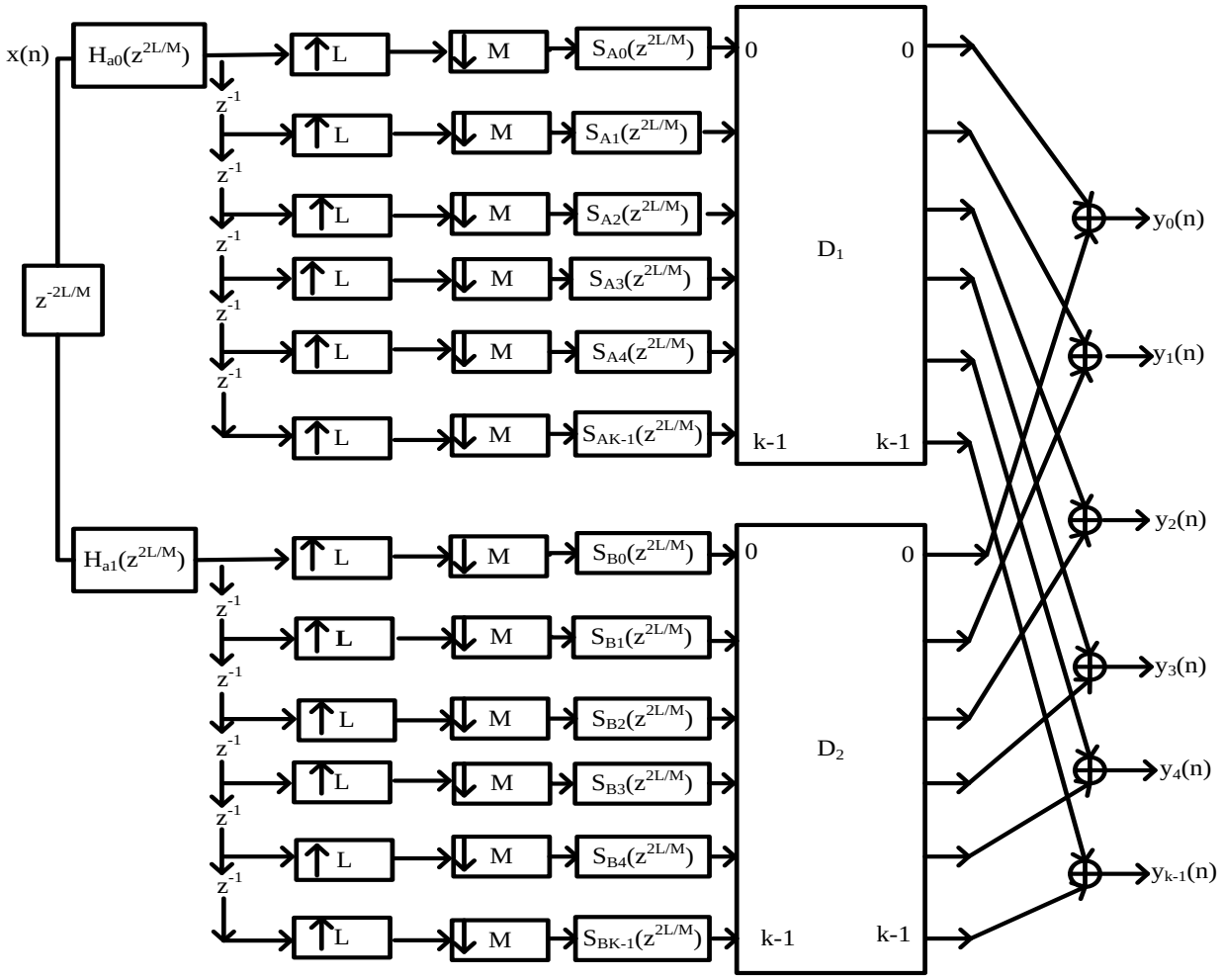

Figure 4.3. Diagram depicting HFarrow masking channelisation Algorithm.

The centre frequency component of each band is shifted precisely by a phase of fractional delay, μ. The phase to be shifted is at π and $-\pi$ for different fractional delays, while the transition band of H-Farrow FB is centred at $\frac{\pi}{2}$ rad. The H-Farrow FB design is made up of three filtering stages, namely; the base filter, $H_a(z)$, the coefficient decimation filter, $H_{cd}(z)$, and the masking filter $A(z)$. The design used the Parks-McClellan algorithm and the filter is realized using the direct transposed FIR in its implementation.

4.2.5 Hybrid Farrow Channelisation Design Steps

The procedure for the HFarrow channelisation algorithm are outlined below:

1. Normalize all the channel bandwidths (BWs), such that BW_i and transition bandwidth Δ_i specifications range from 0 to 1, and 1 corresponds to $\frac{f_s}{2}$, where f_s is the sampling frequency.
2. Determine the weighing scale value of each channels by varying the value of m.
3. Compute the value of D_1 and D_2
4. Calculate each channel stopband frequency such that $\omega_{si} = \frac{BW_i}{2}$.
5. Calculate the modal bandwidth such that BW_{Modal}=$\frac{GCD(BW_1',BW_2',BW_3')}{2}$. This corresponds to the modal stop band frequency.
6. Calculate the decimation factor, M, of the masking filter using this formula $M = \frac{\pi}{\omega_{ms}}$. The interpolated factor is calculated using the formula $L = \left\lfloor \frac{\pi}{\omega_{ms}} \right\rfloor$, where ω_{ms} is the stopband frequency for each masking channel. Thus the fractional rate for masking filter μ_a can be calculated as $\frac{L_{ma}}{M_{ma}}$.
7. Calculate the decimated factor for the complementary filter using the formula $M = \frac{\pi}{\pi+\omega_{mcs}}$. The interpolated factor is calculated using the formula $L = \left\lfloor \frac{\pi}{\pi+\omega_{mcs}} \right\rfloor$. Thus, the fractional rate for complementary filter μ_c can be calculated thus: $\frac{L_{mc}}{M_{mc}}$.
8. Determine transition bandwidth for masking and complementary filter, Δ_i such that:$\Delta_k' = \Delta_k \times \frac{L_k}{M_k}$ where $\frac{L_k}{M_k}$ is the fractional rate for masking or complementary filter.
9. Determine the base modal or complementary modal TBW as:
Δ_{modal}=min$(\Delta_1', \Delta_2', ..., \Delta_n')$. This corresponds to the modal transition width.
10. Calculate the modal, masking and complementary passband width using $\omega_p = \omega_s - \Delta_{modal}$.
11. Determine the passband edge and stop band edge of the modal or prototype filter, masking and complementary filter from Table 3.1, where $m = \left\lfloor \frac{\omega_{mp}\frac{L_{ma}}{M_{ma}}}{2pi} \right\rfloor$ is for masking filter and $m = \left\lceil \frac{\omega_{ms}\frac{L_{mc}}{M_{mc}}}{2pi} \right\rceil$ is for complementary filter.
12. Find the stopband ripple using $\delta_{s1}' = \delta_{s1}\frac{L_i}{M_i}$.
13. The modal passband peak ripple is calculated as: δ_{pmodal}=min$(\delta_{p1}', \delta_{p2}', ..., \delta_{pn}')$.
14. The cut-off frequency of the prototype and masking filter are calculated using Table 3.1.
15. Determine the prototype filter order and the individual channels filter order using Bellanger formular as: $N = \frac{-2log_{10}(\delta_p'\delta_s')}{3\Delta_{TBW}}$ -1 [179].

4.3 IMPROVEMENT OF HFARROW CHANNELISATION ALGORITHM USING DIFFERENT NUMBER SYSTEMS

The previous number systems such as PDA-RNS, PDA-CSRNS, PDA-CSE are applied to the HFarrow Channelisation algorithm in order to reduce the computational capicity of HFarrow filter.

4.3.1 Improvement of HFarrow Channelisation Algorithm with Parallel Distributed Arithmetic Based Residual Number (HFARROW PDA-RNS)

The performance of HFarrow channelisation algorithm can be improved by using parallel residual number system (PDA-RNS), as outlined in section 3.3.1. The results obtained from HFarrow PDARNS FB are optimised further using genetic algorithm. In this way, the same objective function as well as the prototype and masking parameters as in section 3.3.2.1 were used.

4.3.2 Improvement of HFarrow Channelisation Algorithm with Parallel Distributed Arithmetic Based Canonical Signed Residual Number (HFARROW PDA-CSRSN)

Improving the performance of HFarrow channelisation algorithm with parallel canonical signed residual number system (PDA-CSRNS), as outlined in section 3.3.3.1. The results obtained from HFarrow PDACSRNS FB are optimised further using genetic algorithm. In this way, the same objective function as well as the prototype and masking parameters as in section 3.3.3.1 were used.

4.3.3 Improvement of HFarrow Channelisation Algorithm Using Parallel Distributed Based Common Sub-Expression Residual Number System (HFARROW PDA-CSERNS)

Table 4.1 shows the results of the partial products of the HFarrow PDA-CSERNS.

The hybrid common sub-expressions elimination method (HCSE) uses both the horizontal common sub expressions (HCSs) and vertical common sub-expressions (VCSs) pattern bits. The common factored bits pattern used in HCSs are: $[1001]$, $[10001]$, and $[111111111]$ while the common factored bits pattern in VCSs are $[11]$. These hybrid pattern together with their negated versions are used for the filter realisation as indicated.

$$
\begin{aligned}
X_2 &= [01] = 2 \\
X_3 &= [10] = 3 \\
-X_4 &= [11] = 4 \\
X_5 &= [101] = 5 \\
X_6 &= [111] = 6 \\
X_7 &= [1001] = 7 \\
-X_7 &= [100\bar{1}] = -7 \\
X_8 &= [1111] = 8 \\
X_9 &= [111111] = 9
\end{aligned}
$$

Table 4.1. Table showing the partial products of the HFarrow PDA-CSERNS.

Filter Coefficient	R_{31}	R_{32}	R_{33}
b0=b29	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 $\bar{1}$	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 $\bar{1}$
b1=b28	0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0	0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
b2=b27	0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1	0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
b3=b26	0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 $\bar{1}$
b4=b25	1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1	1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
b5=b24	1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1	1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0	1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1
b6=b23	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 $\bar{1}$	1 1 1 1 1 1 1 1 1 1 1 1 0 0 $\bar{1}$ $\bar{1}$
b7=b22	1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1	1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0
b8=b21	1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0	1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0	1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
b9=b20	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 $\bar{1}$	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
b10=b19	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
b11=b18	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
b12=b17	1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 0 0 $\bar{1}$ 0	1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
b13=b16	1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0	1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
b14=b15	1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1

Table 4.2. The partial products of the HFarrow FB using PDA-CSERNS.

R_{31}	R_{32}	R_{33}
8000800060007000	8000800080007000	8000800060007000
0000000000005000	0000000000080000	0000000000-7000
00000000000000004	0000000000000500	0000000000004000
0000000000040040	0000000000000000	0000000000-7000
9999999999904000	9999999999930300	9999999999940020
0000000000000404	0000000000025000	0000000000020304
0000000000003000	0000000000007000	0000000000070000
0000000000027000	0000000000000030	0000000000020000
0000000000005000	00000000000020000	00000000000700000
0000000000005000	00000000000020000	0000000000002000
0000000000020000	0000000000007000	0000000000000500
0000000000000040	0000000000000300	0000000000208000
9999999999930400	99999999999-70000	9999999999700000
0000000000020000	0000000000020000	0000000000080000
0000000000030430	0000000000040440	0000000000700020

4.4 CONCLUDING REMARKS

In this chapter, HFarrow Channelisation algorithm was developed for the design of low complexity multi-standard SDR based receiver. The hybrid Farrow filter was designed by modulating conventional Farrow Per Channel filter. The modulated filter was cascaded with with frequency response masking interpolated coefficient decimated filter. This is an efficient realisation for both uniform and non-uniform channelisation. The performance of the HFarrow channelizer was further optimized using three different digital filter number representations. These are the PDA-RNS, PDA-CRNS and PDA-CSERNS. A case application study was used to investigate the performances of the developed HFarrow and the improvements made on it. However, there were noticeable rounding errors when the different number systems representations were used. Genetic algorithm method was deployed into the modulated Farrow filter bank in order to minimize the passband ripples while maximizing the stopband attenuation using the derived objective functions.

CHAPTER 5 MULTI-RATE, MULTI-STAGE FILTER

5.1 INTRODUCTION

Single stage digital filter implementation of multi-standard channels on SDR requires a higher filter order which contributes to the complexities of the system. The complexities of the filter restrict the efficiency of implementing multi-standard channels on SDR receiver. Implementing multi-stage, multi-rate filter design for multi-standard channel receivers on SDR can achieve remarkably lower computational complexity. The process involves dividing the large filter into stages and cascading the smaller filters together using different sampling rates. The resultant effect of this is that newly generated filter with relaxed filter specifications is produced.

Multi-rate, multi-stage filter decimates the input signal to the desired output channel signal. Decreasing the sampling rate resulted in a significant reduction in the number of filter coefficients. However, by implementing the poly phase decomposition of the multi-rate, multi-stage filter reduced the computational overload to the absolute in terms of number of the samples operation per second and the filter order.

The section discusses two different multi-rate, multi-stage algorithms, namely, multi-rate, multi-stage hybrid generalized discrete Fourier transform (MHGDFT) and the multi-rate, multi-stage hybrid farrow (MHFarrow) channelisation algorithms. The first section discusses the MHGDFT channelisation algorithm. Under this section, the design procedure to implement the algorithm will be described. The second section explains the design steps required to carry out the multi-rate, multi-stage hybrid Farrow (MHFarrow) algorithm. Other similar channelisation algorithm in literature such as coefficient decimated filter bank (CDFB), improved coefficient decimated filter bank (ICDM FB) and non-maximally decimated filter bank (NMDUFB) are also used for benchmarking. The chapter concludes with the concluding remarks about the performance of the two multi-stages algorithms and their comparison with HGDFT and HFarrow algorithms.

5.2 MULTI-RATE, MULTI-STAGE HGDFT (MHGDFT) CHANNELISATION ALGORITHM

The computational complexity of HGDFT filter can be reduced further by using the multi-rate, multi-stage design. Multiplier effects can be reduced in the HGDFT channelisation algorithm by utilising the multi-rate, multi-stage HGDFT channelisation algorithm. The single rate filter with sharp cut off filter specifications is factorized into multi-stages, which when cascaded together will produce the original filter with the same filter specifications.

Relaxing the filter specifications in HGDFT algorithm can result in a better transition width, with smaller filter coefficients. The procedure for the MHGDFT algorithm is outlined. Multi-rate, multi-stage design involves the use of rational sampling rates in which the different decimators are cascaded with the interpolator. The interpolation is often proceeded by the decimation structures in order to prevent degradation due to aliasing. Interpolation increases sampling rate while decimation filters decrease the sampling rate of the signal. The gradual decrease in the sampling rate of the system leads to different simplified filtering stages which must satisfy the overall filter requirements as indicated in Figure 5.1. Passband ripples, stop band attenuation and transition bandwidth determine the complexity of filter.

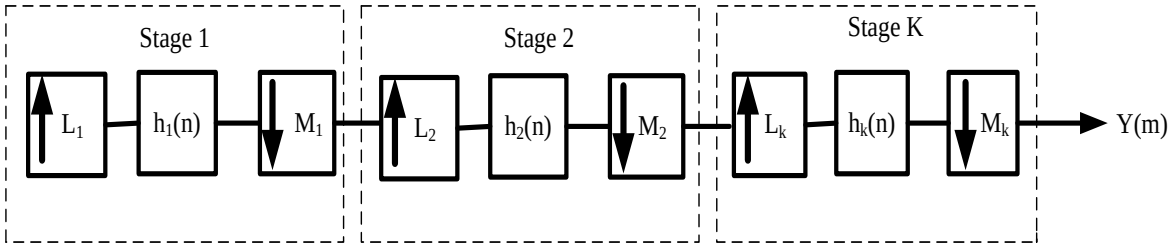

Figure 5.1. Multi-stage structures for MHGDFT Algorithm.

5.2.1 Design Steps for Multi-rate, Multi-stage HGDFT Channelisation Algorithm (MHGDFT)

The design steps for the MMHGDFT filter bank are as outlined:

1. Calculate the decimation factor, M, of the masking filter using the formula $M=\frac{\pi}{\omega_{s_i}}$. The interpolation factor is calculated using the formula $L=\left\lfloor \frac{\pi}{\omega_{si}} \right\rfloor$, where ω_{si} is the stopband frequency for each channels. Thus the fractional factor, d^k, for masking filter can be calculated here as $\frac{L_i}{M_i}$.
2. From the calculated rational rate converter, d^k, determine the multiples of the given decimation and the multiples of the given interpolation.

3. The number of stages will depend on the decimation factors and interpolation factors. The decimation factor is an even product of the decimation factors associated with different stages in a multi-stage decimation. Equation (5.1) shows how this is accomplished.

$$\begin{aligned} M &= \prod_{i=1}^{K} M_i \\ &= M1 \times M_2 \times ..., M_K \end{aligned} \tag{5.1}$$

The interpolation factor is a product of interpolation factors associated with different stages in the multi-stage interpolator.

$$\begin{aligned} L &= \prod_{i=1}^{K} L_i \\ &= L1 \times L_2 \times ..., L_K \end{aligned} \tag{5.2}$$

4. For each new stage in the base, masking and complementary filter, calculate the final sampling frequency for decimation and interpolation factor using Equation (5.3) and Equation (5.4), where $F_{s,i}$ is the initial frequency at stage i and $F_{s,K}$ is the final frequency at stage K.

$$\begin{aligned} F_{s,K} &= \frac{F_{s,i}}{M} \\ &= \frac{F_{s,i}}{\prod_{i=1}^{K} M_i} \end{aligned} \tag{5.3}$$

$$\begin{aligned} F_{s,K} &= F_{s,i} L \\ &= F_{s,i} \prod_{i=1}^{K} L_i \end{aligned} \tag{5.4}$$

5. For each stage, normalize all the channel bandwidths (BWs), such that BW_i and transition bandwidth Δ_i specifications range from 0 to 1, where $BW_{s,k} = \frac{BW_{s,j}}{F_{s,k}}$.
6. Calculate the modal bandwidth by first calculating the individual bandwidth such that $BW_i' = \frac{BW_i}{2}$. The modal bandwidth is calculated as: $BW_{Modal} = \frac{GCD(BW'_{s,1}, BW'_{s,2}, BW'_{s,3})}{2}$. This corresponds to the stop band frequency. The modal bandwidth is the greatest common divisor of all the individual bandwidth.
7. Calculate the stop band frequency such that

$$f_{stop} \leq \frac{F_{s,K}}{2} \tag{5.5}$$

In order to prevent degradation due to aliasing into the desired signal band, the stop band cut-off frequency of the i^{th} stage must conform to this relation.

$$f_{stop,i} \leq F_{s,i} - f_{stop} \tag{5.6}$$

8. Normalize the transition bandwidth, Δ_k such that it is equal to $\frac{\Delta_i}{M_i}$ for decimated signal and $\Delta_i L_i$ for interpolated signal.
9. Determine transition bandwidth for stage i using $\Delta_{s,i}$ such that the transition bandwidth for decimation factor will be equal to $\Delta'_{s,k}$=$\frac{\Delta_{s,i}}{M_k} \times e^{d^k}$. The next stage j will be defined as $\Delta'_{s,j}$=$\frac{\Delta_{s,k}}{M_k} \times e^{d^k}$. The transition bandwidth for interpolation factor is equal to $\Delta'_{s,k}$=$\Delta_{s,i} L_k \times e^{d^k}$. The next stage j will be defined as $\Delta'_{s,j}$=$\Delta_{s,k} L_k \times e^{d^k}$
10. Determine the modal transition bandwidth as: Δ_{modal}= $\min(\Delta'_i, \Delta'_2, ..., \Delta'_n)$. This corresponds to the modal transition width.
11. Calculate the modal passband width as f_{Mpass}= f_{stop} -$\Delta_{tranmodal}$.
12. Calculate the passband width as f_{pass}= $f_{stop} - \Delta'_{s,k}$.
13. Determine the new pass band ripple using the following Equation.

$$\delta'_p = \frac{\delta_p}{2 \times M} \tag{5.7}$$

Let the given stop band attenuation remains constant.

14. Determine the prototype filter order and the individual channels filter order using Bellanger formula as:

$$N = \frac{-2log_{10}(\delta'_p \delta'_s)}{3\Delta_{TBW}} - 1. \tag{5.8}$$

5.3 MULTI-RATE, MULTI-STAGE HFARROW (MHFARROW) FILTER BANK

The computational complexity of HFarrow filter can be reduced further by using multi-rate, multi-stage design algorithm.

5.3.1 Design Steps for Multi-rate, Multi-stage HFarrow Channelisation Algorithm (MHFARROW)

The design steps for the MHFarrow filter bank are as outlined:

1. Calculate the decimation factor, M, of the masking filter using this formula M=$\frac{\pi}{\omega_{si}}$. The interpolated factor is calculated using the formula L=$\left\lfloor \frac{\pi}{\omega_{si}} \right\rfloor$, where ω_{si} is the stopband frequency for each channels. Thus the fractional factor, d^k, for masking filter can be calculated here as $\frac{L_i}{M_i}$.
2. From the calculated fractional rate converter, determine the multiples of the given decimators while the given interpolated factor is constant.
3. The number of stages will depend on the decimation factors and interpolated factor. The decimation factor is an even product of the decimation factors associated with different stages in a multi-stage decimation. Equation (5.9) shows how this is accomplished.

$$\begin{aligned} M &= \prod_{i=1}^{K} M_i \\ &= M1 \times M_2 \times ..., M_K \end{aligned} \tag{5.9}$$

$$\begin{aligned} L &= \prod_{i=1}^{K} L_i \\ &= L1 \times L_2 \times ..., L_K \end{aligned} \tag{5.10}$$

4. For each new stage in base, masking and complementary filter, calculate the final sampling frequency for interpolation and decimation factor using Equation (5.11), where $F_{s,i}$ is the initial frequency at stage i and $F_{s,K}$ is the final frequency at stage K.

$$\begin{aligned} F_{s,K} &= \frac{F_{s,i}}{M} \\ &= \frac{F_{s,i}}{\prod_{i=1}^{K} M_i} \end{aligned} \tag{5.11}$$

$$\begin{aligned} F_{s,K} &= F_{s,i} L \\ &= F_{s,i} \prod_{i=1}^{K} L_i \end{aligned} \tag{5.12}$$

5. For each stage, normalise all the channel bandwidths (BWs), such that BW_i and transition bandwidth, Δ_i, specifications range from 0 to 1, and where $BW_{s,k} = \frac{BW_{s,j}}{F_{s,k}}$.
6. Calculate the modal bandwidth by first calculating the individual bandwidth such that $BW'_i = \frac{BW_i}{2}$. The modal bandwidth is thus calculated as: $BW_{Modal} = \frac{GCD(BW'_{s,1}, BW'_{s,2}, BW'_{s,3})}{2}$. This corresponds to the stop band frequency.
7. Calculate the stop band frequency sucht that

$$f_{stop} \leq \frac{F_{s,K}}{2} \tag{5.13}$$

In order to prevent degradation due to aliasing into the desired signal band, the stop band cut-off frequency of the i^{th} stage must conform to this relation.

$$f_{stop,i} \leq F_{s,i} - f_{stop} \tag{5.14}$$

8. Normalize the transition bandwidth, Δ_k such that it is equal to $\frac{\Delta_i}{M_i}$ for decimated signal and $\Delta_i L_i$ for interpolated signal.
9. Determine transition bandwidth for stage i using $\Delta_{s,i}$ such that the transition bandwidth will be equal to $\Delta'_{s,k} = \frac{\Delta_{s,i}}{M_k} \times d^k$. The next stage j will be defined as $\Delta'_{s,j} = \frac{\Delta_{s,k}}{M_k} \times d^k$. The transition bandwidth for interpolation factor is equal to $\Delta'_{s,k} = \Delta_{s,i} L_k \times e^{d^k}$. The next stage j will be defined as $\Delta'_{s,j} = \Delta_{s,k} L_k \times e^{d^k}$

10. Determine the modal TBW as: Δ_{modal}=min$(\Delta'_1, \Delta'_2, ..., \Delta'_n)$. This corresponds to the modal transition width.
11. Calculate the passband width as f_p= $_{stop} - \Delta'_{s,k}$.
12. Determine the new pass band ripple using the following Equation.

$$\delta'_p = \frac{\delta_p}{2 \times M} \tag{5.15}$$

13. Determine the prototype filter order and the individual channels filter order using Bellanger formula as:

$$N = \frac{-2log_{10}(\delta'_p \delta'_s)}{3\Delta_{TBW}} - 1[179] \tag{5.16}$$

.

5.4 CONCLUDING REMARKS

This section focuses on developing multi-rate, multi-stage HGDFT (MHGDFT) and multi-rate, multi-stage Farrow (MHFarrow) channelisation algorithms. The approach involves multiple stages HGDFT and HFarrow channelisation implementation with varying sampling rates cascaded together. Two up-sampler or interpolator with higher sampling rates are proceeded by four decimator filters in the MMHGDFT and MMHFarrow filter. The interpolator increases the sampling rate while the decimator filter reduces the sampling rate. Filter with stringent filter specifications are produced with better transition bandwidth, optimised pass band ripples and smaller filter coefficient in comparison with single filter implementation. The method is an effective approach to further reduces the complexity of filter.

CHAPTER 6 RESULTS AND DISCUSSIONS

6.1 INTRODUCTION

This Chapter presents the results and discussions of three algorithm developments with their various improvement approaches for reducing complexity in multi-standards SDR based receiver.

This work will be categorised under three main sections. Section 6.2 contains the results and discussion of developed hybrid generalised discrete Fourier transform (HGDFT) algorithm; followed by the result and interpretations of the improvement approaches on the algorithm. Section 6.3 explains the results and discussion of another algorithm approach described as hybrid Farrow (HFarrow) channelisation algorithms with its improvement techniques while Section 6.4 reports the results and the discussion obtained for multi-rate, multi-stage hybrid generalised discrete Fourier transform (MHGDFT) and multi-rate, multi-stage Farrow (MHFarrow) channelisation algorithms.

6.2 RESULTS AND INTERPRETATIONS OF HYBRID GENERALISED DISCRETE FOURIER TRANSFORM CHANNELISATION ALGORITHM (HGDFT)

Section 6.2 focused on the results as well as discussion of the HGDFT channelisation algorithm development explained in Chapter 3. The results of the improvements on HGDFT filter bank (FB) with PDA-RNS were summarised in Section 6.2.1. Section 6.2.2 explained in detail the results obtained when improvement was made on HGDT with PDA-CRNS. While Section 6.2.3 described the results obtained when improvement was made on HGDFT with PDA-CSERNS.

By adopting the design steps in Section 3.2.3, the following filter frequency characteristics in Table 3.2 were realized and briefly summarised. Using the information contained in Table 3.2, the normalized channel bandwidth of BT, ZigBee and WCDMA were 0.05, 0.2 and 0.25 respectively. The bandwidth for the channels were 0.025, 0.1 and 0.125. From Step 2 of Section 3.2.3, the passband width of the prototype filter was set to the greatest common divisor of the signal bandwidth. The modal filter normalised frequency values was 0.025, which corresponds to the stopband frequency. Decimation factor of each channel was calculated using the formula in

Step 3 of Section 3.2.3.

Following the design illustration in Section 3.2.1.2 and extracting the 4^{th} channel from the prototype filter, the filter order was 678 as explained in Table 6.1. BT channels had stopband attenuation of 0.998 and filter order of 230. Zigbee channels had stopband attenuation of 0.989 with filter order of 118. Also, for WCDMA, the stopband attenuation was 0.997 with filter order of 90. Varying the values of m, n and M, new set of frequencies were obtained with higher filter order as shown in Table 6.2. The coefficient of the prototype filter with values of $M = 40$ and $m = 10$, had stopband deviation of 0.0316 with filter order of 240. BT channels had passband deviation and stopband deviation of 0.1223 and 0.0316 had filter order of 240. The pass band group delay was between 62.5 and 63 samples. Zigbee channels had pass band deviation and stopband deviation of 0.1223 and 0.0316 with the filter order of 120 while the pass band group delay was from 62.5 and 63 samples. WCDMA channels had passband deviation and stopband deviation set to 0.001778 and 0.1223 while the filter order was 90 with the group delay of 44.5 and 45.5 samples. The total filter order for the three channels was 690. It can be explained that the filter length generated is too high for multi-standard receiver system.

Table 6.1. The frequency characteristics of Modulated filter when $m = 4$

Filter Bank	m	M	stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples(δ_{ms})	stopband attenuation(δ_{mp})	Filter length
Modal filter, H_a	4	40	0.025	0.0225	0.998	44	235
Blue tooth,H_{ma}	4	40	0.025	0.0225	0.998	44	235
Zigbee, H_{ma}	4	10	0.1	0.09	0.989	42	118
WCDMA,H_{ma}	4	8	0.2	0.175	0.997	56	90

Table 6.2. The frequency characteristics of Modulated filter when $m = 10$

Filter Bank	stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband attenuation(δ_{mp})	stopband ripples(δ_{ms})	Filter length
Modal filter, H_a	0.025	0.0225	0.998	-58	240
Blue tooth,H_{ma}	0.025	0.0225	0.998	-58	240
Zigbee, H_{ma}	0.1	0.09	0.989	-62	120
WCDMA,H_{ma}	0.2	0.175	0.987	-68	90

Following the design steps in Section 3.2.3 to the result obtained in Table 6.1, the following filter frequency characteristics in Table 6.3 were realized. The fractional factor for BT, zigbee and WCDMA were calculated from Step 3 of Section 3.2.3. Prototype low pass filter was computed by varying the fractional factor by $\frac{39}{40}$. The transition bandwidth for the prototype filter was 0.00095, with passband attenuation of 0.1dB, stopband

attenuation of -50dB, and the filter length of 180. When the rational factor of $39/40$ was applied to BT, the transition bandwidth changed to 0.00095, with passband ripple of 0.009751, stopband peak ripple of -39 and filter order of 144. Also, with the fractional factor of $\frac{9}{10}$, the transitional width of zigbee was 0.0041, with passband ripple of 0.009751, stopband peak ripple of -39 and filter order of 24.

Applying the fractional factor of $\frac{7}{8}$ to WCDMA channels, the transition width obtained was 0.01, with passband ripple of 0.0875, stopband peak ripple of -48.125 and filter order of 15. Figure 6.1 shows the magnitude response of the modal filter. Also, when prototype complementary channels was varied with fractional factor of $\frac{8}{9}$, $\frac{8}{9}$, $\frac{8}{9}$ and $\frac{7}{8}$, BT,zigbee, and WCDMA channels were obtained. The complementary transition bandwidth for modal filter, BT, Zigbee and WCDMA were: 0.00258, 0.001, 0.004, 0.01 with the filter order of 200, 148, 24 and 11 respectively. Table 6.4 shows the filter characteristics of complementary masking filter using HGDFT channelisation algorithm. Figure 6.1 shows the magnitude response of the modal filter.

Table 6.5 shows the multiplication complexity of different filter bank algorithms. Multipliers utilised by the developed HGDFT filter were 511, while the multipliers used by interpolated coefficient decimation masking filter (ICDM), coefficient decimated filter bank (CDFB), non- uniform modulated discrete Fourier transform (NU-MDFT) and generalized discrete Fourier transform filter bank (GDFT FB) as seen in [30, 66, 181, 1] were found to be 1745, 1545, and 1090 and 1444, respectively. Thus, number of multipliers utilized by the hybrid GDFT filter bank were compared and found to be lower than those of the CDFB [66], ICDM [30, 181] and GDFT [1] methods although there is trade off in complementary masking filter, H_{mc} in CDFT, ICDM FB and GDFT TB.

Hybrid GDFT filter bank achieved 53 % decrement in the number of multiplications compared to NU-MDFT [181]. Also, the developed HGDFT algorithm attained 71 % decrement in the number of multipliers compared to the CDFB filter [66], 67 % multiplication decrement was achieved when compared to ICDM [30] and 64% multipliers decrement compared to GDFT FB [1]. Figure 6.1, Figure 6.2, Figure 6.3 and Figure 6.4 showed the magnitude responses of modal filter, BT, Zigbee and WCDMA masking filter. The modal filter is a lowpass filter with passband attenuation of 0.1dB, stopband attenuation of -50dB, and the filter length of 180. Figure 6.2 shows the bandpasss filter for BT with passband ripple of 0.009751, stopband attenuation of -39 and filter order of 144. Figure 6.3 represents the bandpass filter for Zigbee with passband ripple of 0.009751, stopband attenuation of -39 and filter order of 24 and Figure 6.4 represents the bandpass filter for WCDMA with passband ripple of 0.0875, stopband attenuation of -48.125 and filter order of 15.

Table 6.3. The frequency characteristics of masking filters implemented using HGDFT filter bank.

Filter Bank	$\frac{L}{M}$	stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband attenuation(δ_{mp})	stopband ripples(δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.1	-50	180
Blue tooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.0975	-39	144
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0959	0.09	-39	24
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.0875	-48.25	15

Table 6.4. The frequency characteristics of complementary masking filter implemented using HGDFT filter bank.

Filter Bank	$\frac{L}{M}$	stopband frequency (ω_{mcs})	Pass band frequency(ω_{mcp})	Pass band attenuation (δ_{mcp})	Stopband ripples (δ_{mcs})	Filter length
Modal filter	$\frac{8}{9}$	0.026	0.024	0.1	-50	200
Bluetooth	$\frac{8}{9}$	0.026	0.024	0.088	-35.5	148
Zigbee	$\frac{8}{9}$	0.104	0.096	0.088	-35.5	24
WCDMA	$\frac{7}{8}$	0.135	0.115	0.0875	-48.25	41

Table 6.5. Multiplication complexity for non-uniform filter bank.

Filter Bank	Filter Order			Total number of multiplications
	H_a	H_{ma}	H_{mc}	
CDFB [66]	3089	400	-	1745
ICDM FB [30]	2929	160	-	1545
NU-MDFT FB [181]	187	430	469	1090
GDFT FB [1]	160	511	-	1444
HGDFT FB	170	158	183	511

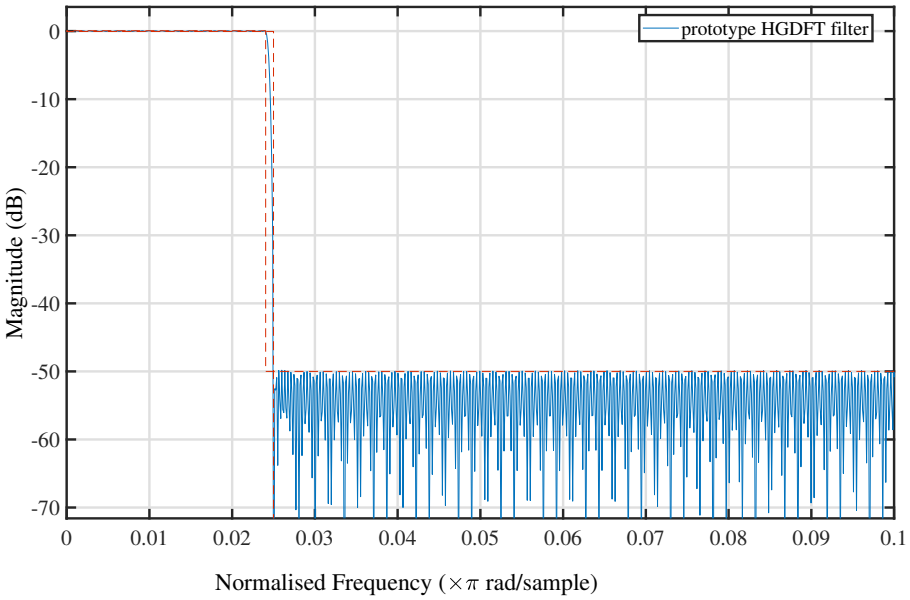

Figure 6.1. Magnitude response for the modal filter using HGDFT channelisation algorithm.

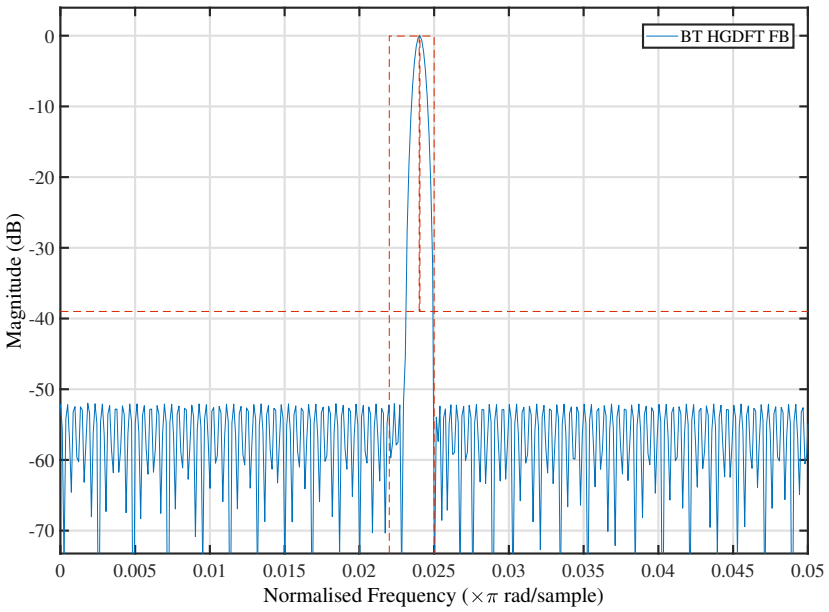

Figure 6.2. Magnitude response for the Blue-tooth masking filter using HGDFT channelisation algorithm.

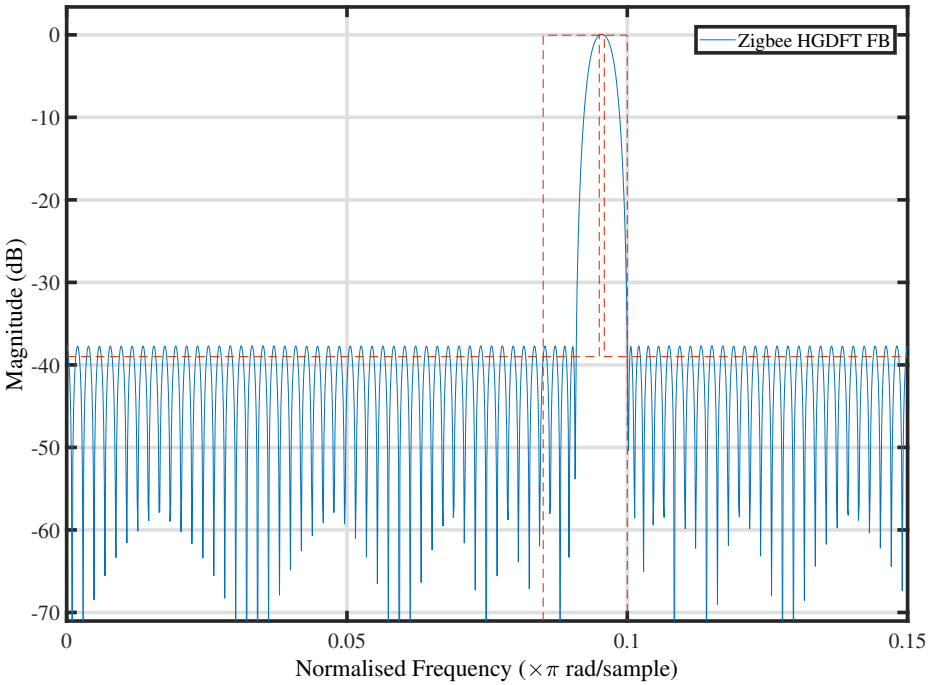

Figure 6.3. Magnitude response for the Zigbee masking filter using HGDFT channelisation algorithm.

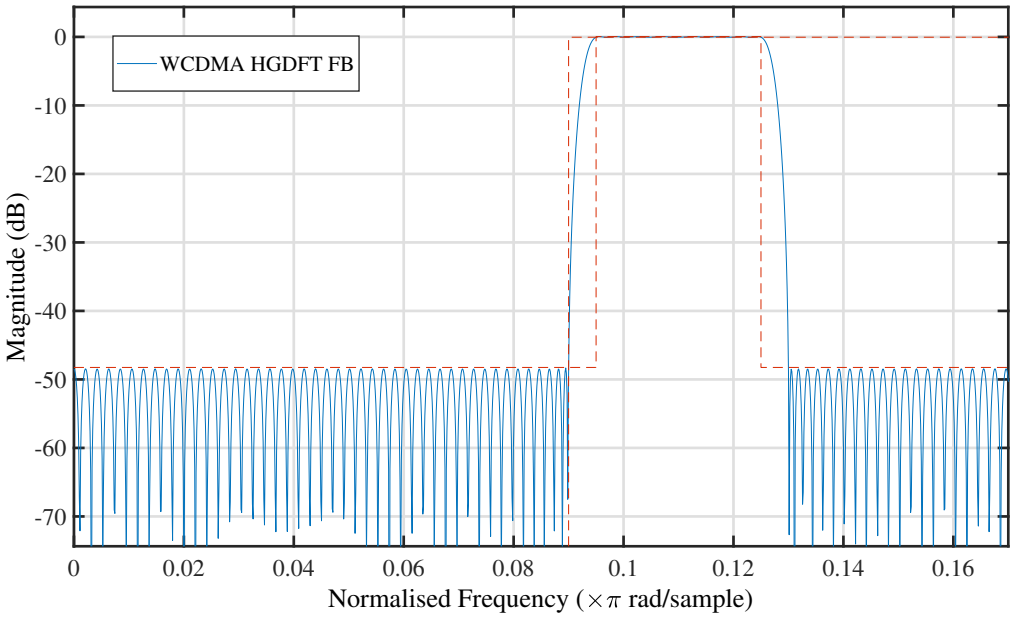

Figure 6.4. Magnitude response for the WCDMA masking filter using HGDFT channelisation algorithm.

6.2.1 Improvement of HGDFT With PDA-RNS filter (HGDFT-PDARNS)

Table 6.6. Comparison of filter specifications, in terms of number of bits.

Filter	Input Bits	Passband Ripples (dB)	Stopband Attenuation	No of Adders	Amplitude Distortion
Prototype	12-bits	3.360	−11.204	185	0.20
filter	14-bits	0.087	−48.146	182	0.205
	16-bits	0.09	-48.146	179	0.19
Blue-tooth	12-bits	3.457	−10.532	182	0.05
	14-bits	0.094	−42.818	181	0.054
	16-bits	0.095	−43.062	144	0.047
Zigbee	12-bits	0.089	−39.095	32	0.164
	14-bits	0.088	−39.094	29	0.164
	16-bits	0.089	-39.094	22	0.164
WCDMA	12-bits	0.089	−49.33	19	0.292
	14-bits	0.089	−50.43	18	0.243
	16-bits	0.0887	−50.34	17	0.242

Table 6.7. The frequency characteristics of masking filters implemented using HGDFT PDARNS filter bank.

Filter Bank	$\frac{L}{M}$	stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples(δ_{mp})	stopband attenuation(δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.09	-48.146	179
Blue tooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.095	-43.062	144
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0959	0.089	-39.094	22
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.0887	-50.34	17

Table 6.8. The three RNS values from the filter coefficient.

Filter Coeff.	Double Precision values	Binary values	Integer values	R_{31}	R_{32}	R_{33}
b0=b29	0.01441862490369894	0000000111011001	473	8	25	11
b1=b28	0.02392873769930442	0000001100010000	784	9	16	25
b2=b27	0.025455867630145797	0000001101000010	834	28	2	25
b3=b26	0.003220390664297086	0000000001101010	106	13	10	7
b4=b25	-0.035174037047374741	1111101101111111	-1153	-6	-1	-31
b5=b24	-0.072204929013503033	1111011011000010	-2366	10	-30	-23
b6=b23	-0.081482771549880470	1111010110010010	-2670	-4	-14	-30
b7=b22	-0.041374486392402265	1111101010110100	-1356	-23	-12	-3
b8=b21	0.031642040537188047	0000010000001101	1037	14	13	14
b9=b20	0.109687967473366120	0000111000001010	3594	29	10	30
b10=b19	0.128854319531854060	0001000001111110	4222	6	30	31
b11=b18	0.083665355981792339	0000101010110101	2741	13	21	2
b12=b17	-0.026818540295058590	1111110010010001	-1391	-27	-15	-5
b13=b16	-0.121692802585422390	1111000001101100	-3988	-20	-20	-28
b14=b15	-0.026818540295058590	1111010101100101	-2843	-22	-27	-5

The results obtained in Section 6.2 were further improved using PDARNS. By replacing the HGDFT with the HGDFT PDA-RNS filter, there was a 100% decrement in the number of multipliers used, from 511 to 0, in which case the use of multipliers was totally eliminated.

Table 6.6 shows the comparison of filter specifications in terms of number of bits. Bits 8, 12 and 16 were used during the simulation for comparing the prototype and the masking filters. The varied parameters were passband ripples, $\delta_{p'}$, stopband attenuation, δ_s, the number of adders and amplitude distortion. It was observed from Table 6.6 that the 16-bits showed a closer margin to the continuous filter values of HGDFT FB. Application of HGDFT PDARNS FB to the 16-bits filter values using the procedure in Section 3.3.1, resulted in the filter specifications shown in Table 6.7. Binary presentations of the HGDFT PDA-RNS frequency specifications are shown in Table 6.8 and Table 6.9.

Table 6.9. The Partial products in binary for the HGDFT PDA-RNS.

Filter Coeff.	R_{31}	R_{32}	R_{33}
b0=b29	0000000000001000(8)	0000000000011001(25)	0000000000001011(11)
b1=b28	0000000000001001(9)	0000000000010000(16)	0000000000011001(25)
b2=b27	0000000000011100(28)	0000000000000010(2)	0000000000001001(25)
b3=b26	0000000000001101(13)	0000000000001010(10)	0000000000000111(7)
b4=b25	1111111111111010(-6)	1111111111111111(-1)	1111111111100001(-31)
b5=b24	1111111111110110(10)	1111111111100010(-30)	1111111111101001(-23)
b6=b23	1111111111111100(-4)	1111111111110010(-14)	1111111111100010(-30)
b7=b22	1111111111101001(-23)	1111111111110100(-12)	1111111111111101(-3)
b8=b21	0000000000001110(14)	0000000000001101(13)	0000000000001110(14)
b9=b20	0000000000011101(29)	0000000000001010(10)	0000000000011110(30)
b10=b19	0000000000000110(6)	0000000000011110(30)	0000000000011111(31)
b11=b18	0000000000001101(13)	0000000000010101(21)	0000000000000010(2)
b12=b17	1111111111100101(-27)	1111111111110001(-15)	1111111111111011(-5)
b13=b16	1111111111101100(-20)	1111111111101100(-20)	1111111111100100(-28)
b14=b15	1111111111101010(-22)	1111111111100101(-27)	1111111111111011(-5)

6.2.1.1 Optimisation of HGDT-PDARNS with Genetic Algorithm (HGDFT PDARNS-GA)

The results obtained in Table 6.7 were further optimised using PDARNS-GA. Table 6.10 shows the frequency specifications for the masking and complementary masking filter using HGDFT PDARNS-GA. The RNS values of the HGDFT PDARNS-GA FB were indicated in Table 6.12 with its equivalent binary values as shown in Table 6.13. Table 6.14 and Table 6.15 showed device utilisation comparison. The total hardware resources occupied by HGDFT PDA-RNS were presented as follows: 1531 total slices, 3897 slices of LUTs, 3990 flip-flops, with total power of 325.71 mW and total delay of 3.122ns. The total slices occupied by NU-MDFT CSD with Pareto ABC were 2406, slice LUTs utilised were 8950, flip-flops consumed were 8980, and total power consumed was 1751 with total delay of 3.75ns. The performance of NU-MDFT filter optimized with SID-CSE showed that the total slices consumed were 1633, slice LUTs utilised were 5901 with flip-flops of 5911 and total power of 1281 while the delay time was 2.6ns.

Analysis and evaluation of these performances, as depicted in Figure 6.5 showed that HGDFT PDA-RNS utilised 22% of the total LUT, 17% of the slices LUT, 17.3% of the flip-flops, 8.7% power consumption and 25% delay

in the execution time when compared with NU-MDFT CSD with Pareto ABC [181] and SID-CSE [181].

The filter achieved 78% decrement in the number of occupied slices from 2406 to 1531 slices. There was 91.3% decrement in power consumption and 75% decrement in execution delay time.

Table 6.10. The frequency characteristics of masking filters implemented using HGDFT PDARNS-GA filter bank.

Filter Bank	$\frac{L}{M}$	stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples(δ_{mp})	stopband attenuation (δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.99984	-49.999984	176
Blue-tooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.097484	-38.999984	143
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0959	0.089984	-38.999984	22
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.087484	-48.249984	16

Table 6.11. The frequency characteristics of complementary masking filter implemented using HGDFT PDARNS-GA filter bank.

Filter Bank	$\frac{L}{M}$	stopband frequency(ω_{mcs})	Passband frequency(ω_{mcp})	Passband ripples (δ_{mcp})	stopband attenuation (δ_{mcs})	Filter length
Modal filter	$\frac{8}{9}$	0.026	0.024	0.099984	-49.999	180
Blue-tooth	$\frac{8}{9}$	0.026	0.024	0.087984	-35.4999	142
Zigbee	$\frac{8}{9}$	0.104	0.096	0.087984	-35.4999	24
WCDMA	$\frac{7}{8}$	0.135	0.115	0.087484	-48.249984	4

Table 6.12. The three RNS values obtained from the filter coefficient of HGDFT PDARNS-GA.

Filter Coeff.	Double Precision values	Binary values	Integer values	R_{31}	R_{32}	R_{33}
b0=b29	0.014434814453125000	0000000111011001	473	8	25	11
b1=b28	0.02392578125000000	0000001100010000	784	9	16	25
b2=b27	0.025451660156250000	0000001101000010	834	28	2	25
b3=b26	0.003204345703125000	0000000001101010	105	12	9	6
b4=b25	-0.035186767578125000	1111101101111111	-1153	-6	-1	-31
b5=b24	-0.07220458984375000	1111011011000010	-2366	10	-30	-23
b6=b23	-0.0814881933593750000	1111010110010010	-2670	-4	-14	-30
b7=b22	-0.041381835937500000	1111101010110100	-1356	-23	-12	-3
b8=b21	0.0316467285156250000	0000010000001101	1037	14	13	14
b9=b20	0.10968017578125000	0000111000001010	3594	29	10	30
b10=b19	0.128845214843750000	0001000001111110	4222	6	30	31
b11=b18	0.083648681640625000	0000101010110101	2741	13	21	2
b12=b17	-0.0268249511718750	11111100 10010001	-879	-20	-17	-12
b13=b16	-0.121704101562500000	1111000001101100	-3988	-20	-20	-28
b14=b15	-0.0268249511718750	1111 1100 1001 0001	-879	-20	-17	-12

From Table 6.15, total devices utilisation were compared. Slice registers utilised by HGDFT PDA-RNS filter bank were 3828 out of 419328. This showed drastic decrement in slice registers when compared with NU MDFT FB in [181] which consumed 29,797 out of 301,440 and in SDR [182] which used 15,295 out of 58,880 and also in SDR channelizer [183] which used up 29,797 out of 301,440. Also, the total slice LUTs used by HGDFT PDA-RNS filter was found to be 3369 out of 209664, while the hardwarwe utilised by NU MDFT filter in [181] was observed to be 5901 out of 298,600 and the device consumption rate in [183] was discovered to be 21,169 out of 150,720 and [182] used up 14,726 out of 58,880.

Table 6.13. Partial products in binary for the HGDFT PDARNS-GA.

Filter Coeff.	R_{31}	R_{32}	R_{33}
b0=b29	0000000000001000(8)	0000000000011001(25)	0000000000001011(11)
b1=b28	0000000000001001(9)	0000000000010000(16)	0000000000011001(25)
b2=b27	0000000000011100(28)	0000000000000010(2)	0000000000001001(25)
b3=b26	0000000000001100(12)	0000000000001001(9)	0000000000000110(6)
b4=b25	1111111111111010(-6)	1111111111111111(-1)	1111111111100001(-31)
b5=b24	1111111111110110(10)	1111111111100010(-30)	1111111111101001(-23)
b6=b23	1111111111111100(-4)	1111111111110010(-14)	1111111111100010(-30)
b7=b22	1111111111101001(-23)	1111111111110100(-12)	1111111111111101(-3)
b8=b21	0000000000001110(14)	0000000000001101(13)	0000000000001110(14)
b9=b20	0000000000011101(29)	0000000000001010(10)	0000000000011110(30)
b10=b19	0000000000000110(6)	0000000000011110(30)	0000000000011111(31)
b11=b18	0000000000001101(13)	0000000000010101(21)	0000000000000010(2)
b12=b17	1111111111101100(-20)	111111111110 11111(-17)	1111111111101111(-12)
b13=b16	1111111111101100(-20)	1111111111101100(-20)	1111111111100100(-28)
b14=b15	1111111111101100(-20)	111111111110 11111(-17)	1111111111101111(-12)

Table 6.14. Device and power utilisation.

Parameter	Continuous Coefficients	HGDFT PDA-RNS	HGDFT PDARNS-GA	CSD with Pareto ABC[181]	SID-CSE [181]
Total slices	2507	1531	1520	2406	1633
Slice LUTs	8694	3897	3900	8950	5901
Flip- flops	10313	3990	3989	8980	5911
Total Power (mW)	1865	325.71	325	1751	1281
Total Delay (ns)	45.125	3.122	3.21	3.75	2.6

Table 6.15. Comparison of device utilisation.

Parameter	SDR in [182]	SDR channelizer in [183]	SDR channelizer in NU MDFT FB [181]	HGDFT PDARNS filter	HGDFT PDARNS-GA
Slice Registers	15,295 out of 58,880	29,797 out of 301,440	5880 out of 597,200	3828 of 419328	3820 of 419328
Slice LUTs	14,726 out of 58,880	21,169 out of 150,720	5901 out of 298,600	3369 out of 209664	3369 out of 209664

Thus, Table 6.14 proves that the hardware resources utilisation for the implementation of HGDFT PDA-RNS filter bank were found to be less than that in SDR channelizer [181, 182, 183]. The lower filter order in this design, coupled with the modularity in RNS, clearly contributed to lower slice requirement, lower power consumption when compared to design in [120, 181].

Also, from Table 6.14 and Table 6.15, the device utilisation were compared. The total hardware resources occupied by HGDFT PDARNS-GA were presented as follows: total slices occupied were 1520; slice LUTs utilised were 3900; the flip flops used were 3989; with total power of 325 mW and total delay of 3.21 ns.

Analysis and evaluation of these performances, as depicted in Figure 6.5 showed that improving HGDFT PDARNS-GA filter utilised 21% of the total LUT, 15% of the slices LUT, 18% decrement in the flip-flops used, 9% power consumption and 25% delay in the execution time when compared with HGDFT PDA-RNS, NU-MDFT CSD with Pareto ABC [181] and SID-CSE [181]. The filter achieved an 79% decrement in the number of occupied slices from 2406 to 1520 slices. There was 90% decrement in power consumption and 75% decrement in execution delay time. Figure 6.7 shows the bandpasss filter for BT with passband ripple of 0.097484, stopband attenuation of -38.999984 and filter order of 143. Figure 6.6 represents the bandpass filter for Zigbee with passband ripple of 0.089984, stopband attenuation of -38.999984 and filter order of 22 and Figure 6.8 represents the bandpass filter for WCDMA with passband ripple of 0.087484, stopband attenution of -48.249984 and filter order of 16.

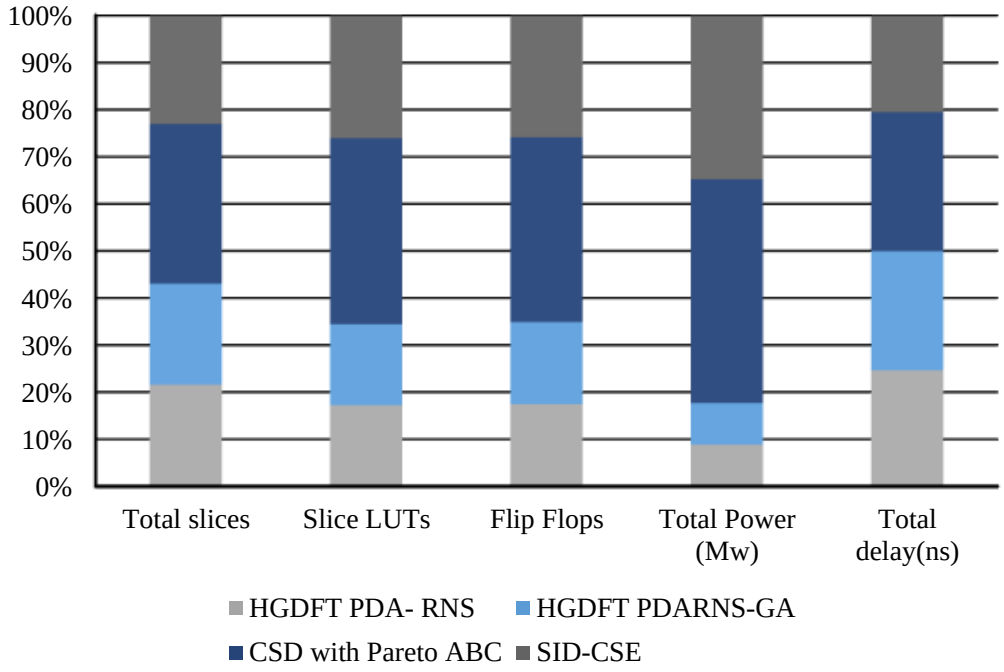

Figure 6.5. Resources utilisation of HGDFT PDA-RNSGA with other filter bank.

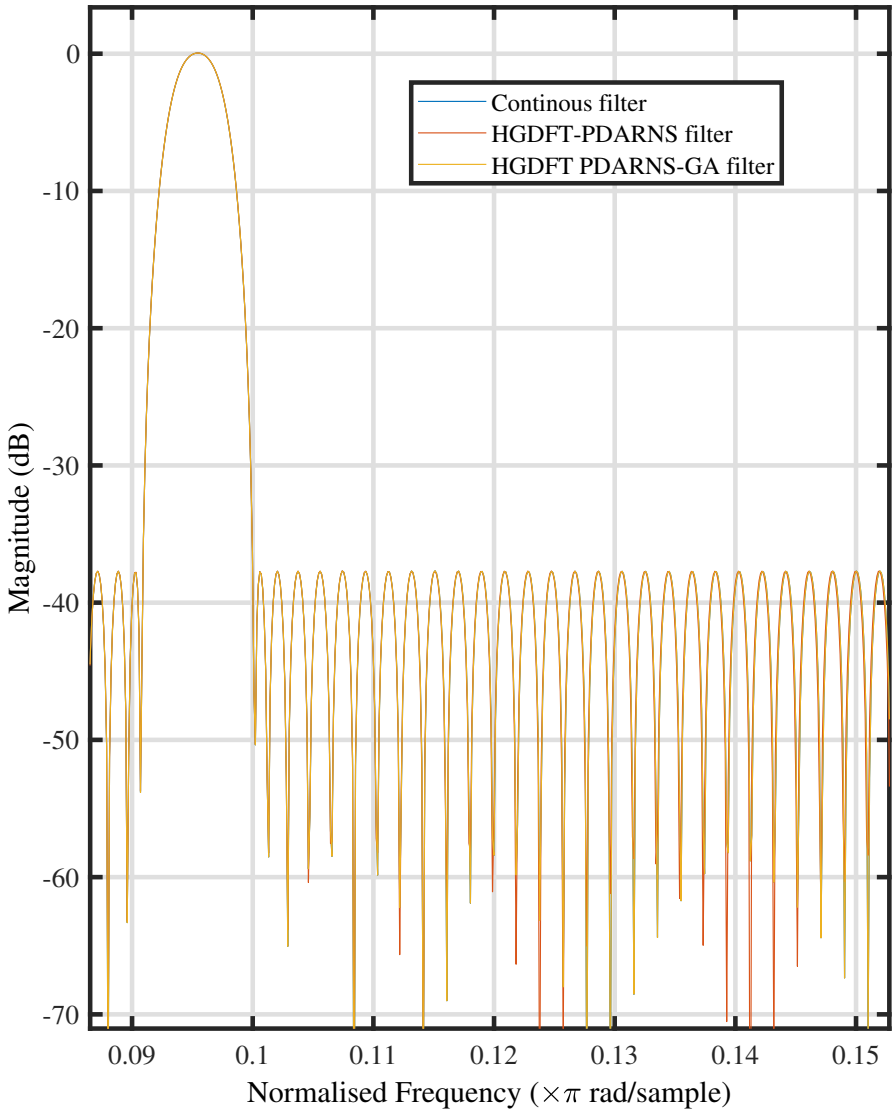

Figure 6.6. Frequency response of Zigbee channels using HGDFT PDARNS-GA masking filter.

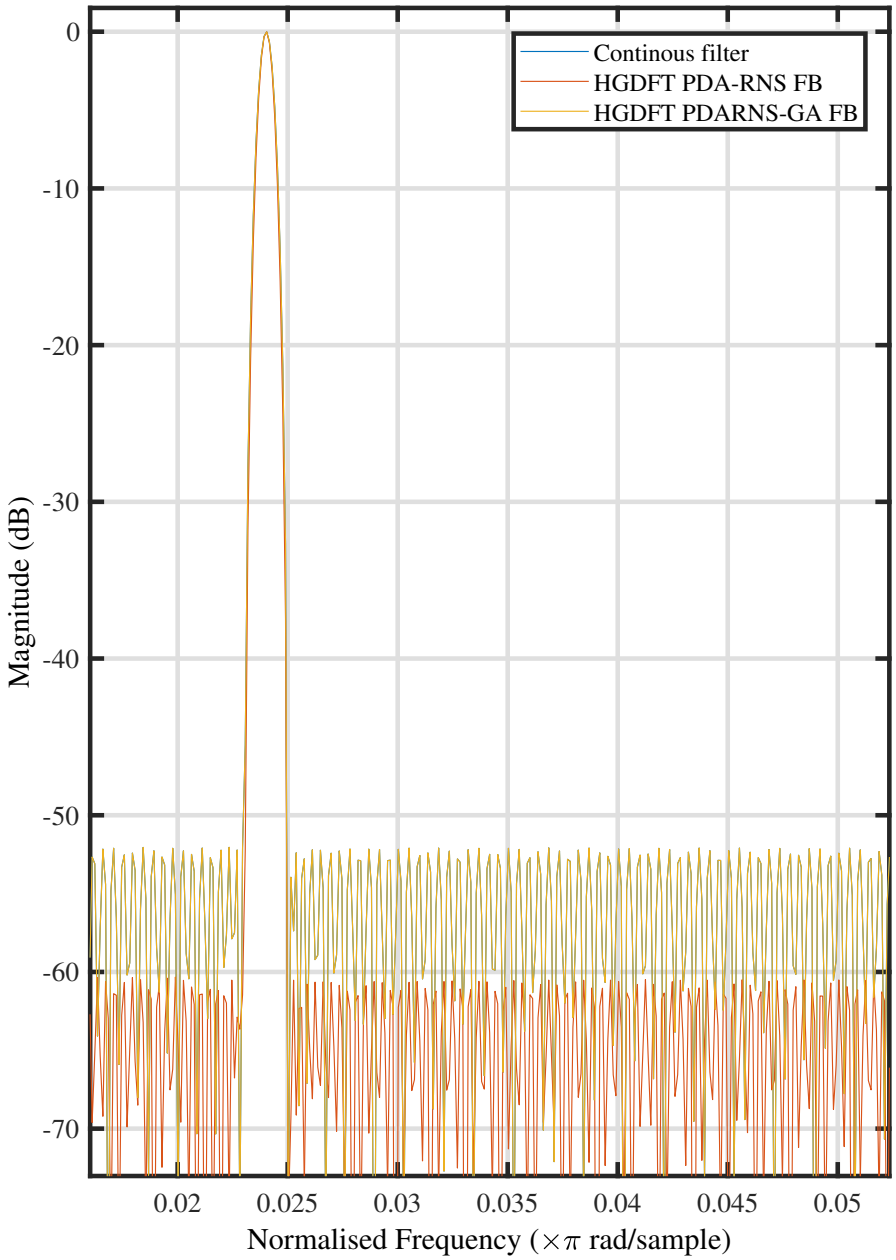

Figure 6.7. Frequency response of BT channels using HGDFT PDARNS-GA masking filter.

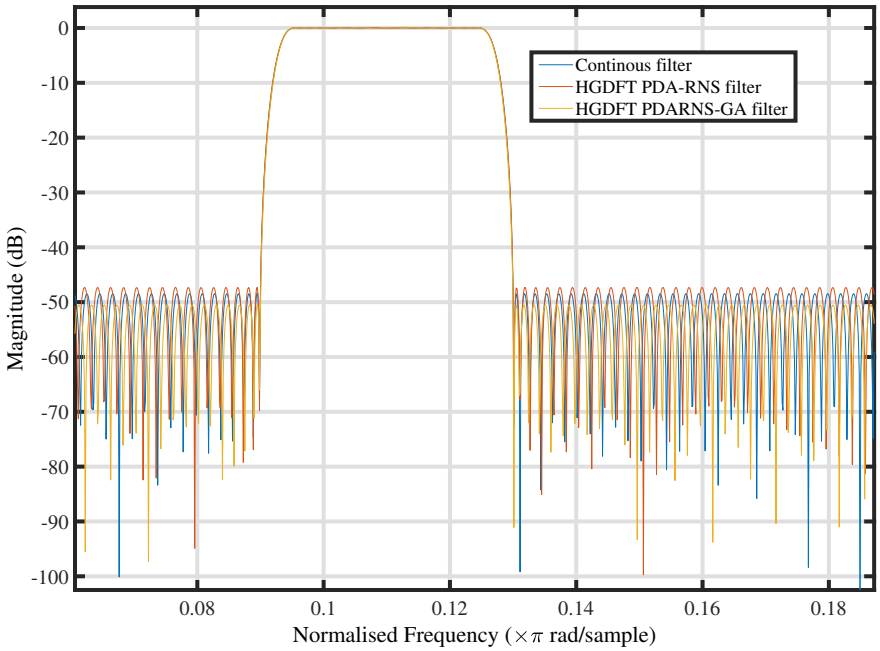

Figure 6.8. Frequency response of WCDMA using HGDFT PDARNSGA masking filter in comparison with HGDFT PDA-RNS masking filter.

6.2.2 Improvement of HGDFT with Parallel Distributed Arithmetic Based Canonical Signed Residual Number System (PDA-CSRNS)

A 29-tap FIR filter coefficients generated for HGDFT filter designed from Section 3.2.4 was used for the implementation. The results obtained in Section 6.2 were further improved using PDACSRNS.

Table 6.16 shows the comparison of filter specifications in terms of number of bits. Bits 8, 12 and 16 were used during the simulation for comparing the prototype and masking filters. The varied parameters were passband ripples, δ_p, stopband attenuation, δ_s, the number of adders and amplitude distortion. It was observed from Table 6.16 that the 16-bits showed a closer margin to the continuous filter values of HGDFT FB. Application of HGDFT PDACSRNS FB to the 16-bits filter values using the procedure in Section 3.3.1, resulted in the filter specifications shown in Table 6.17. The generated filter coefficients were converted into binary formats. This was later converted into CSD. Table 6.19 showed the generated CSD bits. The number of 1's generated have been reduced by 34.5% which leads to decrement in the complexity of the filter bank.

Table 6.16. Comparison of CSRNS filter specifications using different word length.

Filter	Input Bits	Passband Ripples (dB)	stopband Attenuation (dB)	No of Adders from SPT	Amplitude Distortion
Prototype filter	12-bits	3.355	−15.211	180	0.211
	14-bits	0.089	−48.243	179	0.212
	16-bits	0.093	-48.24	176	0.21
Blue-tooth	12-bits	4.565	−11.521	179	0.0465
	14-bits	0.0932	−42.82	197	0.0432
	16-bits	0.09	−41.856	140	0.044
Zigbee	12-bits	0.0889	−39.12	29	0.166
	14-bits	0.085	−39.11	27	0.165
	16-bits	0.082	-40	20	0.168
WCDMA	2-bits	0.085	−43.638	17	0.28
	14-bits	0.084	−50	17	0.2745
	16-bits	0.083	−52.2	15	0.223

Table 6.17. Filter frequency specifications and hardware complexity of HGDFTPDA-CSRNS FB.

Filter	SPT Terms used	Passband ripples (dB)	Stopband attenuation	Total number of Adders	Amplitude distortion
Prototype filter	7 SPTs	0.093	−48.243	168	0.211
	6 SPTs	0.0893	−48.243	169	0.2112
	5 SPTs	0.096	-48.221	174	0.21
	4 SPTs	0.095	-48.13	173	0.212
Blue-tooth	7 SPTs	0.09	−41.856	140	0.044
	6 SPTs	0.091221	−41.234	138	0.0429
	5 SPTs	0.0912	−41.22	139	0.0411
	4 SPTs	0.0911	−41.12	139	0.0409
Zigbee	7 SPTs	0.082	−40	19	0.165
	6 SPTs	0.082	−39.43	18	0.167
	5 SPTs	0.0831	−38.78	18	0.165
	4 SPTs	0.0841	−38.54	19	0.166
WCDMA	7 SPTs	0.084	−51.62	18	0.2744
	6 SPTs	0.0835	−51.67	18	0.2745
	5 SPTs	0.0834	−51.93	18	0.2746
	4 SPTs	0.0832	−51.91	19	0.2745
	3 SPTs	0.0832	−51.91	19	0.2743

Table 6.18. The frequency characteristics of masking filters implemented using HGDFT PDA-CSRNS filter bank.

Filter Bank	$\frac{L}{M}$	stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband Ripples (δ_{mp})	stopband Attenuation (δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.096	-48.221	169
Bluetooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.0912	−41.22	138
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0959	0.0831	−38.78	18
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.0834	−51.93	18

Table 6.19. Partial products of the HGDFT FB using PDA-CSRNS.

Filter Coeff.	R_{31}	R_{32}	R_{33}
b0=b29	0000000000001000(8)	000000000010$\bar{1}$001(25)	000000000010$\bar{1}$0$\bar{1}$(11)
b1=b28	0000000000001001(9)	0000000000010000(16)	000000000010$\bar{1}$001(25)
b2=b27	000000000100$\bar{1}$00(28)	0000000000000010(2)	0000000000001001(25)
b3=b26	000000000010$\bar{1}$01(13)	0000000000001010(10)	00000000000100$\bar{1}$(7)
b4=b25	1111111111111010(-6)	1111111111111111(-1)	1111111111100001(-31)
b5=b24	111111111111010(10)	1111111111100010(-30)	1111111111101001(-23)
b6=b23	1111111111111100(-4)	1111111111110010(-14)	1111111111100010(-30)
b7=b22	1111111111101001(-23)	1111111111110100(-12)	1111111111111101(-3)
b8=b21	000000000100$\bar{1}$0(14)	000000000010$\bar{1}$01(13)	000000000100$\bar{1}$0(14)
b9=b20	000000000100$\bar{1}$01(29)	0000000000001010(10)	000000000100$\bar{1}$0(30)
b10=b19	00000000000010$\bar{1}$0(6)	000000000100$\bar{1}$0(30)	00000000010000$\bar{1}$(31)
b11=b18	000000000010$\bar{1}$01(13)	0000000000010101(21)	0000000000000010(2)
b12=b17	1111111111100101(-27)	1111111111110001(-15)	111111111111110$\bar{1}$(-5)
b13=b16	111111111110$\bar{1}$00(-20)	111111111110$\bar{1}$00(-20)	1111111111100100(-28)
b14=b15	1111111111101010(-22)	1111111111100101(-27)	111111111111110$\bar{1}$(-5)

6.2.2.1 Optimisation Of HGDFT PDACSRNS Genetic Algorithm (HGDFT PDACSRNS-GA)

The results obtained in Table 6.17 were further optimised using PDACSRNS-GA. Table 6.20 shows the frequency specifications for the masking and complementary masking filter using HGDFT PDACSRNS-GA. The RNS values of the HGFTB PDACSRNS-GA FB were indicated in Table 6.21 with its equivalent CSD bits were as shown in Table 6.22. Table 6.23 and Table 6.24 showed device utilisation comparison.

The total slices occupied by HGDFT PDA-CSRNS were 1470; slice LUTs utilised were 3472; flip-flops consumed were 3602; total power consumed was 341.36 with total delay of 3.22ns. Also, the total slices occupied by NU-MDFT CSD with Pareto ABC were 2406, slice LUTs utilised were 8950, flip-flops consumed were 8980, total power consumed was 1751 with total delay of 3.75ns. The performance of NU-MDFT filter optimized with SID-CSE showed that the total slices consumed were 1633, slice LUTs utilised were 5901 with flip- flops of 5911 and total power of 1281 while the delay time was 2.6ns.

Table 6.20. The frequency characteristics of masking filters implemented using HGDFT PDACSRNS-GA filter bank.

Filter Bank	$\frac{L}{M}$	Stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples(δ_{mp})	Stopband attenuation (δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.0999864	-49.999864	167
Bluetooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.097486	-38.999	137
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0959	0.089986	-38.9999861	16
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.087486	-48.124986	17

Table 6.21. The three HGDFT PDACSRNS-GA values from the filter coefficient.

Filter Coeff.	Double Precision values	Binary values	Integer values	R_{31}	R_{32}	R_{33}
b0=b29	0.014439826589188383	0000000111011001	473	8	25	11
b1=b28	0.023925294525676558	0000001100010000	784	9	16	25
b2=b27	0.025452250396302972	0000001101000010	834	28	2	25
b3=b26	0.003219876117672146	0000000001101010	106	13	10	7
b4=b25	-0.0351689729181176006	1111101101111111	-1152	-6	-1	-31
b5=b24	-0.072194689289169181	1111011011000010	-2366	10	-30	-23
b6=b23	-0.081471081849437965	1111010110010010	-2670	-4	-14	-30
b7=b22	-0.041368668067425840	1111101010110100	-1356	-23	-12	-3
b8=b21	0.031637637425677975	0000010000001101	1037	14	13	14
b9=b20	0.109672239834519690	0000111000001010	3594	29	10	30
b10=b19	0.128836669712955410	0001000001111110	4222	6	30	31
b11=b18	0.08353350917422709	0000101010110101	2741	13	21	2
b12=b17	-0.026814421305748543	11111100 10010001	-2715	-13	-5	-24
b13=b16	-0.121675252764629870	1111000001101100	-3987	-12	-13	-6
b14=b15	-0.08285881403315159	1111 1100 1001 0001	-879	-20	-17	-12

Table 6.22. The partial products in binary for the HGDFT PDACSRNS-GA.

Filter Coeff.	R_{31}	R_{32}	R_{33}
b0=b29	0000000000001000(8)	000000000010$\bar{1}$001(25)	000000000010$\bar{1}$0$\bar{1}$(11)
b1=b28	0000000000001001(9)	0000000000010000(16)	000000000010$\bar{1}$001(25)
b2=b27	000000000100$\bar{1}$00(28)	0000000000000010(2)	0000000000001001(25)
b3=b26	000000000010$\bar{1}$01(13)	0000000000001010(10)	000000000000100$\bar{1}$(7)
b4=b25	1111111111111010(-6)	1111111111111111(-1)	1111111111100001(-31)
b5=b24	111111111111$\bar{1}$0$\bar{1}$0(10)	1111111111100010(-30)	1111111111101001(-23)
b6=b23	1111111111111100(-4)	1111111111110010(-14)	1111111111100010(-30)
b7=b22	1111111111101001(-23)	1111111111110100(-12)	1111111111111101(-3)
b8=b21	000000000010$\bar{1}$0$\bar{1}$0(14)	000000000010$\bar{1}$01(13)	000000000010$\bar{1}$0$\bar{1}$0(14)
b9=b20	000000000100$\bar{1}$01(29)	0000000000001010(10)	000000000100$\bar{1}$0$\bar{1}$0(30)
b10=b19	000000000000$\bar{1}$0$\bar{1}$0(6)	000000000100$\bar{1}$0$\bar{1}$0(30)	000000000100000$\bar{1}$(31)
b11=b18	000000000010$\bar{1}$01(13)	0000000000010101(21)	0000000000000010(2)
b12=b17	111111111111010$\bar{1}$(-13)	111111111111 110$\bar{1}$(-5)	1111111111101000(-12)
b13=b16	111111111111$\bar{1}$00(-20)	111111111111$\bar{1}$00(-20)	1111111111100100(-28)
b14=b15	111111111111$\bar{1}$00(-20)	11111111111 0000$\bar{1}$(-17)	1111111111111000$\bar{1}$(-12)

Analysis and evaluation of these performances, as depicted in Figure 6.9 showed that improving HGDFT with PDA-CSRNS utilised 21% of the total LUT, 15% of the slices LUT, 17.3% of the flip-flops used, 9% power consumption and 26% delay in the execution time when compared with NU-MDFT CSD with Pareto ABC [181].

The filter achieved a 79% decrement in the number of occupied slices from 2406 to 1470 slices. There was 91% decrement in power consumption and 75% decrement in execution delay time.

From Table 6.24, total devices utilised were compared. Slice registers utilised by HGDFT with PDA-CSRNS filter bank were 3723 out of 419328. This showed drastic decrement in slice registers when compared with NU MDFT FB in [181] which consumed 29,797 out of 301,440 and in SDR [182] which used 15,295 out of 58,880 and also in SDR channelizer [183] which used up 29,797 out of 301,440. Also, the total slice LUTs used by HGDFT with PDA-CSRNS filter were found to be 3723 out of 209664, while the hardware utilised

by NU MDFT filter in [181] was observed to be 5901 out of 298,600 and the device consumption rate in [183] was discovered to be 21,169 out of 150,720 and [182] used up 14,726 out of 58,880. Thus, Table 6.24 proves that the hardware resources utilisation for the implementation of HGDFT PDA-CSRNS filter bank were found to be less than that in SDR channelizer [181, 182, 183]. The lower filter order in this design, coupled with the modularity in RNS and the decrement in the generated number of ones used, clearly contributed to lower slice requirement, lower power consumption when compared to design in [120, 181]. Figure 6.10 shows the bandpasss filter for BT with passband ripple of 0.0912, stopband attenuation of -41.22 and filter order of 138. Figure 6.11 represents the bandpass filter for Zigbee with passband ripple of 0.0831, stopband attenuation of -38.78 and filter order of 18.

The total hardware resources occupied by HGDFT PDACSRNS-GA were presented in Table 6.23. HGDFT PDACSRNS-GA occupied 1440 total slices; 3445 slices of LUTs; 3602 flip-flops with total power of 324.98 mW and total delay of 2.791. Analysis and evaluation of these performances, as depicted in Figure 6.5 showed that improving HGDFT PDACSRNS-GA filter bank utilised 20% of the total LUT, 15% of the slices LUTs, 15% of the flip-flops used, 8.9% power consumption and 24% delay in the execution time when compared with HGDFT PDA-RNS, NU-MDFT CSD with Pareto ABC [181] and SID-CSE [181].

The filter achieved an 80% decrement in the number of occupied slices from 2406 to 1440 slices. There was 91.1% decrement in power consumption and 76% decrement in execution delay time.

Table 6.23. Device and Power utilisation.

Parameter	Continuous Coefficients	HGDFT PDA-CSRNS	HGDFT PDACSRNS -GA	CSD with Pareto ABC[181]	SID-CSE [181]
Total slices	2507	1470	1440	2406	1633
Slice LUTs	8694	3472	3445	8950	5901
Flip- flops	10313	3586	3602	8980	5911
Total Power (mW)	1865	341.36	324.98	1751	1281
Total Delay (ns)	45.125	3.22	2.791	3.75	2.6

Table 6.24. Comparison of device utilisation.

Parameter	SDR in [182]	SDR channelizer in [183]	SDR channelizer using NU MDFT FB [181]	HGDFT PDA-CSRNS FB	HGDFT PDA CSRNS-GA
Slice Registers	15,295 out of 58,880	29,797 out of 301,440	5880 out of 597,200	3723 of 419328	3723 of 419328
Slice LUTs	14,726 out of 58,880	21,169 out of 150,720	5901 out of 298,600	3409 out of 209664	3409 out of 209664

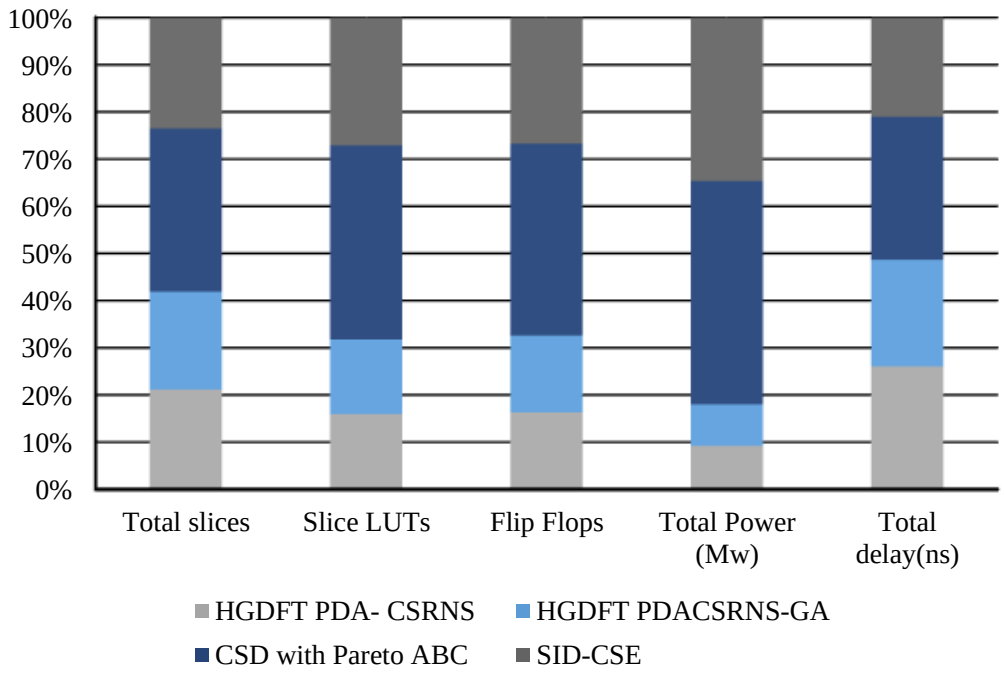

Figure 6.9. Resources utilisation for different filter algorithm relative to HGDFT PDA-CSRNSGA.

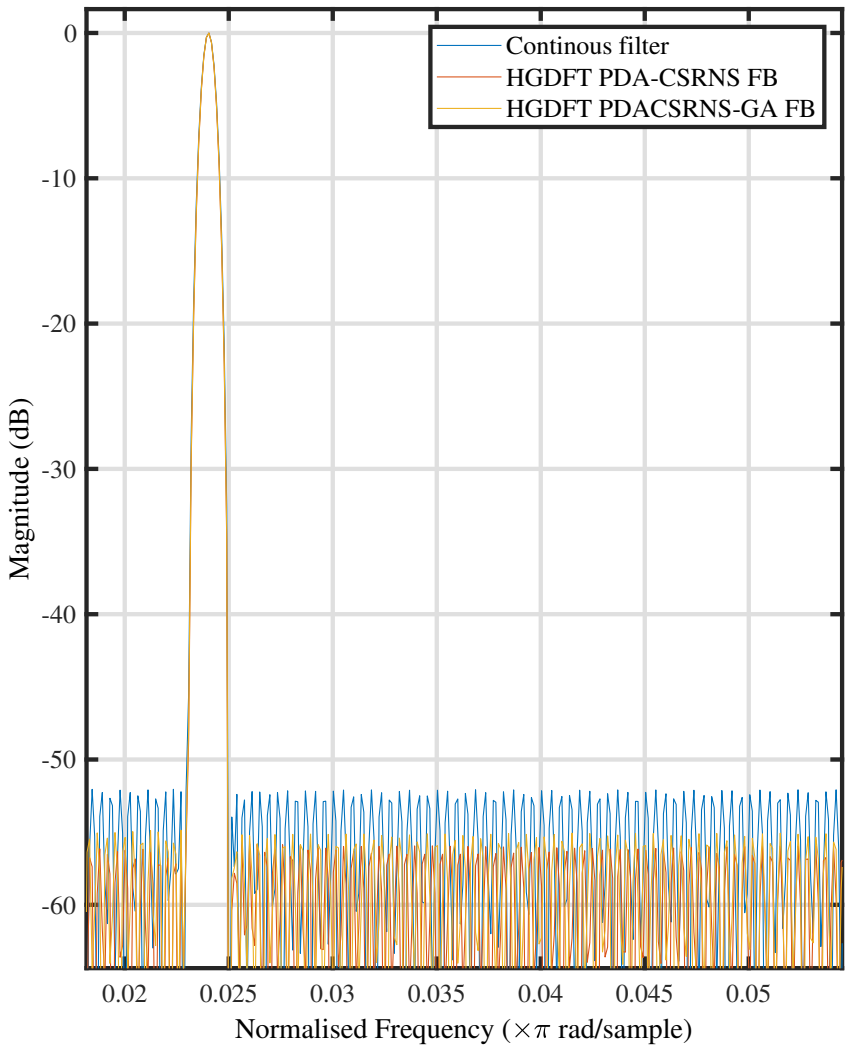

Figure 6.10. Frequency response of BT channels using HGDFT PDACSRNS-GA in comparison with HGDFT PDA-CSRNS masking filter and Continuous filter.

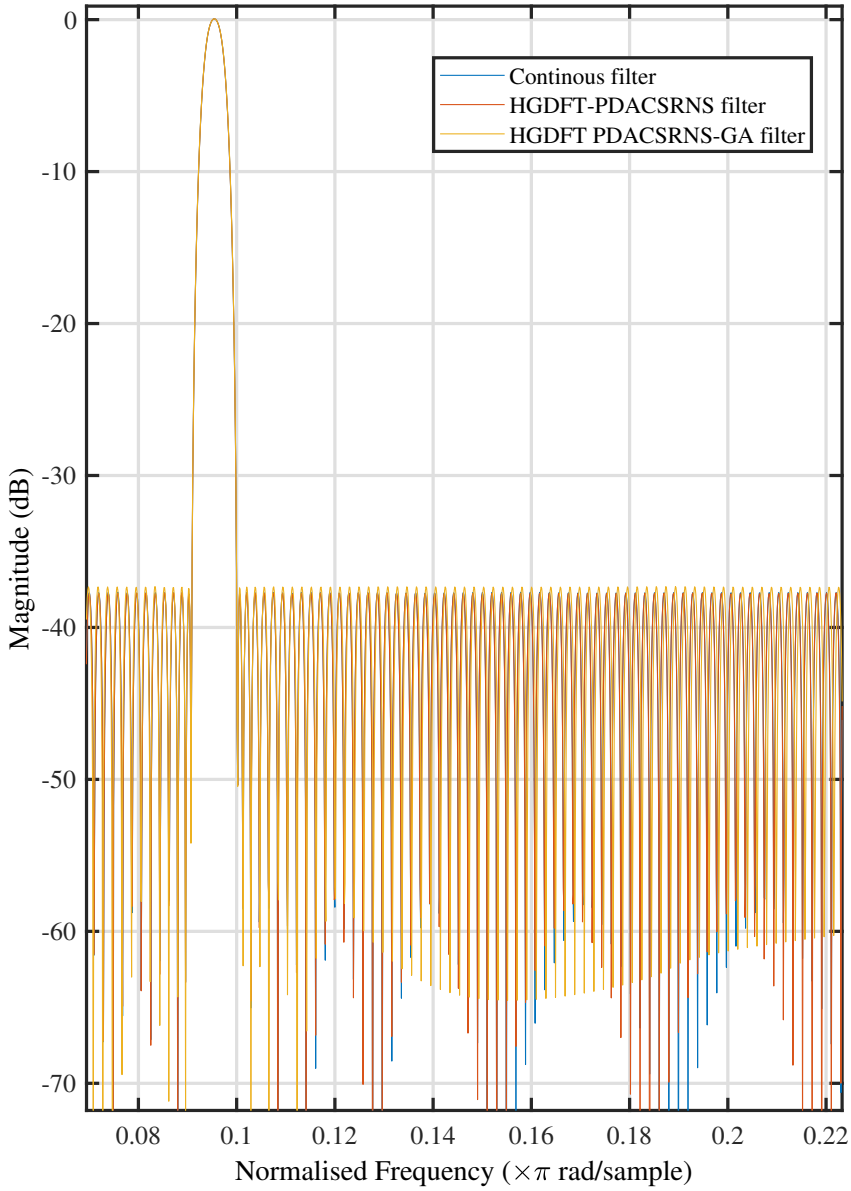

Figure 6.11. Frequency response of Zigbee channels using HGDFT PDACSRNS-GA in comparison with HGDFT PDA-CSRNS masking filter and Continuous filter.

6.2.3 Improvement of HGDFT Parallel Distributed Arithmetic Based Common Sub-Expression Elimination RNS (PDA-CSERNS) Technique

Also, by replacing the multiplier in Table 6.3 with HGDFT PDA-CSERNS, there was 100% decrease in multipliers consumed from 511 to 0, in which case the use of multipliers were totally eliminated. Only adders

were used during the implementation. Table 6.25 and Table 6.26 showed silicon area complexity for the hybrid GDFT with PDA-CSERNS filter.

The HGDFT with PDA-CSERNS method utilised 22% of the total slices, 16% of the slices LUTs, 18% of the flip-flops used, 8.2% power consumption and 25% delay in the execution time. It was observed from Figure 6.12 that this filter achieved a 78% decrement in the number of occupied slices from 2406 to 1456. There was 91.8% decrement in power consumption and 75% decrement in execution delay time. All these were in comparison with [30, 181]. The lower filter order in this design clearly contributed to lower slice requirement, lower power consumption and lower execution time compared to design using [120, 181]. Optimising with HGDFT PDA CSERNS-GA improves the performances of HGDFT PDA-CSERNS filter. It was observed from Figure 6.12 that this HGDFT PDACSERNS-GA filter achieved a 79% decrement in the number of occupied slices from 2406 to 1455. There was 93% decrement in power consumption and 76% decrement in execution delay time.

Table 6.25. Device and power utilisation.

Parameter	HGDFT PDA-CSERNS FB	HGDFT PDACSERNS-GA	CSD with Pareto ABC	SID-CSE
Total slices	1456	1455	2406	1633
Slice LUTs	3360	3360	8950	5901
Flip- flops	3540	3537	8980	5911
Total Power (mW)	334	330	1751	1281
Total Delay (ns)	3.18	3.14	3.75	2.6

Table 6.26. Comparison of device utilisation for HGDFT with PDA-CSERNS.

Parameter	SDR channelizer in [7]	SDR channelizer in [8]	SDR channelizer using NU-MDDFT FB	HGDFT PDA-CSERNS	HGDFT PDA CSERNS -GA
Slice Registers	15,295 out of 58,880	29,797 out of 301,440	5880 out of 597,200	3500 out of 419328	3500 out of 419328
Slice LUTs	14,726 out of 58,880	21,169 out of 150,720	5901 out of 298,600	3318 out of 209664	3316 out of 209664

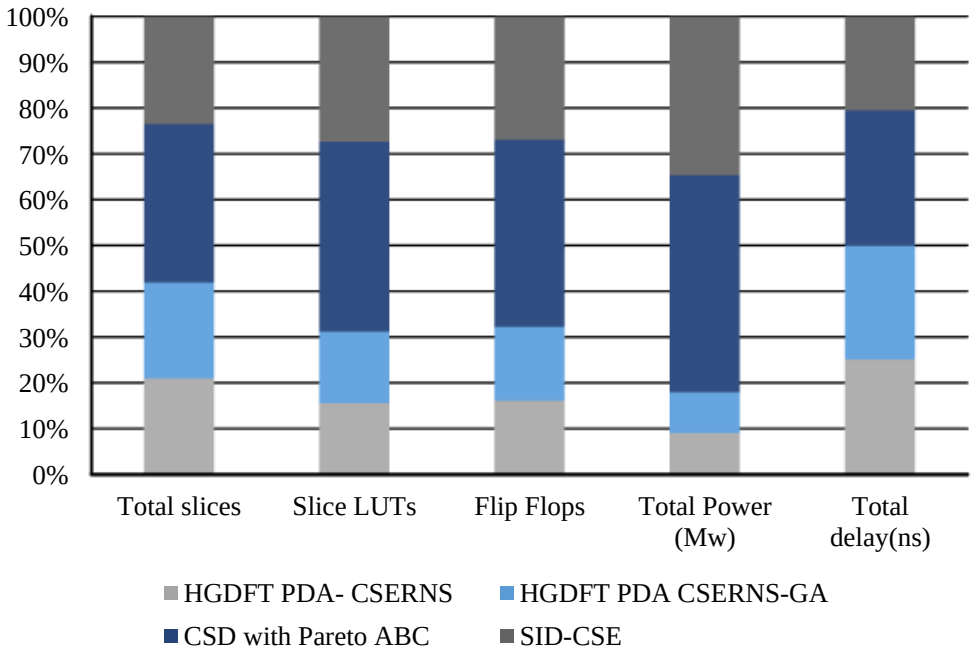

Figure 6.12. Plot of HGDFT FB with PDA-CSERNS.

6.3 RESULTS AND DISCUSSIONS FOR HFARROW CHANNELISATION ALGORITHM

This Section presents the results as well as discussion of the HFarrow algorithm development and its various improvement methods as discussed in Chapter Four. Section 6.3 discusses the result obtained from the development of HFarrow channelisation algorithm. Section 6.3.1 presents the results of the improvements made on HFarrow with PDA-RNS. Section 6.3.2 explains in detail the results obtained when improvement is made on HFarrow with PDA-CRNS. Section 6.3.3 discusses the results obtained with improvement of HFarrow with PDA-CSERNS.

Using the information contained in Table 6.27, the normalized channel bandwidth of BT, ZigBee and WCDMA were 0.05, 0.2 and 0.25 respectively. The channel bandwidths were 0.025, 0.1 and 0.125. From Step 2 of Section 4.2.5, the passband width of the prototype filter was set to the greatest common divisor of the signal bandwidth. The modal filter normalised frequency values was 0.025, which corresponds to the modal stopband frequency.

Decimation factor for each channel was calculated using the formula in Step 3 of Section 4.2.5. The following were the decimation factors for modal filter, BT, Zigbee, WCDMA: $\frac{39}{40}$, $\frac{39}{40}$, $\frac{9}{10}$, and $\frac{7}{8}$ respectively. The modal decimation factor was found to be $\frac{39}{40}$. When fractional rate, d^k, of $\frac{39}{40}$ was applied to the modal filter, transition bandwidth computed was $0,002375$, with passband peak ripple of 0.1dB, stopband peak ripple of -50dB, and

the filter length of 132. Also, when fractional rate, d^k of $\frac{39}{40}$ was applied to the BT channels, transition bandwidth computed was $0,0026$, with passband peak ripple of 0.00975dB, stopband peak ripple of -39dB, and the filter length of 107.

When the fractional rate of $\frac{9}{10}$ was applied to Zigbee, the transition bandwidth was calculated to be $0,011$, with passband ripple of 0.09, stopband peak ripple of -39 and filter order of 24. When the fractional rate of $\frac{7}{8}$ was applied to WCDMA, the transition bandwidth was calculated to be $0,021$, with passband ripple of 0.0875, stopband peak ripple of -48.125 and filter order of 9.

Also, stopband for complementary masking frequency, ω_{mcs}, was calculated using Table 3.1. The value of stopband edge, passband edge and the fractional rate were calculated using design Steps 5 through to Steps 9 in Section 4.2.5. The complementary masking decimation factor for modal filter, BT, Zigbee and WCDMA were: $\frac{8}{9}$, $\frac{8}{9}$, $\frac{8}{9}$ and $\frac{7}{8}$ respectively. The complementary masking transition bandwidth for modal filter, BT, Zigbee and WCDMA were: 0.00222, 0.00222, 0.0089, 0.021875 with the filter order of 209, 150, 37 and 13. Table 6.28 showed the filter characteristics of complementary masking filter using HGDFT channelisation algorithm. The frequency characteristic input is shown in Table 6.27.

Table 6.27. The frequency characteristics of masking filters implemented using HFarrow filter bank.

Filter Bank	d^k	stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples (δ_{mp})	stopband attenuation (δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.022625	0.1	-50	132
Blue-tooth,H_{ma}	$\frac{39}{40}$	0.025	0.0224	0.0975	-39	107
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.089	0.09	-39	24
WCDMA,H_{ma}	$\frac{7}{8}$	0.2	0.125	0.0875	-48.25	9

Figure 6.14 through to Figure 6.16 showed the magnitude response of the modal filter, Blue-tooth, Zigbee and WCDMA respectively. The modal filter is a lowpass filter with passband attenuation of 0.1dB, stopband attenuation of -50dB, and the filter length of 132 as shown in Figure 6.14. Figure 6.13 shows the bandpasss filter for BT with passband ripple of 0.09751, stopband attenuation of -39 and filter order of 107. Figure 6.15 represents the bandpass filter for Zigbee with passband ripple of 0.09, stopband attenuation of -39 and filter order of 24 and Figure 6.16 represents the bandpass filter for WCDMA with passband ripple of 0.0875, stopband attenuation of -48.125 and filter order of 9. The number of multipliers utilised by the HFarrow filter bank were analysed, compared and found to be lower than CDFB [66] and ICDM [30] methods as indicated in Table 6.29 and Table 6.30.

Table 6.28. The frequency characteristics of complementary masking filter implemented using HFarrow filter bank.

Filter Bank	d^{kc}	stopband frequency(ω_{mcs})	Passband frequency(ω_{mcp})	Passband ripples (δ_{mcp})	stopband attenuation(δ_{mcs})	Filter length
Modal filter	$\frac{8}{9}$	0.027307	0.02269	0.1	-50	147
Blue-tooth	$\frac{8}{9}$	0.027307	0.02269	0.092	-36.92	134
Zigbee	$\frac{8}{9}$	0.1080	0.0911	0.088	-35.5	29
WCDMA	$\frac{7}{8}$	0.2	0.125	0.0875	-48.25	9

Table 6.29. Multiplication complexity for non-uniform filter bank.

Filter Bank	Filter Order			Total number of multiplication
	H_a	H_{ma}	H_{mc}	
Modal filter	279	-	-	187
BT	-	107	134	156
Zigbee	-	24	29	37
WCDMA	-	9	9	9

From the Table 6.30, the total number of multipliers used by HFarrow filter bank were 389 while the ICDM expended 1545 multipliers, NU MDFFB consumed up to 1090, CDFB used up 1745 and FPCC consumed 980. Thus, the total number of multipliers utilised by HFarrow channelisation was 22% of the total number of multipliers in CDFB algorithm, while it depleted 25% of the total number of multipliers in ICDM. The percentage of multipliers used by HFarrow algorithm in comparison with NU MDFT FB was 37%. while HFarrow algorithm used 39% of FPCC FB. HFarrow showed decrement in the following: 78% in CDFB, 75% ICDM, 63% in NU MDFT and 61% of FPCC. There was a remarkable decrement in the number of multipliers used in HFarrow compared with the algorithms used for the comparison as shown in the literature, although there is trade of comolementary masking filter, H_{mc} in CDFT, ICDM FB and FPCC TB.

Table 6.30. Comparison of different multiplication complexities for non-uniform filter bank.

Filter Bank	Filter Order			Total number of multiplication
	H_a	H_{ma}	H_{mc}	
CDFB [66]	3089	400	-	1745
ICDM FB [30]	2929	160	-	1545
NU-MDFT FB[181]	187	430	469	1090
FPCC FB [1]	-	-		980
HFarrow filter Bank	187	100	102	389

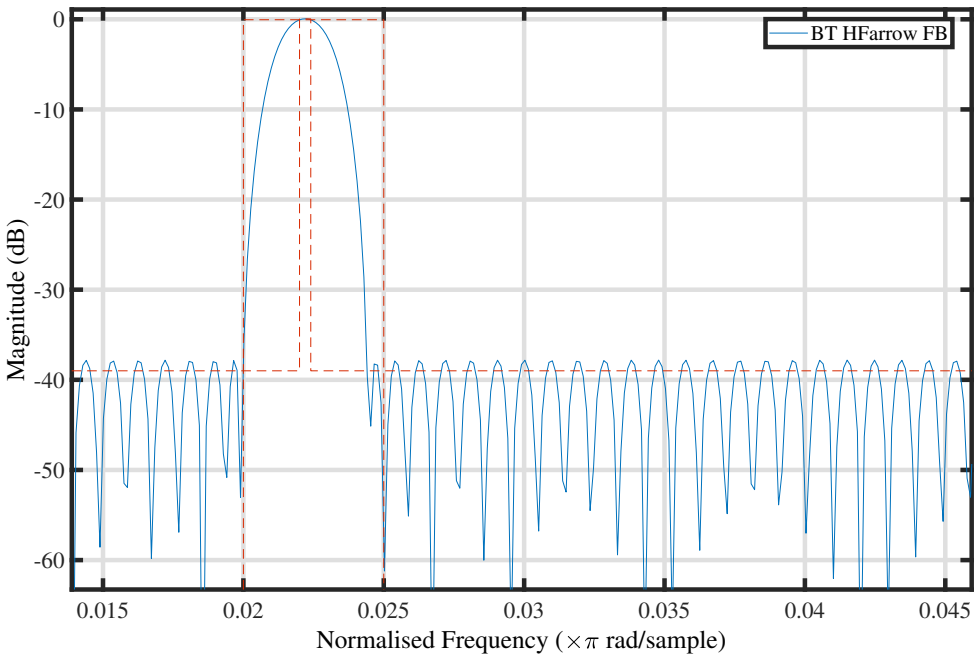

Figure 6.13. Magnitude response for the Bluetooth masking filter using HFarrow algorithm.

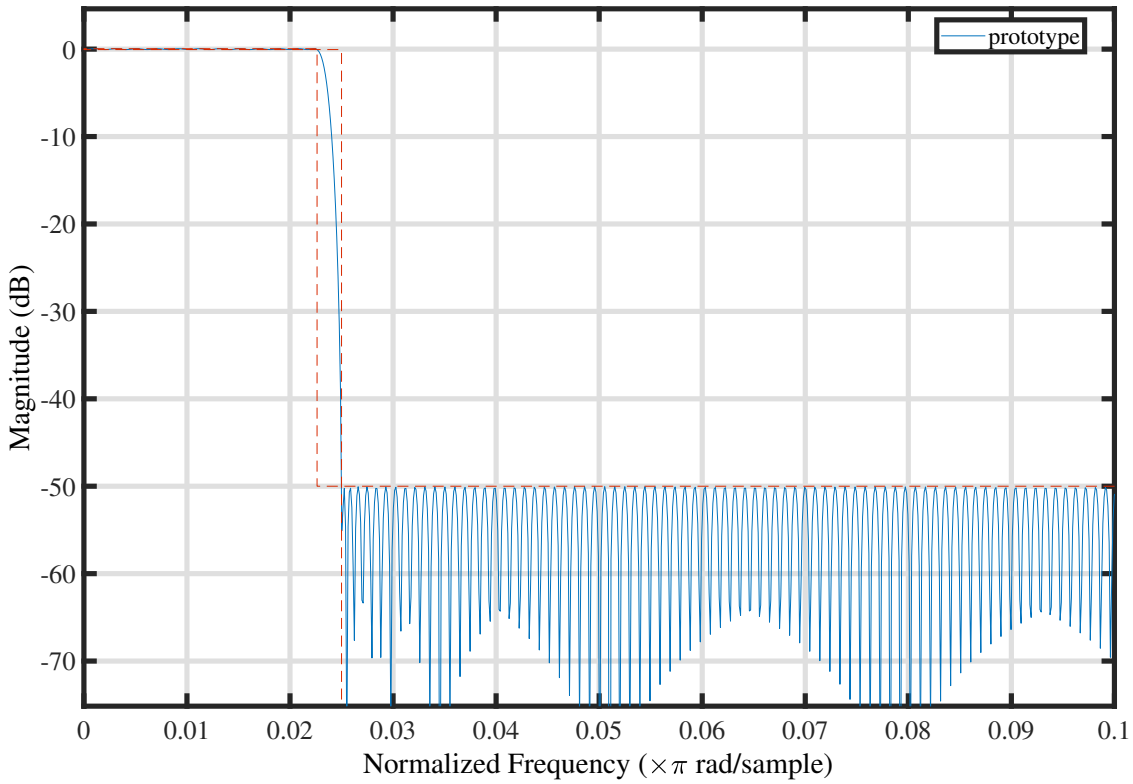

Figure 6.14. Magnitude response for the modal filter using HFarrow algorithm.

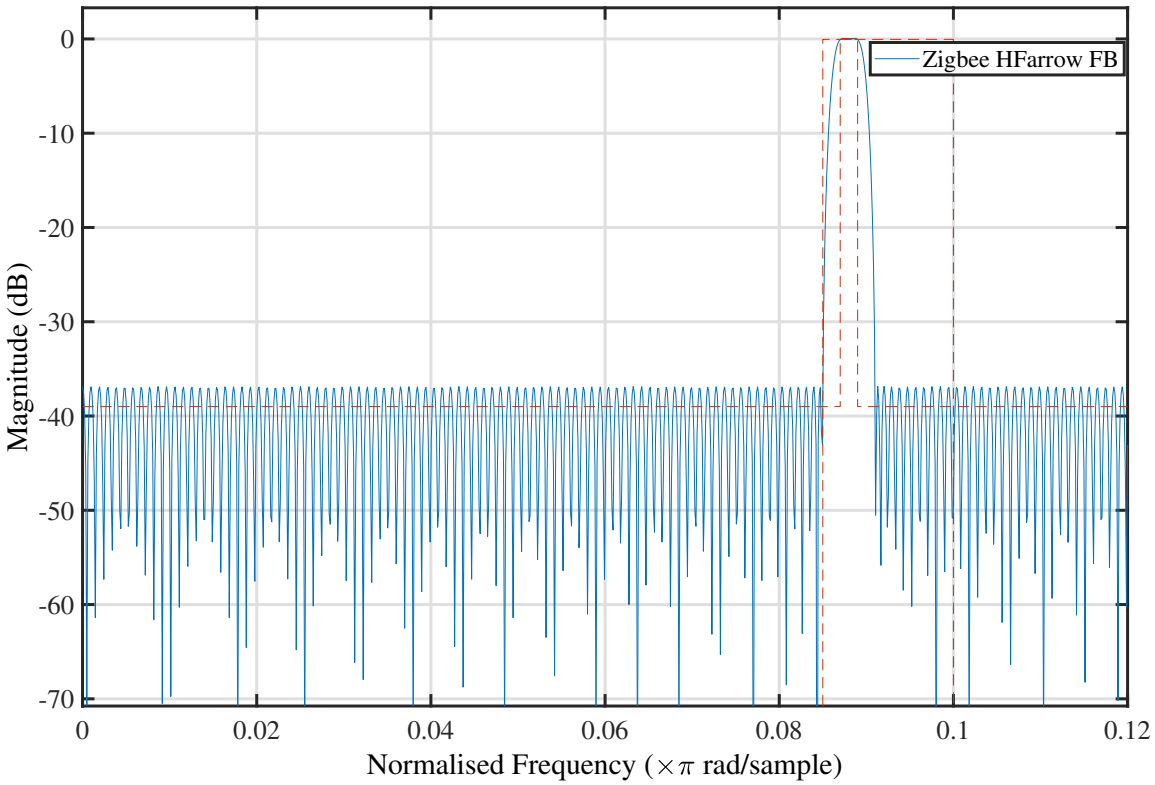

Figure 6.15. Magnitude response for the Zigbee masking filter using HFarrow algorithm.

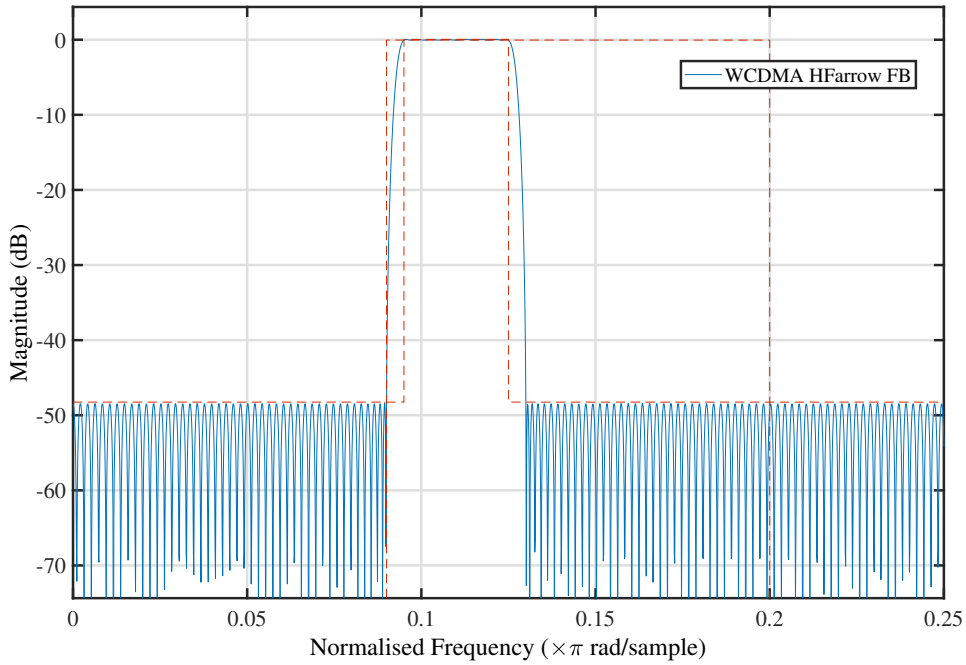

Figure 6.16. Magnitude response for the WCDMA masking filter using HFarrow algorithm.

6.3.1 Improvement of HFarrow Channelisation Algorithm with Parallel Distributed Arithmetic Based Residual Number System (HFarrow PDA-RNS)

The results obtained in Table 6.27 were further improved using PDARNS. Table 6.32 compared the filter specifications in terms of number of bits. Bits 8, 12 and 16 were used during the simulation for comparing the prototype and masking filters. The varied parameters were passband ripples, δ_p, stopband attenuation, δ_s, number of adders and amplitude distortion. It was observed from Table 6.32 that the 16-bits showed a closer margin to the continuous filter values of HFarrow FB. Application of HFarrow PDA-RNS FB to the 16-bits filter requirement using the procedure in Section 3.3.1, resulted in the filter specifications shown in Table 6.32. Table 6.33 and Table 6.34 presented the results obtained by improving HFarrow PDA-RNS filter.

Table 6.31. Comparison of filter specifications, in terms of number of bits.

Filter	Input Bits	Passband Ripples (dB)	Stopband attenuation (dB)	No of adders	Amplitude distortion
Prototype	12-bits	14	−60.911	455	0.030
filter	14-bits	0.083	−47.242	152	0.205
	16-bits	0.085	-48.146	130	0.19
Bluetooth	12-bits	0.0786	−38.214.	133	0.104
	14-bits	0.0785	−38.213	133	0.103
	16-bits	0.0779	−38.221	102	0.105
Zigbee	12-bits	0.0897	−38.103	36	0.104
	14-bits	0.0851	−38.154	34	0.1106
	16-bits	0.085	-38.124	25	0.115
WCDMA	12-bits	0.0891	−47.234	14	0.03
	14-bits	0.0906	−47,235	16	0.0348
	16-bits	0.0891	−47,228	7	0.031

Table 6.32. The frequency characteristics of 16- bits masking filters implemented using HFarrow PDARNS filter bank.

Filter Bank	$\frac{L}{M}$	Stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples(δ_{mp})	Stopband attenuation(δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.085	-48.146	130
Bluetooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.0779	-38.221	102
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0959	0.085	-38.124	25
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.0891	-47.228	7

From these performances, the plot in Figure 6.17 depicted that HFarrow with PDA-RNS utilised 18% of the total LUT, 11% of the slices LUT, 12.2% decrement in the flip-flops used, 8% power consumption and 25% delay in the execution time when compared with NU MDFT CSD with Pareto ABC [181].

The filter achieved a 88% decrement in number of occupied slices from 2406 to 1206 slices. There was 92% decrement in power consumption and 75% decrement in execution delay time. From Table 6.39, the total devices utilised were compared. Slice registers utilised by HFarrow with PDA-RNS filter bank were 2800 out of 419328. This showed a drastic decrement in slice registers when compared with NU MDFT FB in [181] which consumed 29,797 out of 301,440 and in SDR channelizer [182] which used 15,295 out of 58,880 and also in SDR [183] which used up 29,797 out of 301,440. Also, the total slice LUTs used by HFarrow algorithm with PDA-RNS filter were found to be 2799 out of 209664, while the hardware utilised by NU MDFT filter in [181] was observed to be 5901 out of 298,600 and the device consumption rate in [183] was discovered to be 21,169 out of 150,720 and [182] used up 14,726 out of 58,880. Thus, Table 6.38 proved that the hardware resources utilisation for the implementation of HFarrow with PDA-RNS filter bank were found lesser than that in SDR channelizer [181, 182, 183].The lower filter order in this design, coupled with the modularity in RNS, clearly contributed to lower slice requirement, lower power consumption when compared to design in [120, 181].

Table 6.33. The three RNS values from the HFarrow PDA-RNS filter coefficient.

Filter Coefficient	Double precision values	Binary values	Integer values	R_{31}	R_{32}	R_{33}
b0=b29	0.01441862490369894	0000000111011001	473	8	25	11
b1=b28	0.023928737699304842	0000001100010000	784	9	16	25
b2=b27	0.025455867630145794	0000001101000010	834	28	2	25
b3=b26	0.003220390664297086	0000000001101010	106	13	10	7
b4=b25	-0.035174037047374741	1111101101111111	-1153	-6	-1	-31
b5=b24	-0.072204929013503033	1111011011000010	-2366	10	-30	-23
b6=b23	-0.081482771549880470	1111010110010010	-2670	-4	-14	-30
b7=b22	-0.041374486392402265	1111101010110100	-1356	-23	-12	-3
b8=b21	0.031642040537188047	0000010000001101	1037	14	13	14
b9=b20	0.109687967473366120	0000111000001010	3594	29	10	30
b10=b19	0.128854319531854060	0001000001111110	4222	6	30	31
b11=b18	0.083665355981792339	0000101010110101	2741	13	21	2
b12=b17	-0.026818540295058590	1111110010010001	-1391	-27	-15	-5
b13=b16	-0.121692802585422390	1111000001101100	-3988	-20	-20	-28
b14=b15	-0.082869641432214508	1111010101100101	-2843	-22	-27	-5

Table 6.34. The partial products in binary for HFarrow PDA-RNS.

Filter Coefficient	R_{31}	R_{32}	R_{33}
b0=b29	0000000000001000(8)	0000000000011001(25)	0000000000001011(11)
b1=b28	0000000000001001(9)	0000000000010000(16)	0000000000011001(25)
b2=b27	0000000000011100(28)	0000000000000010(2)	0000000000001001(25)
b3=b26	0000000000001101(13)	0000000000001010(10)	0000000000000111(7)
b4=b25	1111111111111010(-6)	1111111111111111(-1)	1111111111100001(-31)
b5=b24	1111111111110110(10)	1111111111100010(-30)	1111111111101001(-23)
b6=b23	1111111111111100(-4)	1111111111110010(-14)	1111111111100010(-30)
b7=b22	1111111111101001(-23)	1111111111110100(-12)	1111111111111101(-3)
b8=b21	0000000000001110(14)	0000000000001101(13)	0000000000001110(14)
b9=b20	0000000000011101(29)	0000000000001010(10)	0000000000011110(30)
b10=b19	0000000000000110(6)	0000000000011110(30)	0000000000011111(31)
b11=b18	0000000000001101(13)	0000000000010101(21)	0000000000000010(2)
b12=b17	1111111111100101(-27)	1111111111110001(-15)	1111111111111011(-5)
b13=b16	1111111111101100(-20)	1111111111101100(-20)	1111111111100100(-28)
b14=b15	1111111111101010(-22)	1111111111100101(-27)	1111111111111011(-5)

6.3.1.1 Optimising the Performance of HFarrow PDA-RNS with Genetic Algorithm (HFarrow PDARNS-GA)

Optimising the frequency specifications of HFarrow PDARNS filter with genetic algorithms, the following Table 6.35 was generated. The corresponding RNS values and its binary representations are shown in Table 6.36 and Table 6.37. Following the procedure explained in Section 6.3.1, and implementing HFarrow filter coefficients with PDA-RNS filter, there was 100% decrease in the multipliers utilised. The multipliers reduced from 526 to 0, in which case the use of multipliers was totally eliminated. Table 6.38 and Table 6.39 showed silicon area complexity for the HFarrow with PDA-RNS filter.

The total hardware resources occupied by HFarrow with PDA-RNS were presented as follows: 1206 total slices, 2073 slices of LUTs, 2338 flip-flops, with total power of 333.53 mW and total delay of 3.328ns. The total slices occupied by NU-MDFT CSD with Pareto ABC were 2406, slice LUTs utilised were 8950, flip-flops consumed were 8980, total power consumed was 1751 with total delay of 3.75ns. The performance of NU-MDFT filter optimized with SID-CSE showed that the total slices consumed were 1633, slice LUTs utilisation were 5901

with flip-flops of 5911 and total power of 1281 while the delay time was 2.6ns.

Table 6.35. The frequency characteristics of 16- bits masking filters implemented using HFarrow PDARNS-GA filter bank.

Filter Bank	$\frac{L}{M}$	Stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples (δ_{mp})	Stopband attenuation(δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.0998	-49.146	130
Bluetooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.0974876	-38.2314	101
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0899876	0.085	-38.915	22
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.0874	-47.223	6

‘

Table 6.38 and Table 6.39 showed device utilisation comparison. The total hardware resources occupied by HFarrow with PDARNS-GA were presented as follows: 1201 total slices, 2072 slices of LUTs, 2330 flip-flops, with total power of 332 mW and total delay of 3.31 ns.

Analysis and evaluation of these performances, as depicted in Figure 6.17 showed that improving HFarrow with PDA-RNSGA utilised 18% of the total LUT, 10% of the slices LUT, 11% in the flip-flops used, 8.3% power consumption and 17% delay in the execution time when compared with NU-MDFT CSD with Pareto ABC [181] and SID-CSE [181].

The filter achieved a 90% decrement in the number of occupied slices from 2406 to 1201 slices. There was 91.7% decrement in power consumption and 83% decrement in execution delay time. Figure 6.18, Figure 6.19 and Figure 6.20 showed the magnitude responses of BT, Zigbee and WCDMA masking filter. Figure 6.18 shows the bandpasss filter for BT with passband ripple of 0.00974876, stopband attenuation of -38.2314 and filter order of 101. Figure 6.19 represents the bandpass filter for Zigbee with passband ripple of 0.085, stopband attenuation of -38.915 and filter order of 22 and Figure 6.20 represents the bandpass filter for WCDMA with passband ripple of 0.0874, stopband attenuation of -47.223 and filter order of 6.

Table 6.36. The three RNS values from the filter coefficient of HFarrow PDARNS-GA.

Filter Coefficient	Double Precision values	Binary values	Integer values	R_{31}	R_{32}	R_{33}
b0=b29	0.014070901481142770	0000000111001101	461	27	13	32
b1=b28	0.024284784907422027	0000001100011100	796	21	28	4
b2=b27	0.025411988546488323	0000001101000001	833	27	1	8
b3=b26	0.004184957106717582	0000000010001001	137	13	9	5
b4=b25	-0.035688346150032906	1111101101101111	-1169	-9	-15	-19
b6=b23	-0.083206120226861024	1111010101011010	-2726	-2	-26	-13
b7=b22	-0.041002650704817206	1111101011000000	-1344	-20	0	-9
b8=b21	0.030419600003954141	0000001111100101	997	5	5	7
b9=b20	0.111713827678263960	0000111001001101	3661	3	13	31
b10=b19	0.128680005559823810	0000111001001101	4216	0	24	25
b11=b18	0.086238745715614962	0000101100001010	2826	5	10	31
b12=b17	-0.028272776806474137	1111110001100010	-926	-4	-2	-31
b13=b16	-0.121142675770288870	1111000001111111	-3969	-30	-31	-24
b14=b15	-0.086126817088436447	1111010011111010	-926	-4	-2	-31

Table 6.37. The partial products in binary for the HFarrow PDARNS-GA.

Filter Coeff.	R_{31}	R_{32}	R_{33}
b0=b29	000000000001 1011	0000000000001101	0000000000100000
b1=b28	0000000000010101	000000000001 1100	0000000000000100
b2=b27	0000000000011011	0000000000000001	0000000000001000
b3=b26	0000000000001101	0000000000001001	0000000000000101
b4=b25	1111111111110111	1111 1111 1111 0001	1111111111101101
b5=b24	1111111111110110	1111111111100010	1111111111101001
b6=b23	1111 1111 1111 1110	111111111110 0110	1111 1111 1111 0011
b7=b22	1111 1111 1110 1100	0000000000000000	1111111111110111
b8=b21	0000000000000101	0000000000000101	0000 0000 0000 0111
b9=b20	0000000000000011	0000000000001101	0000000000011111
b10=b19	0000000000000000	0000000000011000	0000000000011001
b11=b18	0000000000000101	0000000000001010	0000000000011111
b12=b17	1111 1111 1111 1100	1111 1111 1111 1110	1111 1111 1110 0001
b13=b16	1111 1111 1110 0010	1111 1111 1110 0001	1111 1111 1110 1000
b14=b15	1111 1111 1111 1100	1111 1111 1111 1110	1111 1111 1110 0001

Table 6.38. Device and power utilisation.

Parameter [181]	Continuous Coefficients	HFarrow PDARNS	HFarrow PDARNS-GA	CSD	SID-CSE ABC[181]
Total slices	2507	1206	1201	2406	1633
Slice LUTs	8694	2073	2072	8950	5901
Flip- flops	10313	2338	2330	8980	5911
Total Power (mW)	1865	333.53	332	1751	1281
Total Delay (ns)	45.125	3.328	3.31	3.75	2.6

Table 6.39. Comparison of device utilisation.

Parameter	SDR in [182]	SDR channelizer in [183]	SDR channelizer using NU MDFT FB[181]	HFarrow PDA-RNS FB	HFarrow PDARNS -GA
Slice Registers	15,295 out of 58,880	29,797 out of 301,440	5880 out of 597,200	2800 out of 419328	2789 out of 41938
Slice LUTs	14,726 out of 58,880	21,169 out of 150,720	5901 out of 298,600	2799 out of 209664	2793 out of 209664

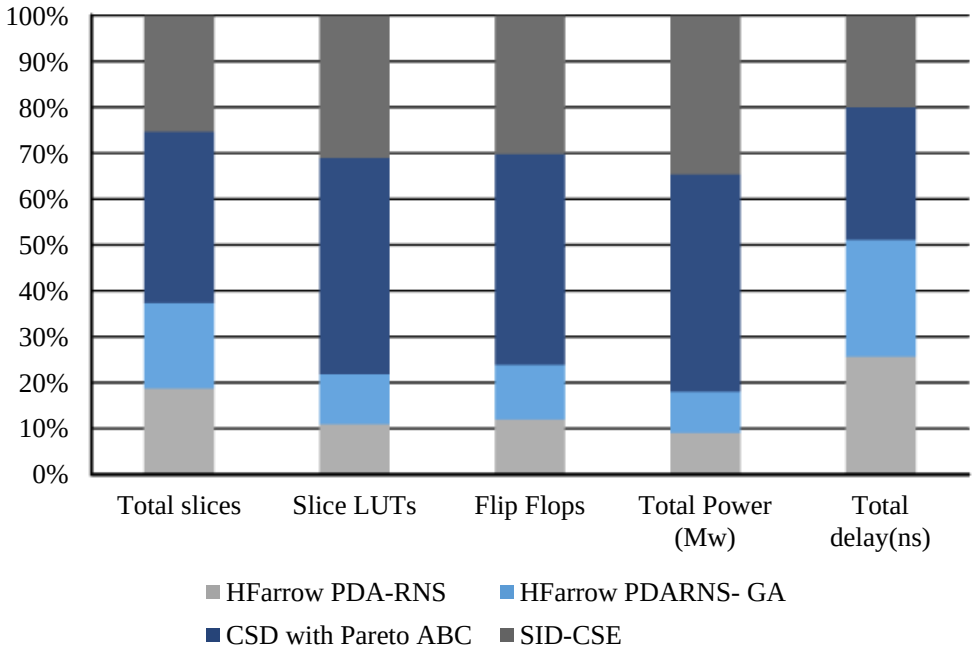

Figure 6.17. Plot of resources utilisation for HFarrow PDA-RNS filter.

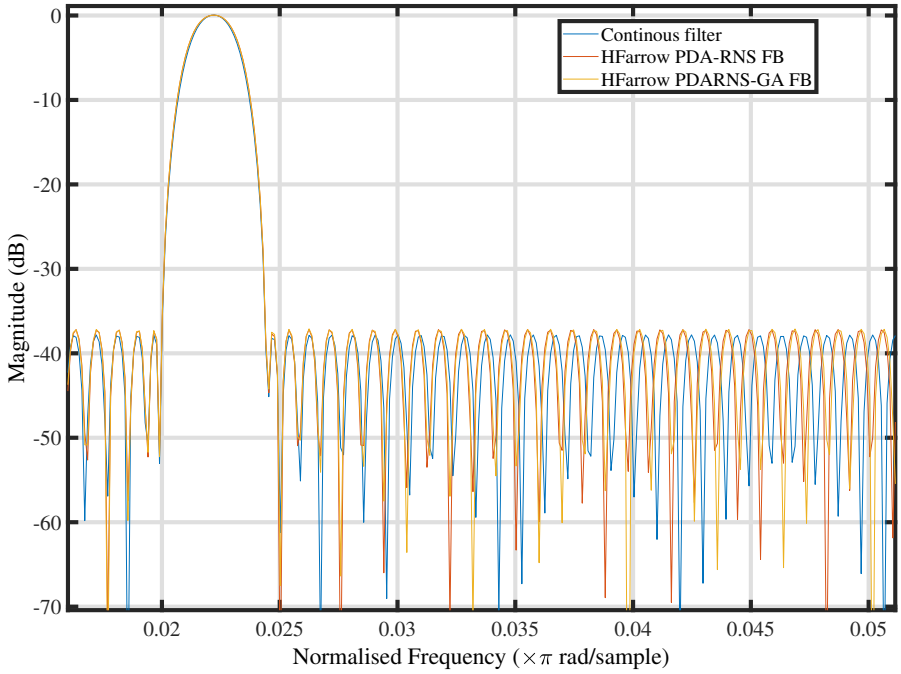

Figure 6.18. Plot of Bluetooth frequency response of HFarrow PDARNS-GA masking filter.

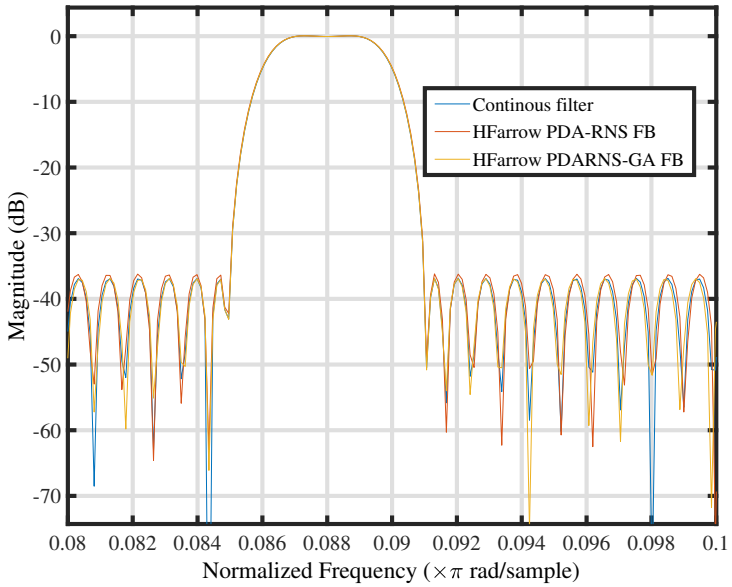

Figure 6.19. Plot of Zigbee frequency response using HFarrow PDARNS-GA masking filter.

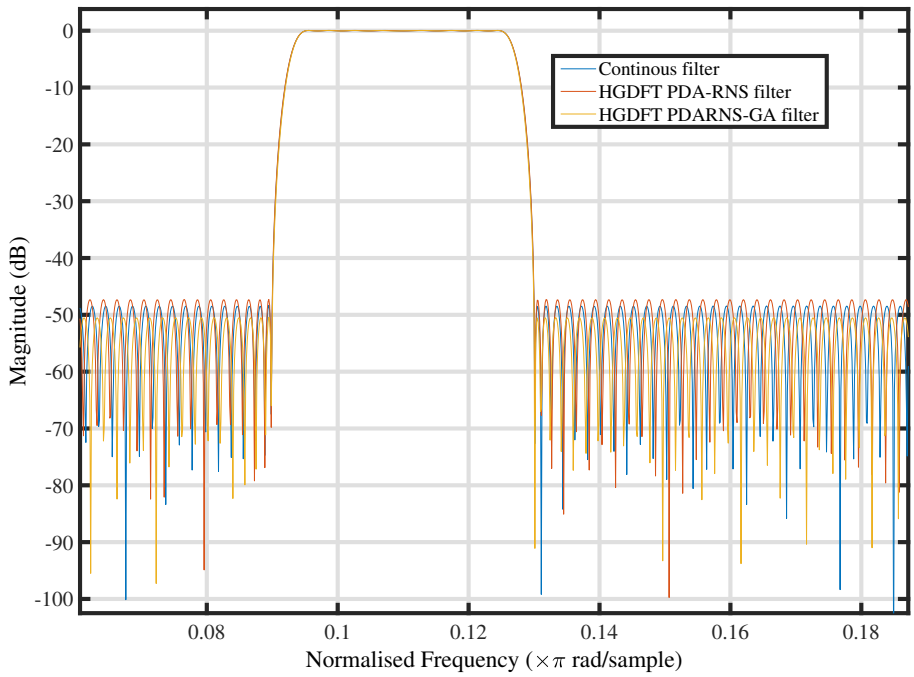

Figure 6.20. Plot of WCDMA frequency response using HFarrow PDARNS-GA masking filter.

6.3.2 Improvement of HFarrow Channelisation Algorithm with Parallel Canonical Signed Residual Number System (PDA-CSRSN)

Improving the performances of Table 6.27 using parallel canonical signed residual number system (PDA-CSRNS) results in filter specifications shown in Table 6.40. Table 6.41 showed the frequency specifications and hardware complexities for different SPT terms. Average of five SPT terms were used during the simulation. When compared with what was obtained from the Table 6.30, the HFarrow filter utilised 30% of the total number of multipliers used in CDFB filter; 34% of the entire mmultipliers used by ICDM and it uses 50% of the multipliers NU MDFT filter. From Table 6.42, the number of multipliers used by HFarrow PDA-CSRNS have been totally eliminated. Therefore, the total number of adders used during the implementation was 325.

Table 6.40. Comparison of filter specifications, in terms of number of bits.

Filter	Input Bits	Passband Ripples (dB)	Stopband Attenuation (dB)	No of Adders	Amplitude Distortion
Prototype	12-bits	3.360	−11.204	404	0.20
filter	14-bits	0.083	−47.242	165	0.205
	16-bits	0.085	-48.146	121	0.19
Bluetooth	12-bits	0.0905	38.233	132	0.112
	14-bits	0.0902	−38.231	134	0.106
	16-bits	0.0901	−38.2315	100	0.119
Zigbee	12-bits	0.0739	−38.261	24	0.111
	14-bits	0.0851	−38.9149	23	0.1106
	16-bits	0.085	38.915	22	0.115
WCDMA	12-bits	0.0891	−47.234	7	0.03
	14-bits	0.089	−47,235	7	0.0348
	16-bits	0.0897	−47,228	6	0.031

Table 6.41. Filter frequency specifications and hardware complexity of HFarrow PDACSRNS FB.

Filter	SPT Terms used	Passband Ripples (dB)	Stopband Attenuation	Total number of Adders	Amplitude distortion
Prototype filter	7 SPTs	0.085	−48.146	159	0.19
	6 SPTs	0.086	−48.1433	157	0.19
	5 SPTs	0.087	-48.138	145	0.19
	4 SPTs	0.089	-48.138	162	0.187
Bluetooth	7 SPTs	0.0901	−38.231	138	0.119
	6 SPTs	0.09112	−38.231	119	0.119
	5 SPTs	0.0923	−38.435	132	0.119
	4 SPTs	0.09235	−34.5	144	0.119
Zigbee	7 SPTs	0.085	−38.915	30	0.115
	6 SPTs	0.0848	−36.123	35	0.115
	5 SPTs	0.0841	−38.732	34	0.117
	4 SPTs	0.0831	−38.65	31	0.117
WCDMA	7 SPTs	0.0897	−47.228	14	0.031
	6 SPTs	0.0896	−47.226	14	0.031
	5 SPTs	0.0894	−47.342	14	0.031
	4 SPTs	0.08942	−45.24	13	0.031

Table 6.42. The frequency characteristics of 16- bits masking filters implemented using HFarrow PDACSRNS filter bank.

Filter Bank	$\frac{L}{M}$	Stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples(δ_{mp})	Stopband attenuation(δ_{ms})	Filter length (Number of Adders)
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.087	-48.138	145
Bluetooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.0923	-38.435	132
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0959	0.0841	-38.732	34
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.0894	-47.342	14

Table 6.43. The partial products in binary for the HFarrow PDACSRNS.

Filter Coeff.	R_{31}	R_{32}	R_{33}
b0=b29	0000000000001000(8)	000000000010$\bar{1}$001(25)	000000000000010$\bar{1}$0$\bar{1}$(11)
b1=b28	0000000000001001(9)	0000000000010000(16)	000000000010$\bar{1}$001(25)
b2=b27	000000000100$\bar{1}$00(28)	0000000000000010(2)	0000000000001001(25)
b3=b26	000000000010$\bar{1}$01(13)	0000000000001010(10)	00000000000100$\bar{1}$(7)
b4=b25	1111111111111010(-6)	1111111111111111(-1)	1111111111100001(-31)
b5=b24	111111111111$\bar{1}$0$\bar{1}$0(10)	1111111111100010(-30)	1111111111101001(-23)
b6=b23	1111111111111100(-4)	1111111111110010(-14)	1111111111100010(-30)
b7=b22	1111111111101001(-23)	1111111111110100(-12)	1111111111111101(-3)
b8=b21	000000000100$\bar{1}$0(14)	000000000010$\bar{1}$01(13)	000000000100$\bar{1}$0(14)
b9=b20	000000000100$\bar{1}$01(29)	0000000000001010(10)	000000000100$\bar{1}$0(30)
b10=b19	00000000000010$\bar{1}$0(6)	000000000100$\bar{1}$0(30)	0000 0000 0010 000$\bar{1}$(31)
b11=b18	000000000010$\bar{1}$01(13)	0000000000010101(21)	0000000000000010(2)
b12=b17	1111111111100101(-27)	1111111111110001(-15)	111111111111110$\bar{1}$(-5)
b13=b16	111111111111$\bar{1}$00(-20)	111111111111$\bar{1}$00(-20)	1111111111100100(-28)
b14=b15	1111111111101010(-22)	1111111111100101(-27)	111111111111110$\bar{1}$(-5)

6.3.2.1 Optimisation of HFarrow PDA-CSRNS with Genetic Algorithm (HFarrow PDACSRNS-GA)

Table 6.44. Comparison of HFarrow PDACSRNS-GA filter specifications, in terms of number of bits.

Filter	Input Bits	Passband Ripples (dB)	Stopband Attenuation	No of Adders	Amplitude Distortion
Prototype filter	12-bits	3.360	−11.204	404	0.20
	14-bits	0.083	−47.242	152	0.205
	16-bits	0.085	-48.146	156	0.19
Blue-tooth	12-bits	0.0767	38.233	130	0.112
	14-bits	0.0787	−38.231	133	0.106
	16-bits	0.0787	−38.2315	133	0.119
Zigbee	12-bits	0.0739	−38.261	30	0.111
	14-bits	0.0851	−38.9149	35	0.1106
	16-bits	0.085	38.915	34	0.115
WCDMA	12-bits	0.0891	−47.234	14	0.03
	14-bits	0.089	−47,235	14	0.0348
	16-bits	0.0897	−47,228	13	0.031

Table 6.45. The frequency characteristics of 16- bits masking filters implemented using HFarrow PDACSRNS-GA filter bank.

Filter Bank	$\frac{L}{M}$	stopband frequency (ω_{ms})	Passband frequency(ω_{mp})	Passband ripples (δ_{mp})	stopband attenuation(δ_{ms})	Filter length
Modal filter, H_a	$\frac{39}{40}$	0.025	0.02405	0.0998	-49.9999941	179
Blue-tooth,H_{ma}	$\frac{39}{40}$	0.025	0.02405	0.0974892	-38.2315	159
Zigbee, H_{ma}	$\frac{9}{10}$	0.1	0.0959	0.0899892	-38.915	36
WCDMA,H_{ma}	$\frac{7}{8}$	0.125	0.1	0.0889592	-47.228	14

Table 6.46. The three RNS values from the filter coefficient of HFarrow PDA CSRNS-GA.

Filter Coeff.	Double Precision values	Binary values	Integer values	R_{31}	R_{32}	R_{33}
b0=b29	0.014071000751575113	0000000111001101	461	27	13	32
b1=b28	0.024285099060693826	0000001100011100	796	21	28	4
b2=b27	0.025412233654139922	0000001101000001	833	27	1	8
b3=b26	0.004185091179568477	0000000010001001	137	13	9	5
b4=b25	-0.03568829729818995	1111101101101111	-1169	-9	-15	-19
b5=b24	-0.0371915687490131136	1111101101101111	-2356	0	-12	-120
b6=b23	-0.083207125257840270	1111010101011010	-2726	-2	-26	-13
b7=b22	-0.041002941417636510	1111101011000000	-1344	-20	0	-9
b8=b21	0.030419712982115037	0000001111100101	997	5	5	7
b9=b20	0.111715176666545730	0000111001001101	3661	3	13	31
b10=b19	0.128681250470345290	0000111001001101	4216	0	24	25
b11=b18	0.086239997640985028	0000101100001010	2826	5	10	31
b12=b17	-0.028273354723455074	1111110001100010	-926	-4	-2	-31
b13=b16	-0.121144093925718830	1111000001111110	-3770	-12	-6	-25
b14=b15	-0.086127412946896853	1111010011111010	-926	-4	-2	-31

From Table 6.48, the following were the devices and power utilisation of HFarrow PDA-CSRNS: total slices of 1192; slice LUTs of 2014; flip-flops of 2286; power consumption of 333.56mW and total delay of 3.33ns. The device and power utilisation of CSE with Pareto ABC were as follows: 2406 total slices; 8950 slice LUTs; 8980 flip-flops; 1751mW total power and 3.75ns total delay and SID-CSE utilised the following hardware resources: 1633 of total slices; 5901 of slice LUTs; 5911 of flip-flops; 1281 of total power and 2.6ns of total delay. There was a remarkable cut down in the number of resources utilised in HFarrow compared with other algorithms.

Also, from Figure 6.21, the number of slices utilised by HFarrow algorithm was 17% of the whole slices; 10.1% of the slice LUTs was consumed during the channelisation operation. The power consumption was 8.7% while the total operational delay was 25.5%.

Thus, there was considerable decrement in hardware resources of 83% in the total slices, 89.1% in the slice

LUTs; 91.3% in the power and 74.5% decrement in execution time. There was a decrement in resources utilised with HFarrow PDA-CSRNS, compared to when HFarrow PDARNS was used. Table 6.48 and Table 6.49 showed device utilisation comparison. The total hardware resources occupied by HFarrow PDACSRNS-GA were presented as follows: 1190 total slices, 2016 slices of LUT, 2265 flip-flops, with total power of 333.24 mW and total delay of 3.33 ns. Figure 6.22, Figure 6.23 and Figure 6.24 showed the magnitude responses of BT, WCDMA and Zigbee masking filter using HFarrow PDA CSRNS-GA algorithm. Figure 6.22 shows the bandpasss filter for BT with passband ripple of 0.00974876, stopband attenuation of -38.2314 and filter order of 159. Figure 6.24 represents the bandpass filter for Zigbee with passband ripple of 0.0899892, stopband attenuation of -38.915 and filter order of 36 and Figure 6.23 represents the bandpass filter for WCDMA with passband ripple of 0.0889592, stopband attenuation of -47.228 and filter order of 14.

Table 6.47. The partial products in binary for the HFarrow PDACSRNS-GA.

Filter Coeff.	R_{31}	R_{32}	R_{33}
b0=b29	0000000000001 1011	000000000001$0\bar{1}$01	0000000000100000
b1=b28	0000000000010101	0000000000001 1100	0000000000000100
b2=b27	000000000010$0\bar{1}0\bar{1}$	0000000000000001	0000000000001000
b3=b26	000000000010$0\bar{1}$01	0000000000001001	000000000000010
b4=b25	111111111111$00\bar{1}$	1111 1111 1111 0001	111111111110$\bar{1}$01
b5=b24	111111111111$0\bar{1}0$	1111111111100010	1111111111101001
b6=b23	1111 1111 1111 1110	111111111110 $10\bar{1}0$	1111 1111 1111 0011
b7=b22	1111 1111 1111$0\bar{1}00$	0000000000000000(0)	111111111111$00\bar{1}$
b8=b21	0000000000000101	0000000000000101	0000 0000 0000 $100\bar{1}$
b9=b20	000000000000$0\bar{1}$	000000000001$0\bar{1}$01	000000000010000$\bar{1}$
b10=b19	0000000000000000	000000000010$\bar{1}$000	000000000010$\bar{1}$001
b11=b18	0000000000000101	0000000000001010	000000000010000$\bar{1}$
b12=b17	1111 1111 1111 1100	1111 1111 1111 1110	1111 1111 1110 0001
b13=b16	1111 1111 1111 0100	1111 1111 1111 1010	1111 1111 1110 $100\bar{1}$
b14=b15	1111 1111 1111 1100	1111 1111 1111 1110	1111 1111 1110 0001

Analysis and evaluation of these performances, as depicted in Figure 6.21, showed that improving HFarrow PDA-CSRNSGA utilised 16% of the total LUT, 10.5% of the slices LUT, 10% of the flip-flops used, 8.3% power consumption and 24.5% delay in the execution time when compared with HFarrow with PDA-RNS, NU-MDFT CSD with Pareto ABC [181] and SID-CSE [181].

The HFarrow PDACSRNS-GA filter achieved an 83% decrement in the number of occupied slices from 2406 to 1190 slices. There was 91.7% decrement in power consumption and 75.5% decrement in execution delay time. This is attributed to the shrinkages in the available number of one's required for computation.

Table 6.48. Device and power utilisation.

Parameter	Continuous Coefficients	HFarrow with PDA-CSRNS	HFarrow PDA CSRNS -GA	CSD with Pareto ABC [181]	SID-CSE [181]
Total slices	2507	1192	1190	2406	1633
Slice LUT	8694	2014	2016	8950	5901
Flip- flops	10313	2286	2265	8980	5911
Total Power (mW)	1865	333.56	333.24	1751	1281
Total Delay (ns)	45.125	3.33	3.33	3.75	2.6

Table 6.49. Comparison of device utilisation.

Parameter	SDR in [182]	SDR Channelizer in [183]	SDR Channelizer using NU MDFT FB	HFarrow PDA-CSRNS FB	HFarrow PDACSRNS -GA
Slice Register	15,295 out of 58,880	29,797 out of 301,440	5880 out of 597,200	2254 out of 239616	2255 out of 239616
Slice LUT	14,726 out of 58,880	21,169 out of 150,720	5901 out of 298,600	1976 out of 119808	1972 out of 119808

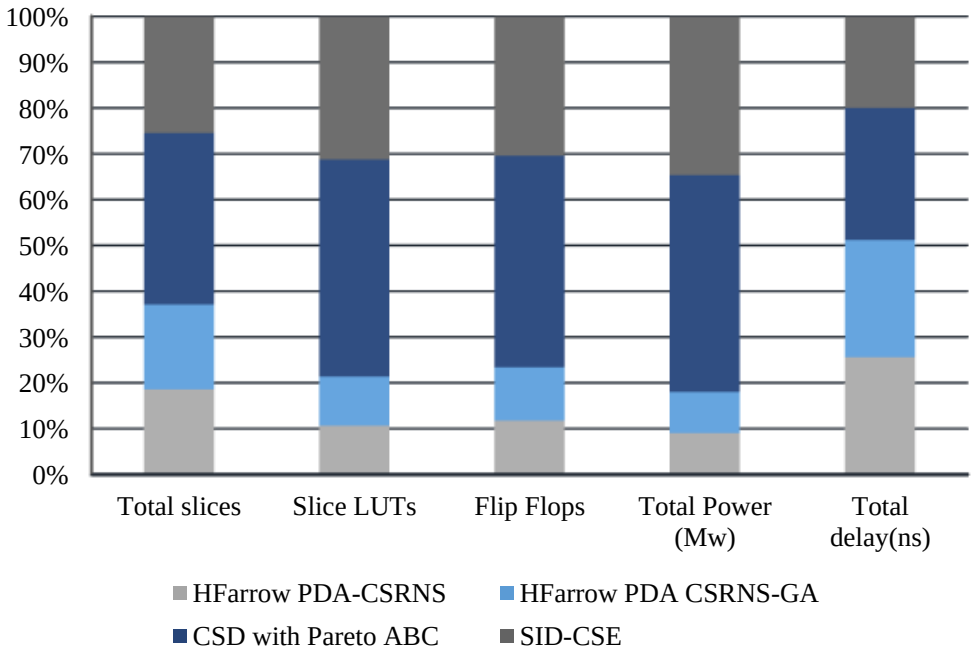

Figure 6.21. Plot of resources utilisation for HFarrow PDA-CSRNS Channelisation algorithm.

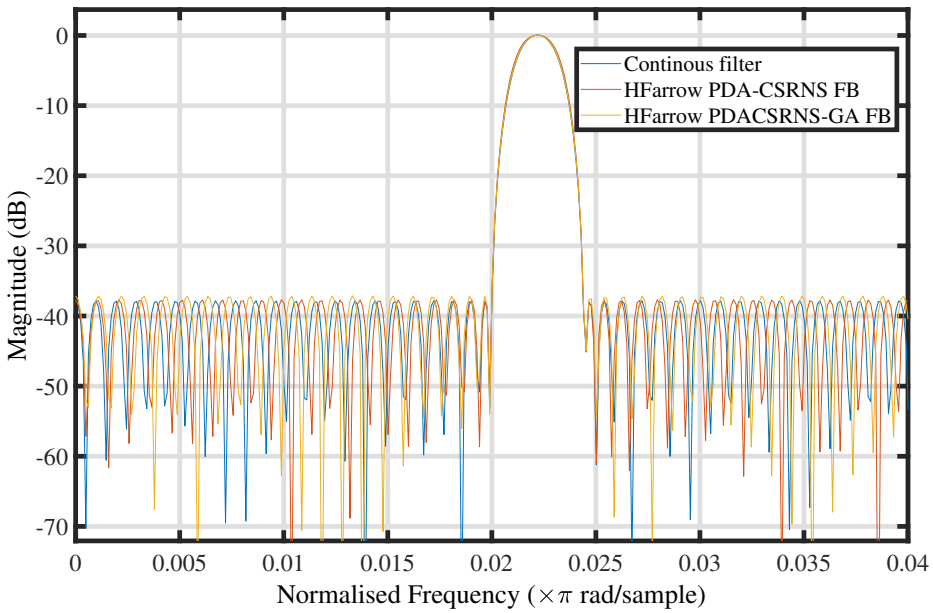

Figure 6.22. Plot of BT frequency response using HFarrow PDACSRNS-GA masking filter.

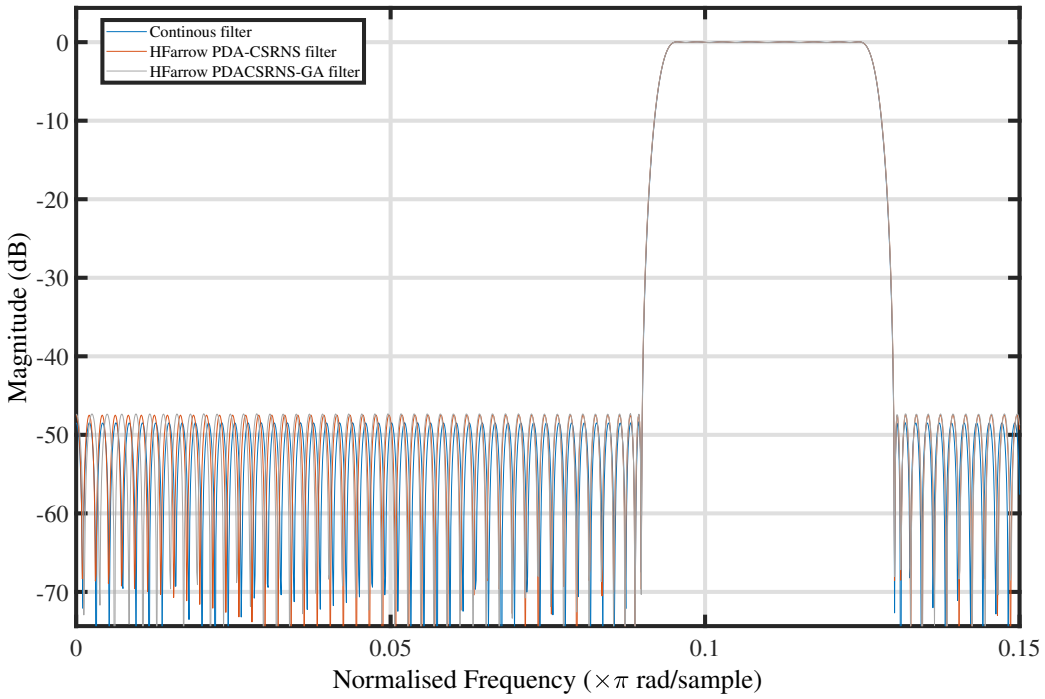

Figure 6.23. Plot of WCDMA frequency response using HFarrow PDACSRNS-GA masking filter.

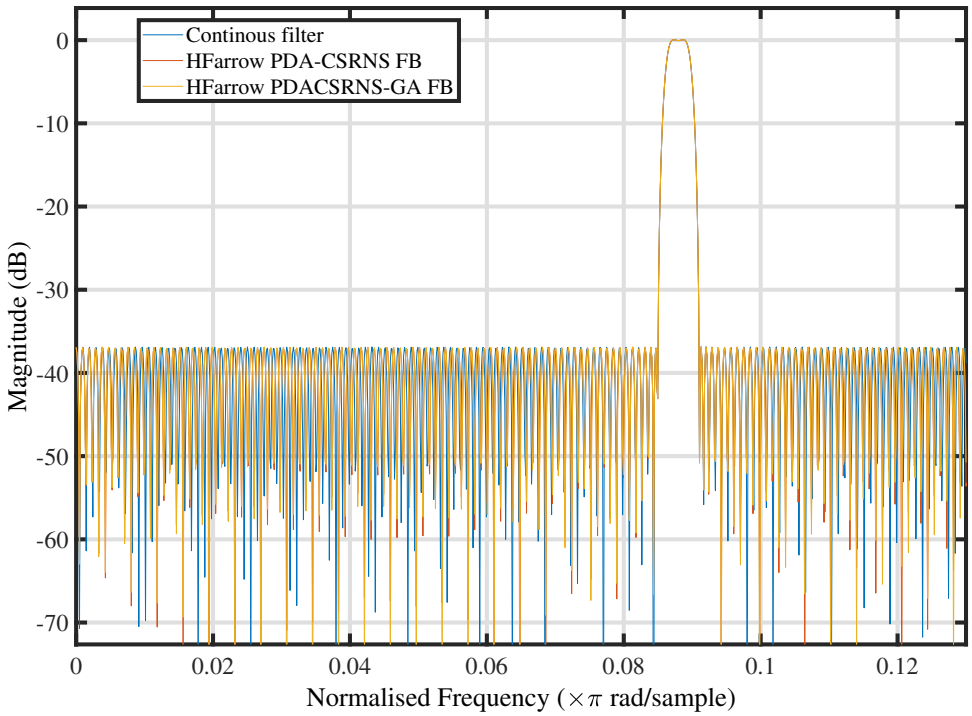

Figure 6.24. Plot of Zigbee frequency response using HFarrow PDACSRNS-GA masking filter.

6.3.3 Improvement of HFarrow Channelisation Algorithm with Parallel Distributed Arithmetic Based Common Sub-Expression Residual Number System (HFarrow PDA-CSERNS)

When parallel distributed arithmetic based common sub-expressions residual filter (PDA-CSERNS) was applied to improve the performance of HFarrow filter, the following results were obtained as depicted in Table 6.50. Hardware resources occupied by HFarow with PDA-CSERNS were as follows: 923 total slices, 1692 slice LUTs, 1725 flip-flops, 333.51mW power and 3.55ns total delay. The CSD with pareto ABC used the following device and power: 2406 total slices, 8950 slice LUTs, 8980 flip- flops, 1751mW power and 3.75 ns total delay. Also, the resources utilisation of SID-CSE were: 1633 total slices, 5901 slice LUTs, 5911 flip-flops, 1281mW power and 2.6ns delay.

Figure 6.25 shows that HFarrow with PDA-CSERNS utilised 84% of the total slices, 80.1% in slice LUTs, 91.4% in power and 75% in delay time. Table 6.50 and Table 6.51 showed device utilisation comparison. The total hardware resources occupied by HFarrow with PDACSERNS-GA were presented as follows: 920 total slices, 1689 slices of LUTs, 1729 flip-flops, with total power of 333.51 mW and total delay of 3.52 ns.

Analysis and evaluation of these performances, as depicted in Figure 6.25 showed that improving HFarrow with PDACSERNS-GA utilised 15% of the total LUT, 8.2% of the slices LUTs, 8.3% of the flip-flops used, 8.6% power consumption and 24% delay in the execution time when compared with HFarrow with PDA-CSRNS, NU-MDFT CSD with Pareto ABC [181] and SID-CSE [181].

The filter achieved an 85% decrements in the number of occupied slices from 2406 to 920 slices. There was a 91.9% decrements in power consumption and 76% decrements in execution delay time.

Table 6.50. Device and power utilisation.

Parameter	Conti-nous coeff.	HFarrow PDA-CSERNS	HFarrow PDA-CSERNS using GA	CSD with Pareto ABC [181]	SID-CSE[181]
Total slices	2507	923	920	2406	1633
Slice LUTs	8694	1692	1689	8950	5901
Flip- flops	10313	1725	1729	8980	5911
Total Power (mW)	1865	333.51	333.51	1751	1281
Total Delay (ns)	45.125	3.55	3.52	3.75	2.6

Table 6.51. Comparison of device utilisation.

Parameter	SDR in [182]	SDR Channe-lizer [183]	SDR Channelizer using NU MDFT FB [181]	HFarrow PDA-CSERNS	HFarrow PDA CSERNS -GA
Slice Registers	15,295 out of 58,880	29,797 out of 301,440	5880 out of 597,200	2125 out of 239616	2118 out od 239616
Slice LUTs	14,726 out of 58,880	21,169 out of 150,720	5901 out of 298,600	1641 out of 119808	1637 out of 119808

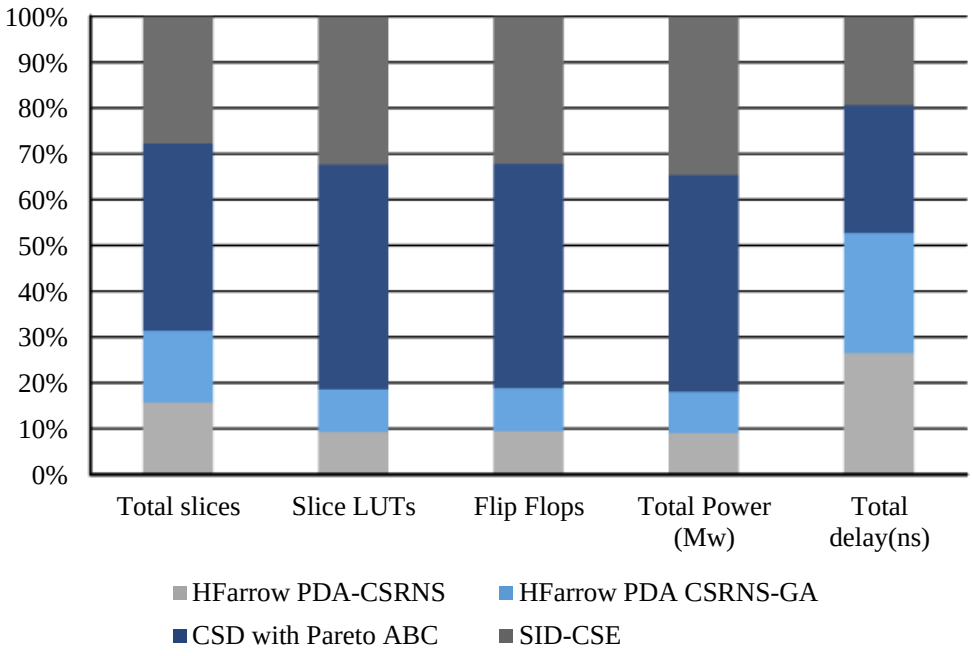

Figure 6.25. Resources Utilization of HFarrow PDA-CSERNS in comparison with other algorithms.

6.4 MULTI-RATE, MULTI-STAGE HYBRID GENERALISED DISCRETE FOURIER TRANSFORM (MHGDFT) AND HYBRID FARROW (MHFARROW) FILTER BANKS

The Section presents results of MHGDFT and MHFarrow algorithms approaches for realising low complexity multi-standard channelisation algorithm in SDR receiver. Section 6.4.1 discusses and presents the results

of MHGDFT algorithm while Section 6.4.2 discusses and interprets the results obtained for MHFarrow algorithm.

6.4.1 Results and Discussion of MHGDFT ALGORITHM

Four different filtering stages were used for the realization. Table 6.52 through to Table 6.55 showed filter specifications for multi-rate, multi-stage prototype and masking filters. Filter specifications were relaxed and this leads to a decrement in the multipliers used as evidenced in the lower filter order as shown in Table 6.56. The total number of multipliers used in MHGDFT filter bank were 111, while HGDFT filter obtained 511 multipliers, CDFB [66] and ICDM filter bank [184] have 1745 and 1545 respectively. The lower filter order and multipliers are due to cascade of different stages of filter.

Table 6.52. Filter specifications for the first stage MHGDFT algorithm.

Filter Bank	M	L	ω_s	ω_p	$\delta_{p'}$	$\delta_{s'}$	filter Length
Modal filter	-	39	0.4875	0.3964	0.0975	39	18
Bluetooth	-	39	0.4875	0.3964	0.0975	39	14
Zigbee	-	9	0.45	0.3592	0.09	39	4
WCDMA	-	7	0.438	0.229	0.0875	48.25	2

Table 6.53. Filter specifications for the second stage MHGDFT algorithm.

Filter Bank	M	L	ω_s	ω_p	$\delta_{p'}$	$\delta_{s'}$	filter length
Modal filter	2	-	0.1219	0.097	0.0992	48,25	6
Blue-tooth	2	-	0.1219	0.0991	0.0926	39	4
Zigbee	2	-	0.1141	0.08462	0.855	39	6
WCDMA	4	-	0.10958	0.08275	0.08531	48.25	6

Table 6.54. Filter specifications for the third stage MHGDFT algorithm.

Filter Bank	M	L	ω_s	ω_p	$\delta_{p'}$	$\delta_{s'}$	filter length
Modal filter	4	-	0.4875	0.3037	0.0926	39	3
Bluetooth	4	-	0.01523	0.01236	0.0950	39	1
Zigbee	5	-	0.01141	0.008462	0.08775	39	10
WCDMA	2	-	0.01370	0.01034	0.08531	48.25	11

Table 6.55. Filter specifications for the four stage MHGDFT algorithm.

Filter Bank	M	L	ω_s	ω_p	$\delta_{p'}$	$\delta_{s'}$	filter length
Modal filter	5	-	0.02	0.0165	0.0975	39	13
Bluetooth	5	-	0.001523	0,001034	0.09555	39	13
Zigbee	-	-	-	-	-	-	-
WCDMA	-	-	-	-	-	-	-

Table 6.56. Multiplication complexity for non-uniform filter bank using MHGDFT algorithm.

Filter Bank	Filter Order			Total number of Multiplications
	H_a	H_{ma}	H_{mc}	
CDFB [66]	3089	400		1745
ICDM [30]				
FB	2929	160		1545
HGDFT FB	170	158	183	511
MHGDFT	40	71	71	111

6.4.2 Results and Discussion of MHFarrow Algorithm

Applying the design steps given in Section 5.3.1 to Table 6.27, the following multi-rate, multi-stage filter results as indicated in Table 6.57 through to Table 6.60 were obtained respectively. With this procedure, cascading four multi-rates, multi-stages filter together result in relaxation of the modal and masking filters with reduced filter orders. Comparing the performance with the filter orders seen in Table 6.30. The total number of multipliers used by MHFarrow were 106 compared to 389 found in HFarrow as illustrated in Table 6.61 and CDFB [66], ICDM [30] and NU MDFT filter bank [181] with a higher number of multiplications. Fewer multipliers seen in MHFarrow are the consequences of the relaxed filter specifications.

Table 6.57. The filter specifications for the first stage MHFarrow algorithm.

Filter Bank	M	L	ω_s	ω_p	$\delta_{p'}$	$\delta_{s'}$	filter Length
Modal filter	-	39	0.4875	0.44118	0.1	50	16
Bluetooth	-	39	0.4875	0.4368	0.975	39	13
Zigbee	-	9	0.45	0.4005	0.09	39	4
WCDMA	-	7	0.438	0.4375	0.0875	48.25	2

Table 6.58. The filter specifications for the second stage MHFarrow algorithm.

Filter Bank	M	L	ω_s	ω_p	$\delta_{p'}$	$\delta_{s'}$	filter length
Modal filter	2	-	0.1219	0.097	0.0992	48,25	5
Blue-tooth	2	-	0.1219	0.1092	0.0926	39	3
Zigbee	2	-	0.1141	0.1001	0.855	39	4
WCDMA	4	-	0.10958	0.0547	0.08531	48.25	5

Table 6.59. Filter specifications for the third stage MHFarrow algorithm.

Filter Bank	M	L	ω_s	ω_p	$\delta_{p'}$	$\delta_{s'}$	filter length
Modal filter	4	-	0.4875	0.3037	0.0926	39	2
Bluetooth	4	-	0.01523	0.01365	0.0950	39	1
Zigbee	5	-	0.01141	0.01001	0.08775	39	8
WCDMA	4	-	0.01370	0.0068	0.08531	48.25	8

Table 6.60. Filter specifications for the fourth stage MHFarrow algorithm.

Filter Bank	M	L	ω_s	ω_p	$\delta_{p'}$	$\delta_{s'}$	filter length
Modal filter	5	-	0.02	0.0165	0.0975	39	12
Bluetooth	5	-	0.001523	0.001365	0.09555	39	12
Zigbee	-	-	-	-	-	-	-
WCDMA	-	-	-	-	-	-	-

Table 6.61. Multiplication complexity for non-uniform filter bank using MHFarrow filter.

Filter Bank	Filter Order			Total number of Multiplications
	H_a	H_{ma}	H_{mc}	
CDFB [66]	3089	400	-	1745
ICDM FB [30]	2929	160	-	1545
NU-MDFT FB[181]	187	430	469	1090
HFarrow filter bank	187	100	103	389
MHFarrow filter Bank	41	66	66	106

Table 6.62 and Table 6.63 showed resources utilisation of HGDFT and HFarrow filter with various improvement methods. HFarrow PDACSERNS GA filter shows most optimised resources utilisation. Also, it can be deduced from Table 6.64 that multi rate, multi stage hybrid Farrow (MMHFarrow) filter exhibits the lowest filter computational complexity and it is to be preferred for channelisation of multi-standard receivers in SDR.

Table 6.62. Comparison of device and power utilisation of HGDFT filter and its improvement algorithms.

Algorithms	Total slice	SLice LUTs	Flip flops	Total Power (mW)	Total delay (ns)
HGDFT PDARNS	1531	3897	3990	325.71	3.1222
HGDFT PDARNS-GA	1520	3990	3989	325	3.21
HGDFT PDA CSRNS	1470	3472	3586	341.36	3.22
HGDFT PDACSRNS-GA	1440	3445	3602	324.98	2.799
HGDFT PDACSERNS	1456	3360	3540	334	3.18
HGDFT PDACSERNS-GA	1455	3360	3537	330	3.14

Table 6.63. Comparison of device and power utilisation for HFarrow filter and its improvement algorithms.

Algorithms	Total slice	SLice LUTs	Flip flops	Total Power (mW)	Total delay (ns)
HFarrow PDARNS	1206	2073	2338	333.53	3.328
HFarrow PDARNS-GA	1201	2072	2330	332	3.31
HFarrow PDA CSRNS	1192	2014	2286	333.56	3.33
HFarrow PDACSRNS-GA	1190	2016	2265	333.24	3.33
HFarrow PDACSERNS	923	1692	1725	333.51	3.55
HFarrow PDACSERNS-ga	920	1689	1729	333.51	3.52

Table 6.64. Comparison of the three filter architectures in terms of multiplication complexity for non-uniform filter bank.

Filter Bank	Filter Order			Total number of Multiplications
	H_a	H_{ma}	H_{mc}	
CDFB [66]	3089	400	-	1745
HGDFT	170	158		511
HFarrow filter bank	187	100	103	389
MHGDFT	187	430	469	111
MHFarrow filter Bank	41	66	66	106

6.5 CONCLUDING REMARKS

In Section 6.2, HGDFT Channelisation algorithm was simulated for designing a low complexity multi-standard SDR based receiver and the results presented. The method employed combination of modulated GDFT filter and interpolation coefficient decimation filter in a process described as a hybrid GDFT filter. This is an efficient realisation for both uniform and non-uniform channelisation.

From Section 6.2, the HGDFT channelisation algorithm achieved: 54%, 61% and 86% in comparison with coefficient decimation filter bank (CDFB), interpolated coefficient decimated filter (ICDM) and non-uniform modulated discrete Fourier transform (NU MDFT). The HGDFT channelisation algorithm was further improved upon as explained: HGDFT with PDA-RNS filter as discussed in Section 6.2.1 achieved a 72.21% decrement in the number of occupied slices from 2406 to 1531 slices. There was 90.29% decrement in power consumption and 67.1% decrement in execution delay time when compared with NU-MDFT CSD with Pareto ABC [181]

and SID-CSE [181]. Optimisation of HGDFT PDARNS-GA filter achieved a 79% decrement in the number of occupied slices from 2406 to 1520 slices. There was 90% decrement in power consumption and 75% decrement in execution delay time.

Improving HGDFT with PDA-CSRNS as explained in Section 6.2.2 achieved a 73.32% decrement in the number of occupied slices from 2406 to 1470 slices. There was a decrement of 83% in power consumption and 89.88% decrement in execution delay time. From the simulation results, the performance of HGDFT with PDA-CSRNS filter shows high efficiency in terms of area and speed in comparison with the PDA-RNS. There was remarkable decrement in the total slices, LUT utilisation, power and delay when the HGDFT channelisation algorithm was improved with these number representations compared with the other methods in the literature. Optimisation of the HGDFT PDA-CSRNS filter with HGDFTPDACSRNS-GA further reduces the computational complexities with 80% decrement in the number of occupied slices from 2406 to 1440 slices. There was a decrement of 91.1% in power consumption and 76% decrement in execution delay time.

However, improvement of HGDFT with PDA-CSERNS as described in Section 6.2.3 showed higher performances in terms of computational complexity decrement, resources used, processing speed, and power consumption.

In Section 6.3, HFarrow Channelisation algorithm was simulated for designing a low complexity multi-standard SDR based receiver and the results presented. The method employ a combination of modulated Farrow filter, coefficient decimation filter and the FRM filter, in a process described as a hybrid coefficient decimation filter. This is an efficient realisation for both uniform and non-uniform channelisation. From the simulation results, HFarrow achieved multipiers decrement in the following: 70% in CDFB, 64% ICDM and 50% in NU MDFT. Moreover, the computational complexities were reduced futher using parallel distributed arithmetics with different number systems.

Improvement of HFarrow filter with PDA-RNS filter presented in Section 6.3.1 showed that the filter achieved a 82% decrement in number of occupied slices from 2406 to 1206 slices. There was a 96% decrement in power consumption and 62% decrement in execution delay time in comparison with NU-MDFT CSD with Pareto ABC [181] and SID-CSE [181]. Optimisation of the HFarrow PDARNS-GA resulted in a decrement in the following: 82% in total LUT used; 90% slice LUTs, 89% flip-flop and 83% delay.

Improvement of HFarrow with PDA-CSRNS algorithm presented in Section 6.3.2 showed a decrement in hardware resources of 87% in the total slices, 94.9% in the slice LUTs; 98% in the power and 64%.

Also, when HFarrow with PDA-CSERNS was improved upon as discussed in Section 6.3.3, there were decrement

of: 90% in the total slices, 96% in slice LUTs, 98% in power and 60% in delay time. Optimisation of HFarrow PDACSRNS-GA filter achieved an 83% decrement in the number of occupied slices from 2406 to 1190 slices. There was a 91.7% decrement in power consumption and 75.5% decrement in execution delay time.

The HFarrow-FB outperformed the existing channelisation as a result of the flexibility in its fractional delay. This allows free selection of the cut off frequency and easy adjustment to the passband ripples while improving the stopband attenuation. This results in suppression of the large lobes in the stopband and the cut off frequency of the filter can be readily adjusted. Thus, low complexity filter bank is possible with the HFarrow FB and it can be used to channelise signals from multi-standards signals. There was a remarkable decrement in the total slices, LUT utilisation, power and delay when the HFarrow channelisation algorithm was improved with these number representations, compared with the other methods in the literature.

However, HFarrow with PDA-CSERNS showed higher performances in terms of computational complexity decrement, thus, a decrement of 84% in the total slices, 80.1% in slice LUTs, 90.4% in power and 75% in delay time. Although, it was seen that the delay time of SID CSE was lower than the PDA-CSERNS, it can still be proven that the HFarrow PDA-CSERNS showed better performances than the other improvement methods for HFarrow filter.

In Section 6.4, two new multi-rates, multi-stages approaches for reducing the filter order, the number of multiplications and the computational complexities are presented. The two approaches are: MHGDFT and MHFarrow and they used fractional rate conversion methods. Employing fractional rate transformation in MHGDFT and MHFarrow channelisation algorithm outperformed the single rate HGDFT and HFarrow filter bank. Cascading filters in stages tend to relax the passband width and stopband attenuation, which invariably results in reduced filter order and multipliers and computational complexities.

MHGDT filter bank accomplished 71% multipliers decrement when compared with HGDFT while MHFarrow filter bank achieved up to 73% multipliers decrement rate in comparison with HFarrow filter bank. Table 6.64 shows comparison between the three proposed algorithms in terms of multiplication complexity. It can be deduced from Table 6.64 that multi rate, multi stage hybrid Farrow (MMHFarrow) filter exhibits the lowest filter computational complexity and it is to be preferred for channelisation of multi-standard receivers in SDR.

CHAPTER 7 CONCLUSION

The main aim of this work is to develop a low computational complexity channelisation algorithm for multi-standard receiver platforms with the purpose of reducing computational load, increasing speed of operation and reducing power consumption of the filter bank. The general objective is to explore combinations of low complexity channelisation algorithms and number system with genetic algorithm that should give a flexible multi-standard channelisation algorithm with low computational complexity, high speed, and low power consumption. Major achievement upon execution of the set objectives are:

- Development of an hybrid generalised discrete Fourier transform filter bank and further improvement using different number systems and genetic algorithms.
- Development of an hybrid Farrow channelisation algorithm and further improvement using different number systems and genetic algorithm, by reducing the number of adders and multipliers used.
- Development of a multi-rate, multi-stage hybrid generalised discrete Fourier transform and hybrid Farrow filter with computation complexity of the filter reduced to absolute minimum.

Some details of these achievement are summarized below.

7.1 HYBRID GENERALISED DISCRETE FOURIER TRANSFORM FILTER BANK (HGDFT-FB)

Modulated GDFT filter bank was developed as substitute for conventional GDFT filter bank. HGDFT channelisation algorithm was formed by hybridising modulated GDFT, FRM and coefficient decimation 1 (CD-1) method. An algorithm was written to these effects. The algorithm was simulated on multi-standard (three) channels and the performance assessed based on parameters such as passband width, stopband width, stopband attenuation, passband ripples, filter order and the total number of multiplications.

HGDFT channelisation algorithm showed multiplier reduction in the following: 54% in CDFB, 61% ICDM and 86% in NU MDFT. The HGDFT filter takes advantage of GDFT modulation, symmetrical impulse response, fractional sample rate converter from frequency response masking (FRM) interpolator coefficient decimation factors to adjust the filter specifications, that result in relatively lower computational load. This achievement was further improved using different number systems and genetic algorithm.

7.1.1 Improvement of HGDFT Algorithm with Parallel Distributed based Residual Number System (PDA-RNS)

Major factor contributing to relatively high filter order seen in HGDFT filter are the multipliers and adders. An approach to improve the performance of HGDFT channelisation algorithm by reducing the multipliers used during filtering operations was considered.

Quantized 16-bit binary values obtained from HGDFT filter coefficients were transformed into integer values. The integer values were partitioned into three residual numbers based on moduli values $2n$, $2n-1$ and $2n+1$. Parallel distributed arithmetic architecture was used for implementation of the three residual values. It was observed that HGDFT PDA-RNS filter achieved 78% reduction in number of occupied slices from 2406 to 1531 slices. There was 91.3% reduction in power consumption and 75% reduction in execution delay time. Performance of the HGDFT PDA-RNS was attributed to the processing approach used by parallel distributed arithmetic and the modular nature of residual number system. Improving the performance of HGDFT PDARNS-GA resulted further in lower computational complexity as evidenced in the resources utilised. HGDFT PDARNS-GA filter achieved an 79% reduction in number of occupied slices from 2406 to 1520 slices. There was 91% reduction in power consumption and 75% reduction in execution delay time.

7.1.2 Improvement of HGDFT Algorithm with Parallel Canonical Residue number system (PDA-CSRNS)

Another approach was employed to improve the performance of HGDFT channelisation algorithm.

The algorithm was written and implemented on 16-bit quantized residual values obtained from the HGDFT filter bank. The binary values of the three RNS values obtained were converted into canonical signed digits by scanning the binary bits from left to right for the number of consecutive ones. In this way, the number of computational operations were reduced since the number of ones were reduced. This filter achieved a 79% reduction in the number of occupied slices from 2406 to 1470 slices. There was 91% reduction in power consumption and 74% reduction in execution delay time.

The optimised HGDFT PDA-CSRNSGA filter achieved a 80% reduction in the number of occupied slices from 2406 to 1470 slices. There was 91.1% reduction in power consumption and 76% reduction in execution delay

time.

7.1.3 Improvement of HGDFT Channelisation Algorithm using parallel distributed based common sub-expression elimination (HGDFT PDA-CSE)

Hybrid generalised discrete Fourier transform parallel distributed arithmetic based common sub-expression residual number systems (HGDFT PDA-CSERNS) was found very efficient in [Friend]realising and improving the performance of HGDFT-filter bank. This is done by hybridizing the horizontal and vertical common sub-expression elimination methods together and finding the most used set of bit patterns.

This improvement of HGDFT PDA-CSERNS achieved a 78% reduction in number of occupied slices from 2406 to 1456. There was 91.2% reduction in power consumption and 76% reduction in execution delay time.

Adders in HGDFT filter bank were downsized to remarkable extent and this resulted in reduced logical depth as well as the critical paths. The reduced adder depth produces lesser computational complexity on the digital filter. Simulations showed that the algorithm achieved 0:95% reduction in total slices, 3:23% reduction in slice LUTs, 1.8% reduction in flip-flops used, and 2% reduction in power consumed. Although the HGDFT filter algorithms looks big, it is not more complex than other filter algorithm in the literature and thus the cost implication is bearable.

7.2 HYBRID FARROW CHANNELISATION ALGORITHM (HFARROW)

Another method described here as HFarrow was explored for designing low computational complexity channelisation algorithm. Firstly, a low complexity modulated Farrow filter was developed. This was hybridized with FRM, interpolation coefficient decimation factor. The algorithm was tested on three multi-standard receiver channels with distinct filter specifications as indicated in Section 3.2.4.

Total number of multipliers used by HFarrow filter bank was 389 while ICDM expended was 1545, NU MDFFB consumed up to 1090 , and CDFB used up 1745. The percentage of multipliers used by HFarrow algorithm in comparison with NU MDFT FB was 50%. HFarrow showed multipliers reduction in the following: 70% in CDFB, 64% ICDM and 50% in NU MDFT. There was remarkable reduction in the number of multipliers used in HFarrow compared with the algorithms used in literature.

7.2.1 Improvement of HFarrow Algorithm with PDA-RNS (HFarrow PDA-RNS)

HFarrow PDA-RNS utilised the following resources: 18% of the total LUT, 11% of the slices LUT, 12.2% of the flip-flops used, 8% power consumption and 25% delay in the execution time, when compared with NU-MDFT

CSD Pareto ABC. This translates to a 82% reduction in number of occupied slices from 2406 to 1206 slices. There was 92% reduction in power consumption and 75% reduction in execution delay time.

Optimisation of the HFarrow PDA-RNSGA resulted in 82%, 90 %, 89 % and 83% reductions respectively in total LUT used, LUT slices, flip-flop, and execution time delay.

7.2.2 Improvement of HFarrow Channelisation Algorithm with PDA-CSRSN (HFarrow PDA-CSRNS)

This method improves the resources utilisation of HFarrow filter bank. The computational capabilities of the HFarrow PDA-CSRNS improve remarkably compared to the HFarrow PDA-RNS algorithm.

HFarrow with PDA-CSRNS utilised 17% of the total LUT, 10.1% of the slices LUT, 11.7% of the flip-flops used, 8.7% power consumption and 25.5% delay in the execution time when compared with NU-MDFT CSD with Pareto ABC. The filter achieved a 83% reduction in number of occupied slices, from 2406 to 1201 slices. There was 91.3% reduction in power consumption and 74.5% reduction in execution delay time.

Optimisation using HFarrow with PDA-CSRNS resulted in 84% reduction in number of occupied slices from 2406 to 1190. There was 91.7% reduction in power consumption and 75.5% reduction in execution delay time. It was observed that the HFarrow PDACSRNS-GA filter achieved an 83% reduction in the number of occupied slices from 2406 to 1190 slices. The design is efficient in terms of hardware resources , power utilization and delay.

7.2.3 Improvement of HFarrow Channelisation Algorithm with PDA-CSERNS (HFarrow PDA-CSERNS)

PDA-CSERNS improves performance of HFarrow algorithm due to the ability to cut down the adder depth and so reduce computational load of the filter. The improvement of HFarrow PDA-CSERNS over HFarrow PDA-RNS can be explained in terms of resources utilisation.

Devices and power utilisation are 16% of the total slices, 9.9% of the slice LUTs, 8.6% of the power usage, but the delay was 0.5% higher than HFarrow PDA-CSRNS.

Optimisation using HFarrow PDA-CSERNSGA resulted in 85% reduction in number of occupied slices from 2406 to 920. There was 91.9% reduction in power consumption and 76% reduction in execution delay time. It can be proven that the optimising HFarrow PDA-CSERNS or with PDA-CSERNSGA gives better improvement than HFarrow PDA-RNS.

HGDFT channelisation algorithm showed reduction in the following: 54% in CDFB, 61% ICDM and 86% in NU MDFT while HFarrow showed reduction in the following: 70% in CDFB, 64% ICDM and 50% in NU MDFT. The complexity of the two channelisation algorithms were reduced more when improved with PDA-CSERNS method. Although the HFarrow filter algorithms looks big, it is not more complex than other filter algorithm in the literature and thus the cost implication is bearable.

7.3 MULTI-RATE, MULTI-STAGE HYBRID GDFT (HGDFT) AND HYBRID FARROW (HFARROW) ALGORITHM

In Chapter 5.1, two different multi-rate, multi-stage approaches known as MHGDFT and MHFarrow filter banks were presented to reduce the computational capability of HGDFT and HFarrow filter banks. MHGDFT and MHFarrow filter bank achieved outstanding improvement over the single rate HGDFT and HFarrow filter bank.

The MHGDFT filter bank have 71% lesser filter coefficients than HGDFT filter bank while MHFarrow filter achieved 73% fewer filter coefficients than HFarrow filter bank. The new filter designs have significantly reduced filter coefficients and filter multipliers.

7.4 FUTURE RESEARCH

The following extensions should be considered in future for this research work.

- Use of multi-stage cascaded integrator comb (CIC) and Inverse Sinc filter: The research in this thesis was accomplished using FIR-based filter to reduce the algorithms computational complexities. Multi-stage CIC and Inverse Sinc filter should be considered for low computational complexities of SDR channelisation algorithm in the future. Comparision should be made between single stage CIC and multi-stage CIC filter as well as the inverse Sinc filter. Also the multiplierless single stage and multi-stage CIC filter should also be studied. Multiplierless filter could be expected to reduce the computational capabilities of SDR filter to minimum.
- Improvement of the Hybrid algorithm using Serial or multi-bit serial architecture: The Hybrid algorithms deployed in the research thesis should incorporate serial or multi-bit serial distributed based design. The algorithm should include pipelining of multi-bit serial distributed filter. The system architecture and the system clock should be investigated. This could improve the performance of SDR computational complexity.
- Real time practical implementations of the hybrid algorithms on field programmable gate array (FPGA): The research in the thesis was carried out using the MATLAB and FPGA co-simulations. In the future works, the real time practical implementations should be considered. Two added benefits could be derived

from this approach. The first merit of using FPGA is the full digitization of the architecture and secondly is the faster response time.

7.5 CONCLUDING REMARKS

Channelisation algorithms for processing multi-standard receivers are computationally intensive processes due to high volume of digital filtering and high sampling rate involved at the digital front-end. The effect on the Software Defined Radio (SDR) receiver are increase in power consumption, higher delay which manifests in form of data loss or signal distortion, and higher resources utilisation.

The complexity increment in GDFT filter is attributed to extra three phase modifications added to is time and frequency offset which produce multiplicative complexity in the filter. The resultant effect is higher filter order and higher delay in software defined radio (SDR) mobile systems.
Also, in FPCC, complexity compromise is reached during the process of approximating higher frequency band with high level of distortion at passband ripples and stop band attenuation. A substantial higher filtering operation is required in order to approximate a given design specification with prescribed tolerance. This resulted to higher filter order and higher delay in multi-standard receivers software defined radio.

Developed hybrid Generalised Discrete Fourier Transform (HGDFT), hybrid Farrow (HFarrow), multi-rate, multi-stage HGDFT (MHGDFT) and multi-rate, multi-stage Farrow (MHFarrow) are found as an effective methods for reducing the computational complexities of the conventional GDFT and Farrow filter in SDR channelisation algorithm for multi-standard based receiver platform. The developed algorithms undergo different processes such as modulation process of conventional GDFT and Farrow filter, symmetrical behaviour of impulse response, fractional delay from interpolated coefficient decimation filter enhances low computational capability of the developed filter.

In order to cater for the upcoming development in the wireless receiver, the HGDFT and the HFarrow algorithms are further improved using these number systems approaches. These are parallel distributed arithmetic based residual number system (PDA-RNS), parallel distributed arithmetic's based canonical signed residual number sytem (PDA-CSRNS), parallel distributed based diminished 1 canonical signed residual number system (PDA-DCRNS) and parallel distributed based common sub-expression elimination method (PDA-CSERNS).

However, representing the HGDFT and HFarrow channelisation algorithm with these parallel distributed number systems resulted in distortion of the pass-band ripples and stopband attenuation. Therefore, genetic algorithm is deployed to HGDFT and HFarrow parallel distributed number systems. Remarkable improvements in filter complexities were achieved as the pass-band ripples are minimised while the stop-band attenuation are maximised.

The results obtained were simulated using MATLAB and co-simulated on Altera FPGA using the very high speed integrated circuits (VHSIC) hardware descriptive language.

Reduction in filter orders and multipliers as seen in HGDFT PDARNS-GA and HFarrow PDARNS-GA indicated that the algorithms are effective for executing low complexity channelisation in multi-standard receiver systems. Moreover, the total replacement of the multipliers with adders as seen in HGDFT PDACSRNS-GA, HGDFT PDADCRNS-GA, HGDFT-PDACSE-GA resulted in lower complexities as evidenced by the resources allocations, slice LUT, ALUT registers, power consumption and delay execution time. Implementing HFarrow PDACSRNS-GA and HFarrow PDA-CSE GA algorithms achieved lower complexities than HGDFT parallel distributed based number sytems equivalent developed.

Optimum lower complexities reduction were recorded when multi-rate, multi-stage HGDFT (MHGDFT) and HFarrow (MHFarrow) algorithms were implemented for multi-standard based SDR receiver.

REFERENCES

[1] Á. P. Navarro, "Channelization for multi-standard software-defined radio base stations," Ph.D. dissertation, National University of Ireland Maynooth, 2011.

[2] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Prentice Hall, 1983.

[3] F. R. J. Palomo-Navarro, A. and R. Villing, "Combined FRM and GDFT filter bank designs for improved nonuniform DSA channelisation," *Wireless Communications and Mobile Computing*, vol. 16, pp. 1440–1456, 2016.

[4] U. Meyer-Baese and U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*. Springer, 2007.

[5] S. A. Khan, *Digital Design of Signal Processing Systems: A Practical Approach*. John Wiley & Sons, 2011.

[6] W. A. Abu-Al-Saud and G. L. Stuber, "Efficient wideband channelizer for software radio systems using modulated PR filterbanks," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2807–2820, 2004.

[7] R. E. Crochiere and L. R. Rabiner, "Multirate digital signal processing," *Applied Signal Processing: Theory and Design*, vol. 2, pp. 127–192, 1987.

[8] M.-L. Boucheret, I. Mortensen, and H. Favaro, "Fast convolution filter banks for satellite payloads with on-board processing," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 238–248, 1999.

[9] J. Lillington, "Comparison of wideband channelisation architectures," in *International Signal Processing Conference (ISPC)*. ISPC, 2003, pp. 2042–2079.

[10] T. P. Deshmukh, P. Deshmukh, and P. Dakhole, "Design of cyclotomic Fast Fourier Transform architecture over Galois field for 15 point DFT," in *Industrial Instrumentation and Control (ICIC)*. IEEE, 2015, pp. 607–611.

[11] B. H. Tietche, O. Romain, and B. Denby, "Sparse channelizer for FPGA-based simultaneous multichannel DRM30 receiver," *IEEE Transactions on Consumer Electronics*, vol. 61, pp. 151–159, 2015.

[12] P. Duhamel and M. Vetterli, "Fast Fourier transforms: a tutorial review and a state of the art," *Signal Processing*, vol. 19, pp. 259–299, 1990.

[13] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965.

[14] S. Kalathil and E. Elias, "Non-uniform cosine modulated filter banks using meta-heuristic algorithms in CSD space," *Journal of Advanced Research*, vol. 6, pp. 839–849, 2015.

[15] Z. Chang, A. Vinod, and P. Meher, "Reconfigurable architectures for low complexity software radio channelizers using hybrid filter banks," in *Communication Systems (ICCS)*. IEEE, 2006, pp. 1–5.

[16] F. Harris and R. McGwier, "A receiver structure that performs simultaneous spectral analysis and time series channelization," in *Proceedings of the Technical Conference and Product Exposition (SDR)*. SDR, 2009, pp. 1–5.

[17] S. Sudharman, A. D. Rajan, and T. Bindiya, "Design of a Power-Efficient Low-Complexity Reconfigurable Non-maximally Decimated Filter Bank for High-Resolution Wideband Channels," *Circuits, Systems, and Signal Processing*, vol. 38, pp. 2703–2735, 2019.

[18] A. Ambede, K. Smitha, and A. P. Vinod, "Flexible low complexity uniform and nonuniform digital filter banks with high frequency resolution for multistandard radios," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 23, pp. 631–641, 2014.

[19] P. Lovina and K. A. Manjusha, "SDR Applications using VLSI Design of Reconfigurable Devices," *International Journal of Scientific Research in Science and Technology*, vol. 4, pp. 1148–1153, 2018.

[20] X. Peng, J. Li, X. Zhou, Q. Lin, and Z. Chen, "Analysis and design of M-channel hybrid filter bank with digital calibration," *IEEE Access*, vol. 6, pp. 24 606–24 616, 2018.

[21] P. S. Diniz, E. A. Da Silva, and S. L. Netto, *Digital Signal Processing: System Analysis and Design*. Cambridge University Press, 2010.

[22] A. P. Navarro *et al.*, "Overlapped polyphase DFT modulated filter banks applied to TETRA/TEDS SDR base station channelization," in *Royal Irish Academy Communication and Radio Science Colloquium (RIA)*. RIA communication and Radio Science, 2010, pp. 1–7.

[23] M. Blok, "Fractional delay filter design for sample rate conversion," in *Computer Science and Information Systems (FedCSIS)*. IEEE, 2012, pp. 701–706.

[24] V. Valimaki and A. Haghparast, "Fractional delay filter design based on truncated Lagrange interpolation," *IEEE Signal Processing Letters*, vol. 14, pp. 816–819, 2007.

[25] T. Karp and N. J. Fliege, "MDFT filter banks with perfect reconstruction," in *Circuits and Systems (ISCAS)*. IEEE, 1995, pp. 744–747.

[26] C. W. Farrow, "A continuously variable digital delay element," in *Circuits and Systems (ISCAS)*. IEEE, 1988, pp. 2641–2645.

[27] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Pearson Education India, 1993.

[28] B. Kuldeep, A. Kumar, and G. Singh, "Design of quadrature mirror filter bank using Lagrange multiplier method based on fractional derivative constraints," *International Journal on Engineering Science and Technology*, vol. 18, pp. 235–243, 2015.

[29] B. Brannon, D. Efstathiou, and T. Gratzek, "A look at software radios: Are they fact or fiction," *Electronic Design*, vol. 46, pp. 117–122, 1998.

[30] A. Ambede, S. Shreejith, A. P. Vinod, and S. A. Fahmy, "Design and realization of variable digital filters for software-defined radio channelizers using an improved coefficient decimation method," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, pp. 59–63, 2016.

[31] E. Dahlman, C. Oestges, A. C. Bovik, B. A. Fette, K. Jack, F. Dowla, S. Parkvall, J. Skold, C. DeCusatis, E. da Silva *et al.*, *Communications Engineering Desk Reference*. Academic Press, 2009.

[32] J. H. Reed, *Software Radio: A Modern Approach to Radio Engineering*. Prentice Hall Professional, 2002.

[33] N. Chandran and M. C. Valenti, "Three generations of cellular wireless systems," *IEEE Potentials*, vol. 20, pp. 32–35, 2001.

[34] T. J. Rouphael, *RF and Digital Signal Processing for Software-Defined Radio: A Multi-Standard Multi-Mode Approach*. Elsevier Science, 2009.

[35] J. R. Machado-Fernandez, "Software defined radio: Basic principles and applications," *Revista Facultad de Ingenieria*, vol. 24, pp. 79–96, 2015.

[36] P. K. Bondyopadhyay, "Guglielmo Marconi-The father of long distance radio communication-An engineer's tribute," in *Microwave Conference (IMC)*. IEEE, 1995, pp. 879–885.

[37] J. J. Green, *The Apparatus for Wireless Telegraphy*. University of Notre Dame, 1933.

[38] D. Raychaudhuri and N. B. Mandayam, "Frontiers of wireless and mobile communications," *IEEE on Wireless Communication*, vol. 100, pp. 824–840, 2012.

[39] L. Kleinrock, "An early history of the internet [History of Communications]," *IEEE Communications Magazine*, vol. 48, pp. 14–20, 2010.

[40] N. Abramson, "Development of the ALOHANET," *IEEE Transactions on Information Theory*, vol. 31, pp. 119–123, 1985.

[41] R. Frenkiel and M. Schwartz, "Creating cellular: A history of the AMPS project (1971-1983)[History of Communications]," *IEEE Communications Magazine*, vol. 48, pp. 14–24, 2010.

[42] C.-Y. Chen, F.-H. Tseng, K.-D. Chang, H.-C. C. Chao, J.-L. Chen *et al.*, "Reconfigurable software defined radio and its applications," *Journal of Applied Science and Engineering*, vol. 13, pp. 29–38, 2010.

[43] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, vol. 33, pp. 26–38, 1995.

[44] R. Janani, A. Manikandan, and V. Venkataramanan, "Software defined radio implementation in 3GPP systems," *Journal of Engineering and Applied Sciences*, vol. 10, pp. 5537–5541, 2015.

[45] R. Lackey and D. W. Upmal, "Speakeasy: the military software radio," *IEEE Communications Magazine*, vol. 33, pp. 56–61, 1995.

[46] A. Palomo Navarro, R. Villing, and R. Farrell, "Practical Non-Uniform Channelization for Multistandard Base Stations," *ZTE Communications*, vol. 9, pp. 115–124, 2011.

[47] D. Babic, V. Lehtinen, and M. Renfors, "Discrete-time modeling of polynomial-based interpolation filters in rational sampling rate conversion, Thailand," in *Circuits and Systems (ISCAS)*. IEEE, 2003, pp. 145–145.

[48] B. Chi, Z. Wang, and S. S. Wong, "A superheterodyne receiver front-end with on-chip automatically Q-tuned notch filters," in *Radio Frequency Integrated Circuits (RFIC)*. IEEE, 2007, pp. 21–24.

[49] T. Neu, "Direct RF conversion: From vision to reality," *Texas Instruments Journal*, vol. 1, pp. 1–8, 2015.

[50] D. M. Akos and J. B. Tsui, "Design and implementation of a direct digitization GPS receiver front end," *IEEE Transactions on Microwave Theory and Techniques*, vol. 44, pp. 2334–2339, 1996.

[51] M. L. Psiaki, J. Thor, and D. Akos, "A comparison of direct RF sampling and downconvert sampling GNSS receiver architectures," in *International Technical Meeting of the Satellite Division of The Institute of Navigation (ION)*. ION, 2003, pp. 1941–1952.

[52] M. L. Psiaki, S. P. Powell, H. Jung, and P. M. Kintner, "Design and practical implementation of multifrequency RF front ends using direct RF sampling," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, pp. 3082–3089, 2005.

[53] T. Tsukahara, R. Ito, and K. Arimura, "Complex signal processing used in modern RF transceivers," in *International Symposium on Radio-Frequency Integration Technology (RFIT)*. IEEE, 2015, pp. 100–102.

[54] R. H. Walden, "Analog-to-digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 539–550, 1999.

[55] P. Roblin, C. Quindroit, N. Naraharisetti, S. Gheitanchi, and M. Fitton, "Concurrent linearization: The state of the art for modeling and linearization of multiband power amplifiers," *IEEE Microwave Magazine*, vol. 14, pp. 75–91, 2013.

[56] V. Giannini, J. Craninckx, and A. Baschirotto, *Baseband Analog Circuits for Software Defined Radio*. Springer Science & Business Media, 2008.

[57] T. Hentschel and G. Fettweis, "Sample rate conversion for software radio," *IEEE Communications Magazine*, vol. 8, pp. 142–150, 2000.

[58] T. Hentschel, M. Henker, and G. Fettweis, "The digital front-end of software radio terminals," *IEEE Personal Communications*, vol. 6, pp. 40–46, 1999.

[59] J. Kaiser, "Nonrecursive digital filter design using the IO-sinh window function," *Proceedings of the IEEE*, vol. 2, pp. 20–23, 1974.

[60] T. Saramäki, S. Mitra, and J. Kaiser, *Handbook for Digital Signal Processing*. New York Wiley Interscience, 1993.

[61] L. Milic, *Multirate Filtering for Digital Signal Processing: MATLAB Applications: MATLAB Applications*. IGI Global, 2009.

[62] F. J. Harris, *Multirate Signal Processing for Communication Systems*. Prentice Hall PTR, 2004.

[63] A. Mehrnia, M. Dai, and A. N. Willson, "Efficient halfband FIR filter structures for RF and IF data converters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, pp. 64–68, 2016.

[64] A. Croisier, D. Esteban, M. Levilion, and V. Riso, "Digital filter for PCM encoded signals," 1973.

[65] R. Mehra and S. Verma, "FPGA Based Design of Direct Form FIR Polyphase Interpolator for Wireless Communication," *International Journal of Electrical Electronics &Telecommunication Engineering*, vol. 44, pp. 1108–1113, 2013.

[66] R. Mahesh, A. P. Vinod, E. M. Lai, and A. Omondi, "Filter bank channelizers for multi-standard software defined radio receivers," *Journal of Signal Processing Systems*, vol. 62, pp. 157–171, 2011.

[67] W. Zhang, C. Zhang, Z. Zhao, F. Liu, and T. Chen, "Low-complexity channelizer based on frm for passive radar multi-channel wideband receiver," *Circuits, Systems, and Signal Processing*, vol. 39, no. 1, pp. 420–438, 2020.

[68] F. Cruz Garrido and J. Torres Gómez, "Fpga design of a time-variant coefficient filter," *Ingeniería Electrónica, Automática y Comunicaciones*, vol. 41, no. 2, pp. 51–62, 2020.

[69] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay [FIR/all pass filters design]," *IEEE Signal Processing Magazine*, vol. 13, pp. 30–60, 1996.

[70] H. Johansson and P. Lowenborg, "On the design of adjustable fractional delay FIR filters," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 50, pp. 164–169, 2003.

[71] H. Johansson, "Farrow-structure-based reconfigurable bandpass linear-phase FIR filters for integer sampling rate conversion," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, pp. 46–50, 2011.

[72] M. A. Alrmah and S. Weiss, "Filter bank based fractional delay filter implementation for widely accurate broadband steering vectors," in *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2013, pp. 332–335.

[73] P. Murphy, A. Krukowski, and A. Tarczynski, "An efficient fractional sample delayer for digital beam steering," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1997, pp. 2245–2248.

[74] C. S. Park, S. Kim, J. Wang, and S. Park, "Design and implementation of a farrow-interpolator-based digital front-end in lte receivers for carrier aggregation," *Electronics*, vol. 10, p. 231, 2021.

[75] S. J. Darak, A. P. Vinod, and E. M. Lai, "A new variable digital filter design based on fractional delay," in *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2011, pp. 1629–1632.

[76] S. Dhabu and V. A. Prasad, "Design of modified second-order frequency transformations based variable digital filters with large cutoff frequency range and improved transition band characteristics," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, pp. 413–420, 2015.

[77] K. Rajalakshmi, S. Gondi, and A. Kandaswamy, "A fractional delay fir filter based on lagrange interpolation of farrow structure," *Int. J. Elect. Electron. Eng*, vol. 1, pp. 103–107, 2012.

[78] T. Devis and M. Manuel, "A low-complexity 3-level filter bank design for effective restoration of audibility in digital hearing aids," *Biomedical Engineering Letters*, vol. 10, pp. 593–601, 2020.

[79] R. Crochiere, *Multirate Digital Signal Processing*. Pearson Hall, 1983.

[80] K. F. C. Yiu, N. Grbić, S. Nordholm, and K. L. Teo, "A hybrid method for the design of oversampled uniform DFT filter banks," *Signal Processing*, vol. 86, pp. 1355–1364, 2006.

[81] K. R. Rao, D. N. Kim, and J. J. Hwang, *Fast Fourier Transform-Algorithms and Applications*. Springer Science & Business Media, 2011.

[82] T. Bindiya and E. Elias, "Design of totally multiplier-less sharp transition width tree structured filter banks for non-uniform discrete multitone system," *International Journal of Electronics and Communications*, vol. 69, pp. 655–665, 2015.

[83] Á. P. Navarro, R. Villing, and R. J. Farrell, "Practical Non-Uniform Channelization for Multistandard Base Stations," *ZTE Communications*, vol. 9, pp. 15–24, 2015.

[84] E. Elias, P. Löwenborg, H. Johansson, and L. Wanhammar, "Tree-structured IIR/FIR uniform-band and octave-band filter banks with very low-complexity analysis or synthesis filters," *Signal Processing*, vol. 83, pp. 1997–2009, 2003.

[85] S. Kalathil and E. Elias, "Design of multiplier-less sharp non-uniform cosine modulated filter banks for efficient channelizers in software defined radio," *International Journal Engineering science and technology*, vol. 19, pp. 147–160, 2016.

[86] S. Pandey and S. Nemade, "A review of discrete hartley transform using delay and number of slice count," *International Journal of Electronics Communication and Computer Engineering*, vol. 8, pp. 88–91, 2017.

[87] D. F. Chiper, "Radix-2 fast algorithm for computing discrete hartley transform of type iii," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, pp. 297–301, 2012.

[88] G. Bi and Y. Chen, "Fast generalized dft and dht algorithms," *Signal Processing*, vol. 65, pp. 383–390, 1998.

[89] L. Arnaldi and H. Dellavale, "Oversampled filter bank channelizer for cryogenic detectors," *Review of Scientific Instruments*, vol. 92, p. 023304, 2021.

[90] H. Yi, P. Ouyang, T. Yu, and T. Zhang, "An algorithm for morlet wavelet transform based on generalized discrete fourier transform," *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 17, p. 1950030, 2019.

[91] J. Sun and Z. Zhang, "Generalized orthogonal discrete w transform and its fast algorithm," *Discrete Dynamics in Nature and Society*, vol. 2021, 2021.

[92] Y. Dai, H. Yi, and Y. Tao, "An algorithm for linear convolution based on generalized discrete fourier transform," *Chin. J. Eng. Math*, vol. 36, pp. 106–114, 2019.

[93] W.-L. Hsue and W.-C. Chang, "Real discrete fractional fourier, hartley, generalized fourier and generalized hartley transforms with many parameters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, pp. 2594–2605, 2015.

[94] G. Bongiovanni, P. Corsini, and G. Frosini, "One-dimensional and two-dimensional generalised discrete fourier transforms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, pp. 97–99, 1976.

[95] R. Daher and O. Tyr, "Weighted approximation for the generalized discrete fourier–jacobi transform on space math," *Journal of Pseudo-Differential Operators and Applications*, vol. 11, pp. 1685–1697, 2020.

[96] T. Murakami and Y. Ishida, "Generalized sliding discrete fourier transform," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 99, pp. 338–345, 2016.

[97] J. J. Soltis, "Elliptically generalized discrete fourier transform with applications," in *Intelligent Robots and Computer Vision XVI: Algorithms, Techniques, Active Vision, and Materials Handling*, vol. 3208. International Society for Optics and Photonics, 1997, pp. 65–73.

[98] S. W. Bergen, "A design method for cosine-modulated filter banks using weighted constrained-least-squares filters," *Digital Signal Processing*, vol. 18, pp. 282–290, 2008.

[99] X.-Q. Gao, Z.-Y. He, and X.-G. Xia, "The theory and implementation of arbitrary-length linear-phase cosine-modulated filter bank," *Signal Processing*, vol. 80, pp. 889–896, 2000.

[100] J. Kliewer, T. Karp, and A. Mertins, "Processing arbitrary-length signals with linear-phase cosine-modulated filter banks," *Signal Processing*, vol. 8, pp. 1515–1533, 2000.

[101] X. Chen, X. Hu, H. Zhou, and N. Xu, "Fxpnet: Training a deep convolutional neural network in fixed-point representation," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2494–2501.

[102] F. Cabello, J. León, Y. Iano, and R. Arthur, "Implementation of a fixed-point 2D Gaussian Filter for Image Processing based on FPGA," in *Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*. IEEE, 2015, pp. 28–33.

[103] T. I. Karimov, D. N. Butusov, V. S. Andreev, V. G. Rybin, and D. I. Kaplun, "Compact Fixed-Point Filter Implementation," in *Conference of Open Innovations Association (FRUCT)*. IEEE, 2018, pp. 73–78.

[104] D. Mukherjee and S. Mukhopadhyay, "Fast hardware architecture for fixed-point 2D Gaussian filter," *International Journal of Electronics and Communications*, vol. 105, pp. 98–105, 2019.

[105] Z. Yu, M.-L. Yu, K. Azadet, and A. Willson, "The use of reduced two's-complement representation in low-power DSP design," in *Circuits and Systems (ISCAS)*. IEEE, 2002, pp. 78–80.

[106] Z. Yu, M.-L. Yu, K. Azadet, and A. N. Willson, "A low power adaptive filter using dynamic reduced 2's-complement representation," in *Custom Integrated Circuits Conference (CICC)*. IEEE, 2002, pp. 141–144.

[107] O. Salomon, J.-M. Green, and H. Klar, "General algorithms for a simplified addition of 2's complement numbers," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 839–844, 1995.

[108] R. Zimmermann, "Efficient VLSI implementation of modulo $2^n + 1$ addition and multiplication," in *Computer Arithmetic (ARITH)*. IEEE, 1999, pp. 158–167.

[109] W. Wang, M. Swamy, and M. O. Ahmad, "Moduli selection in RNS for efficient VLSI implementation," in *Circuits and Systems (ISCAS)*. IEEE, 2003, pp. 241–258.

[110] H. T. Vergos, C. Efstathiou, and D. Nikolos, "High speed parallel-prefix modulo $2n+1$ adders for diminished-one operands," in *Computer Arithmetic (ARITH)*. IEEE, 2001, pp. 211–217.

[111] V. Indumathi and P. Nagaraju, "Design of Radix-4 Signed Digit Encoding for Pre-Encoded Multipliers Using Verilog." *International Journal of Engineering and Techniques*, vol. 4, pp. 351–356, 2018.

[112] S. Motla, "FPGA Implementation of Pre Encoded Multipliers," Ph.D. dissertation, Thapar University Punjab, 2017.

[113] R. Keote, P. Karule, and V. Nimgade, "An efficient design of redundant binary modified partial product generator," in *Conference on Advances in Signal Processing (CASP)*. IEEE, 2016, pp. 446–449.

[114] R. Prathiba, P. Sandhya, and R. Varun, "Design of high performance and low power multiplier using modified booth encoder," in *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE, 2016, pp. 794–798.

[115] E. B. Krishna and T. Reshma, "Area And Power Efficient of PreEncoded Multipliers using NR4SD," *International Journal of VLSI system, Design and Communication Systems*, vol. 5, pp. 1054–1058, 2017.

[116] S. B. G. Mala and P. Soundarya, "Design of Fast and Low Area Pre-Encoded Multipliers Based on NR8SD Encoding technique for DSP/Multimedia applications," *International Journal of Research*, vol. 5, pp. 2273–2283, 2018.

[117] D. Husinga, "Design of optimized filters using CSD coefficient representation," Ph.D. dissertation, University of California, Davis, 1996.

[118] E. Grosswald, *Topics from the Theory of Numbers*. Springer Science & Business Media, 2010.

[119] A. R. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation*. World Scientific, 2007.

[120] K. Srinivasa Reddy and S. K. Sahoo, "An approach for fixed coefficient RNS-based FIR filter," *International Journal of Electronics*, vol. 104, pp. 1358–1376, 2017.

[121] K. V. S. Bindiya, TS and E. Elias, "Design of low power and low complexity multiplier-less reconfigurable non-uniform channel filter using genetic algorithm," *Global Journal of Reseach Engineering*, vol. 12, pp. 7–19, 2012.

[122] T. Bindiya and E. Elias, "Modified metaheuristic algorithms for the optimal design of multiplier-less non-uniform channel filters," *Circuits, Systems, and Signal Processing*, vol. 33, pp. 815–837, 2014.

[123] S. Andraos and H. Ahmad, "A new efficient memoryless residue to binary converter," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1441–1444, 1988.

[124] A. S. Molahosseini and K. Navi, "New arithmetic residue to binary converters," *International Journal of Computer Science and Engineering*, vol. 1, p. 296, 2007.

[125] A. B. Premkumar, "An RNS to binary converter in a three moduli set with common factors," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, pp. 298–301, 1995.

[126] A. B. Premkumar, M. Bhardwaj, and T. Srikanthan, "High-speed and low-cost reverse converters for the $(2n-1, 2n, 2n+1)$ moduli set," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, pp. 903–908, 1998.

[127] C. H. Vun, A. B. Premkumar, and W. Zhang, "A new RNS based DA approach for inner product computation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, pp. 2139–2152, 2013.

[128] D. Živaljević, N. Stamenković, and V. Stojanović, "Digital filter implementation based on the RNS with diminished-1 encoded channel," in *Telecommunications and Signal Processing (TSP)*. IEEE, 2012, pp. 662–666.

[129] H. T. Vergos and C. Efstathiou, "A unifying approach for weighted and diminished-1 modulo $2n+1$ addition," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, pp. 1041–1045, 2008.

[130] R. Singh and R. Mishra, "Design and Simulation of Diminished-One Modulo $2n+1$ Adder Using Circular Carry Selection," in *Proceedings of the World Congress on Engineering (WCE)*. WCE, 2011, pp. 6–8.

[131] P. A. Mohan, "Evaluation of fast conversion techniques for binary-residue number systems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, pp. 1107–1109, 1998.

[132] A. Skavantzos and M. Abdallah, "Implementation issues of the two-level residue number system with pairs of conjugate moduli," *IEEE Transactions on Signal Processing*, vol. 47, pp. 826–838, 1999.

[133] R. Conway and J. Nelson, "Fast converter for 3 moduli RNS using new property of CRT," *IEEE Transactions on Computers*, vol. 48, pp. 852–860, 1999.

[134] K. A. Gbolagade, G. R. Voicu, and S. D. Cotofana, "An Efficient FPGA Design of Residue-to-Binary Converter for the Moduli Set," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 19, pp. 1500–1503, 2011.

[135] A. Hiasat, "A Residue-to-Binary Converter for the Extended Four-Moduli Set," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 25, pp. 2188–2192, 2017.

[136] P. A. Mohan, *Residue Number Systems: Algorithms and Architectures*. Springer Science & Business Media, 2012.

[137] N. S. Szabo and R. I. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*. McGraw-Hill, 1967.

[138] W. Wang, M. Swamy, and M. O. Ahmad, "Moduli selection in RNS for efficient VLSI implementation," in *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2003, pp. 245–258.

[139] K. Kaluri, W. F. Leong, K.-H. Tan, L. Johnson, and M. Soderstrand, "Comparison of RNS and optimized FIR digital filters in Xilinx FPGA's," in *Circuits and Systems Midwest Symposium (MWSCAS)*. IEEE, 2001, pp. 438–441.

[140] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Synthesis of multiplier-less FIR filters with minimum number of additions," in *International Conference on Computer Aided Design (ICCAD)*. IEEE, 1995, pp. 668–671.

[141] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, pp. 677–688, 1996.

[142] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, pp. 456–462, 1974.

[143] N. Sankarayya, K. Roy, and D. Bhattacharya, "Algorithms for low power and high speed FIR filter realization using differential coefficients," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, pp. 488–497, 1997.

[144] C.-Y. Yao, H.-H. Chen, T.-F. Lin, C.-J. Chien, and C.-T. Hsu, "A novel common-subexpression-elimination method for synthesizing fixed-point FIR filters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, pp. 2215–2221, 2004.

[145] F. Xu, C.-H. Chang, and C.-C. Jong, "Contention resolution algorithm for common subexpression elimination in digital filter design," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, pp. 695–700, 2005.

[146] D. R. Bull and D. H. Horrocks, "Realisation techniques for primitive operator infinite impulse response digital filters," in *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1993, pp. 607–610.

[147] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, pp. 569–577, 1995.

[148] A. P. Vinod, E. Lai, D. L. Maskell, and P. K. Meher, "An improved common subexpression elimination method for reducing logic operators in FIR filter implementations without increasing logic depth," *Integration*, vol. 43, pp. 124–135, 2010.

[149] I. Hatai, "A Reconfigurable digital up converter architecture and its hardware Implementation for Software Defined Radio System," Ph.D. dissertation, IIT, Kharagpur, 2017.

[150] M. Marimuthu and D. B. Rajendiran, "High speed constant multiplier design based on VHBCSE algorithm with parallel prefix adders," *International Journal of Advanced Science and Engineering Research*, vol. 3, pp. 554–561, 2018.

[151] D. Stevenson, "A proposed standard for binary floating-point arithmetic," *Computer*, vol. 14, pp. 51–62, 1981.

[152] A. Kaivani and S. Ko, "Floating-point butterfly architecture based on binary signed-digit representation," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 24, pp. 1208–1211, 2015.

[153] J. Eilert, A. Ehliar, and D. Liu, "Using low precision floating point numbers to reduce memory cost for MP3 decoding," in *IEEE Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2004, pp. 119–122.

[154] P. Lindstrom, S. Lloyd, and J. Hittinger, "Universal coding of the reals: alternatives to IEEE floating point," in *Proceedings of the Conference for Next Generation Arithmetic (CoNGA)*. ACM, 2018, pp. 1–14.

[155] M. Pietras, "Error analysis in the hardware neural networks applications using reduced floating-point numbers representation," in *American Institute of Physics conference Proceedings (AIP)*. AIP Publishing LLC, 2015, p. 660005.

[156] S. Badave and A. Bhalchandra, "Critical Path Reduction of Distributed Arithmetic Based FIR Filter," *International Journal of Advanced Computer Science and Applications*, vol. 7, pp. 71–78, 2016.

[157] S. Singh and S. Saini, "Performance Analysis of FIR Filter Using Distributed Arithmetic," *Global Journal of Computers and Technology*, vol. 5, pp. 310–313, 2017.

[158] N. S. Naik and K. A. Gupta, "An efficient reconfigurable FIR digital filter using modified distribute arithmetic technique," *International Journal of Emerging Technology and Advanced Engineering*, vol. 6, pp. 152–156, 2017.

[159] H. Yoo and D. V. Anderson, "Hardware-efficient distributed arithmetic architecture for high-order digital filters," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2005, pp. 5–125.

[160] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE Transactions on Acoustics, Speech, and Signal Processing Magazine*, vol. 6, pp. 4–19, 1989.

[161] C.-C. Chen, T.-J. Lin, C.-W. Liu, and C.-W. Jen, "Complexity-aware design of a DA-based FIR filters," in *Asia-Pacific Conference on Circuits and Systems (APCCS)*. IEEE, 2004, pp. 445–448.

[162] P. Longa and A. Miri, "Area-efficient FIR filter design on FPGAs using distributed arithmetic," in *International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2006, pp. 248–252.

[163] A. M. M.Iruleswari, M.Sheela Merlin, "Design and Implementation of distributed arithmetic technique based FIR filter using lookup table," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 5, pp. 3480–3488, 2016.

[164] K. J. Roy and R. Ramya, "Low-Power And Low-Area Adaptive FIR Filter Based on Distributed Arithmetic and LMS Algorithm," *International Journal of Scientific and Research Publications*, vol. 4, pp. 1–5, 2014.

[165] D. Kalaiyarasi and T. K. Reddy, "Design and implementation of least mean square adaptive FIR filter using offset binary coding based distributed arithmetic," *Microprocessors and Microsystems*, vol. 71, p. 102884, 2019.

[166] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, pp. 95–99, 1988.

[167] A. Lee, M. Ahmadi, G. Jullien, W. Miller, and R. Lashkari, "Digital filter design using genetic algorithm," in *IEEE Symposium on Advances in Digital Filtering and Signal Processing (IADFSP)*. IEEE, 1998, pp. 34–38.

[168] Z. B. GARIP and A. F. BOZ, "The FIR Filter Design based on Genetic Algorithm," *Balkan Journal of Electrical and Computer Engineering*, vol. 6, pp. 33–36, 2018.

[169] J. E. Cousseau, S. Werner, and P. D. Donate, "Factorized all-pass based IIR adaptive notch filters," *IEEE Transactions on Signal Processing*, vol. 55, pp. 5225–5236, 2007.

[170] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks (ICNN)*. IEEE, 1995, pp. 1942–1948.

[171] F. Serbet, T. Kaya, and M. Ozdemir, "Design of digital IIR filter using Particle Swarm Optimization," in *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2017, pp. 202–204.

[172] I. Sharma, A. Kumar, L. Balyan, and G. K. Singh, "A new hybrid CSE technique for multiplier-less FIR filter," in *International Conference on Digital Signal Processing (DSP)*. IEEE, 2017, pp. 1–5.

[173] M. M. Gowda, M. Namratha, G. Hussain, D. Chandan, and G. Parameshappa, "Design of Digital FIR Low Pass Filter using PSO Algorithm," *International Journal of Scientific Research and Engineering Development*, vol. 3, pp. 1041–1045, 2020.

[174] W. B. Ye, X. Lou, and Y. J. Yu, "Design of low-power multiplierless linear-phase FIR filters," *IEEE Access Scientific Journal*, vol. 5, pp. 23 466–23 472, 2017.

[175] A. J. Yousif, G. J. Ahmed, and A. S. Abbood, "Design of Linear Phase High Pass FIR Filter using Weight Improved Particle Swarm Optimization," *International Journal of Advanced Computer Science and Applications*, vol. 9, pp. 270–275, 2018.

[176] J. Dash, B. Dam, and R. Swain, "Optimal design of linear phase multi-band stop filters using improved cuckoo search particle swarm optimization," *Applied Soft Computing*, vol. 52, pp. 435–445, 2017.

[177] P. Kaur and M. Kaur, "Performance of Optimal Fractional Delay-IIR Filter Using Evolutionary Algorithms," *International Journal of Electronics Engineering Research*, vol. 9, pp. 665–675, 2017.

[178] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *World Congress on Nature & Biologically Inspired Computing (NaBIC)*. IEEE, 2009, pp. 210–214.

[179] M. Bellanger, "On computational complexity in digital filters," in *Proc. European Conference on Circuit Theory and Design (ECCTD)*. ECCTD, 1981, pp. 58–63.

[180] L. Leibowitz, "A simplified binary arithmetic for the Fermat number transform," *IEEE Transactions on Acoustics, Speech, and Signal processing*, vol. 24, pp. 356–359, 1976.

[181] S. V and E. Elias, "Low complexity reconfigurable channelizers using non-uniform filter banks," *Computers & Electrical Engineering*, vol. 68, pp. 389–403, 2018.

[182] P. K. Devi and R. Bhuvaneshwaran, "FPGA implementation of coefficient decimated polyphase filter bank structure for multistandard communication receiver," *Journal of Theoretical and Applied Information Technology*, vol. 64, pp. 128–140, 2014.

[183] S. Cappello, G. Cardarilli, L. Di Nunzio, R. Fazzolari, M. Re, and P. Albicocco, "Flexible channel extractor for wideband systems based on polyphase filter bank," *Journal of Theoretical & Applied Information Technology*, vol. 95, pp. 1450–1560, 2017.

[184] A. Ambede, K. G. Smitha, and A. P. Vinod, "A low-complexity uniform and non-uniform digital filter bank based on an improved coefficient decimation method for multi-standard communication channelizers," *Circuits, Systems, and Signal Processing*, vol. 32, pp. 2543–2557, 2013.