NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

FPGA based Uniform Channelizer Implementation

By Fangzhou Wu

A thesis presented to the

National University of Ireland

in partial fulfilment of the requirements for the degree of

Master of Engineering Science

Department of Electronic Engineering

National University of Ireland Maynooth

March 2016

Research supervisors: Dr. Rudi Villing

Head of department: Dr. Ronan Farrell

Abstract

Channelizers are widely used in modern digital communication systems. Advanced uniform multirate channelization have been theoretically proved to be capable of reducing the computational load, with a better performance. Therefore, in this thesis, we implement these designs on a FPGA board for the sake of the comprehensive evaluation of resource usage, performance and frequency response.

The uniform filter-banks are one of the most essential unit in channelization. The Generalised Discrete Fourier Transform Modulated Filter Bank (GDFT-FB), as an important variant of basic a DFT-FB, has been implemented in FPGA and demonstrated with a better computational saving rather than traditional schemes. Moreover the oversampling version is demonstrated to have a better frequency response with an acceptable amount of extra resources. On the other hand, frequency response masking (FRM) techniques is able to reduce the number of coefficients. Therefore, the full FRM GDFT-FB and alternative narrowband FRM GDFT-FB are both implemented in FPGA platform, in order to achieve a better performance and hardware efficiency.

Declaration

I hereby declare that this thesis is my own work and has not been submitted in any form for another award at any other university or institute of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

_____ _____

Signature Date

Acknowledgements

First of all, I'd like to thank my supervisor Dr. Rudi Villing. He has a strong sense of responsibility. Thank you for your supervision, kindness and patience. I am deeply grateful for all the help I received during the course.

And I am also very thankful to my parents for continuous support, love and understanding.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

DSP (Digital Signal Processing) has been a rapidly developing aspect of modern technology and has become an indispensable part of many products that we use in our modern lives [1]. Multirate signal processing is one of the major branches of DSP. It is a technology that finds use in signal processing systems where various sub-systems with differing sample or clock rates need to be interfaced together. At other times multi-rate processing is used to reduce the computational overhead of a system. It thus confers advantages of: (1) greatly reducing the cost of hardware; (2) providing an improved implementation performance [2]. Multirate digital filters and filter-banks are two widely used applications in multirate signal processing. They are mostly used in the field of speech processing, image processing and communications [3, 4].

Polyphase filters and filter-banks are one of the outstanding channelization examples to represent multirate digital filters [5]. They offer a significant reduction in processing complexity by way of separating the input signals into several channels. Polyphase filter-bank are widely used in industry, such as in the MP3 audio format [6] and in the digital receiver analysers that are discussed in this thesis.

The theory of polyphase filter banks was formed in the 80s and has been further developed since then [7-10], to have more configurations that can be adapted to more complicated tasks [11]. However, only a few of the further developments have been realized on hardware platforms, such as on a field-programmable gate arrays (FPGA) and DSP processors.

The reason of implementing the theoretic polyphase filter-bank on an FPGA is that an FPGA has sufficient resources and a high performance that can allow the implementation of a large number of DSP algorithms very efficiently compared to a single chip processor [12]. In data flow applications, no instructions need to be fetched from memory, and not many read/write operation from memory are required since most of the data has directly

been inserted into the register, because the FPGA input samples are flowing through the programmed logic cells. In addition, FPGA architectures allow developers to exchange resources for speed by configuring more logic resources to perform parallel processing [13].

Several polyphase DFT-FB (Discrete Fourier Transform-FilterBank) FPGA implementations have already been realized [14-17]. However these FPGA DFT-FBs design the architectures from a very basic level. The designs may lack flexibility due to limited number of channels and the amount of resources that can be exchanged for speed in different computational complexity scenarios. In the case of a change in the design requirements, considerable more work could be needed to adjust the design. This could potentially hinder further complicated development based on the implementation of a DFT-FB FPGA architecture. Thus some of the complex and resource demanding algorithm components are replaced by IP cores among the designs in this thesis.

Most of the FPGA implementations are focused on the basic Discrete Fourier Transform-FilterBank (DFT-FB). There are also some further developments based on DFT-FB, which are proved to have a better DSP performance or a better hardware efficiency. However they haven't been implemented on the FPGA platform in literature. Therefore in this thesis, these DFT-FB based designs are going to be built on FPGA, then a thorough evaluation will be evaluated to see if they are practical in terms of being useful for industry applications.

1.1 Thesis objective

The objective in this thesis is to design and implement a new set of polyphase filter-bank-based uniform channelizers on an FPGA platform. The designs will cover critically sampled GDFT-FBs (Generalized DFT modulated Filter Bank), oversampled GDFT-FBs, GDFT-FBs with full FRM technology and GDFT-FBs with narrow-band GDFT-FBs. Every type of filter contains both odd and even stacked channel allocation configurations. Most of the new designs in this thesis take advantage of Xilinx IP cores to simplify and boost the development, and ensure that other larger DSP FPGA designs based on these uniform channelizers are more convenient. We will also discuss the implementation of

the theory that has been proposed already, along with the problems that developers face when realizing the FPGA channelizers and possible solutions to these. Later sections in this thesis will cover the possibility for optimizations in the architecture for better efficiency, along with a performance and results' analysis, an assessment of the hardware resource usage, and an evaluation of whether the design is or is not feasible with current communication standards.

1.2 Thesis contributions

In this thesis, several kinds of polyphase filters and new filter-bank designs based on the polyphase filter-bank will be implemented on an FPGA to evaluate their performance, system complexity, resource usage and their feasibility for industrial scenarios.

Developers are now willing to take advantage of pre-built blocks e.g. IP cores, as the complexity of modern digital systems increases at a remarkable speed that is driven nowadays by the challenging pressures of time-to-market. This is one of the design reuse methodologies [18]. These IP cores give a great convenience when designing or tuning a new FPGA architecture. Complicated processing elements can be designed to either process samples in parallel in order to have an improved processing speed, or process then serially for an efficient resource usage. The works presented in this thesis replace the FIR filters and FFT elements with IP cores of a basic DFT-FB, and introduce further developed polyphase filter-banks (GDFT-FB, oversampled GDFT-FB, full FRM GDFT-FB and alternative narrowband FRM GDFT-FB) implementation by using IP cores, in order to have a significant reduction in the cost of design.

GDFT-FB is a generalized version of DFT-FB, it allows the polyphase filter-bank to have more configurations such as facilitating phase shifting and adjustable centre frequency. This design flexibility leads to an odd-stacked channel allocation filter-bank that has a better spectrum usage. Therefore, the FPGA implementation of GDFT-FB is presented in this thesis. A new designed complex FIR block which can filter samples with complex coefficients is also introduced and explained in detail, which is the essential part of a GDFT-FB.

GDFT-FB also provides the option to design the polyphase filter-bank in an oversampled configuration. This allows a better reconstruction of signals by reducing the aliasing problem between adjacent channels. Therefore the oversampled GDFT-FB FPGA implementation is presented in this work. A mathematical equivalent of a sample distribution model as a theoretical expression and diagram are developed for the FPGA architecture in order to fit IP cores in an oversampled configuration. Furthermore, an oversampled odd-stacked channel allocation GDFT-FB design is also implemented on the FPGA to have a better spectrum usage.

Apart from the polyphase filter-bank, FRM (Frequency Response Masking) filter technology also provides a significant computational reduction to produce an equivalent set of filtering results. The goal of combining into an FPGA design the FRM with the polyphase filter-bank will further reduce the number of coefficients required, and so the even and odd stacked channel allocations can be easily achieved. Thus, eventually a very efficient oversampled GDFT-FB with narrow-band FRM technology design in FPGA (both even and odd stacked) has been realized in this work.

Besides the high level FPGA implementation, some of adjustments and tweaks to fit IP cores to the new development of filter-banks, and hardware optimization to some of the models is also introduced in each design according to the type of the filter-bank.

Then, evaluations are performed of all the new developed filter banks, in a small scale of 16 channels. Evaluation includes frequency response, EVM (Error Vector Measurement), adjacent channel interference and hardware resources, in order to test their accuracy, performance and hardware efficiency.

Lastly, another similar evaluation of 256-channel configuration are performed to critically sampled DFT-FB/GDFT-FB and alternative narrowband FRM DFT-FB/GDFT-FB, in order to find which filter-bank technology is the most feasible and capable regard to industry scenario, because they have fairly good performance and yet with efficient hardware usage.

The summary of contributions of this thesis is as following:

- Replacing FIR filters and FFT elements with pre-built IP cores in the basic DFT-FB design, and introduce further developments of GDFT-FB, oversampled GDFT-FB, full FRM GDFT-FB and alternative narrowband FRM GDFT-FB implementations by using IP cores, in order to have a significant reduction in the cost of design.

- Introducing the FPGA design of generalized version of DFT-FB, i.e. GDFT-FB, which leads a better spectrum usage with the odd-stacked channel allocation.

- New designed oversampled GDFT-FB (in both even and odd stacked channel allocation) has also been designed in FPGA, in order to have a better signal reconstruction by reducing the aliasing between adjacent channels.

- Introducing the FPGA designs of two polyphase filter-banks design combined with FRM filter technology -- Full FRM GDFT-FB and narrow-band FRM GDFT-FB, to have a further efficiency in terms of hardware usage.

- The evaluation with 16-channel configuration are performed of all the new designed filter banks in order to test their accuracy, performance and hardware efficiency. Then the evaluation with 156 configuration are performed to critically sampled DFT-FB/GDFT-FB and narrowband FRM DFT-FB/GDFT-FB, in order to find the most feasible and capable filter-bank technology regard to industry scenario.

1.3 Thesis outline

The rest of the chapters are organized as follows:

Chapter 2 introduces the literature on which the work of this thesis is based. Moreover, an overview of FPGA architecture is presented, along with some material that may be needed to support the discussion of work in this thesis. Furthermore, the concept of polyphase filters is presented, as this is the base of the new design of GDFT-FB, oversampled GDFT-FB, and GDFT-FB applied with FRM.

Chapter 3 mainly presents the implementation of odd-stacked GDFT-FB that was developed from DFT-FB. Moreover, a new method is introduced to deal with the complex coefficients in FPGA design, as the coefficient has been complex modulated. Furthermore, some tweaks and an additional design which helps in adapting the FPGA design to fit the odd-stack configuration and complex operations are presented as well. Finally, there is test of the accuracy of the GDFT-FB design and its hardware usage, through a simulation test on a 16-channel FPGA architecture.

Chapter 4 applies the oversampled design to the polyphase filter-banks in order to have a better recovery of the input signal. The oversampled filter-bank FPGA realization depends on a parallel FIR Filter design. The oversampled configuration has been applied to the odd-stack GDFT-FB as well, which brings further design complexity. The performance and hardware usage is also tested on a FPGA simulation with a 16 channel configuration.

Chapter 5 introduces a computational saving FRM (Frequency Response Masking) technology and a combinational design with a polyphase filter-bank has been implemented on an FPGA. The FRM's two path structure leads to two GDFT-FB filter-banks in the system. The specially designed FIFO can handle the sample rate change due to the 2 stage structure of FRM. Additionally, the complex design allows the FPGA architecture to cope with odd-stack channel allocation. Finally, a very efficient oversampled alternative narrowband FRM GDFT-FB is introduced and developed on the FPGA. The FPGA simulation test is carried out again with both of the new designs that include the FRM for a configuration of 16 channels.

Chapter 6 shows the evaluation and comparison of the new filter-banks designs for a large number of channels. The resource usage, frequency response, EVM (Error Vector Measurement) and adjacent channel interference are the key specifications for analysis. Additionally, this chapter shows the advantages, drawbacks and practicality of these designs.

Chapter 7 gives the summary of the thesis, and point out several future works that can be developed based on the work presented. Finally, the conclusions of the work is presented.

Chapter 2

Background

2.1 Field Programmable Gate Array

2.1.1 Introduction

Field Programmable Gate Array (FPGA) technology continues to advance rapidly since its invention by Xilinx in 1984. The worldwide market of FPGA is anticipated to be 9.8 billion dollars by 2020 [19]. Today FPGAs have become so popular, that in many areas they have replaced custom ASICs (Application Specific Integrated Circuits) and processors in the field of signal processing.

From the most basic point of view, FPGAs are reprogrammable silicon chips. It is an alternative physical architecture to implement digital logic in systems. By using prebuilt logic blocks, the prefabricated silicon chips can be programmed electrically to implement any custom digital hardware functionality by the developer or user. The design is developed in software on a computer, and then compiled to a configuration file that contains the connections of how the components are wired together. In addition, FPGAs can be reconfigured multiple times. The FPGAs are usually programmed and configured using HDL (Hardware Description Languages), such as Verilog and VHDL, like that used for ASICs.

FPGAs not only provide a lot of flexibility to the digital system design, but also give high speed and increased reliability. Unlike processors, FPGAs have a purely parallel processing architecture, which can provide increased speed. Moreover, adding more functions may not affect the speed of the system [20]. Thus FPGA is preferred in a variety applications that are computing intensive - like audio processing, medical electronics and digital signal processing [21-26]. The basic FPGA architecture consists of three important

components: **programmable logic block**, **programmable interconnection** and **I/O blocks** [27]. Figure 2.1 illustrates a typical architecture of a FPGA.

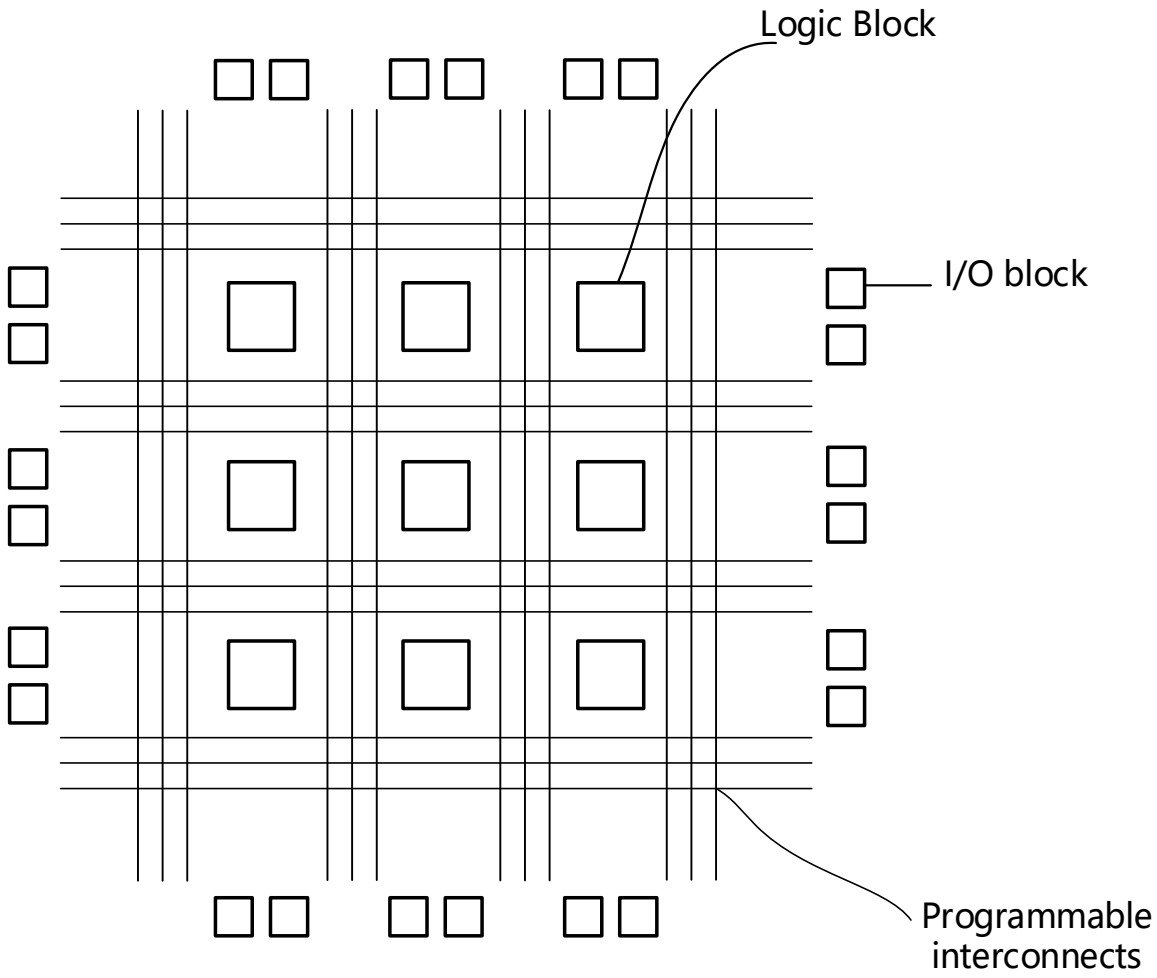

Figure 2.1 The typical internal architecture of FPGA

- **Programmable logic block**

  The programmable logic blocks are aimed to provide the basic functions and storages recourses to the digital system. The FPGA logic blocks are normally based on the combination of transistor pairs called *slices*, which contain basic logic gates like AND or XOR, multiplexers, look-up tables (LUTs) and wide-fanin AND-OR structure. Some modern FPGAs contain a more complex mixture of different of logics which can be used to do certain functions, like multipliers or multiplexers.

- **Programmable interconnection**

  The purpose of the programmable interconnection of a FPGA is to make connections among the logic blocks and I/O blocks to match the user defined in the design. It uses various lengths of wire segments to interconnect through electrically programmable switches. Wire segments may consist of multiplexers, pass transistors and tri-state buffers to form the desired connections.

- **I/O blocks**

  The components of FPGAs, such as logic blocks, require to have interaction with external components off the FPGA chip through the interface called I/O blocks. The I/O blocks are located around the boundary of the FPGA architecture. They play important roles, and occupy about 40% of the FPGA area. Normally they consist of an input buffer and an output buffer with three states, controlled by pull-up and pull-down resistors.

In recent years, further development has been carried out using the commercial FPGA architecture. Block RAMs, DSP48s, multipliers and other special function blocks are embedded into the FPGA chips for some high frequency or multiplications needed scenarios, such as high speed digital signal processing.

2.1.1.1 LUT

Much of the logic in programmable logic block is built up with Look Up Tables (LUTs) by using a small amount of Random Access Memory (RAM). A LUT is basically a table that can determine the output from any given set of inputs. It works just as a truth table in terms of combinational logic. The truth table is a pre-defined output list for any input combinations. Thus no matter how complicated the combinational logics in a FPGA design are, LUTs can implement them with a small amount of resources. Figure 2.2 illustrates an architecture of a 4-input LUT in a programmable logic block.

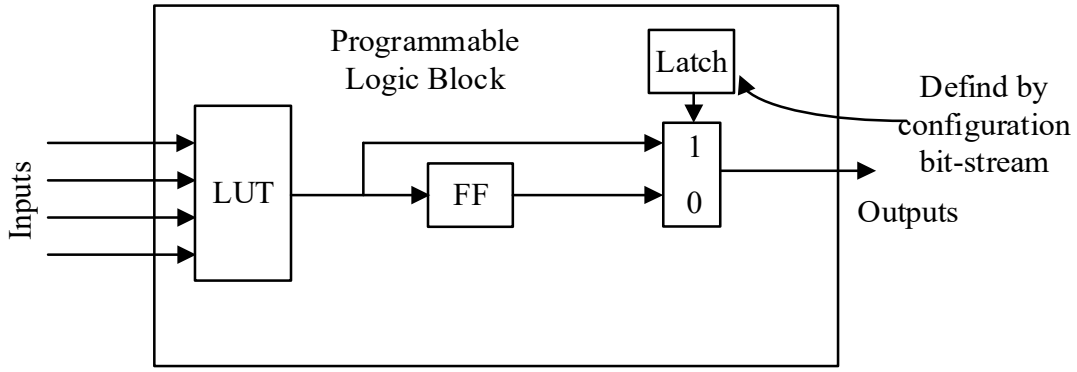

Figure 2.2 FPGA Programmable Logic Block

2.1.1.2 Block RAM

A block RAM is a dedicated two port memory containing Kbs of RAM and can't be used to implement digital logics. It is the RAM embedded throughout the FPGA for data storage. Xilinx FPGA consists of 2 columns of block RAM. Dual-port allows separate reading and writing. It can also be configured to divide the memory into different width sizes: 1x36Kb or 2x18Kb. Block RAM is excellent for First-In/First-Out (FIFO) implementation. Larger memory blocks can be obtained by cascading multiple block RAMs. The maximum word-length data path a block RAM can handle is 18 bits.

2.1.1.3 DSP48s

Modern FPGA architectures have been further developed to increase the speed of multiplication, addition and other operations highly needed for DSP [28]. The Xilinx Virtex-6 family FPGA board has been brought out along with slice embedded in it. The basic structure and procedure of this piece of slice is called Multiply Accumulate (MAC) function, and it is widely used to implement DSP processing in hardware. For example, the DSP48s slice includes adder, subtractors, accumulator and coefficient register, which provide high power efficiency and high performance. Each DSP48s slice is equivalent to more than 500 programmable logic blocks, only consumes about $1/10^{th}$ of the power of the equivalent logic hardware design, and runs up to 600 MHz. In addition, the new added pre-adder in Virtex-6 board can be very useful in symmetric FIR filtering and other particular operations [29].

2.1.2 Verilog HDL

Verilog is one of the two most widely used HDLs (Hardware Description Language) used by integrated designers all over the world. The other is called VHDL.

HDL allows developers simulate their FPGA designs earlier in the development of the product, in order to debug and test designs. Architectures designed in HDL are easy to programme and verify. In addition, HDL is normally more readable compared to schematics, especially for huge scale circuits. [30]

Developers can programme their FPGA modules at 4 levels of design: (1) Algorithmic level, such as if, case and loop statement; (2) RTL (Register-Transfer level) level, to connect registers with Boolean equations; (3) Gate level, to have combinational logic with logic gates like OR and XOR; (4) Switch level, to design the transistor inside switches.

Verilog is also able to define the architecture to control the inputs and outputs of a simulation.

2.1.3 Xilinx IP Cores

Conventional FPGA design would involve the user to manually write all the design code. It may not be the most practical way for producing the best performance for FPGA design. When writing the code for FIR or FFT algorithm manually in HDL, it can cost a lot of time, and it is also harder to verify them. Thus, Xilinx and other FPGA manufacturer provided IP (Intellectual Property) cores to simplify the design procedure.

IP cores are presented with a GUI (Graphic User Interface) and offer a parameterized tool which lets the developer choose and customize certain designs. This gives the developer a greater flexibility and reusability. Furthermore, this tool also have other advantages such as reducing the design risk, less errors, faster and better compiling, more efficient resources usage and better results of the design. Xilinx IP Cores cover a wide field of

designs in the field of DSP, like FIR filters, FFT and shift registers, which play important roles in designs described in this thesis.

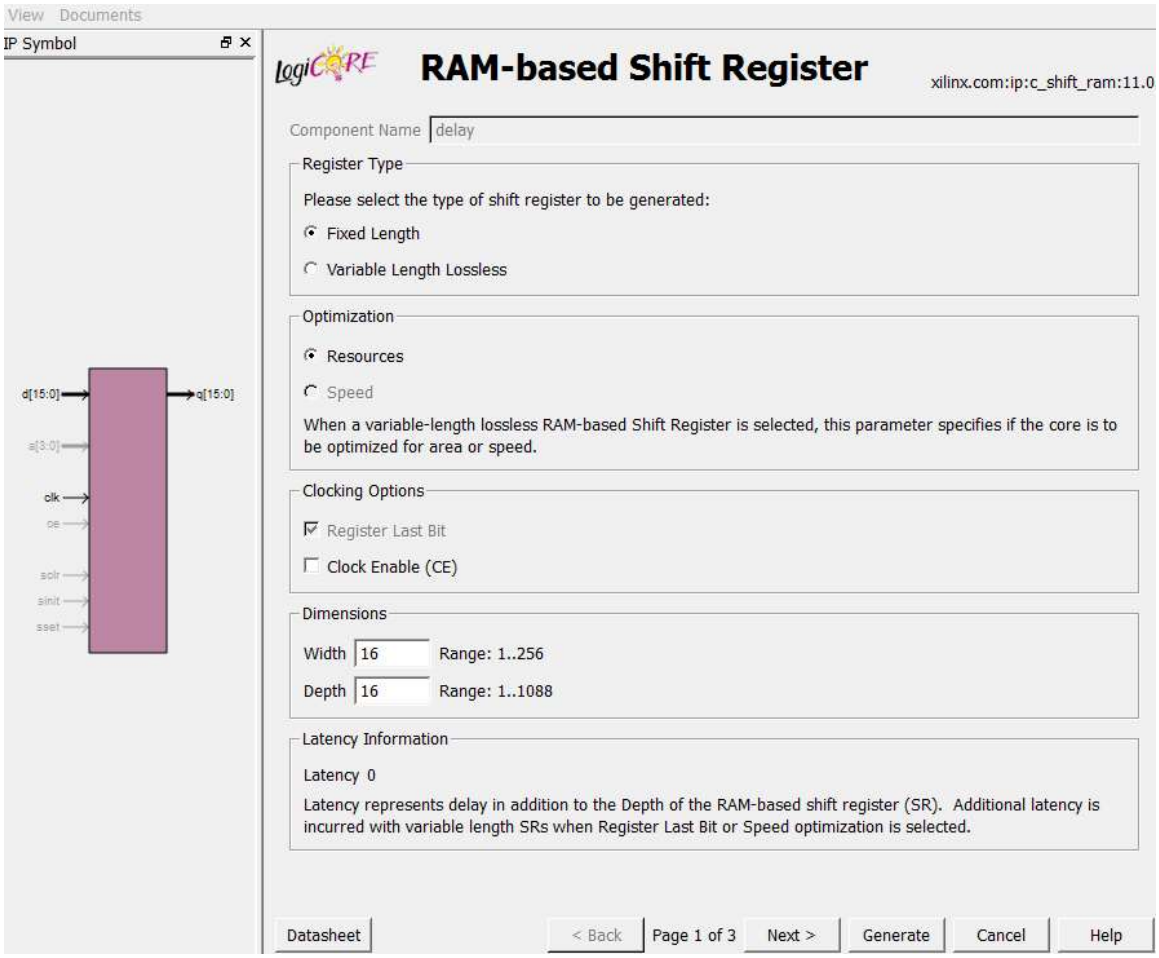

Figure 2.3 Xilinx IP core GUI (Xilinx ISE 14.3)

Figure 2.3 illustrates the example of GUI form a Xilinx IP core. The parameterize factors can determine the control pins, data formation, optimization methods and some other configurations.

2.1.4 Fixed point DSP

Digital signal processing can be separated into two categories – fixed point and floating point [31]. These refer to the format used to store the data in the devices. For a common 16 bits fixed point application, there are up to 65,536 possible bit patterns (2^{16}). Signed fixed point value can use two's compliment to make the value include negative numbers [32]. For a common 32 bits floating point application, there are more bit patterns than

fixed point, which is 2^{32} to be exact. A key feature of floating point presentation is that the numbers are not uniformly spaced [33].

Normally fixed point arithmetic is much faster than floating point in general purpose computers. The internal architecture of the floating point hardware is more complex than the fixed point hardware [31] as all the register and data should be 32 bits word length instead of 16, and all the multipliers and ALU must be able to process floating point arithmetic very fast. As a result, floating point has a better precision and higher dynamic range than fixed point but of greater size and thus cost. Fixed point dividers are usually cheaper than floating point devices.

In terms of performance, the biggest difference between the fixed point processing result and floating point processing result is SNR (Signal-to-Noise Ratio). When storing a 16-bit fixed point value, the original number must be round up or down to its adjacent neighbour by a maximum of half of the gap size. Every time we round a number to fixed point presentation, noise will be added to the signal. Fixed point's rounding noise is much worse than floating point, because the gap between adjacent numbers is much larger than floating point. Usually fixed point has about 3000 times more quantization noise than floating point [31]. Thus these quantization error is a very important criteria in verifying the fixed point in future design.

2.2 Channelization technology

Channelization is part of a digital signal processing that divides the wideband into separate channels, and down converts them to baseband, extracting one or more desired channels. The channels may have uniform or non-uniform allocation. Normally channelization is implemented by a down-converter and a low-pass filer [34]. Figure 2.4 illustrates a 4 channels channelizer. This wideband input signal has 4 interested channels, each of them is being filtered and down-converted to DC (baseband frequency), and is ready for following processing.

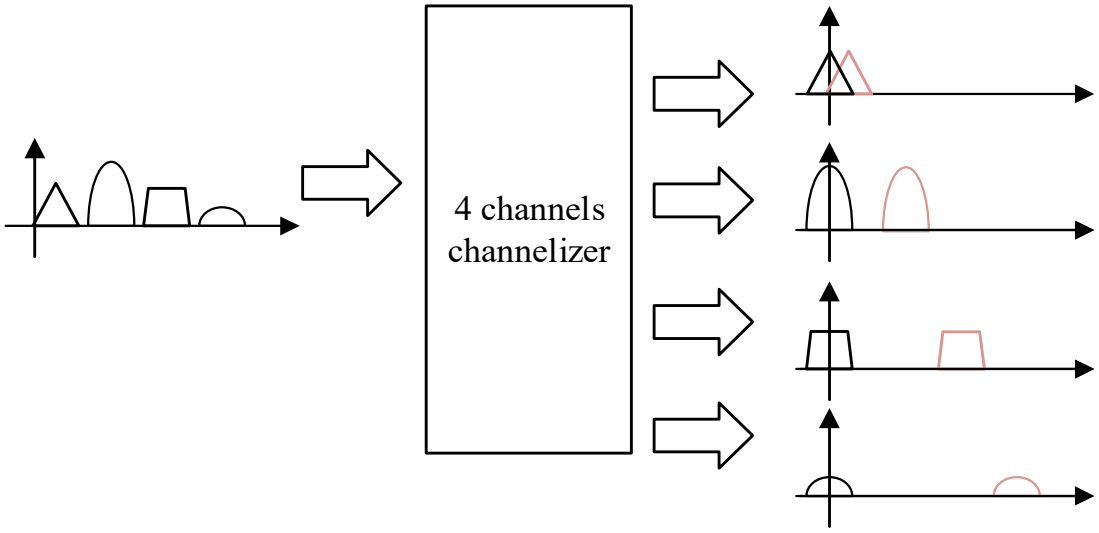

Figure 2.4 A 4 channels channelizer

2.2.1 *Single channel and multi-channel*

Channelization technology is widely applied in the industry. For a mobile base (mobile phones), normally only one channel of signal is required to process. Thus the one-channel channelizer, which contains one down-converter and low-pass (or band-pass) filter can do a good job in this scenario.

2.2.1.1 *Per-channel approach*

A base station needs to extract a large number of channels [35]. There are several ways to implement this job. The 'per-channel approach' is one of the most straightforward solutions for the multi-channel cases [34, 36, 37]. This approach operates K independent one-channel channelizers in parallel, where K is the number of channels. Each sub-channel extracts one channel of interest in the wideband input, as shown in Figure 2.5. The 'per-channel approach' provides a high level of flexibility in the choice of channels' centre frequencies and bandwidths. Channels do not have the constraints of equal bandwidth or that of a uniform allocation. However, this kind of design would need many more hardware resources and power than other efficient designs, like polyphase filter-bank. As the down-converter requires quite a lot of complex multiplications and other operations, and as every channel requires its own down-converter, then as the number of channel K increases, the system complexity greatly increases. In higher sample rate applications, the 'per-channel approach' is not a wise option to implement channelization,

as the current digital signal processer and FPGA cannot provide enough performance for this computational load.

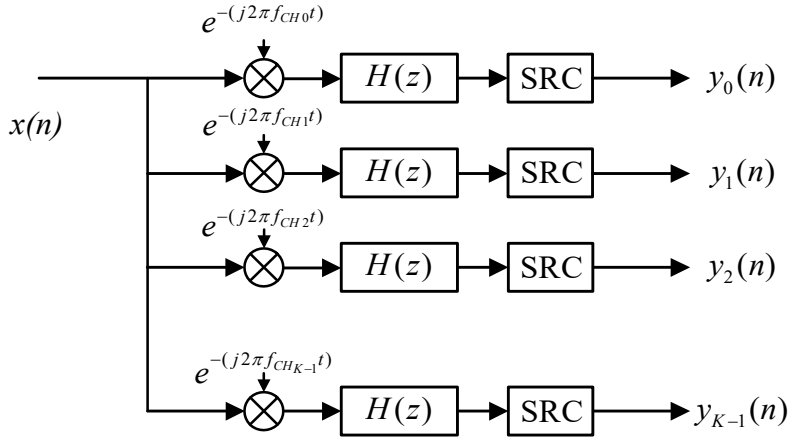

Figure 2.5 Channelizer using per-channel approach to filter channels.

2.2.1.2 Pipelined frequency transform

Another channelization technology is called pipelined frequency transform [38]. This technology occupies a structure, which contains a binary tree of DDCs (Digital Down Converters) followed by a number of SRCs (Sample Rate Converters). Every level of the tree divides the incoming wideband signal into a low frequency half and a high frequency half, and the next level divides these half bands again, until the tree's last level separates out the channels of interest [38]. This structure is also called QMF (Quadrature Mirror Filter) tree [39]. As a result, the system complexity is greatly reduced compared to the per-channel approach, because of the utilisation of the half band symmetry and sample rate reduction at each level.

The pipeline frequency transform offers a more efficient option in terms of hardware usage and power consumption compared to per-channel approach. This is especially in applications where a large number of channels are needed to be separated from the wide-band signal. However, it has weaknesses in terms of flexibility, as all the channels are required to have equal bandwidths and to be uniformly allocated. The diagram of pipeline frequency transform structure is shown in Figure 2.6

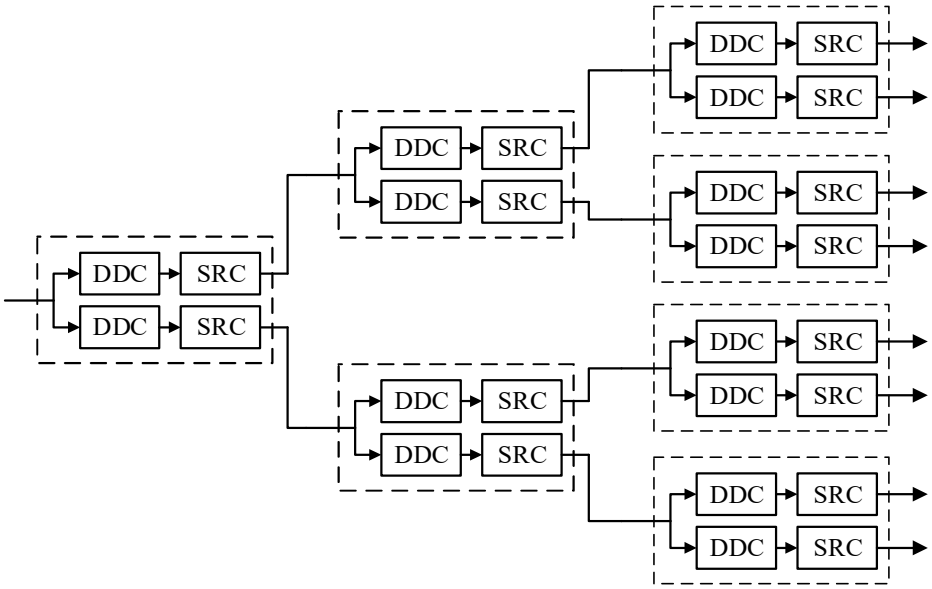

Figure 2.6 Pipeline frequency transform structure of a binary tree with DDC followed by SRC.

2.2.1.3 Polyphase filter-bank

Computational efficient channelizers have been designed by using fast Fourier Transform (FFT) [40-42]. The polyphase filter offers further improvement in terms of efficiency. The operational efficiency and design simplicity is obtained from the fact that only one low-pass filter-bank is needed to be designed, and that the remaining band-pass filters will get their properties automatically after the modulation of the prototype filter. An analysis filter-bank will divide the wide-band signal uniformly (in the even stacked allocation), such that every sub-band would have the same space from its adjacent channels in the receiver side. (Even stacked means that there is one sub-band centred at DC, as shown in Figure 3.3 (a).

The structure of a polyphase filter-bank is shown in Figure 2.7. The input wideband signal samples will first be down-sampled and sent to polyphase decomposed filters. The filtered samples are then extracted using DFT. This type of channelizer are also referred to as DFT-FB (Discrete Fourier Transform Filter-Bank). DFT-FBs require that extracted channels in the wideband signal have to be uniformly allocated, and the signal sample rate has to be an integer multiple of the sub-channel's bandwidth. This is further discussed in chapter 3 as it is the base of the new implementations in the thesis.

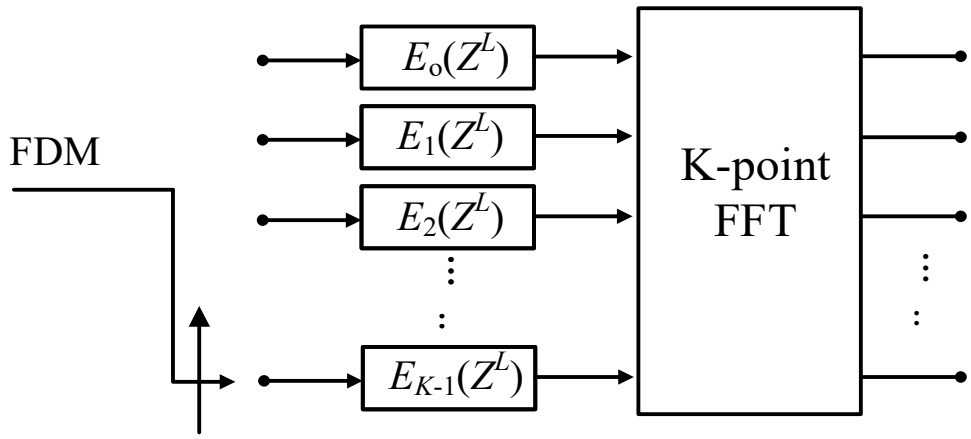

Figure 2.7 The structure of the DFT-FB where L is the oversampling factor and $E_K(z^L)$ are the polyphase components of the prototype filter $H(z)$.

2.2.2 Hardware complexity comparison

Hardware complexity comparisons between these three methods of channelization can be obtained from [17]. The detailed data is shown in Figure 2.1.

The comparisons are focused on the utilizations of LUTs and memory size. The utilization of LUTs is illustrated in Figure 2.8, and the memory size counted in bits comparison is illustrated in Figure 2.9.

Table 2.1 The hardware comparison of per-channel approach, pipeline frequency transform DFT-FB

Channelization method	Number of channels	LUTs	Memory (bits)
Per-channel approach	256	317,498	436,224
	512	650,114	876,544
	1024	1,336,754	1,761,280
Pipeline frequency transform	256	27,930	3,840
	512	32,270	6,529
	1024	36,610	10,625
DFT-FB	256	4,608	4,608
	512	4,793	4,793
	1024	5,345	5,345

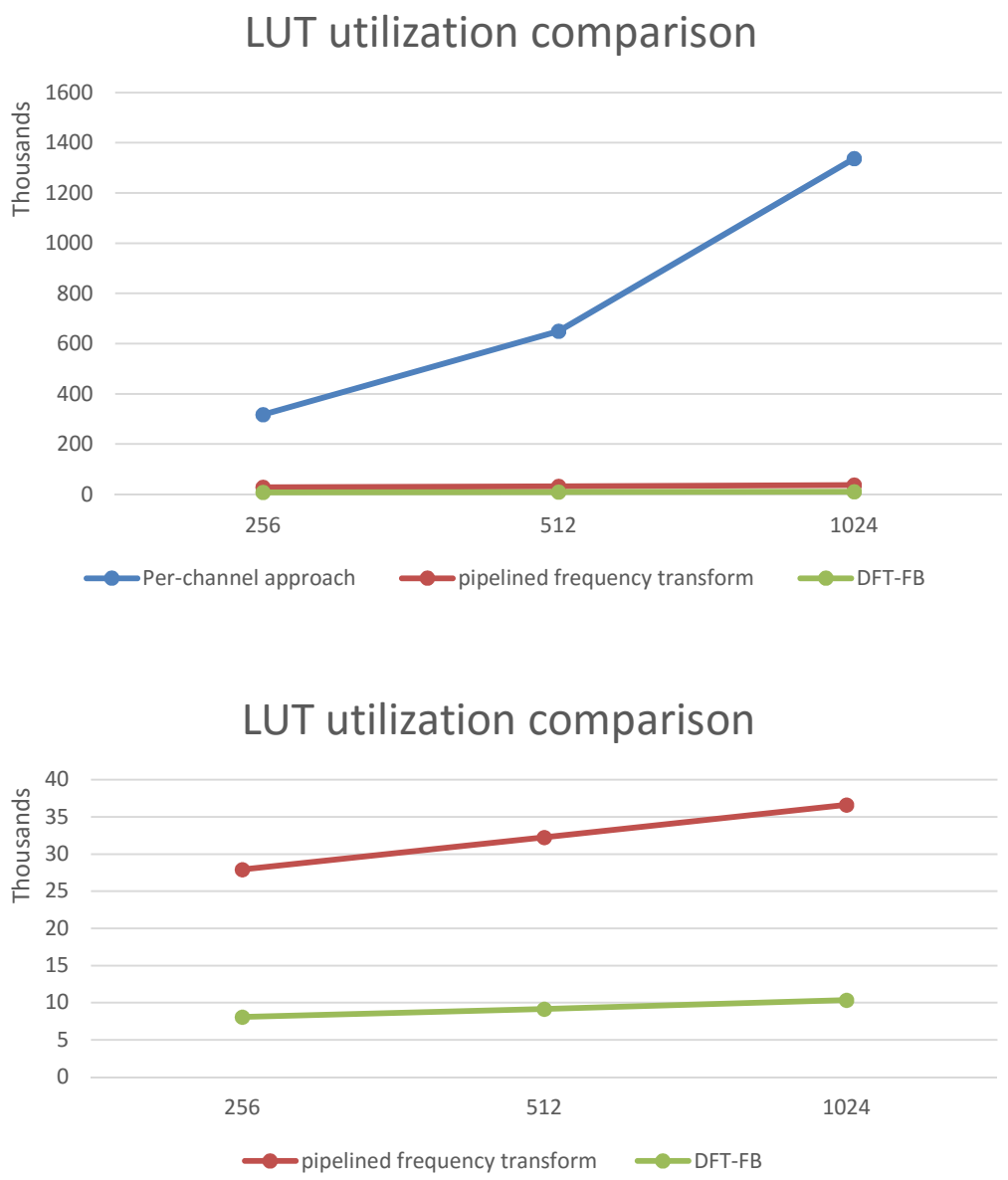

Figure 2.8 The LUT utilization comparison

In the comparison of the LUTs requirements, the Figure 2.8 shows that per-channel approach is a very inefficient channelization approach, it uses more LUT resources, which makes it difficult to compare between DFT-FB and pipelined frequency transform in the upper figure, and the situation will be much worse when the number of channels increases. From the lower plot of Figure 2.8, the clear LUT utilization comparison between DFT-FB and pipelined frequency transform shows that DFT-FB only requires about one third of the LUT resources that pipelined frequency transform uses.

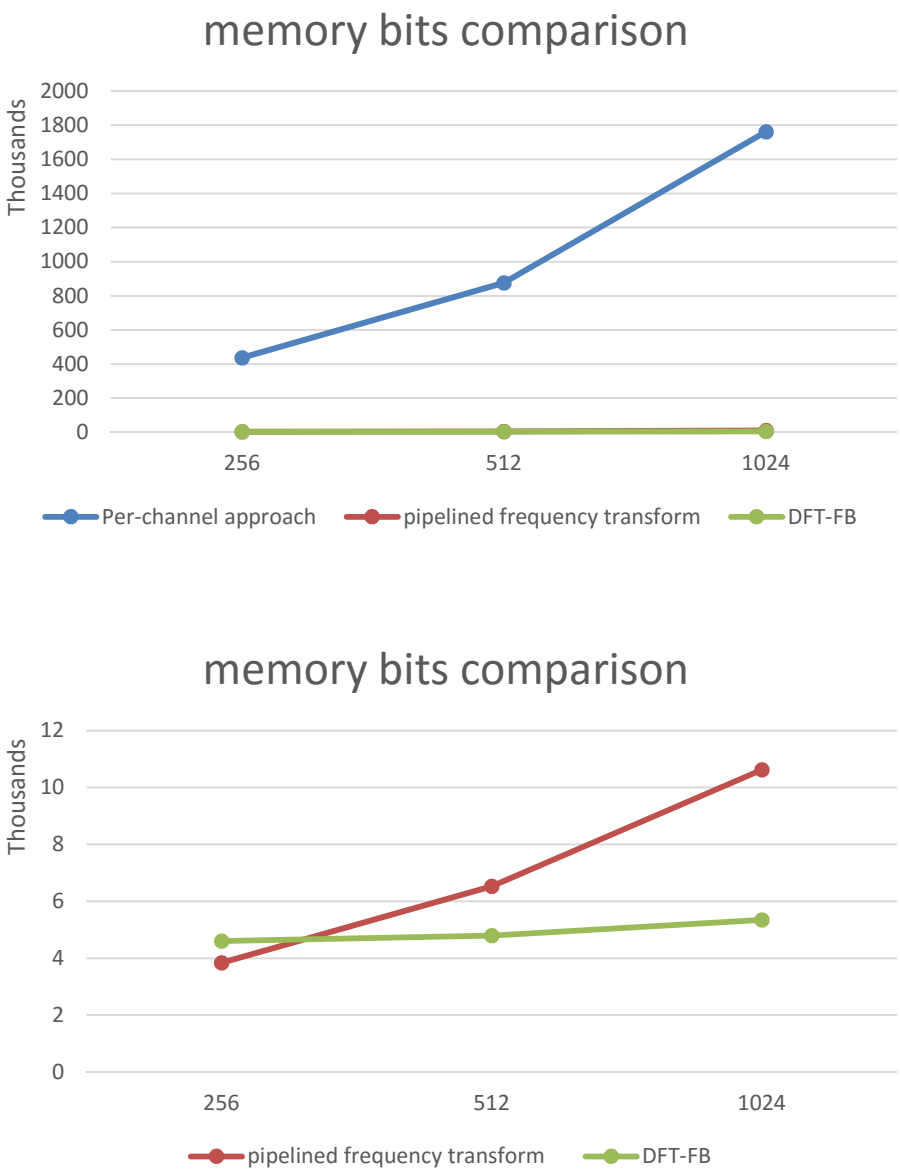

Figure 2.9 the memory bit comparison

In the memory usage comparison, a similar result to LUT resource usage is provided by the data as well. The inefficient per-channel approach still use much more memory resources than other two approaches. The memory comparison between DFT-FB and pipelined frequency transform in the lower plot of Figure 2.9 shows us that if the number of channels is smaller than 256, the pipeline frequency transform will use slightly less memory than DFT-FB; however, for applications of more than 256 channels, the DFT-FB will have a lower memory usage than pipelined frequency transform, and further efficiency will be obtained as the channel number increases.

Based on the analysis of LUTs on memory usage and that the number of channels in industrial application is normally greater than 256, DFT-FB is the preferred approach for further development.

2.2.3 Uniform Versus Non-uniform

In digital signal processing, the down-sample rate and the filter-response in a filter-bank are the same across all the channels in the wideband signal, in this case the system is considered as a uniform filter-bank. An illustration of the frequency response of uniform filter-bank is shown in Figure 2.10.

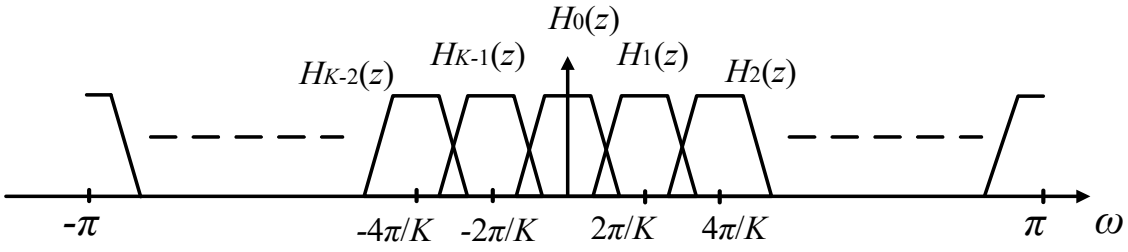

Figure 2.10 The uniform filter-bank's frequency response

There are some further developments based on DFT-FB in the literature that provide non-uniformly bandwidth channel allocation configurations. [7, 43] developed a non-uniform P-GDFT (Parallel GDFT) to achieve DFSA (Dynamic Fragment Sub-band Allocation).

The structure of P-GDFT is shown in Figure 2.11. This type of non-uniform channelizer employs multiple different bandwidth polyphase filter-banks in parallel, to process the same wideband input signal simultaneously. Every filter-bank implements a uniform channel extraction of the wideband by its own filtering specification. The channels of interest are extracted by selecting the needed outputs from each polyphase filter-bank. P-GDFT has a high ratio of non-used channels, and the process of the non-used channels cannot be avoided, because all the channels contribute to the overall computational load. This means that there is a lot of waste in terms of resources usage in the hardware implementation.

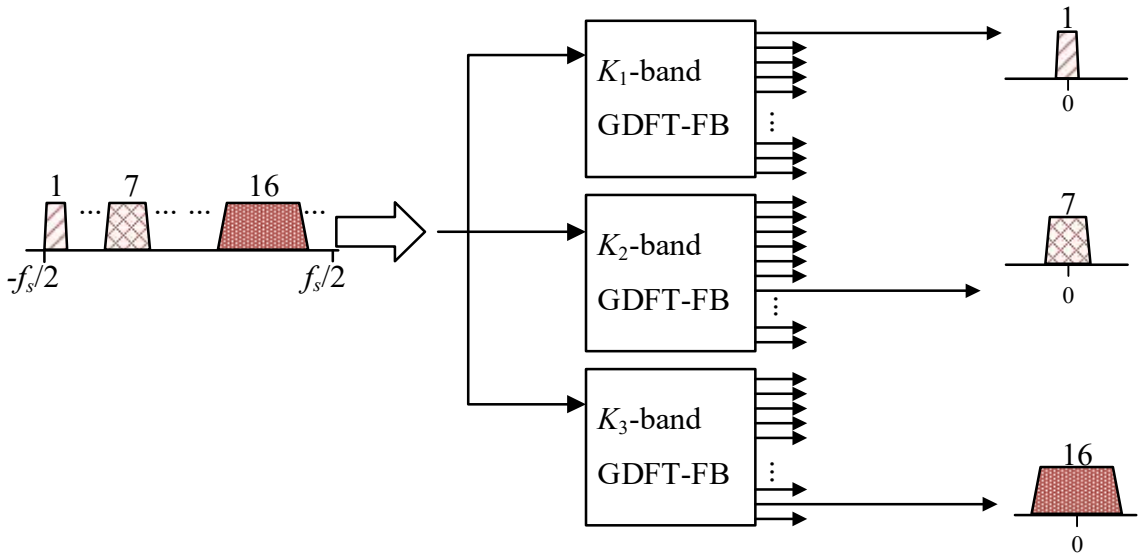

Figure 2.11 P-GDFT non-uniform structure

[7, 43] also carried out another non-uniform channelizer called R-GDFT (Recombined GDFT), as shown in Figure 2.12. The basic idea of R-GDFT is to let a polyphase filter-bank channelize the wideband signal first, and then extract the channel of interest by recombining two or more adjacent channels according to specific requirements of different standards. The bandwidth of the polyphase filter-bank is also known as *granularity band*. The smaller the granularity band is, the more options in bandwidth and centre frequencies of sub-bands.

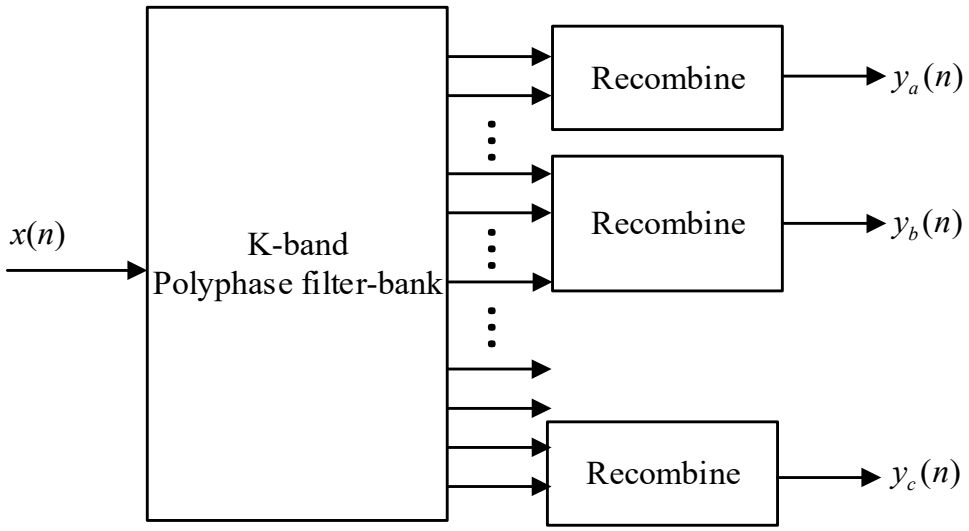

Figure 2.12 Recombined GDFT-FB channelizer

Although R-GDFT can have a better computational efficiency, its re-configurability level is lower than the P-GDFT's, because in the R-GDFT each type of channel can only be centred at the multiple of its channel spacing. In addition, to meet a new standard requirement, P-GDFT only needs partly tuning or adding one more polyphase filter-bank with the new specification.

2.3 Frequency Response Masking (FRM)

The first concept of Frequency Response Masking (FRM) technology was developed in [44], in order to reduce the complexity of designing a linear phase FIR filter with a very sharp transition band. The reduction of FIR design complexity is achieved by employing the cascading connection of an interpolated FIR filter and a FIR filter with a relaxed specification, instead of designing one FIR filter with a very restricted specification. The interpolated FIR filter is obtained by replacing the unit delay Z^{-1} with the delay Z^{-L}, where L is an integer number. In other words, put $L-1$ zeros in every adjacent coefficients of the FIR filter. After this, the FIR filter's frequency response will become periodic. Then the other following relaxed designed FIR filter will mask the duplicate images produced by the interpolated FIR filter.

A much more practical approach has been developed in [45], for applying FRM when designing a sharp linear phase digital filter in a narrow-band or an arbitrary-band. The structure is designed in a parallel form with two branches of cascaded FRM structures mentioned in last paragraph, as shown in the Figure 2.13.

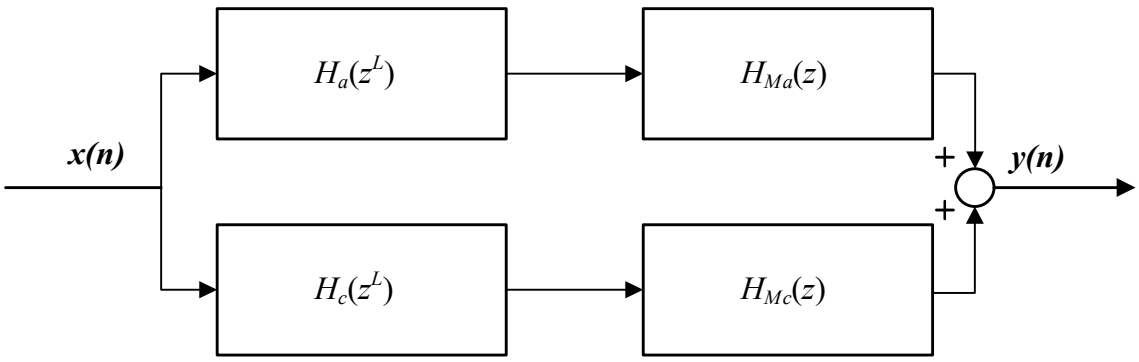

Figure 2.13 Direct form of frequency response masking

In the diagram, the branch on the top is normally called the *positive* branch and the other one on the bottom is normally called the *complementary* branch. All the four linear phase FIR filters have much more relaxed filtering specification compared to the initial single directly designed FIR filter. As a result, fewer non-zero coefficients and multiply operations are required by the FRM structure to have a sharp filter response. The computational load would have a significant reduction along with the computational load of the directly designed FIR filter having the equivalent filtering specification. The process of how to filter in both top and bottom FRM branches, and how to sum them together to result the final desired filtering frequency response is shown in Figure 2.14.

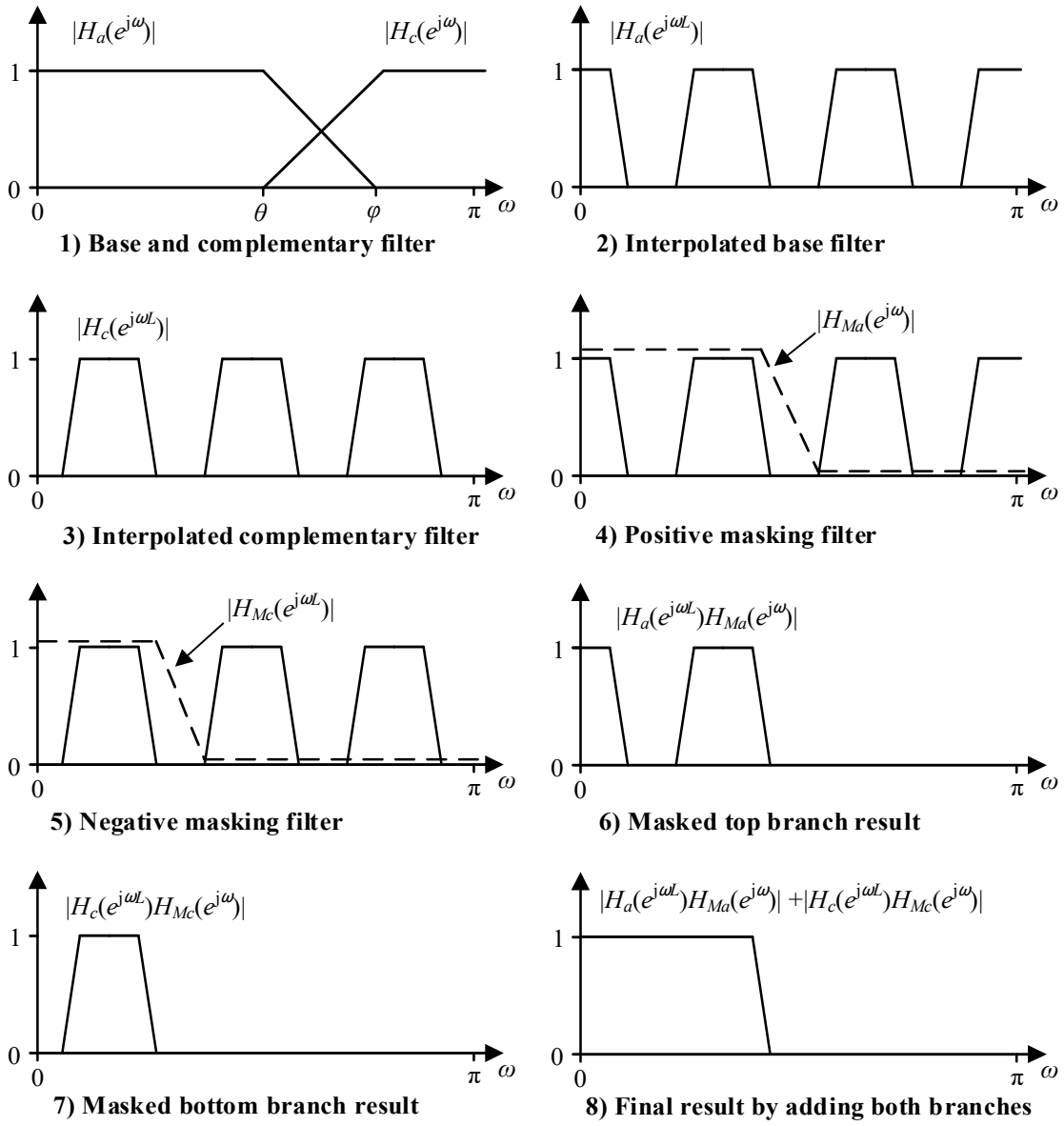

Figure 2.14 The process of two branches filtering base on FRM

In the FRM structure, the base filter is $H_a(z)$ and the complementary filter is $H_c(z)$. Their frequency responses are shown in Figure 2.14 (1) [46]. Both of them are then interpolated by a factor L by adding $L-1$ zeros in every adjacent coefficient of the FIR filter. Thus the passbands of $H_a(z)$ and $H_c(z)$ are reduced L times, and the transition band is L times sharper. All the filtering frequency responses are centred at $2\pi/L$. After that, the masking filters $H_{Ma}(z)$ and $H_{Mc}(z)$ will filter the interpolated frequency responses of base and complementary filter, thus only useful replicas will be left. In the end, by adding useful base and complementary replicas, the desired sharper filtering response will be finally produced.

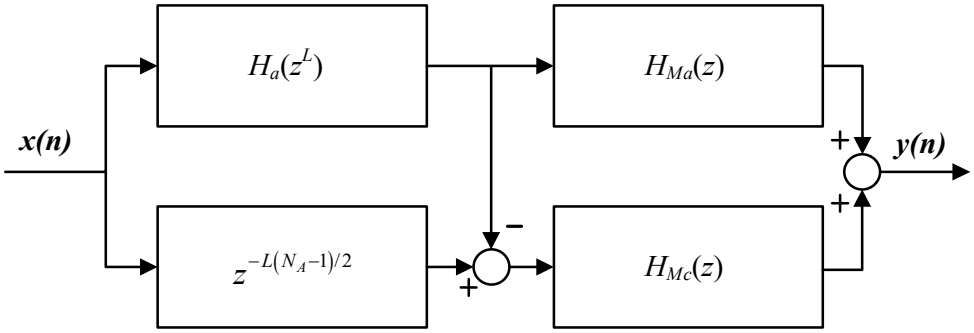

Figure 2.15 Efficient implementation of FRM

The transfer function of the FRM parallel structure is:

$$H(z) = H_a(z^L)H_{Ma}(z) + H_c(z^L)H_{Mc}(z) \tag{2.1}$$

where $H_a(z)$ is the base filter, and $H_c(z)$ is its complementary filter, and $H_{Ma}(z)$, $H_{Mc}(z)$ are the masking filters for interpolated bass filter and complementary filter respectively. If N_a is the order of the base filter, the complementary filter $H_c(z)$ has a relationship with base filter $H_a(z)$ given as:

$$H_c(z) = z^{-(N_a/2)} - H_a(z) \tag{2.2}$$

Hence, the relationship between interpolated base filter and complementary filter can be expressed as:

$$H_c(z^L) = z^{-L(N_a/2)} - H_a(z^L) \tag{2.3}$$

Therefore the complementary filter can be implemented by a chain of delays subtracting the base filter output. This will guide us to an efficient FRM structure as shown in Figure 2.15.

2.3.1 Subclass I filter

It has been discussed in the previous section that the transform function of whole FRM filter is given by equation (2.1). From the description of [45], the specifications of passband ω_p and stopband ω_s of the base filter $H_a(z)$ can be both freely selected. In addition, the complementary filter $H_c(z)$ can be also determined by equation (2.2) in order to implementing a further efficient FRM module illustrated in Figure 2.15.

However, a new class of FRM FIR filters called subclass I filter can be obtained only if we can ensure that the relationship between base filter $H_a(z)$ and complementary filter $H_c(z)$ can be expressed as:

$$H_c(z) = H_a(-z) \tag{2.4}$$

In order to match this condition, this special class of FRM filter must be designed with a base band filter whose transition band includes the normalized frequency $\pi/2$ [47], as shown in Figure 2.16.

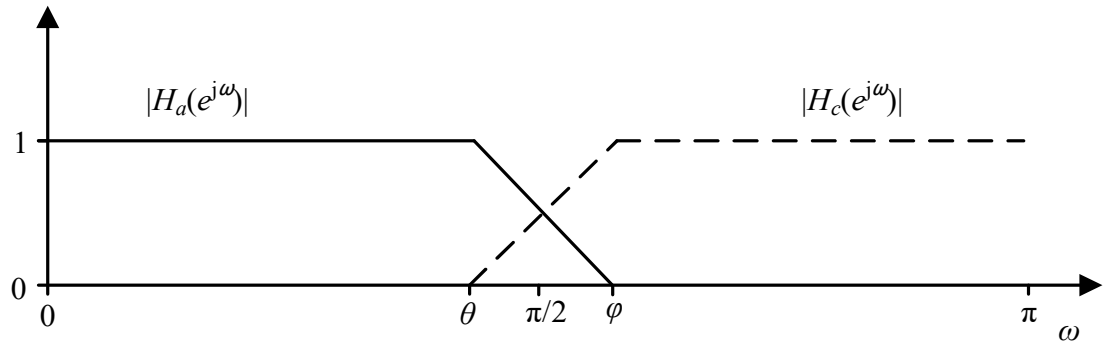

Figure 2.16 Subclass I Filter frequency response

In this circumstance, the computational load of the whole FRM structure could be further reduced if both branches share the same polyphase component as:

$$H_a(z) = H_{a0}(z^2) + z^{-1}H_{a1}(z^2) \tag{2.5}$$

where $H_{a0}(z)$ and $H_{a1}(z)$ are the polyphase components of $H_a(z)$. Then the base filter $H_c(z)$ can be expressed as:

$$H_c(z) = H_{c0}(z^2) + z^{-1}H_{c1}(z^2) = H_{a0}(z^2) - z^{-1}H_{a1}(z^2) \tag{2.6}$$

Thus when (2.6) is applied into (2.1), the whole new efficient design of FRM structure called full FRM is obtained as illustrated in Figure 2.17. This method makes the whole FRM filter design simpler than the design shown in Figure 2.15, as only $H_a(z)$, $H_{Ma}(z)$ and $H_{Mc}(z)$ need to be designed, and the usage of polyphase components of $H_a(z)$ takes the places of two complete filters.

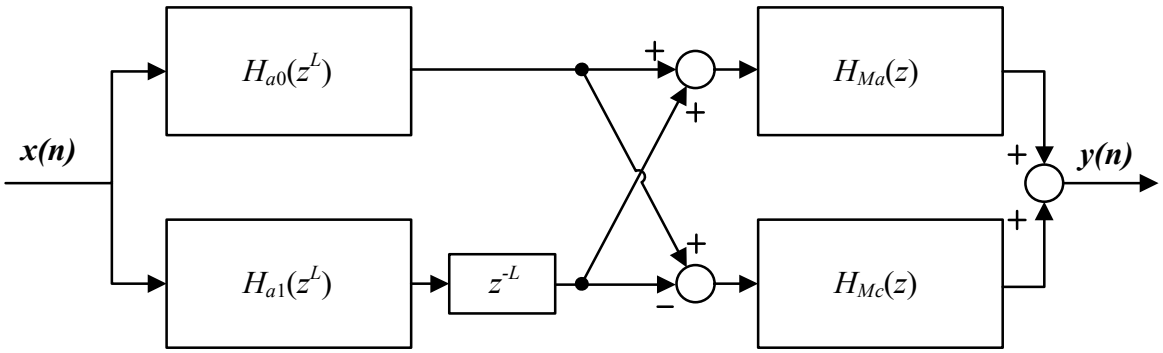

Figure 2.17 The efficient FRM design with polyphase decomposition

2.4 TETRA standard

TETRA (Terrestrial Trunked Radio) is a set of wireless digital telecommunication standards developed by the European Telecommunications Standardisation Institute (ETSI) that describes a common mobile radio communications infrastructure throughout Europe. TETRA provides reliable and robust digital communications to Professional Mobile Radio (PMR) and Public Access Moible Radio (PAMR) applications [48, 49]. These applications are targeted primarily at the mobile radio needs of public safety groups (such as police and fire departments), utility companies, and other enterprises that provide voice and data communications services.

In contrast with existing commercial mobile communication standards, PMR communication systems offer improved communication capabilities such as strong encryption information security, direct-mode to allow end-2-end communication without a base station, very long distance transmission [50]. Furthermore, comparing to commercial communication standards, PMR standards are basically allocated lower frequency bands, thus the wireless channel will produce less free-space attenuation over the transmitted signals.

TETRA is a fully digital system providing consistent voice quality and low bit error rate for data accordingly. It supports voice, circuit switched data and packet switched data services with a wide selection of data transmission rates and error protection levels.

For its modulation, TETRA uses $\pi/4$ Differential Quadrature Phase-Shift Keying (DQPSK). The symbol (baud) rate is 18,000 symbols per second, and each symbol maps to 2 bits, thus resulting in 36 kbit/s gross.

TETRA also uses Time Division Multiple Access (TDMA) technology. The process of TDMA involves digitally modulating a single frequency in order to increase the number of independent communication channels. Specifically it uses 4 channels interleaved into one 25 kHz channel. Instead of just one user being able to use the single 25 kHz channel, it can now be used by up to 4 different users. This creates both a cost savings in frequency needed and base stations or repeaters needed. It can support a gross bit rate of 36 kbits/s, with 7.2k bits/s per TDMA channel. The difference in the 28.8 kbits/s (4*7.2) is from overhead of the TDMA structure.

Each TDMA frame of four slots is grouped further as 18 frames, which, combined, form a multiframe. In circuit mode (as opposed to packet mode) voice and data, is compressed into 17 TDMA frames allowing for a control signaling frame to be used without stopping the flow of data.

2.5 Related work

In this section, we review the relevant works of the polyphase filter-banks.

DFT-FB is widely studied nowadays, and are adopted in several digital systems. A FPGA implementation of DFT-FB in 16 channels with basic optimization of word length and multiplications is presented in [51]. However DFT-FB, by the limit of the theory, this work only has an even-stacked channel allocation. On the other hand, GDFT-FB has the flexibility to channelize the wideband signal in an odd-stack allocation, leads to a better spectrum usage. While, few of GDFT-FB FPGA implementation is mentioned in the literature.

An oversampled DFT-FB has be created in order to reduce aliasing problem [52]. The author designed a new commutator and upsampler to achieve the oversampling filtering. However, if a polyphase filter-bank with different number of channels is needed, the architecture has to be redesigned from a very basic level. A workaround that employs FIR IP core to implement an oversampled DFT-FB or even an oversampled GDFT-FB would be preferred, as implementing FPGA with IP core could greatly simplify the process of design.

Based on DFT-FB, [53] has developed some new FRM applied GDFT-FB designs. The research indicates full FRM GDFT-FB and alternative narrowband FRM GDFT-FB have a better efficiency in terms of computational load compared with DFT-FB theoretically, because FRM technology can greatly reduce the number of coefficients in designing a very narrow-band filter. However nobody has implemented it in a digital signal processer, FPGA or any hardware platform to test these new designs' performance and feasibility for a practical industry system.

2.6 Conclusion

In this chapter, a thorough introduction of the materials and background that may support or be related to the DFT-FB, GDFT-FB, oversampled GDFT-FB full FRM GDFT-FB and alternative narrowband FRM GDFT-FB have been presented in this thesis. There are three main sections: (1) the FPGA background, (2) channelization processing overview and (3) an introduction to the FRM technology. The designs in this thesis will mainly be based on these three sections. In the FPGA background, the basic architecture of FPGA has been discussed. The FPGA's flexibility and process parallelism are the core

advantages compare to other implementation options. Three kinds of FPGA components—LUT, Block RAM and DSP48s are introduced in detail. These components play a very significant role in DSP algorithm implementation. LUT can efficiently get the results from any complex combinational logics. Block RAM can store Kbs of data, and its flexible memory division and read/write strategy make it excellent for building a FIFO. DSP48 is a piece of hardware slice built especially for DSP algorithms. DSP48s' build-in adders, accumulators and registers provide high power efficiency and performance. IP cores are reusable designs that provide resource efficiency and short developing time. Virtex-6 boards are the hardware platform of this thesis and all the code was written in Verilog HDL.

In the channelization technology section, the concept of channelization is briefly covered. Three basic channelization technologies – per-channel approach, pipelined frequency transform and polyphase filter-bank are introduced. Among them all, polyphase filter offers the least silicon cost and power consumption. Polyphase filters are one type of uniform filter-banks, because its sub-bands have the same frequency responses and bandwidths. Two approaches of non-uniformed channelizer (P-GDFT and R-GDFT) based on polyphase filter-bank are introduced.

FRM technology is a computational saving method for designing a very sharp transition band filter. It occupies the cascading connection of an interpolated FIR filter and a FIR filter with a relaxed specification instead of designing just one high order FIR filter. An efficient new class of FRM FIR filter is also discussed in this chapter, as further efficient FPGA designs will be based on this theory.

Chapter 3

Critically Sampled Uniform Wideband Channelization

3.1 Introduction

3.1.1 DFT-Filterbank (DFT-FB)

Different to other channelization technologies, DFT-FBs implement channelization based on complex modulation of the prototype filter in cooperation with the DFT algorithm. Compared to per-channel channelization, a DFT-FB only requires one filter and one DFT matrix, instead of K-1 filters. In addition, using the Fast Fourier Transform (FFT) rather than the DFT further improves computational load efficiency.

When a critically sampled configuration (*D=K*, where *D* is the down-sampling factor and *K* is the number of channels) is being applied, every sub-band's centre frequency is located at:

$$\omega_{CHk} = \frac{2\pi k}{K} = \frac{2\pi k}{D}, \qquad k = 0,1,....,K-1 \tag{3.1}$$

If we consider a prototype low-pass filter, *h(n)*, then the appropriate band pass filters for each sub-band are given by:

$$h_k(n) = h(n)W_K^{kn}, \qquad k = 1,2,3,.....,k-1 \tag{3.2}$$

where

$$W_K = e^{j(2\pi/K)} \tag{3.3}$$

The prototype filter for a K band filter-bank, $H(z)$ in the z-domain, may be divided into K poly-phase components, $E_p(z)$, as follows:

$$H(z) = \sum_{n=-\infty}^{\infty} h(n) z^{-n} = \sum_{p=0}^{K-1} z^{-p} E_p(z^K) \tag{3.4}$$

where

$$E_p(z) = \sum_{n=-\infty}^{\infty} h(nK + p) z^{-n} \tag{3.5}$$

The K sub-band filters are obtained by complex modulation of the prototype filter polyphase components using the DFT algorithm [42] as:

$$H_k(z) = \sum_{p=0}^{K-1} E_p(z^K) z^{-p} W_K^{-kp}, \quad p, k = 0, ..., K-1 \tag{3.6}$$

Figure 3.1 shows the block diagram representation of a DFT-FB analysis bank suitable for use as a uniform channelizer. (In typical implementations the Fast Fourier Transform (FFT) is used instead of the DFT because of its greater computational efficiency.)

In Figure 3.1, an anti-clockwised commutator (considered as a efficient form of delay and downsampling) would deliver the input samples into sub-band (prototype lowpass filter's polyphase component) by turns. Following that, a DFT matrix would implement the W_K^{-kp} factor in the (3.6).

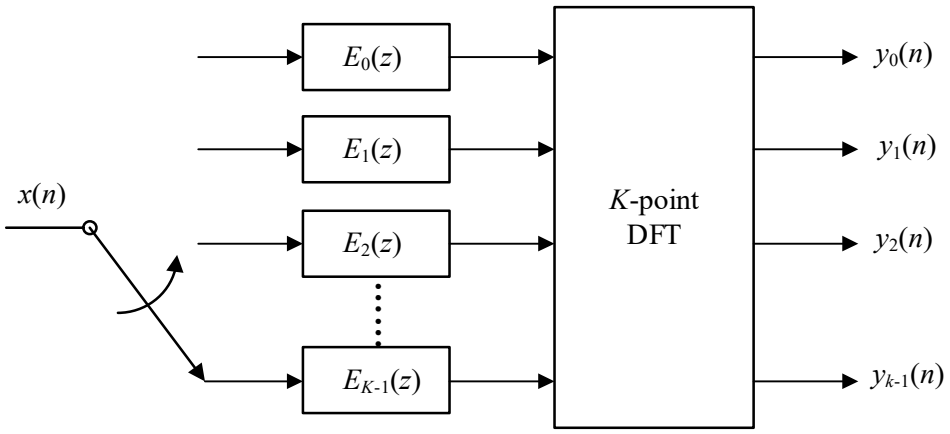

Figure 3.1 The polyphase DFT modulated receiver

For the convience of further analysis and research, the commutator in Figure 3.1 can be replaced with delay followed with down-sampling as shown in Figure 3.2.

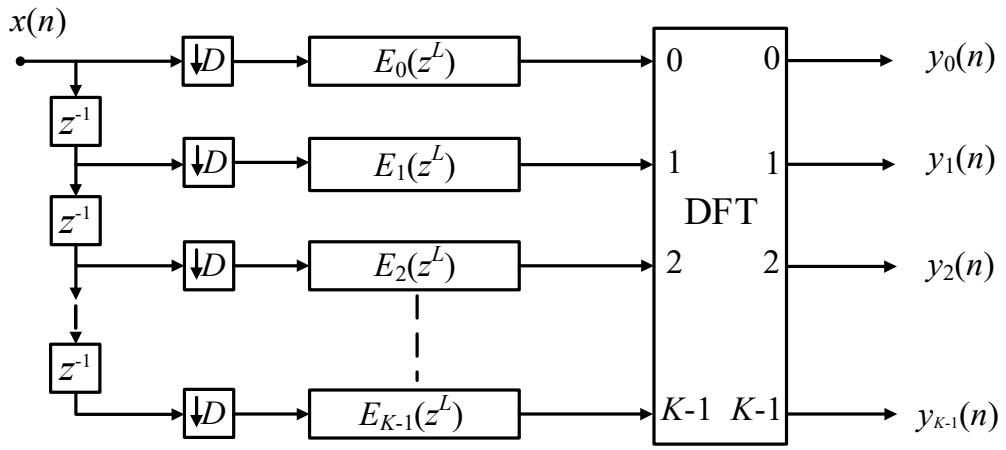

Figure 3.2 DFT modulated filter-bank (DFT-FB)

In the figure L is the oversampling factor of the DFT-FB, defined as:

$$L = K / D \tag{3.7}$$

and the output sample rate of each sub-band, F_s, is related to the input sample rate by:

$$F_s = F_{s,IN} / D \tag{3.8}$$

Therefore, when $L = 1$ the DFT-FB is *critically sampled* whereas when $L > 1$ the filter bank is *oversampled*. Moreover, although the output of each filter, $H_k(z)$ in (3.6), is theoretically decimated after filtering, for efficiency this decimation normally takes place before the filtering operation in a polyphase decimated implementation according to noble identities. In this case the polyphase components are also decimated by D so that instead of $E_p(z^K)$ we have $E_p(z^{K/D}) = E_p(z^L)$.

3.1.2 Generalized DFT Filter-Bank (GDFT-FB)

The DFT can be considered to be a special case of the Generalized DFT in which the sub-band centre frequencies and phases can be more explicitly controlled [41]. The GDFT-FB offers extra flexible channel stacking and phase shifting configurations, so that in

some applications the GDFT-FB may be preferred to the DFT-FB. One of the important reasons is that GDFT-FB can support both even-stacked and odd-stacked channel allocation as shown in Figure 3.3(b) whereas the DFT-FB only supports even-stacked channels.

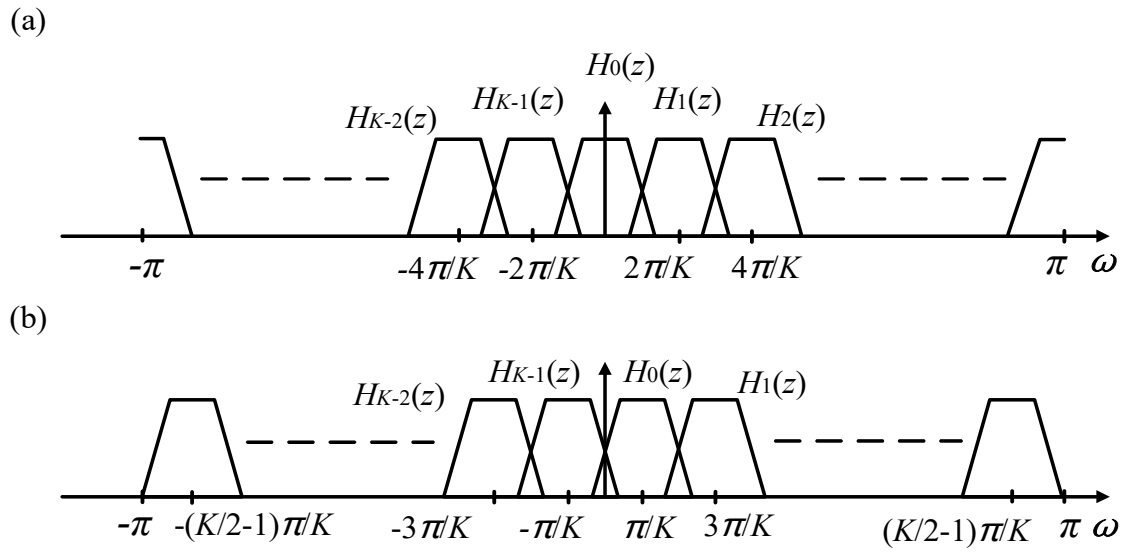

Figure 3.3 (a) Even stacked channels, (b) odd stacked channels

The benefit of odd-stacked channels is that the channelization of the wideband signal will be shifted by half of one sub-band bandwidth to the right. Thus it will eliminate the two half-sub-bands at either end of the wideband spectrum of the even-stacked allocation, like in Figure 3.3 (a). If all sub-bands must be used, this achieves more efficient spectrum usage.

In the GDFT-FB, the configuration of phase shifting and channel stacking flexibility in the implementation of every sub-band filter results from GDFT modulation. Like the DFT-FB, the GDFT-FB obtains its every sub-channel's band-pass filter $H_k(z)$ from complex modulation of the prototype low-pass filter $H(z)$. In the case of the GDFT-FB, this is:

$$H_k(z) = W_K^{-(k+k_0)n_0} \sum_{p=0}^{K-1} E'_p(z^K) z^{-p} W_K^{-kp} \qquad (3.9)$$

where

$$E'_p\left(z^K\right) = E_p\left(z^K W_K^{-k_0 D}\right) W_K^{-k_0 p} \tag{3.10}$$

The GDFT-FB is shown in the Figure 3.4. As before, K is the number of analysis filter-bank channels and D is the decimation factor. The GDFT parameter n_0 determines the possible phasing shifting which can be applied to the output of the filter-banks. For the channelizers in this thesis it always equal to 0. The parameter k_0 determines the stack allocation of the channels in the channelizer wideband input spectrum. When k_0=0 and n_0=0, then the even-stacked configuration is applied, as in Figure 3.3(a). There is one sub-band which is centred at DC, and two half sub-bands at either end of the spectrum. It is exactly the same as DFT-FB. In other words, the DFT-FB is a special case of GDFT-FB [41]. In contrast, if k_0=1/2 and n_0=0, all the channels have been shifted half of one sub-band bandwidth to the right, as in Figure 3.3(b). Thus there is no channel in the centre of DC, and all the sub-bands are complete.

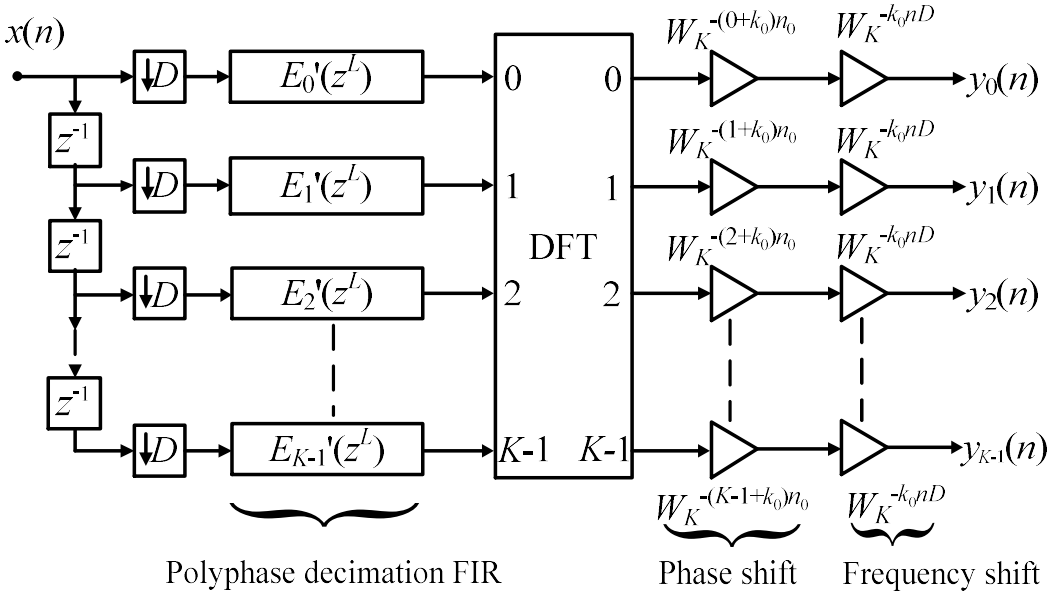

Figure 3.4. GDFT modulated filter bank (GDFT-FB)

The phase shift term can be simplified to a multiplication by 1, because n_0 is zero (as it is when the filter-bank is used to implement a channelizer). Unfortunately the complex modulation terms in the definition of $E'_p\left(z^K\right)$ means that the polyphase components of the prototype filter now have complex rather than real coefficients. In general it is clear

that the flexibility of the GDFT-FB results in some additional complexity and computation relative to the DFT-FB.

3.2 The FPGA implementation

In this section, a critically sampled DFT-FB (even stacked) FPGA implementation is developed on the Xilinx FPGA family using the Xilinx ISE (Integrated Software Environment) tool suite and the reusable IP core library.

3.2.1 Basic DFT-FB channelizer FPGA implementation

For the basic DFT-FB, the FIR compiler IP core provided with the development environment has a number of possible configurations, one of which is 'Polyphase Decimation' mode. In this mode, the IP core will implement the structure of the anti-clockwise commutator and polyphase decomposition of a prototype filter shown in Figure 3.1. It supports designs from 8 channels up to 1024 channels [54].

3.2.1.1 Coefficients Mapping

In Figure 3.1, a K to 1 polyphase decimation filter is illustrated. All the low-pass prototype filter coefficients $a_0, a_1, ..., a_n$ have been mapped to K polyphase sub-channels $h_0(n), h_1(n), ...h_K(n)$ respectively, according to

$$E_p(n) = \sum_{p=0}^{K-1} h(nK + p), \quad p = 0,1,..., K-1 \tag{3.11}$$

If we assume $K=4$, $D=4$, as an example, the polyphase filters $h_k(n)$ will be given by

$$\begin{aligned} h_0(n) &= [a_0, a_4, a_8, a_{12}, ...] \\ h_1(n) &= [a_1, a_5, a_9, a_{13}, ...] \\ h_2(n) &= [a_2, a_6, a_{10}, a_{14}, ...] \\ h_3(n) &= [a_3, a_7, a_{11}, a_{15}, ...] \end{aligned} \tag{3.12}$$

3.2.1.2 Complex Signal Process in FPGA

Typically in the FPGA realization of communication systems, a complex signal cannot be processed directly in the complex form. Instead, before doing any processing on the FPGA, the complex input signal must be divided into two parts: one part is its in-phase (I) component, which is also known as the real part of the signal and the other part is it's quadrature (Q) component, which is also known as the imaginary part of the signal.

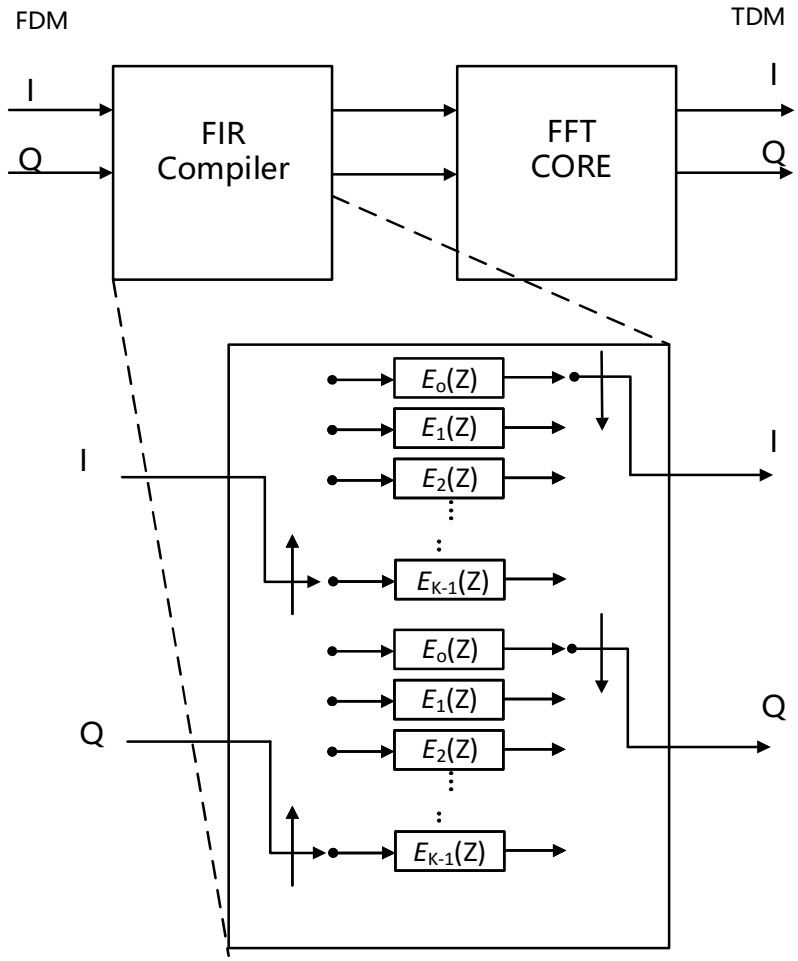

Figure 3.5 The DFT-FB FPGA design for the complex input

The FIR compiler IP core does not deal with complex or I and Q inputs directly, but it does support 2 inputs. Therefore the I and Q components can be supplied as separate (real valued) inputs to the same FIR compiler block. Each of these inputs is filtered with the same (real) filter coefficients in the same sub-band simultaneously. FIR compiler and FFT Core process a complex signal's I/Q components in two path respectively as shown in Figure 3.5.

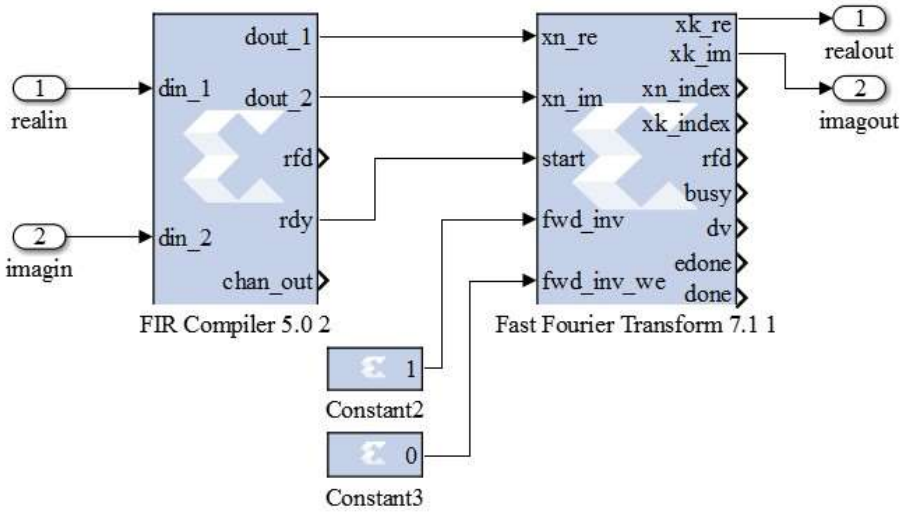

Figure 3.6 The FPGA implementation of DFT-FB

Figure 3.6 illustrates the FPGA implementation of the DFT-FB. We assume the input is already down converted to base band and divided into its I and Q components which go to pins din_1 and din_2 respectively. After filtering by the FIR compiler block samples corresponding to the same time instance in all sub-bands will come out serially as a burst of data, transmitted at the rate of the clock. When the FIR compiler core outputs this burst of data, the 'rdy' (ready) pin will be asserted. The Fast Fourier Transform (FFT) core will be triggered by this signal (connected to its start port), in order to start FFT processing to the output from the FIR core at the right time instant. The FIR compiler outputs dout_1 and dout_2 have a three clock cycles delay after 'rdy' (shown in Figure 3.7). Therefore, the FFT core needs to be configured with a 3 cycle offset on its 'start' port. The FFT core is pre-set to have a FFT transform length of the number of channels. After FFT transform, the final result will be output in the form of separated stream of I and Q components from xk_re and xk_im respectively. The order in which sub-band samples appear in the serial output stream is determined by the FFT butterfly operation.

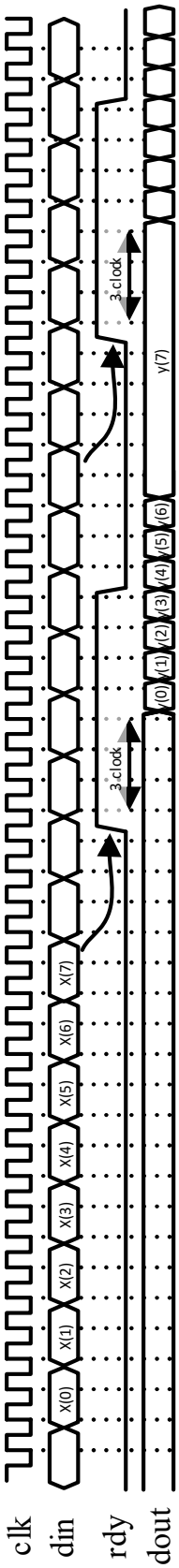

Figure 3.7 The waveform shows that the output of FIR complier has a 3 clock delay from rdy signal

3.2.2 GDFT-FB Channelizer FPGA Implementation

3.2.2.1 Complex modulation of prototype filter coefficients

For the DFT-FB, the prototype low-pass filter $H(z)$ can be designed to have all real coefficients. However, in the GDFT-FB case, the filters of the sub-bands have been subjected to complex modulation as shown in equation (3.9). In fact, this modulation is applied offline during design so that the complex modulated coefficients may be divided into their I and Q components. These real-valued I and Q coefficients are then supplied to two FIR compiler IP Cores. In order to explain how this is done, we need to first convert equation (3.10) to its time domain equivalent form with the interpolation:

$$e'_p(n) = e_p(n) e^{j\frac{2\pi}{K}k_0 Dn} e^{-j\frac{2\pi}{K}k_0 p} \tag{3.13}$$

where

$$e_p(n) = h(nK + p), \quad p = 0,...,K-1 \tag{3.14}$$

As in the case of the critically sampled odd stacked GDFT-FB, where $D = K$ and $k_0 = 1/2$ then equation (3.13) will be reduced to

$$e'_p(n) = e_p(n) e^{j\pi n} e^{-j\pi p/K} \tag{3.15}$$

The modulation of coefficients is applied to each polyphase component independently. Thereafter the modulated component coefficients are interpolated and reassembled as indicated by (3.4), substituting $E'_p(z^K)$ for $E_p(z^K)$, to form the appropriate arrangement of prototype filter coefficients.

3.2.2.2 Complex Filter Coefficients using the FIR Compiler

The GDFT-FB is not as straightforward to implement as the DFT-FB when the parameter k_0 is non-zero [41]. (We will not examine cases where the parameter n_0 is non-zero in this work since it is not required for channelizer design.) When k_0 is non-zero then the coefficients of the prototype filter are subject to complex modulation (see (3.10)) yielding complex filter coefficients which the FIR compiler IP core does not support. Using cross

coupling between real and imaginary signal paths though both real and imaginary part of coefficients could implement complex filtering, as shown in Figure 3.8 (b), where *I*(z) represents the I component of the *H*(z) coefficients and *Q*(z) represents the Q component of the *H*(z) coefficients. It can be easily realized by adding and multiplication operations, plus delay operations in discrete-time FIR filters [55].

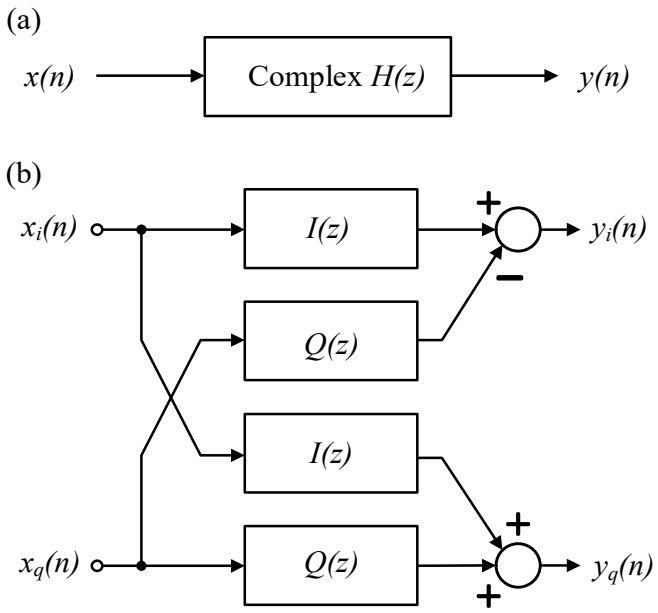

Figure 3.8 Cross coupling of complex signal filtering

To see how this cross-coupled approach works, consider the following example. Assume an input signal consisting of just one sample, $x = x_i + x_q j$, and a filter with only one coefficient $h = h_i + h_q j$.

The filtering resulting from complex convolution (which reduces to one multiplication in this case) is

$$xh = \left(x_i h_i - x_q h_q\right) + \left(x_i h_q + x_q h_i\right) j \tag{3.16}$$

In accordance with this approach, the GDFT-FB has been implemented using two FIR compiler blocks. Each FIR compiler block has the same number of coefficients but different values corresponding to the I (real) component of the coefficients for the first

FIR compiler block and the Q (imaginary) component of each coefficient for the second FIR compiler block.

Thus before passing FIR compiler outputs into the FFT core, they must be combined using the cross coupling approach as shown in Figure 3.9.

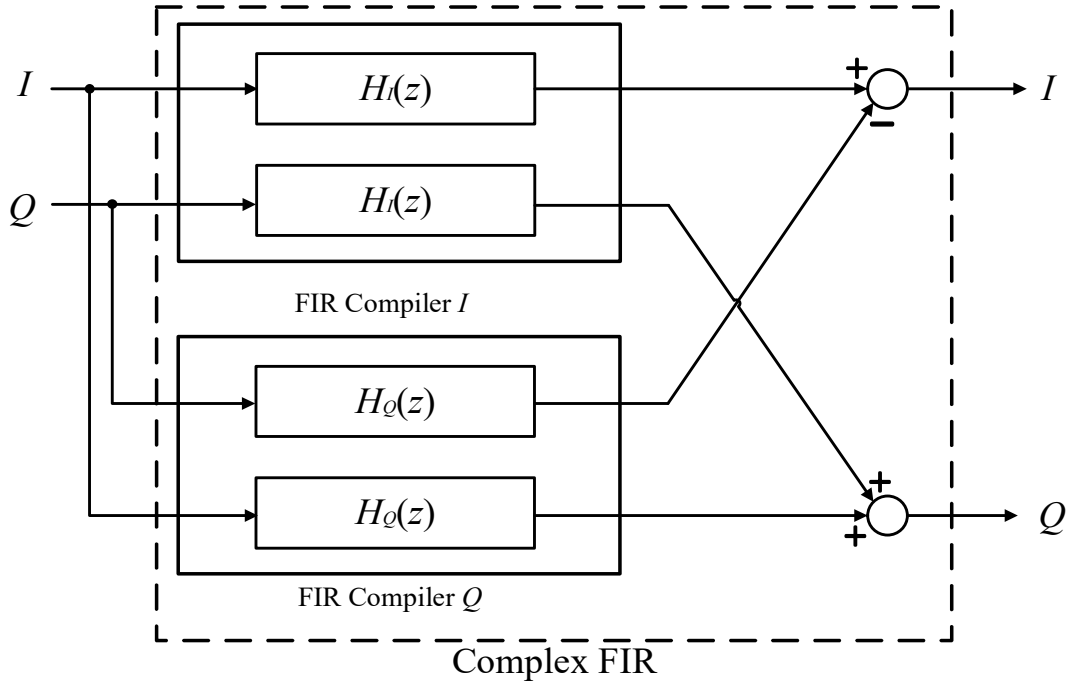

Figure 3.9 Complex FIR implemented using cross-coupled FIR compiler IP core

3.2.2.3 Frequency Shift State Machine

From Figure 3.4, it is clear that the output of the DFT is followed by two separate complex multiplications. The first of these, the phase shift operation, simplifies to multiplication by 1 (that is, no operation required) in our usage because n_0 is zero. The second is a frequency shift operation required to shift the output sub-band down by $F_s/2$ so that it is centred on DC. This corresponds to a mixing operation but close examination of the possible multiplier values shows that it can be efficiently implemented with a state machine.

The multiplication $W_K^{-k_0 nD}$ is expanded as

$$W_K^{-k_0 nD} = e^{-\frac{j2\pi}{K}(k_0 nD)} \qquad n \in \mathbb{N} \tag{3.17}$$

For a GDFT-FB configuration that is odd-stacked, where k_0=1/2, and critically sampled (that is, the oversample factor is $L = K/D = 1$), then the complex multiplication will be reduced to

$$W_K^{-k_0 nD} = e^{-j\pi n} = \begin{cases} 1, & n \text{ even} \\ -1, & n \text{ odd} \end{cases} \text{ for } n \in \mathbb{N} \qquad (3.18)$$

where n is the output sample number.

Equation (3.17) can be efficiently implemented using a state machine which either passes the output through unchanged (for even numbered samples) or negates the output (for odd numbered samples). Since the FFT outputs 1 sample from each of the K output sub-band sequentially, the state machine should change state only after K samples have been output from the FFT.

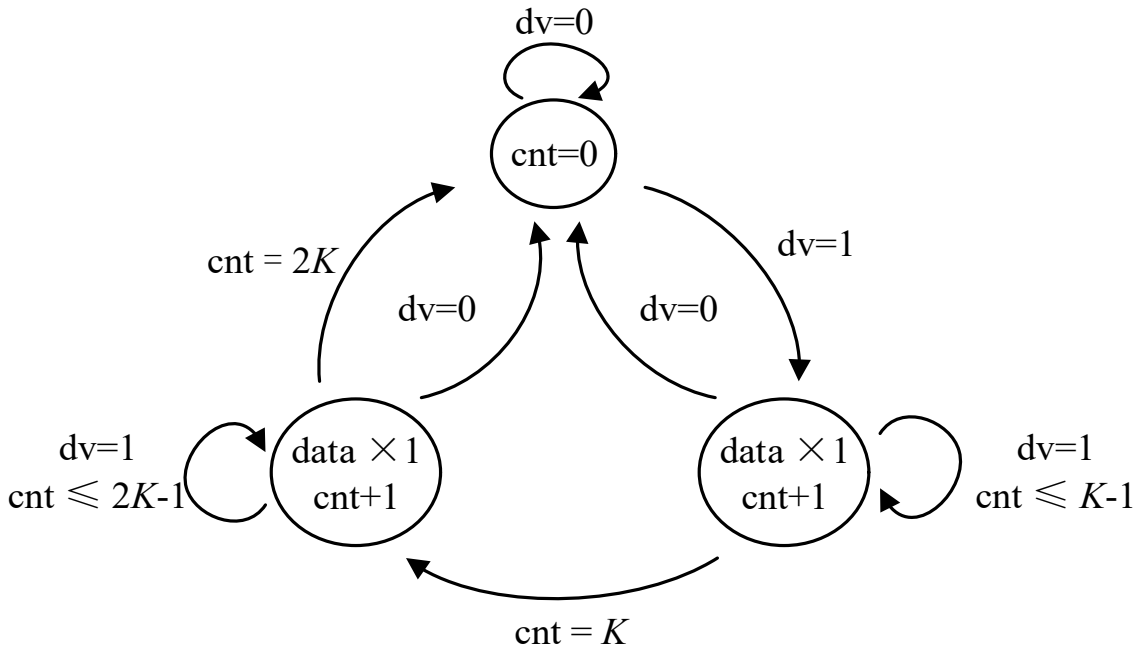

Figure 3.10 Frequency shifting state machine work flow.

A frequency shifting state machine has the work flow as shown in Figure 3.10. The state machine could be triggered by the ‘dv’ signal from FFT core, which is when data is valid.

When the state machine is reading the first round of K samples $x(0)$~$x(K\text{-}1)$ (one sample from each channel, they are all 'n'th sample in each sub-band), it will multiply 1 to these samples. If the state machine is reading the second $x(K)$~$x(2K\text{-}1)$, it will multiply -1 to these samples. After the state machine processed $2K$ samples, the counter will start over again.

To negate digital bits, an 'inversion and adding one'[56] is applied. An extra bit is extended to the MSB of the result register, and initialized with the sign bit (origin highest bit). As an addition operation will lead to the overflow, the extension of sign bit have to be applied to positive value as well. In addition, the resetting pin 'rst_n' is also assigned in the statement to clear the registers.

3.2.2.4 Final Design

Figure 3.11 shows the FPGA implementation block diagram of the theoretical GDFT-FB design shown in Figure 3.4, incorporating each of the steps described above. GDFT-FB could be used to implement both even-stacked and odd-stacked channelizers (with suitable choice of k_0), because it is a general design. Nevertheless, the DFT-FB is the more efficient design for even stacked designs since it does not require the complex FIR (which requires 2 FIR compiler blocks) or the output frequency shift state machine.

It is worth noticing that the addition and subtraction operations in the complex filtering processing cost 1 clock cycle. Thus the delay of filtering comparing to 'rdy'(described in § 3.2.1.2) increases from 3 to 4 clock cycles. For the sake of synchronization, the signal 'rdy' is then delayed for 1 clock cycle. The 'start' pin of FFT core is determined by the logic 'AND' of both 'rdy' pins of two FIR compilers in the case of synchronization. The operation 'AND' is a combinational operation which only takes a small amount of time compari to a clock cycle.

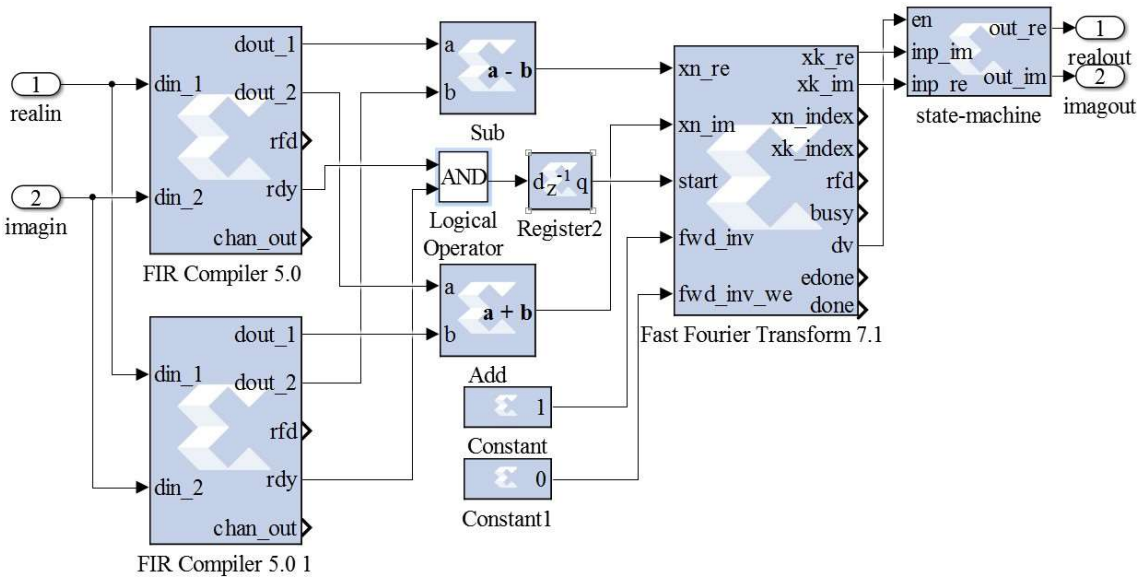

Figure 3.11 FPGA implementation of the GDFT-FB

3.3 FPGA Implementation Evaluation

3.3.1 Implementation and test environment

All the implementation and evaluation is worked on Xilinx Virtex-6 ML605 and Xinlix ISE 14.3 design software. The FPGA filter-banks are programmed using Verilog HDL and cooperate with certain IP cores.

3.3.1.1 Implementation specification

To test the feasibility, demonstrate the performance, and validate the accuracy of the FPGA implementations, the following designs were simulated: the critically sampled DFT-FB (even stacked configuration) and critically sampled GDFT-FB (odd-stacked configuration). All the implementations should have the same design specifications, in order to make a fair and reasonable comparison. All filter-banks should have the same number of channels, channel characteristics (passband ripple, stopband attenuation, and bandwidth) and the same fixed point word-length.

The evaluation criteria focused on the sub-band frequency response, EVM performance, and adjacent channel interference. To make the evaluation concrete, the specifications of

the TETRA (TErrestrial TRunked RAdio) Voice and Data standard (with 25 kHz channels) were used [48].

The passband ripple, stop attenuation, bandwidth and other filtering specifications would vary for different communication standards. To have a comprehensive comparison, the TETRA 25 kHz channel specification is used as a standard among these FPGA filter-bank designs. This standard allows us to have a passband ripple not greater than +/- 2dB, a stop attenuation greater than 55dB and a bandwidth of 25 kHz. Specifications like frequency band in the RF or others will not be considered in this case.

In this test case, the sampling frequency F_s of wideband input signal is 400 kHz, so that both the negative and the positive sides of the spectrum could be used, so there are 400 kHz of spectrum (from - $F_s/2$ to $F_s/2$) could be used, which could contains 16 channels with 25 kHz bandwidth. To meet the requirement of the TETRA standard, the designed prototype filter for both even and odd configuration GDFT-FB has 416 coefficients.

For the even stacked configuration (DFT-FB), the transmitter centre frequencies have been chosen, so that the input wideband signal is also even stacked. Fifteen channels with 25 KHz bandwidth have been allocated to centre frequencies at 0 kHz, +/- 25 kHz, +/- 50 kHz …… +/- 175 kHz. One channel (which appears as two half-sub-bands at either end of the wideband spectrum) is not usable because of the nature of even stacked configuration. For the odd stacked configuration (GDFT-FB) all 16 channels are usable, and the centre frequencies are located at +/- 12.5 kHz, +/- 37.5 kHz, +/- 62.5 kHz, ……+/ 187.5 kHz. In both even and odd stacked configurations, each transmitted channel has an 18k symbols/second of $4/\pi$ DQPSK modulated digital signal.

The channel characteristics were matched (as far as possible) by designing one prototype filter for DFT-FB/GDFT-FBs designs and composite filters having an equivalent frequency response for the FRM based filter banks.

Number of channels: The number of channels is set to be 16 for both of the filter-bank designs.

Word-length: In these implementations, the input samples have 16-bit signed in-phase and quadrature parts, the coefficients are also in 16-bit signed representation. This allows

us to make efficient use of the embedded DSP Blocks on the FPGA. The architecture allows for 16-bit coefficients with a scale value. The scale value must be computed in advance by the user, but is simply a case of finding the maximum dynamic range for each sub-band filter and scaling by a power of 2. In addition, 16 bits is a much more common word-length in the industry, so for all the design, 16 bits is chosen. The specification of filter-banks are presented in Table 3.1.

Table 3.1 Test filter-banks' specifications

Design specification terms	**Value**
Number of channels	16
Stopband ripple	+/- 2 dB
Passband attenuation	-55 dB
Fixed point word-length	16 bits

3.3.1.2 Xilinx Viretx-6 board overview

The Virtex-6 FPGA board is a Xilinx designed programmable platform for developers. It is built on a 40 nm copper process technology and operates on a 1.0V voltage with a 0.9V low-power option. The board has an up to 50% lower power consumption than its previous generation.

The board available in this thesis is Virtex-6 ML605. It uses a XC6VLX240T-1FFG1156 device. XC6VLX240T has a total of 241,152 logic cells, and 37,680 programmable logic slices. The LUT in Virtex-6 can be used as one 6 inputs LUT or two 5 inputs LUTs. Four such LUTs and 8 registers form a slice, and two slices form a programmable logic block. In addition, some part of the slices can configure their LUTs as distributed RAMs, and these can make up to 3,650 Kb of storage.

This development FPGA board also contains 416 dual-port block RAMs, every block RAM can store 36 Kbits of data. A 36Kbit block RAM can be split into two 18Kbit blocks to double Block RAM bandwidth. Each of them has two independent ports which share the stored data.

There are also 768 DSP48E1 Slices to optimize DSP algorithm computations. It is an enhanced architecture with a 25-bit pre-adder, 25x18 multiplier, 48-bit adder and 48-bit accumulator, capable of operating at the clock rate of 600 MHz. The new pre-adders involved in Virtex-6, are typically used for symmetrical filtering, and may have a

significant reduction in usage of logic slices in some certain applications. The resources summary of the Virtex-6 XC6VLX240T-1FFG1156 is shown in Table 3.2.

Table 3.2 Virtex-6 XC6VLX240T-1FFG1156 FPGA board resources summary

Device	Logic cells	Programmable Blocks		Block RAMs			DSP48s
		Slices	Max Distributed RAM(Kb)	18kb	36kb	Max(Kb)	
XC6VLX240T	241,152	37,680	3,650	832	416	14,976	768

3.3.1.3 Implementation and test flow

A diagram of the implementation and test flow for filter-banks development appears in Figure 3.12.

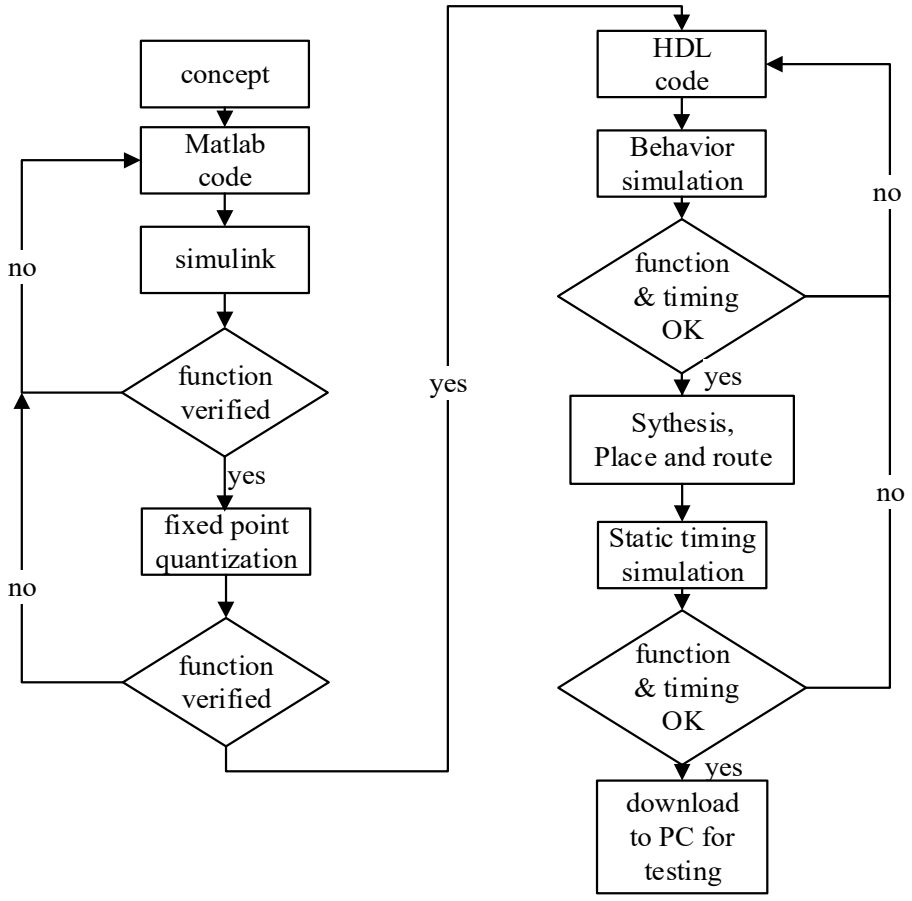

Figure 3.12 The Filter-bank development and testing flow

To implement and test a complex DSP application in FPGA not only need the FPGA design program, but we also need the mathematic programme MATLAB and its additional package Simulink.

First, Filter Design and Analysis Tool (FDATool) of MATLAB was used to design the prototype filters in accordance with the specifications. The resulting filter coefficients are specified with floating point precision. Next a Simulink model of each channelizer was implemented for the appropriate number of channels using the filter coefficients already designed. Simulation of this model was performed using floating point data and parameters to assess whether or not it matches the theoretical performance. After that, the parameters and internal data of every stage of the Simulink model were quantized to 16-bit fixed point values that would be used in the FPGA implementation. This fixed point model was then simulated to assess its performance in comparison to the floating point implementation results. If the fixed point performance matches the specification of desired filter-bank sufficiently well, MATLAB was used to generate the quantized parameter and input in .coe file, required by the FPGA platform.

In the FPGA implementation section, functional FPGA components are written using Verilog HDL. These components and IP cores are wired with each other in a higher level block using Verilog HDL as well. When the system is build up and synthesised, the developer also needs to design a testbench in ISE to provide the simulation environment with all the input, and the testbench also needs to take the results from simulation. A lot of verifications can be done by checking the result waveform. However in DSP applications, the result samples also need to write into files for further verification after downing to PC, like phase and frequency response checking by using MATLAB, because too many samples need to be recorded. If both functional and timing simulation are passed and verified functional, developers can get the report of resources usage and other result.

3.3.2 Evaluation and Results

3.3.2.1 Frequency response

The frequency response of the FPGA based odd-stacked GDFT-FB is shown in Figure 3.13. The reference frequency response was obtained from a floating point simulation of

the GDFT-FB in Simulink. Against this, the FPGA GDFT-FB frequency response can be compared. When zooming into the passband, shown in Figure 3.14, we can see the effect of fixed point quantization in the passband. The passband ripple has increased to 0.0548 dB, from floating point design's 0.005 dB. However the 2 dB limit was still not exceeded.

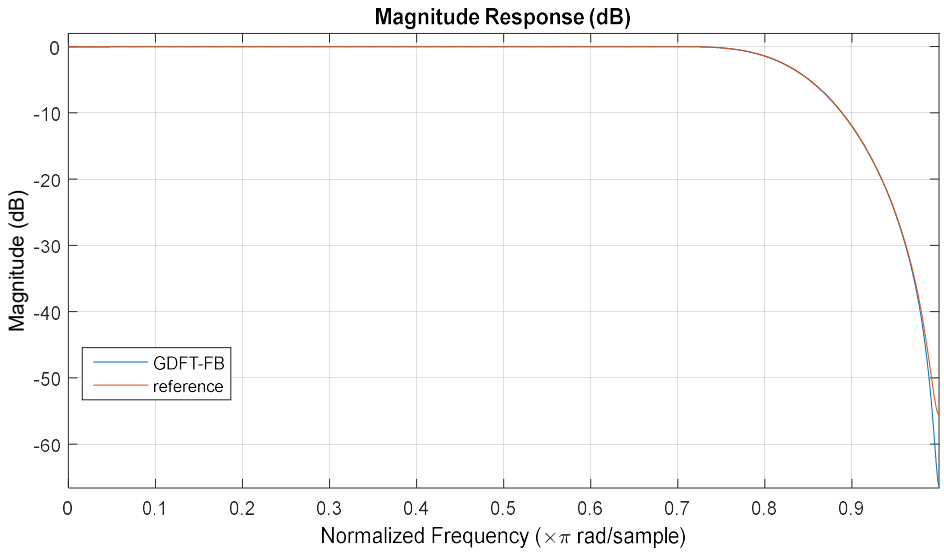

Figure 3.13 Sub-band frequency response of the FPGA Fixed Point GDFT-FB (blue line) compared to a floating point GDFT-FB reference (red line)

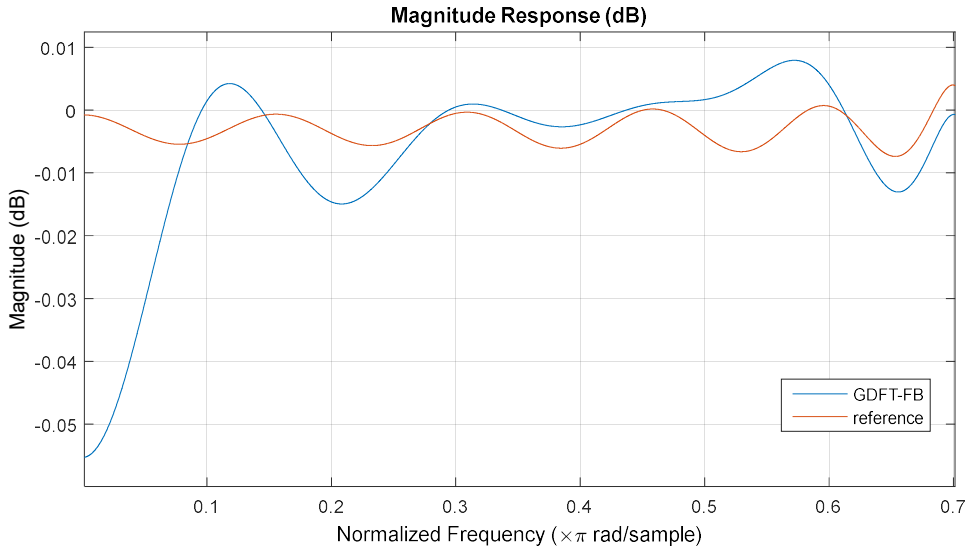

Figure 3.14 Passband comparison between FPGA based GDFT-FB and its floating point reference

3.3.2.2 *EVM result*

Error Vector Magnitude (EVM) is a measurement of error performance in complex DSP systems. Basically it indicates the vector differences between the ideal signal and the received signal. EVM can help to validate the performance of the system in terms of phase noise, I-Q imbalance and filter distortion. The TETRA standard indicates that the Root Mean Square (RMS) EVM shall be less than 0.1, and the peak vector EVM shall be less than 0.3.

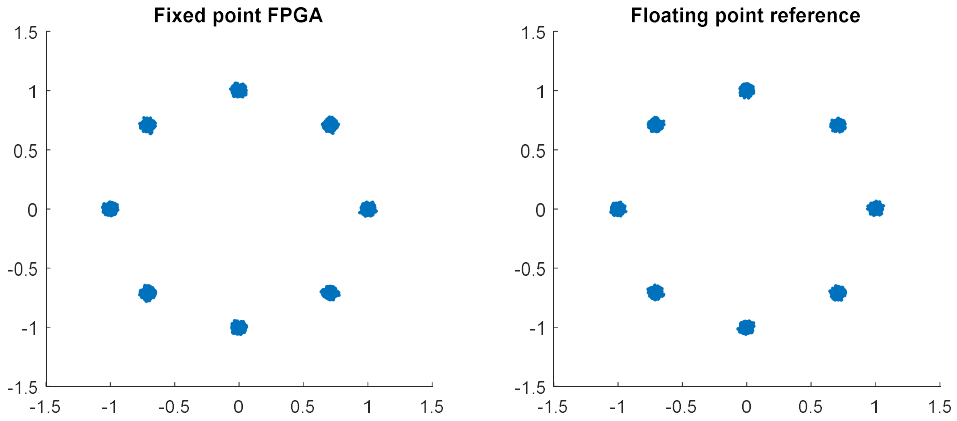

Figure 3.15 The QPSK modulation constellation after the FPGA based GDFT-FB (left), and the QPSK modulation constellation after a floating point GDFT-FB

Figure 3.15 shows the DQPSK modulated signal constellation diagram processed by the FPGA based GDFT-FB and the reference floating point GDFT-FB. The EVM test result is show in the Table 3.3.

Table 3.3 The EVM performance of a 16 channel GDFT-FB based on FPGA

	GDFT-FB on FPGA	Floating point FPGA	TETRA Limits
Peak	0.0732	0.0700	0.1
RMS	0.0298	0.0296	0.3

3.3.2.3 *Adjacent channel interference*

Adjacent channel interference is caused by unwanted power from the signal in the adjacent channel intruding into the channel of interest. The interference will be more

severe, if more energy is added to the adjacent channel, because more unwanted power comes into the channel of interest. The ability of a system to reject adjacent channel interference ultimately affects its ability to deal with a mix of near (higher power) and far (lower power) transmitters. Thus rejection of adjacent channel interference is an important characteristic of a channelizer's sub-band filter performance.

The TETRA specification specifies that the minimum requirement for the value of carrier to adjacent ratio is C/I_a=-45 dB.

To test this, three adjacent channels were simulated. The channel in the middle was the channel of interest. The two channels on either of its sides were used to generate interference. The two interfering channels were set to the maximum amplitude at first while the channel of interest was attenuated to the limits of the specification. EVM measurement of the channel of interest will validate if the RMS and peak EVM meets specifications when C/I_a=-45 dB. Figure 3.16 illustrates the three transmitted channels which contribute to the wide band input signal. Note that in this case, even stacked channel allocation (appropriate for the DFT-FB) was used. The channel of interest has the carrier to adjacent ratio of -45 dB.

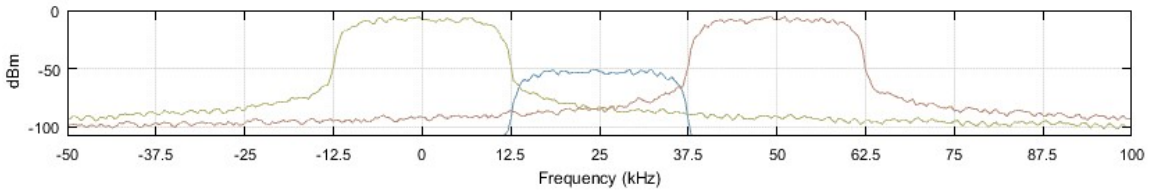

Figure 3.16 The even stacked testing wideband signal of adjacent channel interference when C/I_a = -45 dB

Several different levels of adjacent channel interference were simulated: -10 dB, -20 dB, -30 dB, -40 dB, -45 dB and -50 dB; among these the -45 dB level is the limit required by the TETRA specification. The modulated constellation of the channel of interest extracted by the FPGA GDFT-FB under these adjacent channel interference conditions is displayed in Figure 3.17.

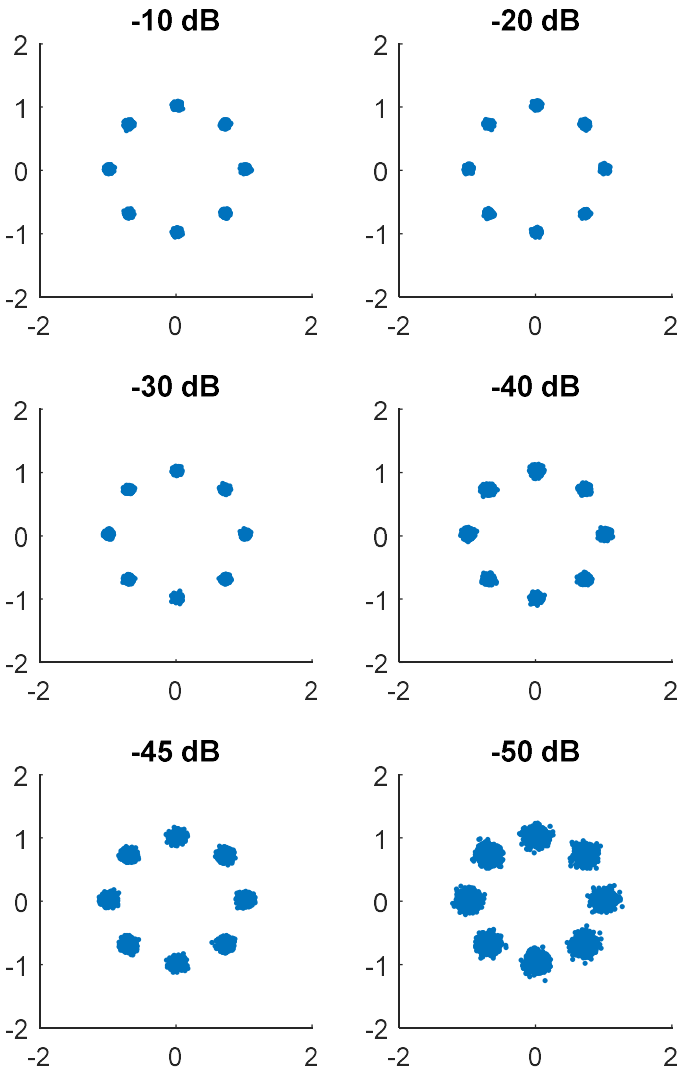

Figure 3.17 Modulation constellation of a channel of interest subjected to different levels of adjacent channel interference after extraction by the FPGA GDFT-FB

The constellation result shows that as the adjacent channel interference increases, the constellation points become more scattered. Table 3.4 shows the numerical EVM results. It is worth noting that when the adjacent channel interference is increased to -45 dB, the RMS and peak EVM are still within the TETRA specified limits. Not until the adjacent channel interference is increased to -50 dB are the RMS and peak limits (0.1 and 0.3 respectively) exceeded.

Table 3.4 RMS and Peak EVM for a channel of interest subjected to different adjacent channel interference level extracted using the FPGA based GDFT-FB

C/I_a (dB)	RMS	Peak
-10	0.0462	0.1109
-20	0.0465	0.1119
-30	0.0479	0.1405
-40	0.0573	0.1539
-45	0.0749	0.2078
-50	0.1159	0.3229

3.3.2.4 Hardware usage

The wideband input signal was digitized at 0.2 megasamples/second with a resolution of 16 bits per sample. The FIR compiler's coefficients, and most of the various input and output buffers, and interim results (like the phase factor) from the FFT would require the use of block RAM. All the multiplications and additions were implemented by DSP48s for maximum performance. Therefore the usage of the block RAM and DSP48 resources are the most important to evaluate. The 16-channel even and odd stacked FPGA GDFT-FB with FPGA conventional per-channel approach filter-bank resource usage is shown in Table 3.5. As the result, odd stacked GDFT-FB will use about 20% percent more resources, but it can have a better spectrum usage. However GDFT-FB FPGA has a very great resources efficiency compared to the per-channel approach, because in per-channel approach design, 16 channels all requires a 416 taps FIR filter, and it's corresponding digital down converter with different centre frequencies.

Table 3.5 Resource usage for the (even stacked) DFT-FB and (odd stacked) GDFT-FB channelizers

Filter-bank Type	Register	LUTs	Block RAM 36	Block RAM 18	DSP48s
Even GDFT-FB	1223	838	0	8	5
Odd GDFT-FB	1435	1270	0	11	7
Per-channel approach	3993	3063	4	48	80
Available	301440	150720	416	832	768

3.4 Chapter conclusion

In this chapter, the FPGA based DFT-FB was implemented using IP cores. It can handle complex input by creating parallel paths for I and Q components through the FIR compilers. The concept and detail of GDFT-FB was introduced. In channelization, its use is motivated by the requirement to extract channels from an odd-stacked channel

allocation. The odd stacked GDFT-FB could be considered to have a better frequency spectrum usage, because it eliminates two of the half sub-bands at either end of the even-stacked channels. Unfortunately, the GDFT-FB requires complex modulated filter coefficients that the FIR compile cannot directly handle. Thus, in the FPGA implementation, this problem is solved using two cross coupled FIR compiler blocks and separating the I and Q components of the complex FIR coefficients such that they can be applied by two FIR compiler blocks (which only accept real-valued coefficients). The GDFT-FB also requires a complex mixer on each output sub-band which was efficiently implemented using a state machine. Nevertheless, the odd-stacked (GDFT-FB) design requires more resources than even-stacked (DFT-FB) design and this is verified in the simulation result section. The simulation results also confirm that with 16-bit fixed point resolution, the critically sampled FPGA based DFT-FB and GDFT-FB can meet the TETRA V&D 25KHz channel specifications, even when subject to interfering adjacent channels at 45 dB higher power on both sides of the channel of interest.

Chapter 4

Oversampled Uniform Wideband Channelization

4.1 Introduction

4.1.1 Aliasing problem and oversampling solution

The critically sampled ($L=K/D=1$) GDFT-FB architecture is straightforward and reliable, as described in the previous chapter. However, a critically sampled filter bank requires a prototype low-pass filter whose pass band and transition band do not exceed the decimated Nyquist frequency of the sub-band if aliasing is to be minimized. If the sub-band filter exceeds the Nyquist frequency, aliasing can become a problem by introducing signal correlated noise which distorts the signal. To avoid the problems of aliasing (in a critically sampled design) it may be necessary to specify that less of the nominal sub-band width is available to the signal (thus creating a wider guard band between signals in adjacent channels), or it may be necessary to design a higher order low-pass prototype filter so that a sharper filter transition can be achieved (to minimize overlap between a filter and its images). These solutions have the disadvantage of either reducing the useful bandwidth of the signal or increasing the resource usage due to the larger numbers of filter coefficients and higher computational load.

To avoid narrowing the available signal bandwidth or unnecessarily increasing the prototype filter order an oversampled filter bank may be used. In an oversampled design, the sub-band Nyquist frequency is now larger than the sub-band spacing ($F_{s,IN}/K$) and it is therefore possible to have sub-band filters which overlap in terms of the input signal

but whose images do not overlap and hence do not cause aliasing after decimation (as shown in Figure 4.1(b). This is described in more detail in [41] .

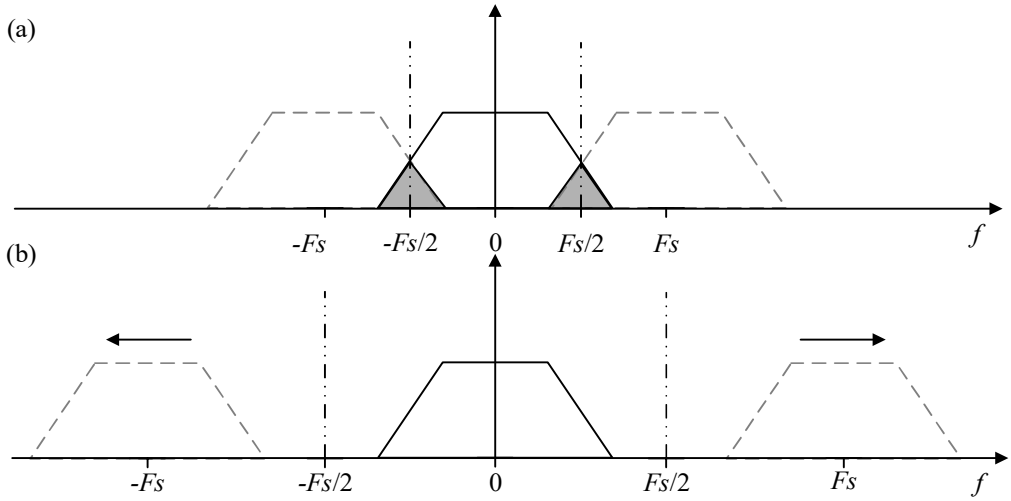

Figure 4.1 The interaction of a filter with its images in the decimated sub-band output (a) exhibits aliasing when critically sampled due to overlapping images whereas (b) oversampling separates the images and greatly reduces aliasing.

4.1.2 Oversampled polyphase decomposition

There are a number of example implementations of the oversampled DFT-FB (even stacked) described in the literature, such as[17, 52, 57, 58]. They all use a similar structure that has integer interpolators in the sub-bands after the operation of a commutator as shown in Figure 4.2.

These designs use the same commutator (or equivalent structure) as in the critically sampled design. Then in every channel an integer-valued interpolator is applied right after the commutator. This approach achieves oversampling the sample rate in every channel by padding L-1 zeros to input samples. In order to achieve this new oversampled input method, a new designed commutator is developed by [52]. In addition, in this oversampled configuration, the polyphase decomposition of the prototype for each channel is given by:

$$E_p(z) = \sum_{n=-\infty}^{\infty} h(nK + p)z^{-n} \tag{4.1}$$

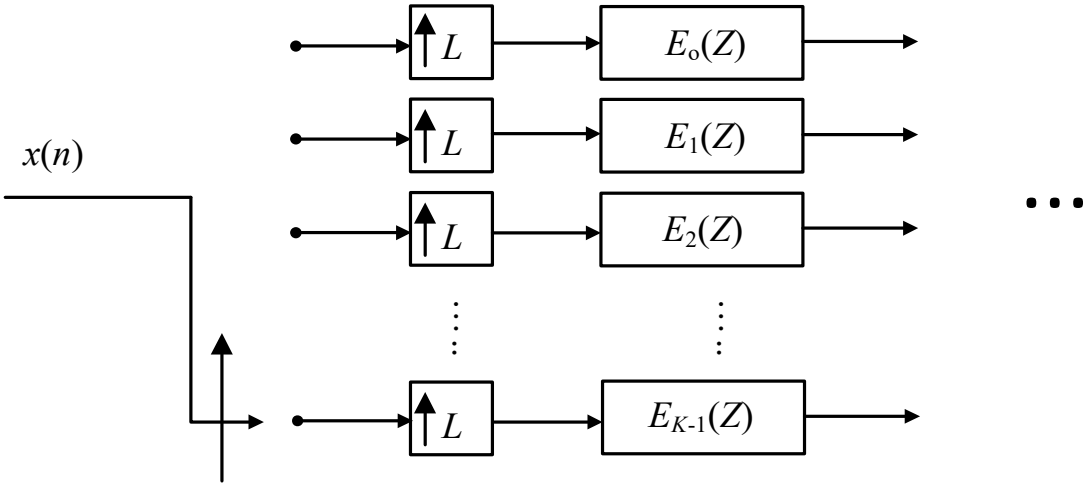

Figure 4.2 Commutator with interpolator in oversampled design

In this thesis, we developed oversampled polyphase filter-bank FPGA implementations based on an equivalent GDFT-FB model described by [7], because we can do extra reconfigurations based on it in order to implement oversampled GDFT-FB with IP cores. Furthermore we can also develop the odd stacked configuration based on it, as described in later sections.

4.2 Oversampled DFT-FB (even stacked)

4.2.1 High level design

The high level structure of the oversampled DFT-FB is very similar to the critically sampled designs shown in Figure 3.2.

Nevertheless, two significant changes must be applied to the filter-bank. First of all, since the up-sampling factor $L = K / D > 1$ in the oversampled case, the decimation factor no longer matches the number of channels. A design limitation of the FIR compiler IP core is that it requires the decimation factor and number of polyphase components to match. For this reason, the implementation of the FIR filtering in the filter bank must be redesigned for oversampling as described in the following section. The second change is to the output frequency shift state machine design. This requires modification since the oversampled signals result in additional possible multiplier values and hence, additional states in the state machine.

4.2.2 Oversampled polyphase decimation FIR

As shown in Figure 3.5, the FIR compiler IP core implements a critically sampled polyphase decimation structure as if using a commutator. Thus like a commutator, the decimation factor in the GDFT-FB must be equal to the number of channels. However, in an oversampled GDFT-FB, the decimation factor must be, by its definition, less than the number of channels (since K/D > 1). The question then, is how to implement an oversampled polyphase decimation FIR using the FIR compiler blocks when they are only available in critically sampled form?

Consider a filter bank where up-sampling factor $L = 2$. First, expand equation (3.4) into

$$H(z) = E_0\left(z^K\right) + z^{-1}E_1\left(z^K\right) + \ldots + z^{-(K-1)}E_{K-1}\left(z^K\right) \tag{4.2}$$

This can be re-written as follows (simply by dividing the terms into two groups)

$$\begin{aligned} H\left(z\right) = {} & E_0\left(z^K\right) + z^{-1}E_1\left(z^K\right) + \ldots + z^{-(K/2-1)}E_{K/2-1}\left(z^K\right) \\ & + z^{-K/2}\left[E_{K/2}\left(z^K\right) + z^{-1}E_{K/2+1}\left(z^K\right) + \ldots + z^{-(K/2-1)}E_{K-1}\left(z^K\right)\right] \end{aligned} \tag{4.3}$$

To generalize equation (4.3) from L=2 into L equals to any integer number, we note that $D = K/L$ from which it can be observed that the polyphase decomposition of the DFT-FB prototype filter in equation (4.3) can be re-written as

$$\begin{aligned} H\left(z\right) &= \sum_{i=0}^{L-1} z^{-iD}\left(\sum_{p=0}^{D-1} z^{-p}E_{p+iD}\left(z^K\right)\right) \\ &= \sum_{i=0}^{L-1} z^{-iD}H_{FIRi}\left(z\right) \end{aligned} \tag{4.4}$$

where

$$H_{FIRi}\left(z\right) = \sum_{p=0}^{D-1} z^{-i}E_{p+iD}\left(z^K\right) \quad i = 0,1,...,L-1 \tag{4.5}$$

With this decomposition, the channels are divided into L groups, defined by $H_{FIRi}(z)$, and there are D channels in each group. The benefit of this grouping is that the number of channels in every group is now equal to the down-sampling factor D. Thus in every group, a quasi-critically sampled decomposition is processed, which makes it compatible with FIR compiler IP Cores.

To see this in practice, consider a 2x oversampled 4-channel DFT-FB (as discussed in § 4.1.2). Using the original GDFT-FB structure, 2x oversampling is achieved in every channel by setting the appropriate decimation factor in each channel, $D=K/L$ (hence $D=2$ in this case). Now applying equation (4.4), the oversampled polyphase components may be grouped into L groups (in this case $L=2$) as shown in Figure 4.3(a) (right hand side).

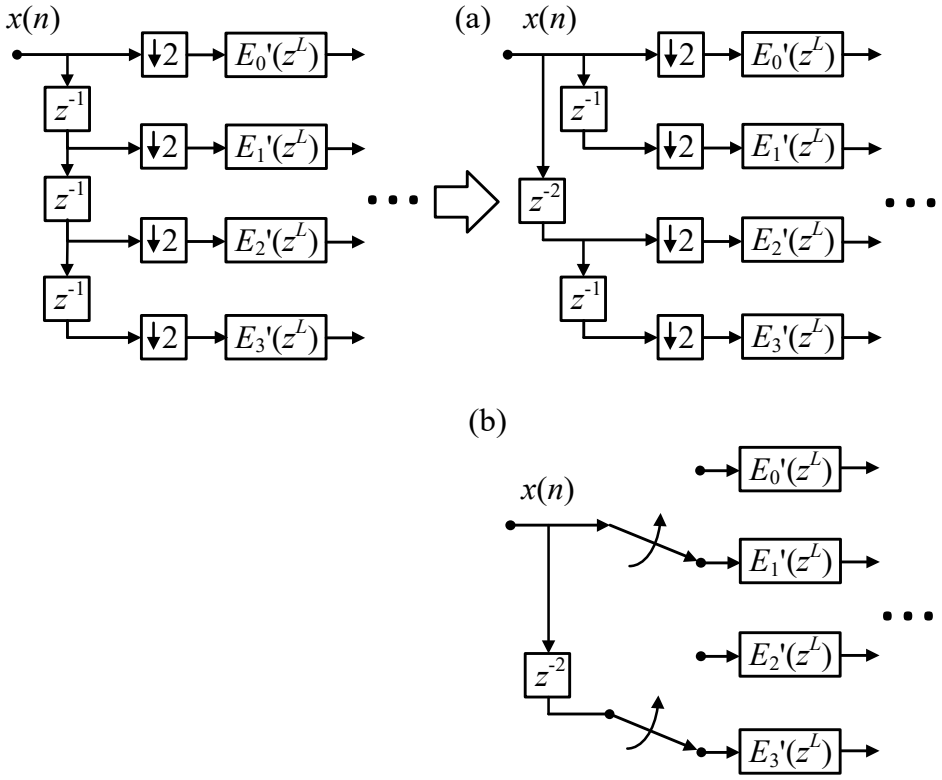

Figure 4.3 Converting 2x oversampled 4 channels GDFT-FB input distribution to (a) a functionally equivalent version based on equation (4.4) and (b) an equivalent version using commutators.

Figure 4.3(b) shows the grouped structure from Figure 4.3(a) but with the delay chain and decimator in every group converted to a commutator. Thus in every group a critically sampled filter bank is obtained. Taking another perspective, the whole system can be seen to achieve 2x oversampling because two commutators are taking the data at the input

sample rate in parallel, thus the whole system is getting twice the number of input samples simultaneously.

The general solution, therefore, is that the oversampled polyphase decimated FIR is implemented using L, the oversampling factor, number of polyphase decimation FIR blocks (real or complex as needed by the DFT-FB or GDFT-FB respectively). The block-specific prototype filter supplied to each of these FIR blocks is created from a subset of the interpolated polyphase components of the original prototype filter in accordance with equation (4.5). In FPGA implementation, each FIR compiler block can only be inserted with the interpolated coefficients of polyphase filters in this group. Normally most applications will only require an oversampling factor of L=2, because oversampling by 2 can already greatly reduce aliasing with adjacent channels. It is likely to be wasteful to oversample by more than this (unless an oversampled output is required for other reasons such as timing synchronisation) as more FIR compiler blocks and more computations will generally be required.

4.2.3 FIR block output samples rearrangement for the FFT

Since each of the FIR blocks executes and produces its outputs in parallel, it is necessary to add an FIR selector state machine which implements the time division multiplexing of multiple FIR block outputs onto the single I and Q inputs to the FFT IP core. Specifically, to implement the DFT-FB or GDFT-FB correctly, the oversampled FIR outputs from branch 0 to branch K of the overall polyphase decomposition must be supplied to the FFT sequentially. First the D samples from 0 to D-1 are selected from FIR_1, then D samples from D to 2D-1 are selected from FIR_2, and so on, until the final D samples from (L-1) D to LD-1 are selected from FIR_L at which point the sequence begins again. An example of an FIR output selector designed for two 4-channel blocks (that is an 8-channel filter bank with an up-sample factor of 2 is shown Figure 4.4.

Figure 4.5 shows the high level implementation of the oversampled polyphase decimation FIR using multiple FIR blocks (based on FIR compiler IP cores), FIFO buffers, and a FIR selector state machine.

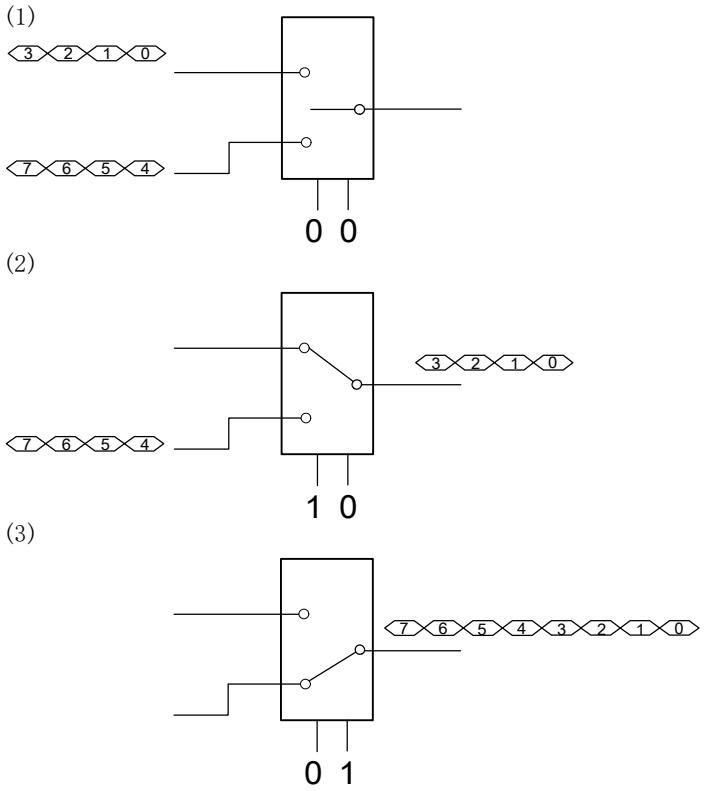

Figure 4.4 FIR selector state machine mapping the output of two FIR blocks to a single TDM output suitable for input to the FFT IP core

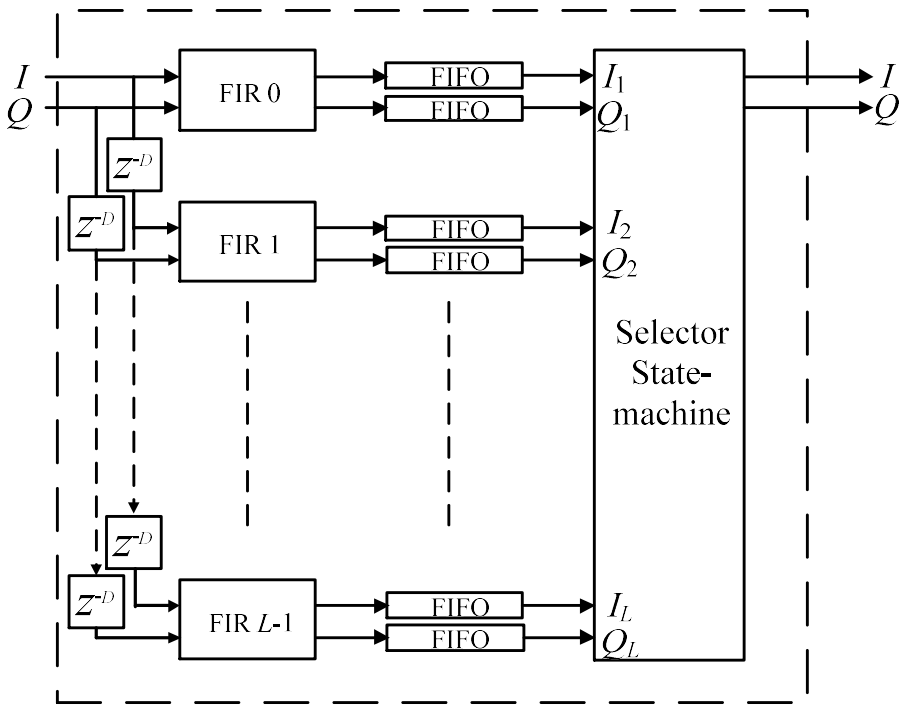

Figure 4.5 Oversampled polyphase decimation FIR implemented using real or complex critically sampled polyphase decimation FIR blocks (based on the FIR compiler IP core)

4.2.4 Oversampled frequency shift state machine

As was the case for the critically sampled GDFT-FB, the sub-band outputs from the DFT (FFT) block require a frequency shift to re-centre each extracted channel on DC. As before, this final frequency shift can be implemented using a state machine, albeit one with more states. The number of states depends on the oversampling factor $L = K / D$. Substituting $k_0 = 1/2$ and $K = LD$ into equation (3.17) yields

$$W_K^{-k_0 nD} = e^{-j\pi n/L} \quad n \in \mathbb{N} \tag{4.6}$$

In the case that $L = 2$ this reduces to just four unique values

$$W_K^{-k_0 nD} = \begin{cases} 1 & n = 4m \\ -j & n = 4m+1 \\ -1 & n = 4m+2 \\ +j & n = 4m+3 \end{cases} \quad n \in \mathbb{N},\ m \in \mathbb{N} \tag{4.7}$$

Therefore the required frequency shift operation can be replaced by a state machine with 4 states. All 4 of these multiplications can be implemented without any multipliers since the operations amount to passing through, negating, or swapping the I and Q components.

Similar state machines can be derived for larger oversampling factors but it is worth noting that some multipliers will be required in this case which is perhaps another reason to consider avoiding higher oversampling factors.

The procedure of how state machine works is shown by Figure 4.6. The state machine has two paths of pins for I and Q components respectively. A counter is used to count the number of inputs taken by the state machine. The port 'dv' of FFT core indicates that FFT core is outputting the result samples, and the state machine starts receiving samples at this moment. If the state machine is running in first round of K samples $x(0)$~$x(K\text{-}1)$ (one sample from each channel, they are all 'n'th sample in each sub-band), it will multiply 1 to these samples. If the state machine is running in samples $x(K)$~$x(2K\text{-}1)$, it will multiply $-j$ to these samples. If the state machine is running in samples $x(2K)$~$x(3K\text{-}1)$, it will multiply -1 to these samples. If the state machine is running in samples $x(3K)$~$x(4K\text{-}1)$,

it will multiply j to these samples. After the state-machine processed $4K$ samples, the counter will start over again.

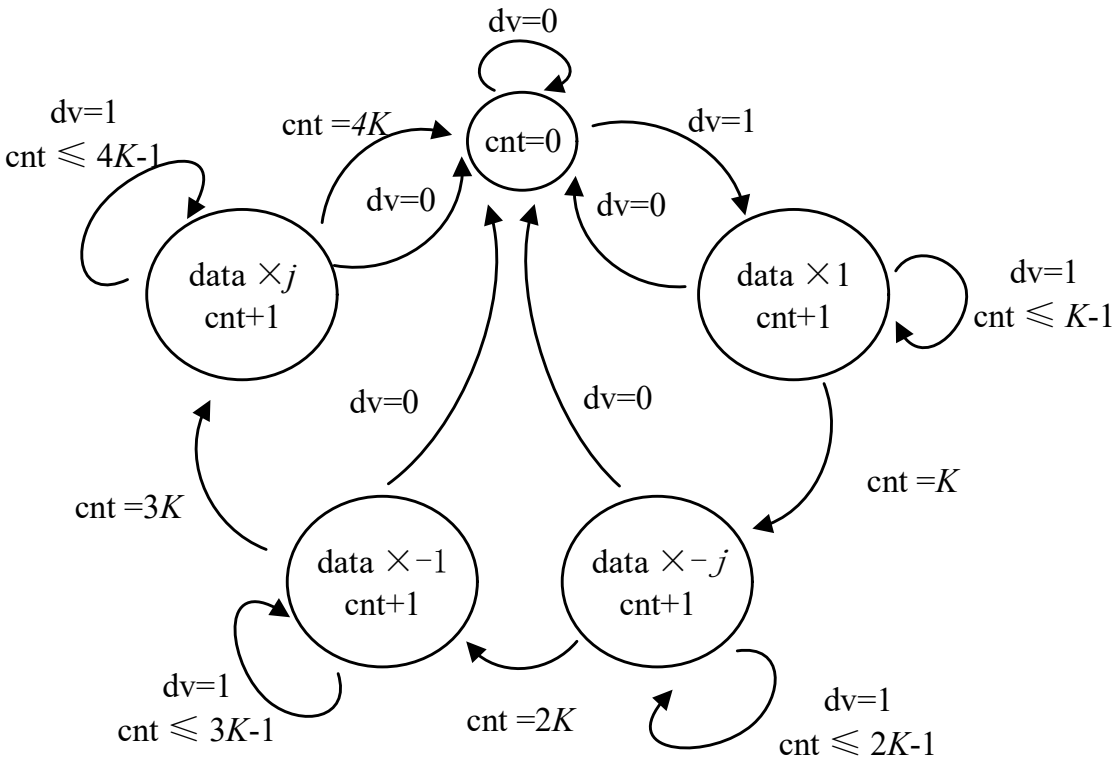

Figure 4.6 Frequency shifting state machine work flow

In FPGA implementation, multiplication by 1, is done by passing through the input. Multiplication by -1, is done by negating both I and Q components using 'inversion and adding one' as mentioned in § 3.2.2.3. Multiplication by j, is done by negating Q component and then swap pins between I and Q. Multiplication by $–j$, is done by negating I components and then swap pins between I and Q. After 32 samples go through the state machine, the counter will reset to 0. The state machine will have the same 4 states for the next 32 samples as a loop.

4.2.5 Final FPGA design

The 2x oversampled DFT-FB FPGA diagram is illustrated in Figure 4.7.

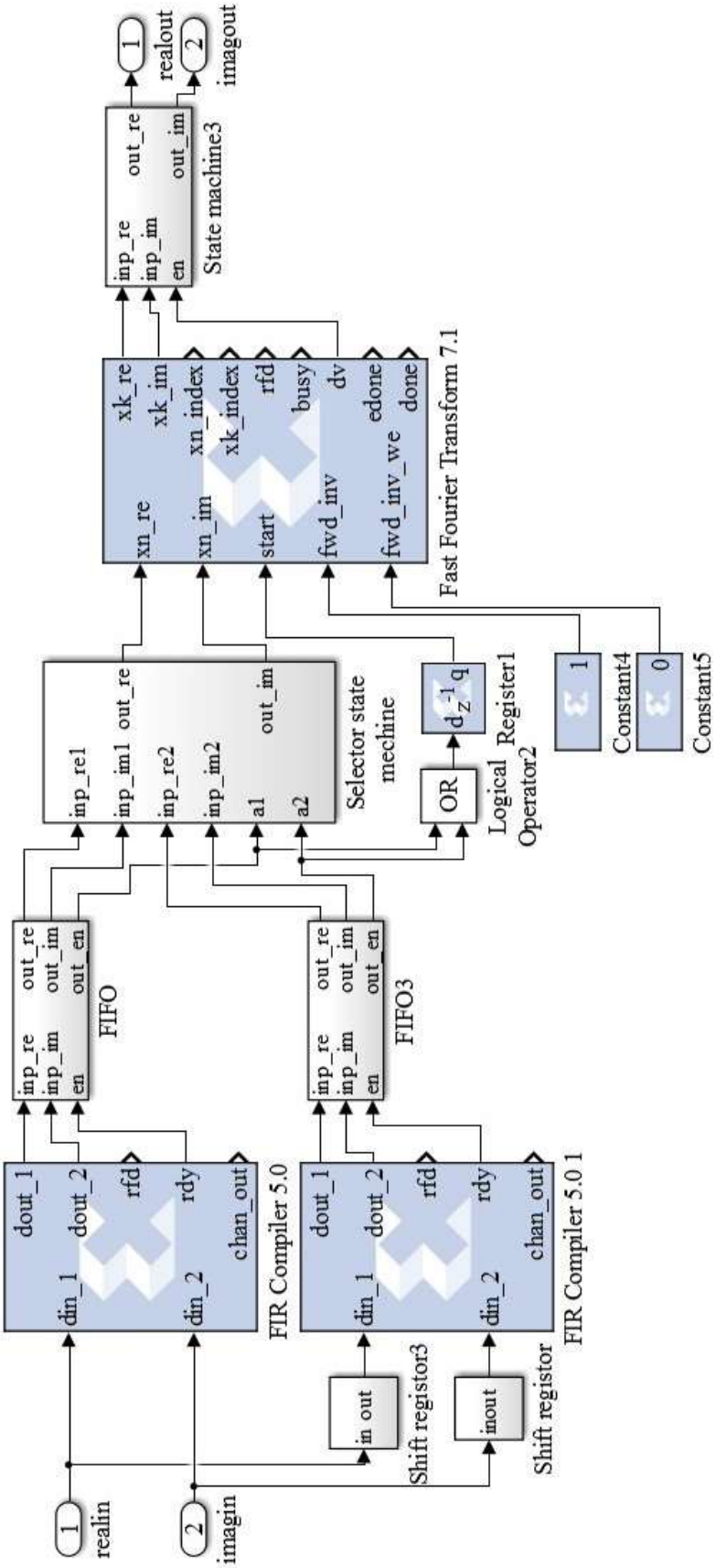

Figure 4.7 the FPGA architecture diagram of 2x oversampled DFT-FB (even stacked)

In Figure 4.7 the delay, Z^{-D}, required between the first and second FIR blocks (see Figure 4.5) is implemented using shift registers (one each for the I and Q components). The first FIR block contains the filter coefficients for polyphase components from $E_0(z^L)$ to $E_{K/2-1}(z^L)$. The remaining filter coefficients are used in the second FIR block. As in the oversampled case, every channel's filter is given by the polyphase component $E_p(z^L)$, there is an L=2 up-sampling factor is applied, thus FIR IP cores are inserted with coefficients padded by zero. The FIFO and selector state machine doing exactly the same thing as in § 4.2.3. It is noticeable that, in the selector machine the outputs are assigned to a register first, though it will have one clock cycle delay to the output. The benefit is that, the output waveforms' edge will be synchronized with the registered 'start' signal of FFT core, to prevent taking the unsafe data to FFT core. The 'start' signal is asserted by the AND operation results of both 'rdy' value of FIR compiler. The rest of the system are almost the same as the critically sampled design, except the state machine has 4 states rather than 2.

4.3 Oversampled GDFT-FB (odd-stacked)

In chapter 3, a critically sampled oddly stacked GDFT-FB was developed. An oversampled GDFT-FB design could produce a channelizer that is more appropriate for systems with odd-stacked channels and suffers less from aliasing in output sub-bands. An expected disadvantage (relative to the even-stacked DFT-FB) is that the odd-stacked design will require more multiplication operations, because the filtering of the FIR compiler is carried out using complex coefficients. In the FPGA implementation, this means adding extra complexity and increased resource usage, because complex signals in the FPGA needed to be separated into I and Q components and filtering will require double the number of FIR compiler in a cross-coupled structure as introduced in the previous chapter.

4.3.1 High level design

The overall structure of the oversampled GDFT-FB design is the same as even-stacked design. The frequency shift state machine is the same for the odd stacked GDFT-FB as it

is for the even stacked DFT-FB. The biggest difference between the odd stacked and even stacked design is that coefficients in the FIR blocks are no longer in real values, and hence a complex FIR design is required to respectively store I and Q components in two FIR compilers similar in last chapter will be applied again.

4.3.2 Oversampled complex polyphase decimation FIR blocks

Similar to the odd-stacked critically sampled GDFT-FB (see previous chapter) the oversampled GDFT-FB requires FIR blocks which can use complex valued coefficients. These coefficients result from the complex modulation of the filter coefficients required to implement the phase and frequency shifts of the GDFT, as in equation (3.9).

We apply the same grouping as in section 4.2.2 to give

$$H_{FIRi}(z) = \sum_{p=0}^{D-1} z^{-i} W_K^{-kp} W_K^{-\frac{1}{2}p} E_{p+iD}\left(z^K W_K^{-\frac{1}{2}D}\right) \quad i = 0,1,...,L-1 \qquad (4.8)$$

The coefficients are once again complex, because of the complex modulation. Therefore, the cross coupling filtering structure like Figure 3.9 is required for every single group (FIR0, FIR1, FIR2…..) in Figure 4.5. As an example, consider an 8-channel 2x oversampled odd stacked GDFT-FB. In this example there would be 2 FIR blocks in the system because of the 2x oversampling. The 8 channels (and hence 8 polyphase components of the prototype filter) would be divided into two groups and each group implemented by one of the FIR blocks. Finally each FIR block would be implemented using two cross-coupled FIR compiler IP cores due to the complex valued coefficients. Therefore, it can be seen that the general design requires $2L$ number of FIR compiler cores for an integer oversampling factor L.

4.3.3 Final FPGA design

An odd-stack 2x oversampled GDFT-FB FPGA diagram is shown in Figure 4.8.

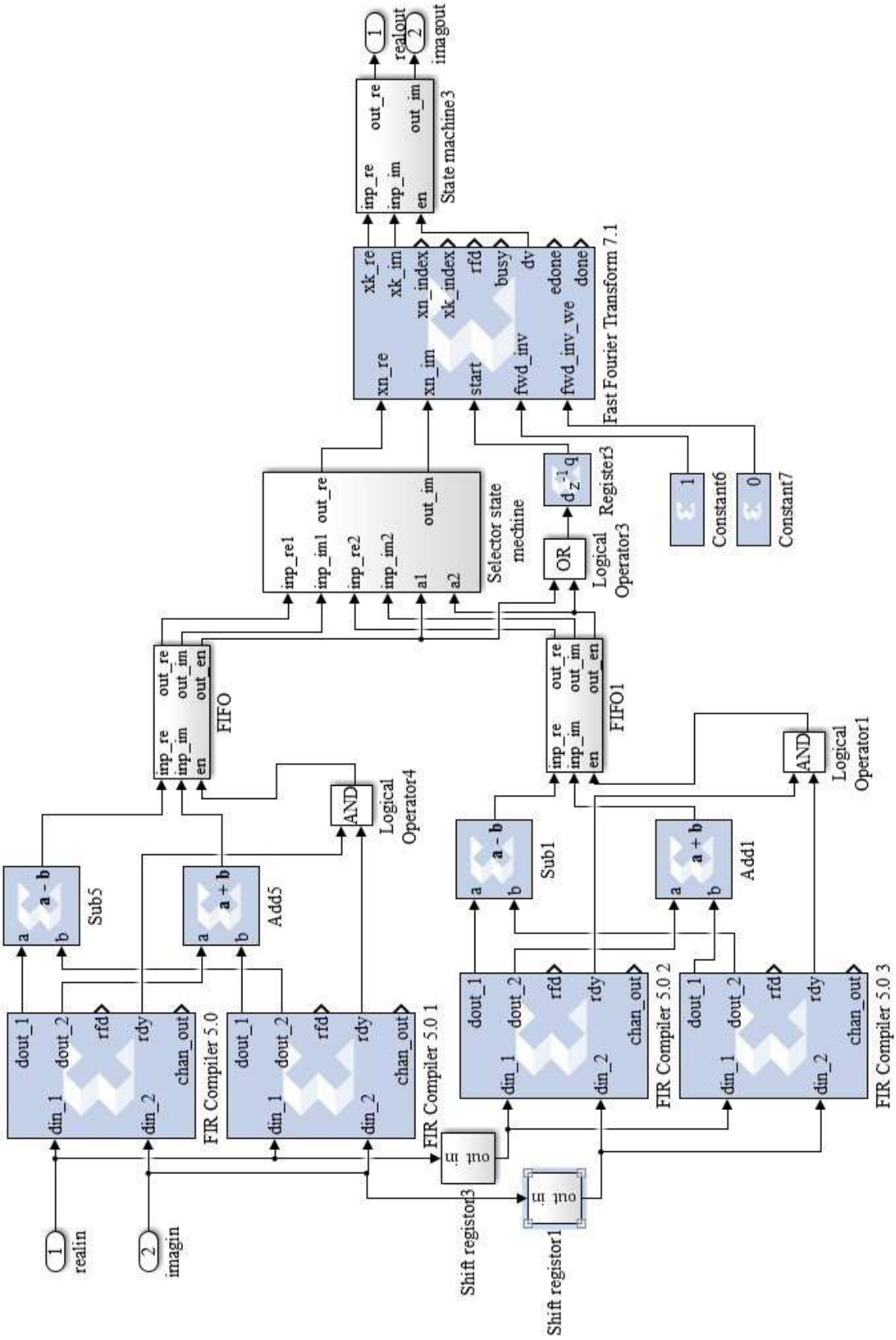

Figure 4.8 the FPGA architecture diagram of 2x oversampled GDFT-FB (even stacked)

In this diagram, FPGA based odd stacked 2x oversampled GDFT-FB occupies the similar structure to the even stacked design as developed in § 4.2.5. The only difference is that, each FIR block in Figure 4.5 employs a 'cross-coupled' FIR compile block in order to make the FIR block compatible with complex coefficients.

4.4 FPGA implementation evaluation

Two 16-channel 2x oversampled filter banks were implemented: one using the FPGA based DFT-FB (even-stacked) and the other using the FPGA based GDFT-FB (odd stacked). As in the previous chapter, the evaluation focused on sub-band frequency response, EVM performance, adjacent channel interference, and resource usage. The simulation setup, based on TETRA 25 kHz channels, was also the same as the previous chapter. The input signal, output signal, and all internal results used 16-bit resolution and fixed point arithmetic.

The prototype filter for both even and odd stacked implementations had 576 coefficients.

Apart from different sub-band allocations the even and odd-stacked implementations have equivalent DFP performance. For this reason only odd-stacked oversampled GDFT-FB filtering result will be discussed. However both implementations will be considered in terms of hardware usage.

4.4.1 Frequency response

The frequency response of a single sub-band of the FPGA-based 2x oversampled odd-stacked GDFT-FB is shown in Figure 4.9. The output of each sub-band is oversampled and therefore the output signal occupies only half of the spectrum, because it is a 2x oversampled design.

It can be seen that the 16-bit fixed point quantization has an impact on the stopband attenuation performance relative to the floating point reference implementation. The FPGA based implementation has around 5 dB less stopband attenuation than the reference implementation. Nevertheless, the FPGA based implementation still meets the TETRA requirement of more than 55 dB attenuation in the stop band.

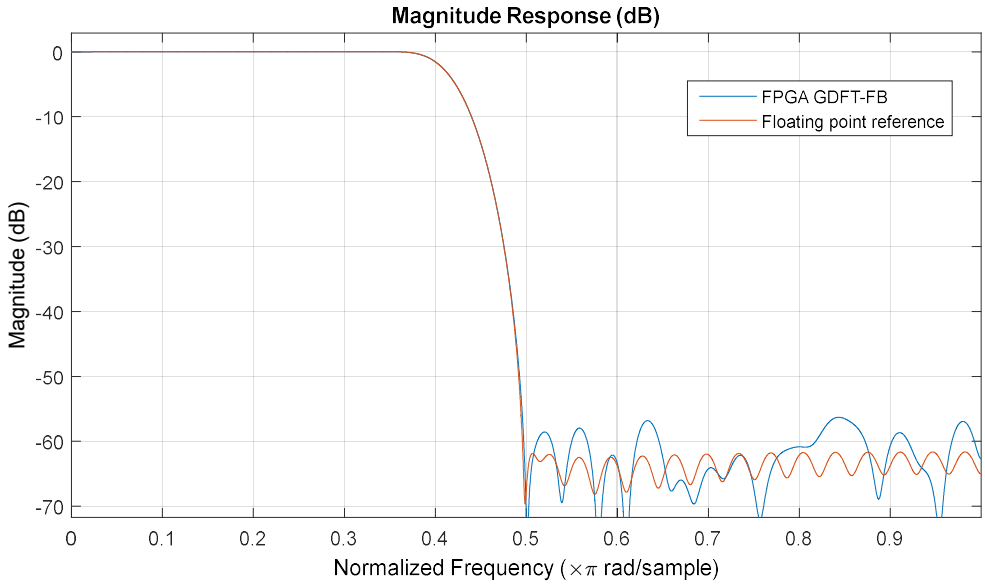

Figure 4.9 Frequency response of one sub-band of the FPGA-based 16-bit 16-channel 2x oversampled GDFT-FB. The FPGA based (fixed point) response (blue) and floating point GDFT-FB reference implementation (red) are both shown.

From the perspective of passband, as shown in Figure 4.10, due to fixed point quantization, the FPGA based GDFT-FB produces slightly more passband ripple (0.06 dB) than the floating point reference implementation (0.01 dB), but it is still much smaller than the 2 dB requirement.

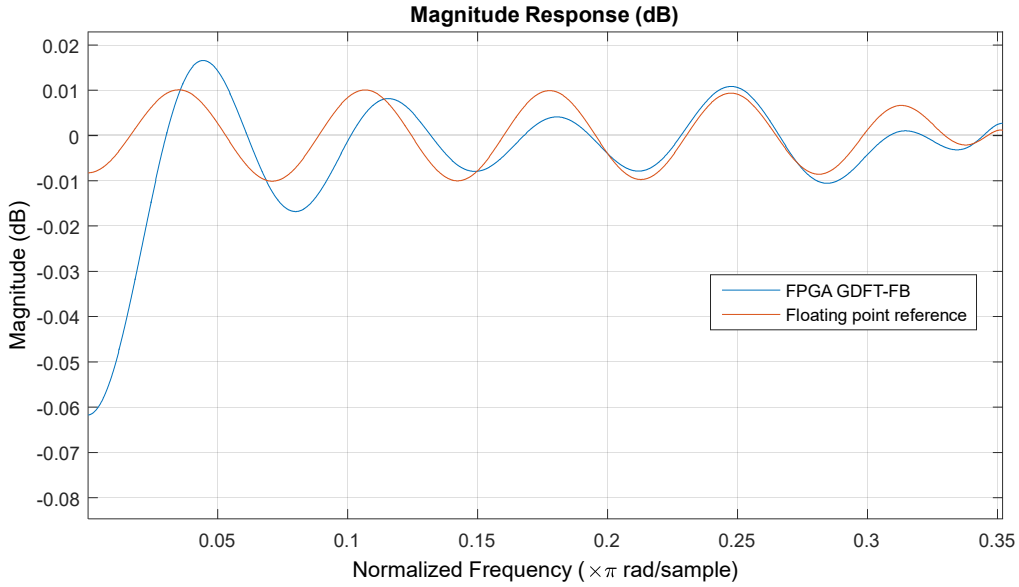

Figure 4.10 Passband comparison between 16-bit FPGA GDFT-FB (blue line) and its floating point reference (red line)

4.4.2 EVM result

As in § 3.3.2.2, we evaluate the EVM of the 16-bit fixed point FPGA based implementation's phase noise, I-Q imbalance, and filter distortion.

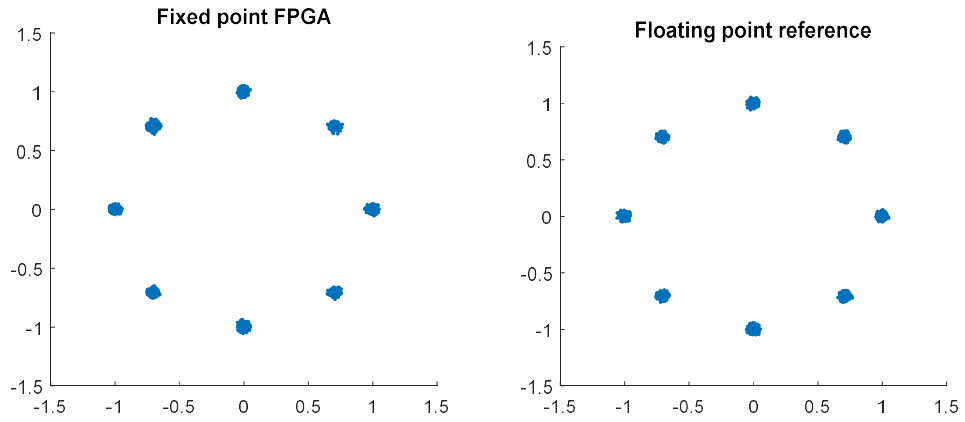

Figure 4.11 The π/4 DQPSK modulation constellation of the FPGA based 2x oversampled GDFT-FB output (left), and the equivalent constellation of the floating point GDFT-FB reference output (right)

Figure 4.11 shows the π/4 DQPSK modulation constellation of the output from the 2x oversampled FPGA based GDFT-FB, and the π/4 DQPSK modulation of the output from the reference floating point GDFT-FB. The EVM result shows that FPGA based implementation performs very similarly to the floating point reference at high signal levels. The numerical results are listed in Table 4.1.

Table 4.1 The EVM performance of an FPGA-based 16-channel 2x oversampled GDFT-FB

	GDFT-FB on FPGA	**Floating point FPGA**	**Limit of TETRA**
Peak	0.0697	0.0662	0.3
RMS	0.0292	0.0292	0.1

4.4.3 Adjacent channel interference

The adjacent channel interference characteristics of the FPGA-based 2x oversampled 16 channel GDFT-FB were also evaluated. Table 4.2 shows the EVM results evaluated at carrier to adjacent channel interfere levels of -10 dB, -20 dB, -30 dB, -40dB, -45dB and -50dB. As expected, the EVM gets worse as the adjacent channel interference level

increases. When the channel of interest power is -45 dB relative to the adjacent interferer, the RMS and peak EVM is still within than the TETRA requirements (0.1 and 0.3 respectively). Only at -50 dB (which exceeds the requirements of TETRA), does the RMS EVM become slightly greater than 0.1. Even here, however, the peak EVM is still less than 0.3.

Table 4.2 EVM result of FPGA 2x oversample GDFT-FB under different adjacent channel interference level

C/I_a (dB)	**RMS**	**PEAK**
-10	0.0465	0.1165
-20	0.0466	0.1207
-30	0.0474	0.1105
-40	0.0569	0.1549
-45	0.0756	0.2025
-50	0.1154	0.2879

4.4.4 Hardware resource usage

The 2x oversampled FPGA-based DFT-FB and GDFT-FB designs are more complicated than the critically sampled designs examined in the previous chapter. More hardware resources can be expected to be required to deal with the following: extra FIR blocks required by the oversampled design; a larger number of coefficients in each block (because the coefficients are interpolated by the oversampling factor and there is no specific optimization in the FIR compiler IP core for zero value coefficients); and finally, the slightly more complicated frequency shifting state-machine. Table 4.3 shows the FPGA hardware resource usage of both even and odd stacked 2x oversampled designs in comparison to a conventional per-channel approach. The result shows that odd stacked 2x oversampled GDFT-FB will use slightly less than twice the number of DSP48s and Block RAM 18s than the even-stacked DFT-FB version, but still much less than a per-channel approach.

Table 4.3 Even and odd stacked 2x oversampled GDFT-FB FPGA resources usage

Filter-bank Type	Register	LUTs	Block RAM 36	Block RAM 18	DSP48s
2x Even DFT-FB	1445	1503	0	7	8
2x Odd GDFT-FB	2121	1570	0	12	15
Per-channel approach	3993	2123	4	48	80
Available	301440	150720	416	832	768

4.5 Chapter conclusion

In this chapter, the FPGA-based oversampled configuration of the DFT-FB and GDFT-FB were designed and evaluated. Such oversampled filter banks are useful to avoid aliasing (which can be a problem for signal reconstruction) or to allow sub-band filters which overlap in frequency.

The oversampled designs that were implemented are closely related to critically sampled designs implemented in the previous chapter. The high level block diagram does not change very much. The only changes are a reduction in the decimation factor applied to the input of the polyphase components and an interpolation of the polyphase components themselves.

However the FPGA-based implementation requires more change than the high level block diagrams would suggest. First of all, a specific solution was developed to support an oversampled polyphase decomposition, because the FIR compiler IP core can only process a critically sampled polyphase decomposition. The solution developed converted the single oversampled polyphase decomposition into a number, L, (e.g. 2 for 2x oversampling) of critically sampled polyphase decompositions running in parallel whose outputs are scheduled appropriately for the DFT by a selector state machine, because it was desirable to continue using the optimized and well tested FIR compiler IP core. Finally, the frequency shift state machine required for the GDFT-FB (odd stacked) had to be modified to allow for more states, specifically 4 states in the case of a 2x oversampled design.

In the evaluation section, the results show that a FPGA-based 16-bit 16 channel 2x oversampled GDFT-FB can achieve essentially the same frequency response, EVM performance and adjacent channel interference resistance as a critically sampled FPGA-based GDFT-FB. As was the case for the critically sampled filter banks evaluated in the previous chapter, the odd-stacked configuration (GDFT-FB) uses more hardware resources than the even-stacked configuration (DFT-FB). However, with the oversampled designs this difference is even more pronounced.

Chapter 5

FRM and the GDFT-FB

5.1 Full FRM applied to the GDFT-FB

5.1.1 Introduction

The FRM approach to filter design can be used to design sharp filters using a cascade of simpler filters with fewer overall coefficients and less stringent design requirements than a single filter would require. For this reason, FRM has been used to implement filter-banks such as the QMFB [59] and CMFB [60-62], with the latter being more commonly implemented.

Basic FRM techniques for filter design were described in some detail in chapter 2. Based on the efficient FRM design discussed in § 2.3.1, a new GDFT based design using FRM technology developed in [53, 63], is introduced here for implementation in FPGA form. The basic idea is to replace both the masking filters of the normal FRM structure, $H_{Ma}(z)$ and $H_{Mc}(z)$, with the GDFT-FB to reduce the complexity of calculation from the direct filtering.

Similar to the FRM based CMFB [64], the full FRM DFT-FB is based on the efficient FRM design with polyphase decomposition shown in Figure 2.17 as well. It is worth mentioning that this efficient design requires a Subclass I filter response (magnitude complementary response) in the base and complementary masking filters. Based on the equations (2.4),(2.5) and (2.6), the prototype filter $H(z)$ of the GDFT-FB can now be expressed in FRM form as:

$$\begin{aligned} H(z) = {} & H_{a0}(z^{2L})H_{Ma}(z) + z^{-L}H_{a1}(z^{2L})H_{Ma}(z) \\ & + H_{a0}(z^{2L})H_{Mc}(z) - z^{-L}H_{a1}(z^{2L})H_{Mc}(z) \end{aligned} \tag{5.1}$$

To simplify this equation, we define

$$\begin{aligned} A(z) &= H_{Ma}(z) + H_{Mc}(z) \\ B(z) &= H_{Ma}(z) - H_{Mc}(z) \end{aligned} \tag{5.2}$$

Then substituting $A(z)$ and $B(z)$ into (5.1) yields

$$H(z) = H_{a0}(z^{2L})A(z) + z^{-L}H_{a1}(z^{2L})B(z) \tag{5.3}$$

Similar to the manner in which the low pass prototype filter is modulated to create the sub-band bandpass filters of the GDFT-FB in equation (3.10), the complex modulation can be applied in the FRM case to create the sub-band filters $H_k(z)$ here also as follows

$$H_k(z) = H_{a0}(z^{2L})A_k(z) + z^{-L}H_{a1}(z^{2L})B_k(z) \tag{5.4}$$

Here, $A_k(z) = A(zW_K^k)$ and $B_k(z) = B(zW_K^k)$. Applying the polyphase decomposition to both $A_k(z)$ and $B_k(z)$ similar to the treatment in chapter 3 yields:

$$\begin{aligned} A(z) &= \sum_{i=0}^{K-1} z^{-i}E_{Ai}(z^K) \\ B(z) &= \sum_{i=0}^{K-1} z^{-i}E_{Bi}(z^K) \end{aligned} \tag{5.5}$$

At last, the bandpass filter in each sub-band of the FRM GDFT-FB can be expressed as:

$$\begin{aligned} H_k(z) = {} & H_{a0}(z^{2L})\sum_{i=0}^{K-1} z^{-i}W_K^{-ki}E_{Ai}(z^K) \\ & +(-1)^k z^{-L}H_{a1}(z^{2L})\sum_{i=0}^{K-1} z^{-i}W_K^{-ki}E_{Bi}(z^K) \end{aligned} \tag{5.6}$$

The whole structure of FRM GDFT-FB is illustrated in Figure 5.1. L_{DFT} is the oversampling factor of the GDFT-FB which is defined as:

$$L_{DFT} = \frac{K}{D} \tag{5.7}$$

As can be seen in the Figure 5.1, there is a phase shift of π in the output of every second sub-band from the DFT on the H_{a1} path of the filter bank. This phase shift is mathematically equal to -1. Thus these sub-band outputs should be subtracted from (rather than added to) the outputs from the DFT on the H_{a0} path of the filter bank. This operation corresponds to the $(-1)^k$ in equation, where k refers to the sub-band index.

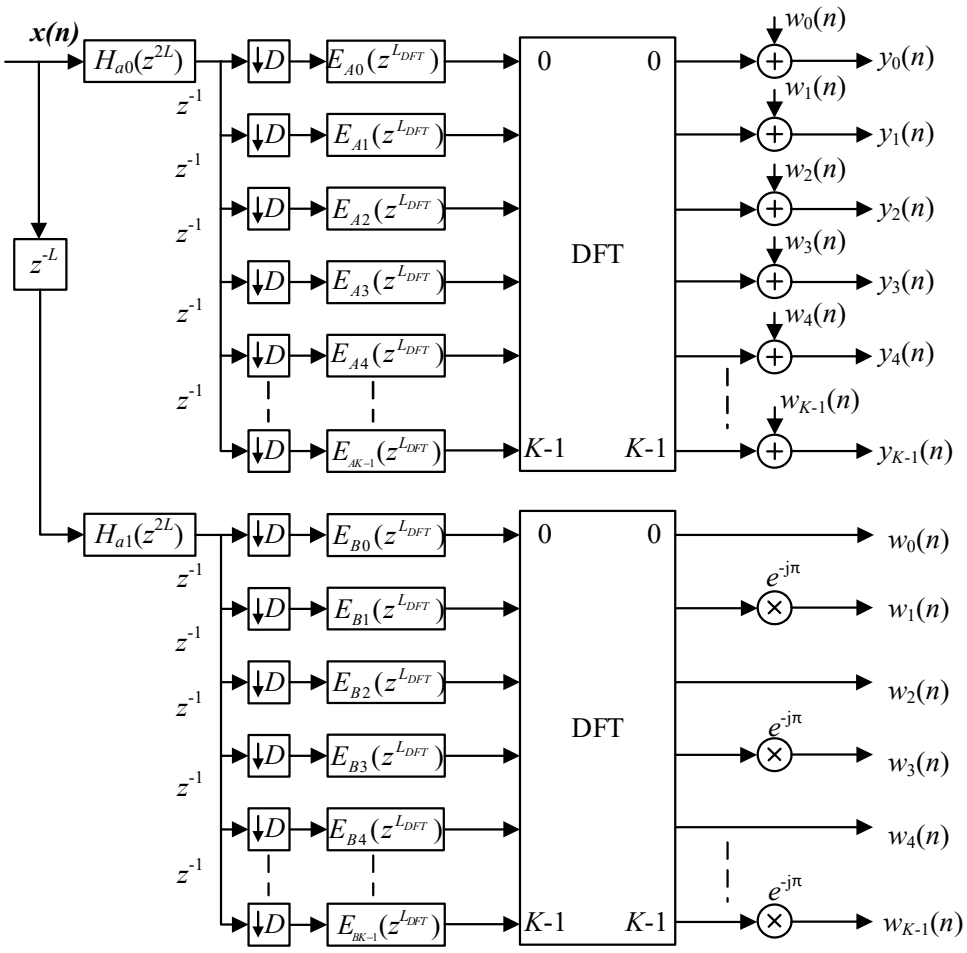

Figure 5.1 Full FRM DFT-FB

5.1.2 The FPGA based full FRM DFT-FB (even stacked)

5.1.2.1 The high level FPGA design

The diagram of an FPGA even stacked full FRM DFT-FB in shown in Figure 5.2.

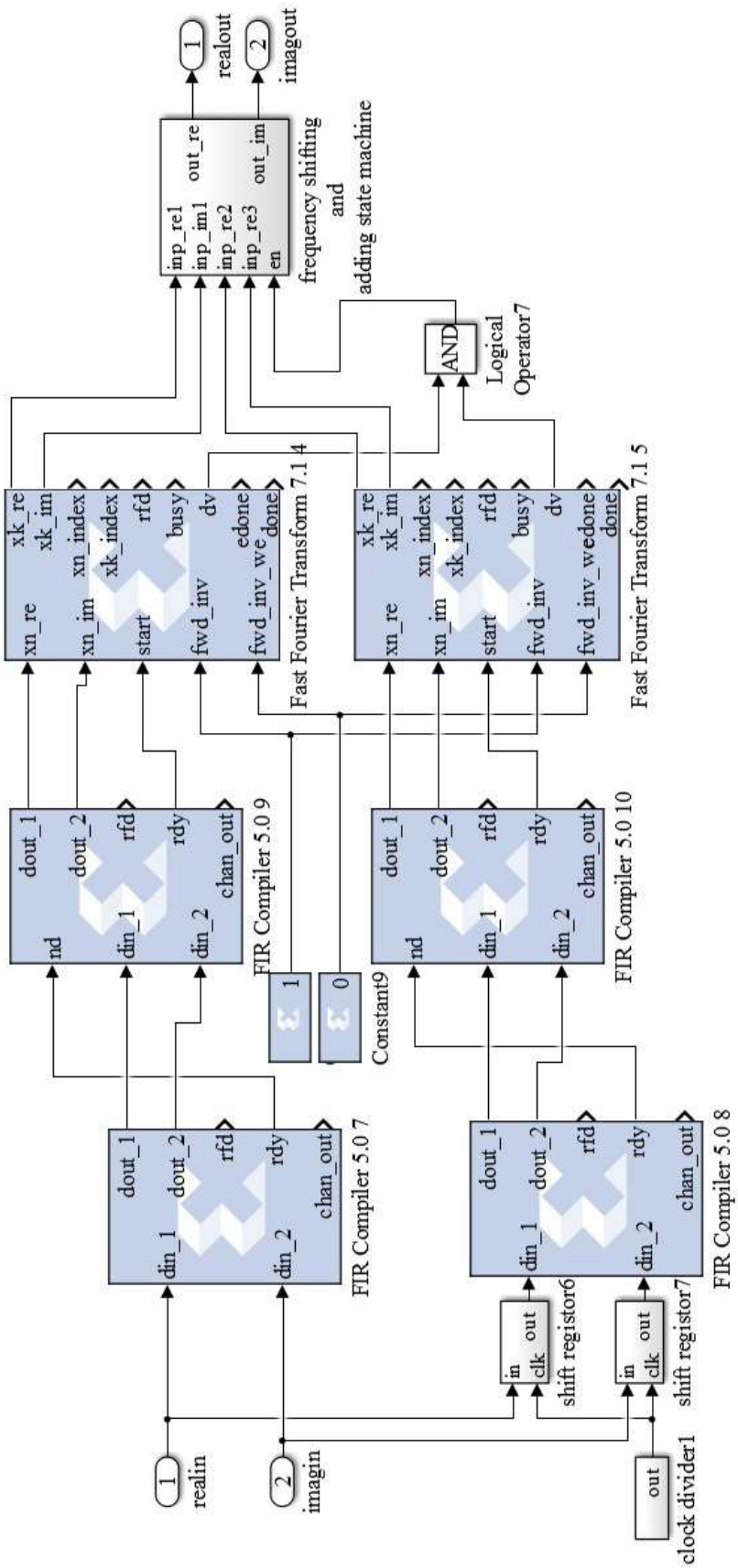

Figure 5.2 The FPGA based even stacked full FRM DFT-FB

The high level design of the DFT-FB with full FRM technology is more complicated than the normal DFT-FB design from chapter 3. First of all, there are two parallel DFT-FB units in the system as the two paths of a full FRM design, as illustrated in Figure 5.1. In addition, more stages will be required in the FPGA architecture, specifically the polyphase decomposed base filter, phase shifting in the output of the second path, and the addition of both DFT outputs to produce the overall channelizer output. It is also necessary to add appropriate delays to ensure the final outputs on each path are synchronized for addition.

5.1.2.2 The delay of second path design with an arbitrary fractional clock divider

The delay z^{-L} prior to the second path of the full FRM design shown in Figure 5.1 subjects the second path to an L sample period delay. The sampling rate of input wideband signal may range from 200 KHz (for an 8 x 25 kHz sub-band channelizer) to the order of a few MHz, according to the filtering specification and number of channels. If we just use a shift register to delay the input for L sample periods triggered by system clock, the required shift register depth will be $L \times clock\_rate/sample\_rate$. In this case L is the interpolation factor for up-sampling the base filter described in § 2.3, and not an oversampling factor for the filter bank itself. The interpolation factor ranges from 10 ~ 100. Thus, the shift register depth on the separate I and Q paths will range from 10's to 100's of thousands. As a result, just directly applying shift registers leads to a huge waste of the register resources.

One of the rational solutions to have an efficient usage of register resources is to trigger the shift register at the input sample rate rather than the system clock rate. Therefore a clock divider is introduced in this work.

The clock divider divides the system clock down to the input sample rate. A simple clock divisions, such as dividing by the value of power of 2 could be realised using a cascading D flip-flops structure. Dividing by arbitrary integer values could simply employ a counter, which outputs the divided clock when it counts to the certain value, or changes state when it reaches half of the value to achieve a 50% duty cycle [65]. Once a non-integer divider is need, however, a different design will be required.

A factional-n clock divider introduced in [66] can be applied to the application here. Assuming the frequency of original clock is N and the desired low speed clock's frequency is D, then the required divider is N/D (since N / (N/D) = D). After the dividing operation, the quotient Q and remainder R can be acquired. A fractional clock division is implemented by combining R number of (Q+1) times dividing and (D-R) number Q times dividing. It is not wise to just put R divided by (Q+1) clock at the beginning, and put (D-R) at the end. To achieve an average between two different frequency clocks, the method is as follows: firstly employ a register m cumulated R at each fast clock raises. Then if the value is less than the value of D, the divider divides the fast clock by Q+1. Otherwise, the divider divides the fast clock by Q. At the same time, subtract m by the value of D. The duty cycle of the divided clock ranges from Q/(2Q+1) to (Q+1)/(2 Q+1). So with a greater value of Q, the duty cycle will be closer to 50%.

At last, the clock divider provides clock triggering the shifting register at the input sample rate, thus the depth of the shift register can be designed to L instead of $L \times clock\_rate/sample\_rate$, which saves a lot of register resources.

5.1.2.3 Polyphase decomposed base filter

Each polyphase component of the base filter is implemented using an FIR compiler IP core. The coefficients of the two polyphase components are chosen in accordance with § 3.2.1.1, following the same procedure as for the prototype filter of the DFT-FB. The notation $H_{a1}(z^{2L})$ indicates an interpolation by 2L, that is, 2L-1 zeros padding are needed between filter coefficients.

5.1.2.4 Phase shifting and addition state machine

The phase shift by $e^{-j\pi}$ in every second sub-band in the H_{a1} path output shown in Figure 5.1, corresponds to negating every output sample on these sub-bands. The final filter bank output result is obtained by adding the H_{a1} path outputs to the H_{a0} path outputs. Remembering that the output of the DFT blocks is time division multiplexed as a serial stream, the addition of output sub-bands with every second sub-band on the H_{a1} path negated, can be efficiently implemented by a state machine controlled addition. The state machine receives samples from both paths of the filter bank. It adds samples from matching even numbered sub-bands but subtracts samples from odd numbered sub-

bands. This reduces the number of hardware resources and processing delay required (in comparison to implementing the operations exactly as shown in Figure 5.1).

5.1.3 The FPGA based full FRM GDFT-FB (odd stacked)

In the last section an even-stack full FRM DFT-FB was developed. Similar to previous chapters, an odd-stacked variant of the filter bank based on the GDFT-FB may be designed. As usual, this introduces the requirement for complex filter coefficients which complicates the filter bank implementation.

The odd-stacked configuration [53] needs the modification similar to that in chapter 3. This is achieved by applying the same complex modulation as that in equation (3.10) to the masking filter's polyphase elements $A_k(z)$ and $B_k(z)$ as:

$$\begin{aligned} A_k(z) &= W_K^{-(k+k_0)n_0} A(zW_K^{(k+k_0)}) \\ B_k(z) &= W_K^{-(k+k_0)n_0} B(zW_K^{(k+k_0)}) \end{aligned} \tag{5.8}$$

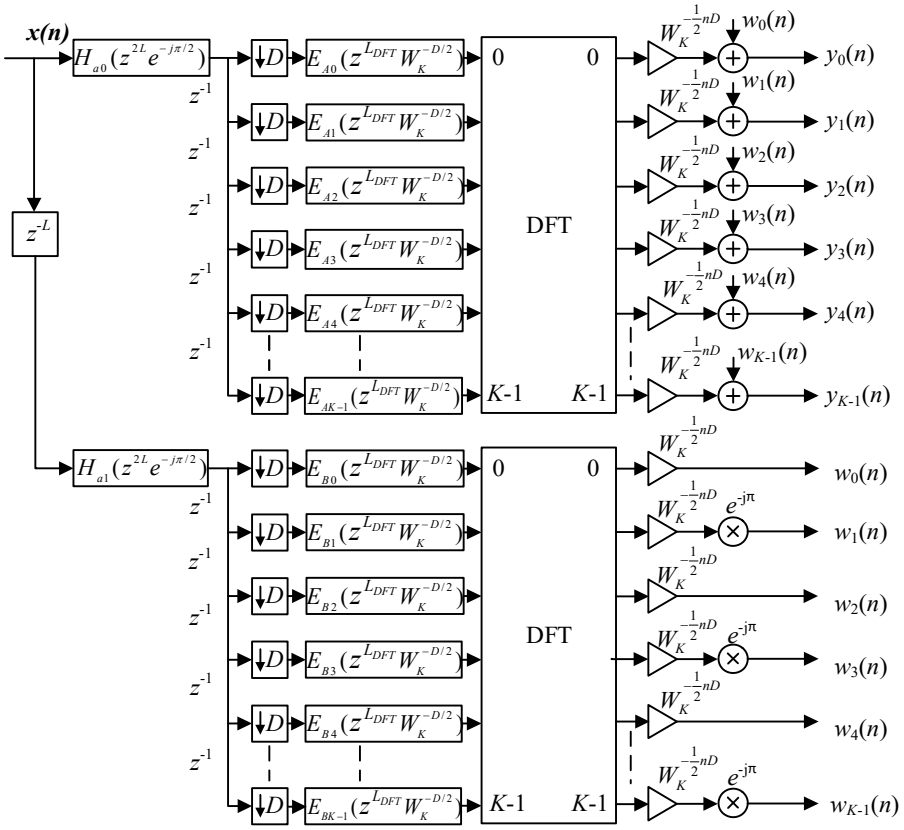

Figure 5.3 Full FRM GDFT-FB (odd stacked, with k_0=1/2 and n_0=0)

The same operation as that of the masking filter will be applied to the base filter as well as shifting the base filter's frequency response from centre at DC to centre at π/2 to match the requirement of odd stacking configuration. Thus Figure 5.3 illustrates the structure of the full FRM odd-stacked GDFT-FB. The expression of the GDFT-FB with FRM technology is shown as:

$$\begin{aligned} H_k(z) = {} & H_{a0}(z^{2L}e_K^{-j\frac{\pi}{2}})\sum_{i=0}^{K-1} z^{-i}W_K^{-ki}W_K^{-\frac{1}{2}i}E_{Ai}(z^K W_K^{-\frac{1}{2}D}) \\ & +(-1)^k z^{-L}H_{a1}(z^{2L}W_K^{-j\frac{\pi}{2}})\sum_{i=0}^{K-1} z^{-i}W_K^{-ki}W_K^{-\frac{1}{2}i}E_{Bi}(z^K W_K^{-\frac{1}{2}D}) \end{aligned} \tag{5.9}$$

It is worth mentioning that, the equation (5.9) and Figure 5.3 are describing the exactly odd stacked case when k_0=1/2 and n_0=0 as in chapter 3. It cannot be applied to all the GDFT-FB cases where arbitrary frequency and phase shifting can be achieved. Therefore k_0 has been replaced by 1/2, and phase shifting factor $W_K^{-(K-1+k_0)n_0}$, like in Figure 3.4, is assigned to 1 in FRM GDFT-FB. Therefore $W_K^{-(K-1+k_0)n_0}$ is omitted in both expression and illustration here.

5.1.3.1 The high level FPGA design

The FPGA design of the full FRM odd stacked GDFT-FB is based on the even stacked version. As part of the conversion to an odd stacked filter-bank, all the FIR filter coefficients become complex values instead of real values. Therefore the I and Q components of the coefficients must be separated into cross coupled FIR compiler IP cores which leads to additional hardware resource usage.

In the case of odd stacked full FRM GDFT-FB, the polyphase decomposition of the masking filter elements, E_A and E_B, are subject to complex modulation to yield

$$E_A(z^K) = \sum_{i=0}^{K-1} E_{Ai}(z^K W_K^{-\frac{1}{2}D}) \tag{5.10}$$

$$E_B(z^K) = \sum_{i=0}^{K-1} E_{Bi}(z^K W_K^{-\frac{1}{2}D}) \tag{5.11}$$

This complex modulation is applied offline at design time so that the modulated filter coefficients are supplied to the FIR compiler IP core. In addition, to let the polyphase decomposed base filters have the odd stacked configuration, a complex frequency shifting $e^{-j\pi/2}$ is applied to them. Again, this is performed at design time so that modified coefficients are used in the corresponding FIR compiler IP cores. That is to say, the separation of I and Q components of coefficients and cross coupling module are employed in two stages of the system.

The polyphase decomposition of filter-banks instead of masking filters employs two of the same structures as the critically sampled odd stacked GDFT-FB, except for the coefficients from the equivalent model in equation (5.8) rather than the prototype low-pass filter.

As for the critically sampled odd-stacked GDFT-FB in chapter 3, the frequency shift $W_K^{-\frac{1}{2}nD}$ applied to the DFT sub-band outputs is implemented using a state machine which either passes or negates samples and changes state every *K* samples.

Besides the frequency shifting just introduced, in odd stacked full FRM GDFT-FB, there is another frequency shifting $e^{-j\pi}$ happening at every second channel of the masking filter in H_{a1} path. So a 'frequency shifting and adding state machine' is developed here to do this frequency shifting, and in addition, it also implements the job of adding, which happens in the end of both paths.

It takes one clock cycle period to process and store the samples to the output, because of the frequency state machine. A one depth register is employed here to delay the trigger of 'en' pin to this state machine to keep the synchronization of both paths of samples.

The diagram of odd stacked full FRM GDFT-FB based on FPGA is shown in Figure 5.4.

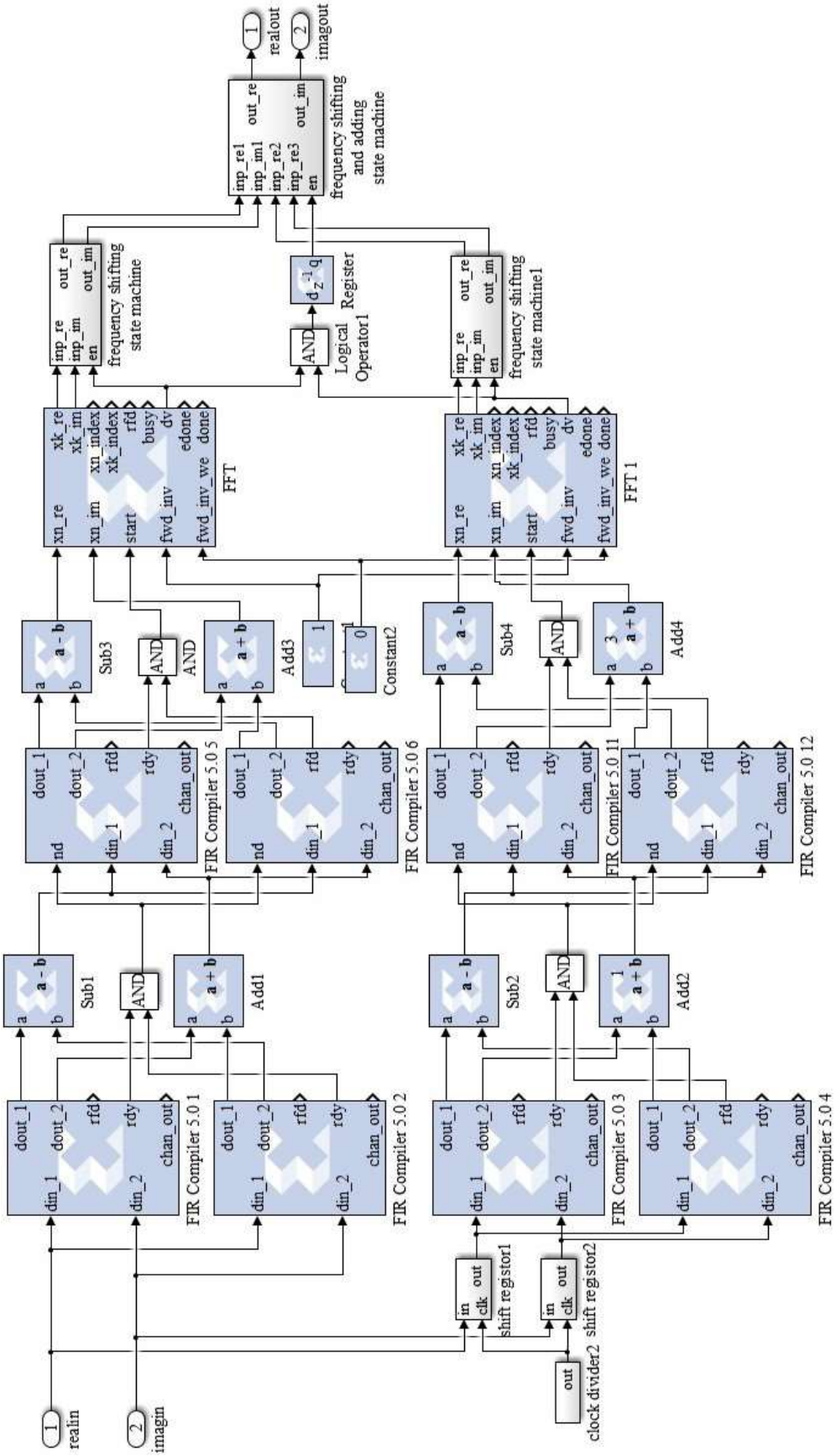

Figure 5.4 The odd stacked full FRM GDFT-FB

5.2 Narrowband FRM applied to the GDFT-FB

5.2.1 Introduction

5.2.1.1 Narrowband FRM

In many channelizer applications it may be sufficient to design the prototype filter response using just the interpolated base filter and a masking filter which rejects all images which result from interpolation [45]. This approach is illustrated in Figure 5.5. In contrast with the full FRM technique, this is known as narrow-band FRM. The narrow-band FRM transfer function is giving by:

$$H(z) = H_a(z^L)H_{Ma}(z) \tag{5.12}$$

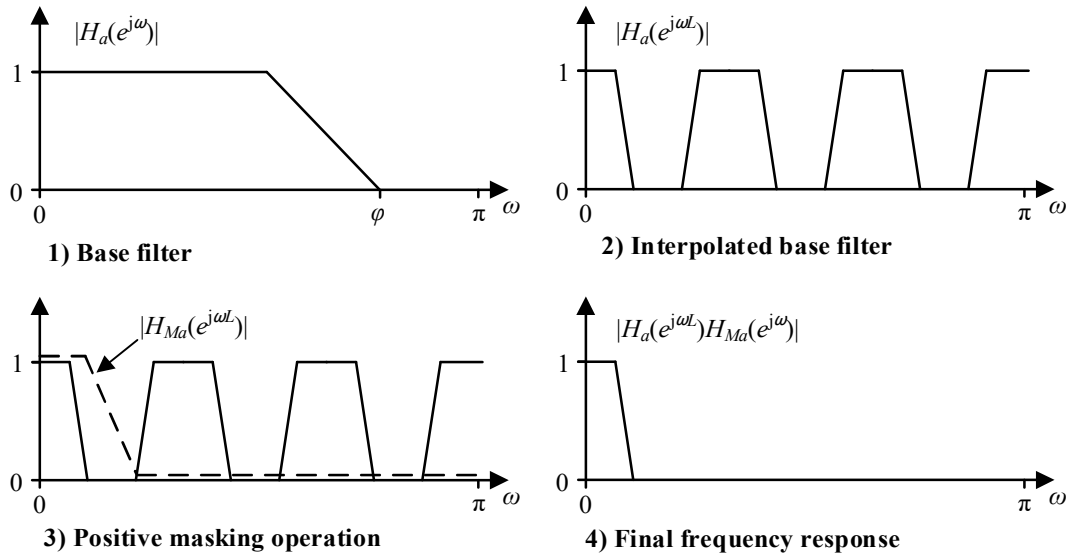

Figure 5.5 The process of narrow-band FRM filter

With narrowband FRM, only the positive branch is occupied, so there is neither a complementary filter nor a complementary masking filter. Therefore equation (5.2) can be simplified to [53]:

$$A(z) = B(z) = H_{Ma}(z) \tag{5.13}$$

First we consider the basic DFT-FB (even-stacked GDFT-FB) with narrowband FRM. The polyphase decomposition can be applied to the base filter and masking filter as:

$$H(z) = H_{a0}(z^{2L})H_{Ma}(z) + z^{-L}H_{a1}(z^{2L})H_{Ma}(z) \tag{5.14}$$

where

$$H_{Ma}(z) = \sum_{i=0}^{K-1} z^{-i}E_{Mai}(z^K) \tag{5.15}$$

Thus every sub-band filter can be expressed as:

$$\begin{aligned} H_k(z) = {} & H_{a0}(z^{2L})\sum_{i=0}^{K-1} z^{-i}W_K^{-ki}E_{Mai}(z^K) \\ & +(-1)^k z^{-L}H_{a1}(z^{2L})\sum_{i=0}^{K-1} z^{-i}W_K^{-ki}E_{Mai}(z^K) \end{aligned} \tag{5.16}$$

From equation (5.16), the masking filter-bank should have the same basic implementation as the full FRM GDFT-FB shown in Figure 5.1, except that the same masking filter components, $E_{Mai}(z^K)$, appear in both the H_{a0} and H_{a1} branches.

For the odd-stacked GDFT-FB with narrowband FRM, the basic procedure is the same as for the full FRM GDFT-FB version. We apply equation (3.10) to each sub-band $H_k(z)$. Furthermore, the frequency response of base filter polyphase components, H_{a0} and H_{a1}, need to be odd stacked by shifting their frequency response from DC to π/2. Thereafter, the narrowband FRM odd stacked GDFT-FB expression can be seen as a modified form of equation (5.16):

$$\begin{aligned} H_k(z) = {} & H_{a0}(z^{2L}e^{-j\frac{\pi}{2}})\sum_{i=0}^{K-1} z^{-i}W_K^{-ki}W_K^{-\frac{1}{2}i}E_{Mai}(z^K W_K^{-\frac{1}{2}D}) \\ & +(-1)^k z^{-L}H_{a1}(z^{2L}e^{-j\frac{\pi}{2}})\sum_{i=0}^{K-1} z^{-i}W_K^{-ki}W_K^{-\frac{1}{2}i}E_{Mai}(z^K W_K^{-\frac{1}{2}D}) \end{aligned} \tag{5.17}$$

The narrowband FRM odd-stacked GDFT-FB structure is very similar to Figure 5.3, with the exception that the masking filter components, $E_{Mai}\left(z^K W_K^{-k_0 K}\right)$, are common to both branches.

5.2.1.2 Alternative structure for oversampled narrowband FRM GDFT-FB

In general, GDFT-FB provides a large computational saving in filtering in comparison to direct per-channel filtering. The benefit is obtained from the polyphase decomposing and noble identity [67]. In the FRM GDFT-FB designs discussed in previous sections, the base filter is placed before the GDFT-FB modulated filters. Therefore, the base filters operate at the wideband input signal sample frequency rather than decimated sub-band sample frequency required for the masking filter polyphase components. Furthermore, with a large number of channels, K, the base filter requires more interpolation so there will be more zero padding and sample delay required in the base filter section.

As an alternative, the base filter can be moved to the output side of the GDFT-FB in order to operate at the lower sub-band output sample rate using the noble identity [53]. A similar alternative implementation to the FRM CMFB has been done [64]. In addition, the FRM interpolation factor L will be applied (in a decimated form) to the base filter when it is moved to the output side of the DFT. This reduces the zero padding and sample delay required. Another benefit of this alternative design is that the base filter coefficients will always be real-valued whether the filter bank is designed for even stacked or odd stacked channels.

To make the alternative narrowband FRM GDFT-FB work, the system must use an oversampled configuration.

The alternative narrowband FRM design contributes one further computational saving. In this configuration, the polyphase decomposition of the base filter is no longer necessary because it follows the DFT. Therefore we no longer need the masking filter polyphase components to be repeated in the two branches resulting from the polyphase decomposition of the base filter. As a consequence, the efficient oversampled narrowband FRM GDFT-FB can be achieved as illustrated in Figure 5.6. As a final optimization, since the base filter doesn't have to be divided into its polyphase components, it may be symmetric thus permitting further efficiency in terms of the number of multiplications required.

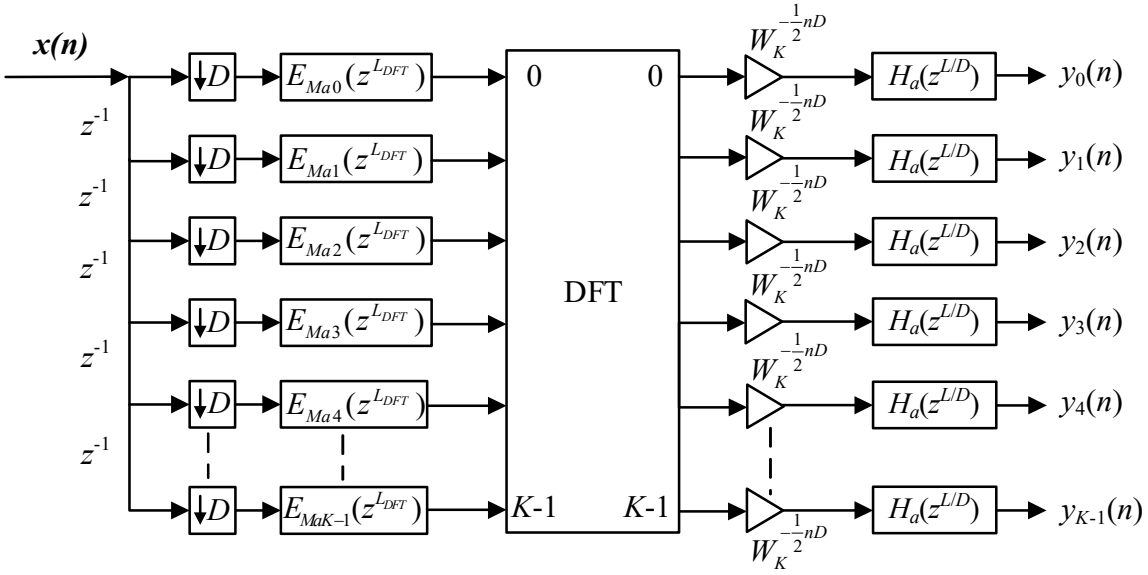

Figure 5.6 Efficient oversample GDFT-FB with narrowband FRM.

5.2.2 The FPGA based alternative narrowband (oversampled) DFT-FB (even stacked)

5.2.2.1 The overall design

The FPGA implementation of the alternative narrowband FRM DFT-FB shares many similarities with the oversampled DFT-FB design in chapter 4. Typically, an oversampling factor of 2 may be used, because it already provides a considerable reduction in terms of aliasing. The masking filter of positive FRM branch replaces the prototype filter of oversampled DFT-FB. The oversampled polyphase decimation FIR introduced in § 4.1.2 which split the sub-bands into two groups must also be employed here. Rearrangement is still required for FFT core to take the right samples. State machine that switches in 4 states is designed to do the equivalent job as frequency shifting, as shown in § 4.2.4. The diagram of a 2x oversampled alternative narrowband FRM GDFT-FB is shown in Figure 5.7. The main difference from the oversampled DFT-FB, is the extra base filter in each output sub-band. An FIR compiler set to ‘single rate’ mode is suitable to be used here, as multiple channels are required to be filtered by the same coefficients at the same sample rate. If the clock is fast enough, this IP core will process these samples serially by reusing DSP48s. FIR compiler can also take advantages of symmetric coefficients, in order to have a further hardware efficiency.

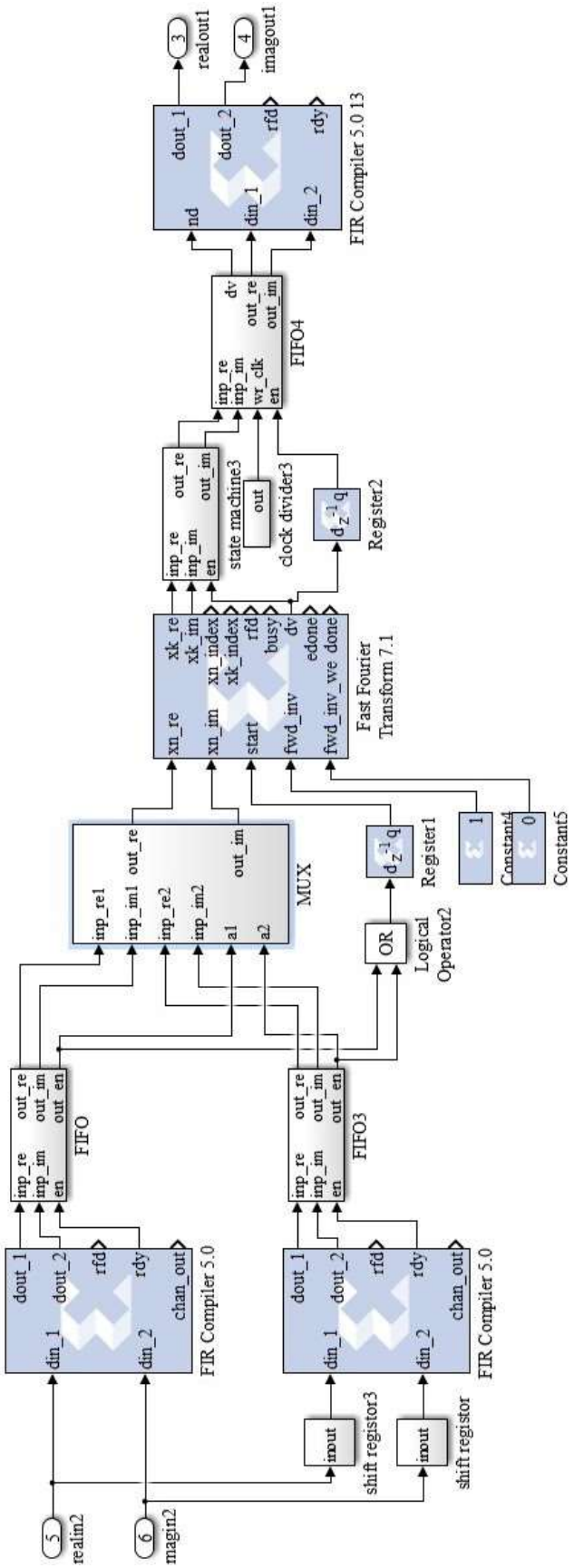

Figure 5.7 2x oversampled alternative narrowband FRM GDFT-FB

5.2.2.2 Efficient FIFO design of base FIR complier

In the oversampled alternative narrowband FRM DFT-FB, the base filters follow the FFT IP core. There is one copy of the base filter on each sub-band and these are implemented using FIR compiler IP cores. The FFT core outputs a burst of data at the clock rate each time it has finished processing *K* input samples as shown in Figure 5.8. It is unwise to have a design in which the FIR compiler receives input samples in short bursts and then does nothing for most of the time waiting for the next round of inputs. The solution is to introduce a FIFO to slow down the pace of the input samples, in order to relieve the pressure on the polyphase decomposition FIR compiler by giving more clock cycles for processing. As a result, a lot of DSP and storage resources may be saved.

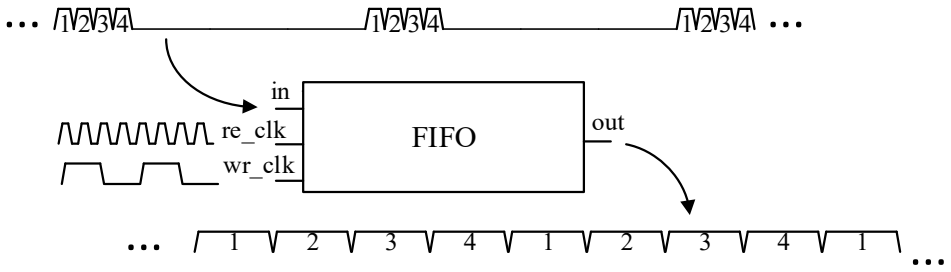

Figure 5.8 The FIFO used to slow the FFT output for sub-band FIR compiler IP cores

The FIFO works at two different clock speeds: re_clk refers to the clock pin that triggers the FIFO to read samples at the fast system clock rate and store them into a shift register; wr_clk triggers the FIFO to write the stored samples at the speed of the sub-band sample rate. Thus the clock tree introduced in the last section is suitable to employ here. The 'en' (enable) input pin is assigned to the FIFO to enable the receiver, and a dv (data_valid) pin is used to indicate the next component when data is available.

5.2.3 The FPGA based alternative narrowband (oversampled) GDFT-FB (odd stacked)

5.2.3.1 Theoretical structure

Converting the oversampled alternative narrowband FRM GDFT-FB to odd stacked channel allocation is similar to converting DFT-FB to GDFT-FB introduced in chapter 3, and converting the oversampled DFT-FB to the oversampled GDFT-FB in chapter 4.

Because the alternative narrowband FRM only employs the positive path of the FRM structure, and swaps the order of the base and masking filter, the expression of every channel band-pass filter in GDFT-FB version can be written as:

$$H_k(z) = \sum_{i=0}^{K-1} z^{-i} W_K^{-ki} E_{Mai}(z^K) H_a(z^{L/D}) \tag{5.18}$$

The odd stacked band-pass GDFT filter is therefore:

$$H_k(z) = \sum_{i=0}^{K-1} z^{-i} W_K^{-\frac{1}{2}i} W_K^{-ki} E_{Mai}(z^K W_K^{-\frac{1}{2}D}) H_a(z^{L/D}) \tag{5.19}$$

The resulting structure of the 2x oversampled odd stacked alternative narrowband FRM GDFT-FB is illustrated in Figure 5.9

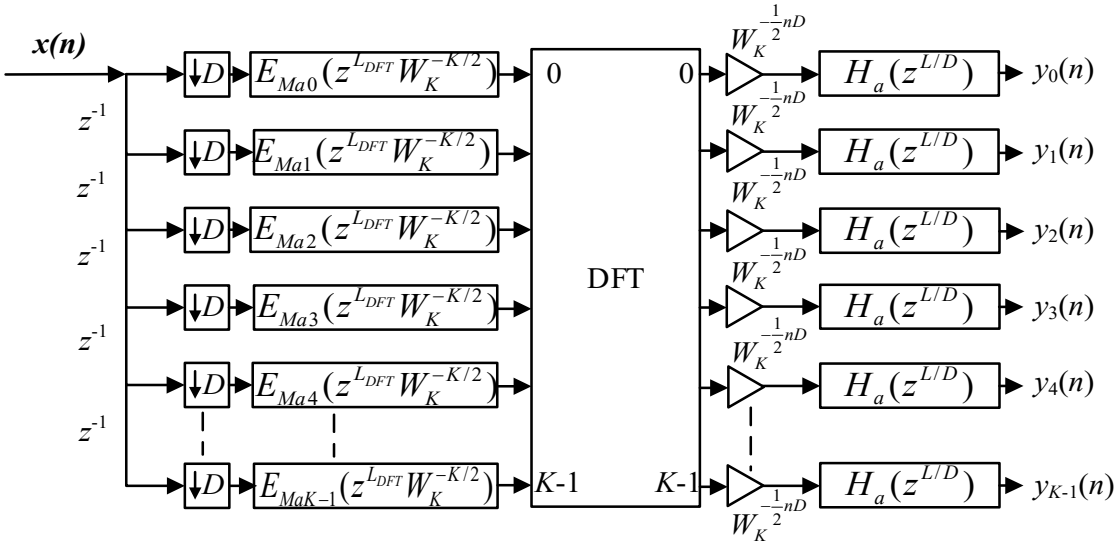

Figure 5.9 Odd stacked GDFT-FB with alternative narrowband FRM technology.

5.2.3.2 FPGA design

As usual for odd-stacked GDFT implementations, the filters at the input of the system are subject to offline complex modulation at design time which yields complex filter coefficients. This in turn requires the FPGA implementation to use 'cross-coupled FIR compiler blocks' described in § 3.2.2.2 to implement the complex-valued coefficients. The diagram of the FPGA implementation of odd stacked alternative narrowband GDFT-FB is shown in Figure 5.10.

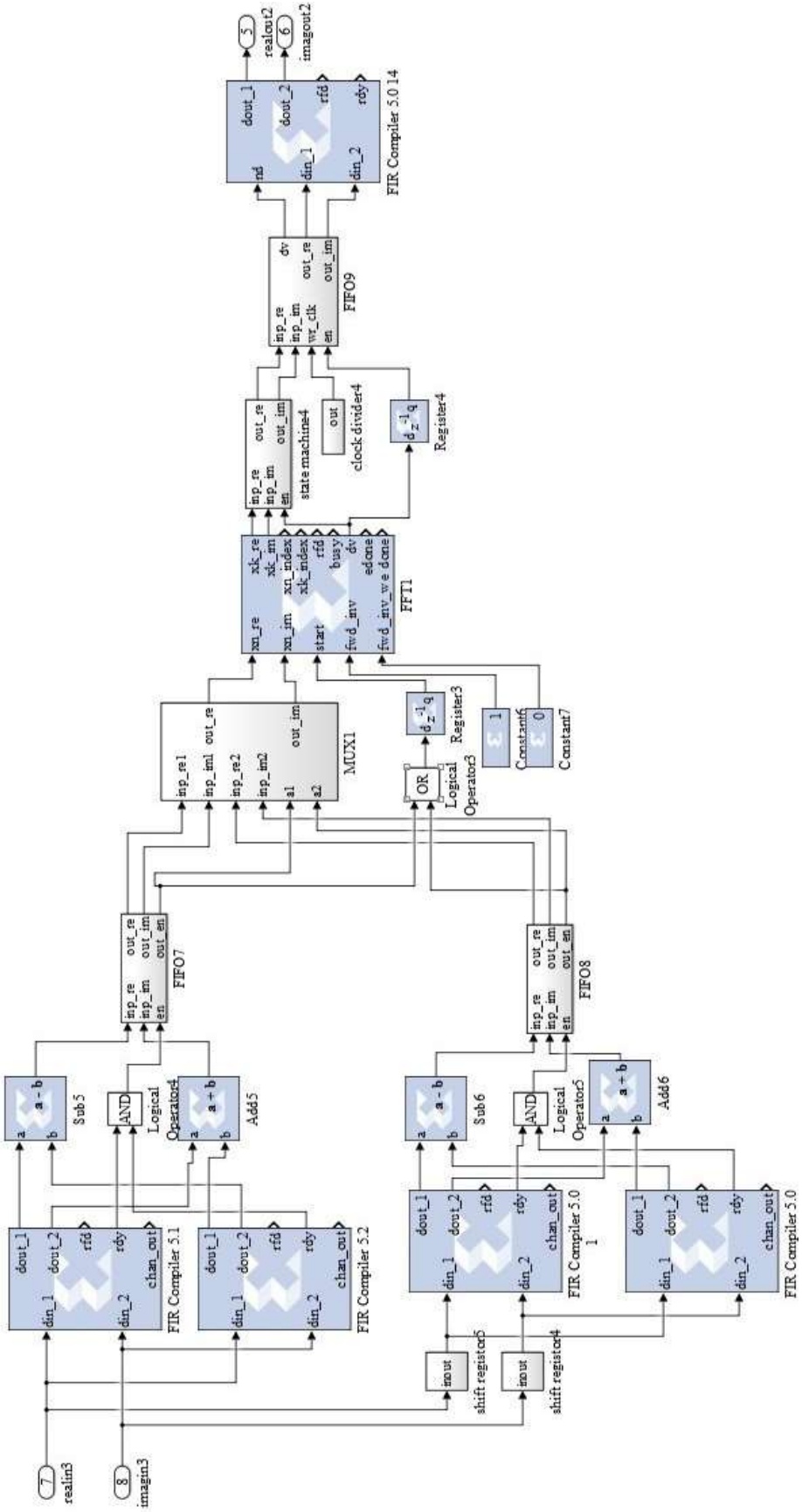

Figure 5.10 FPGA implementation of odd stacked alternative narrowband GDFT-FB

The alternative narrowband FRM design has a considerable reduction of complexity compared to full FRM. Its structure is similar to the oversampled odd stacked GDFT-FB except that there is one additional base filter at the output of each sub-band. Unlike filters at the input to the DFT, these base filters do not need to be complex modulated and therefore no complex valued coefficients are required.

The polyphase decomposition FIR filter $E_{Mai}(z^K W_K^{-\frac{1}{2}D})$ is modulated to complex values. The L_{DFT}-1 zeros padding of coefficients are operated before that of the FIR filters (where $L_{DFT}=K/D$).

As part of the 2x oversampled GDFT-FB a frequency shifting state machine processes the data with the 4 states described in § 4.2.4. As before, FIFOs are used to achieve a steady stream of data from the FFT at the output sample rate.

5.3 Evaluation and results

In this chapter, we have designed the FPGA implementation of the full FRM DFT-FB/GDFT-FB and alternative narrowband FRM DFT-FB/GDFT-FB. As in previous chapters, the evaluation of these filter banks uses test signals based on the TETRA voice and data, 25 kHz channel specifications. All signals and filter bank internal values are quantized to 16-bit fixed point resolution. The evaluation focuses on sub-band frequency response, EVM, performance with adjacent channel interference and hardware resource usage (which ultimately defines how scalable the design is).

The FRM technique for filter design means that the prototype filters used here cannot be identical to those in chapter 3 or chapter 4 although they have been designed to match the specifications of those filters as closely as possible. Moreover, the alternative narrowband FRM approach to filter design differs from the full FRM approach and therefore these two FRM approaches have different prototype filter designs. For a 16-channel full FRM GDFT-FB designed to meet the specifications in § 3.3.1.1, the order of the base filter H_a is 20, the order of the masking filters H_{ma} and H_{mc} is 145 each, and the FRM interpolation factor L is 24. In contrast, for the equivalent 16-channel alternative narrowband FRM GDFT-FB, the order of the base filter H_a is 41, the order of the masking

filter H_{mc} is 41, and the FRM interpolation factor L is 8. The full FRM design is critically sampled whereas the alternative narrowband FRM design is oversampled by 2.

Odd and even-stacked variants of the filter banks will vary in hardware resource usage and channel allocation but will be based on the same filters in either case. Therefore only the filtering results of the odd stacked design are discussed here. The hardware usage is evaluated for even and odd stacked designs separately.

5.3.1 Frequency response

The frequency response of one sub-band of the 16-channel based full FRM GDFT-FB is shown in Figure 5.11. Zooming into the pass band, as shown in Figure 5.12, indicates that the fixed point quantization has an impact on the pass band response. However the passband ripple doesn't increase much. It is about 0.312dB, compared to floating point reference implementation's 0.31dB, and this is still within the 2 dB specification.

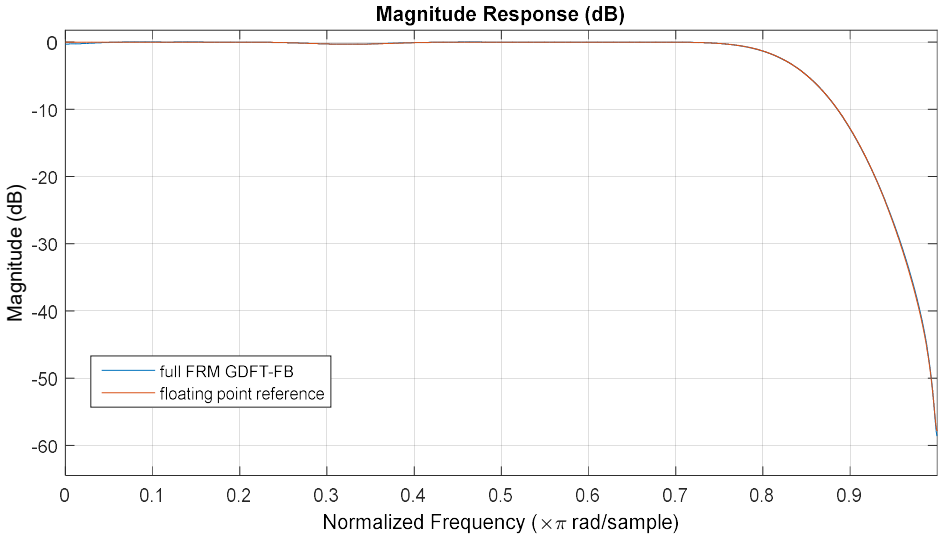

Figure 5.11 Frequency response of the FPGA based full FRM GDFT-FB sub-band (blue) compared to the equivalent floating point reference implementation (red)

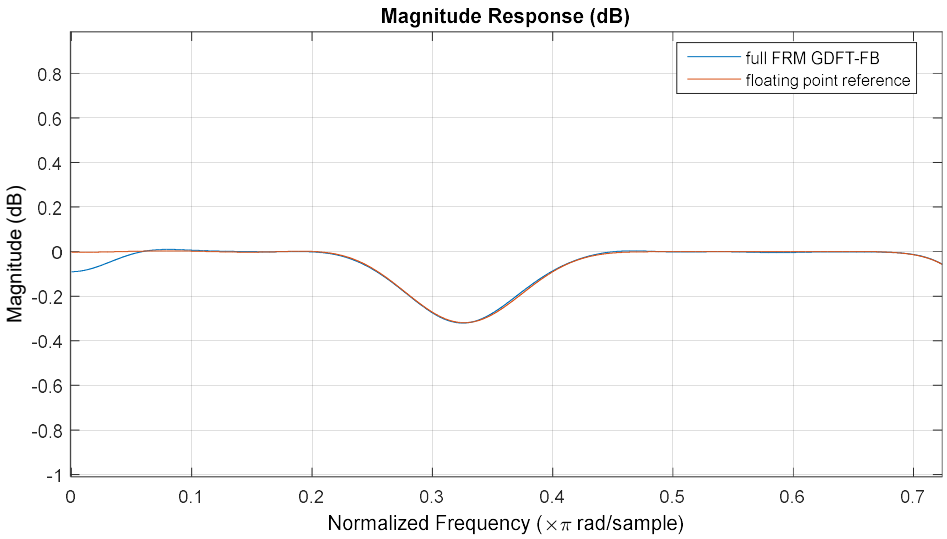

Figure 5.12 Passband comparison between the FPGA based full FRM GDFT-FB (blue) and its equivalent floating point reference implementation (red)

The frequency response of the FPGA based 16-channel alternative narrowband FRM GDFT-FB is shown in Figure 5.13. Since this design is oversampled by 2 the output signal pass band only occupies half of the sub-band spectrum.

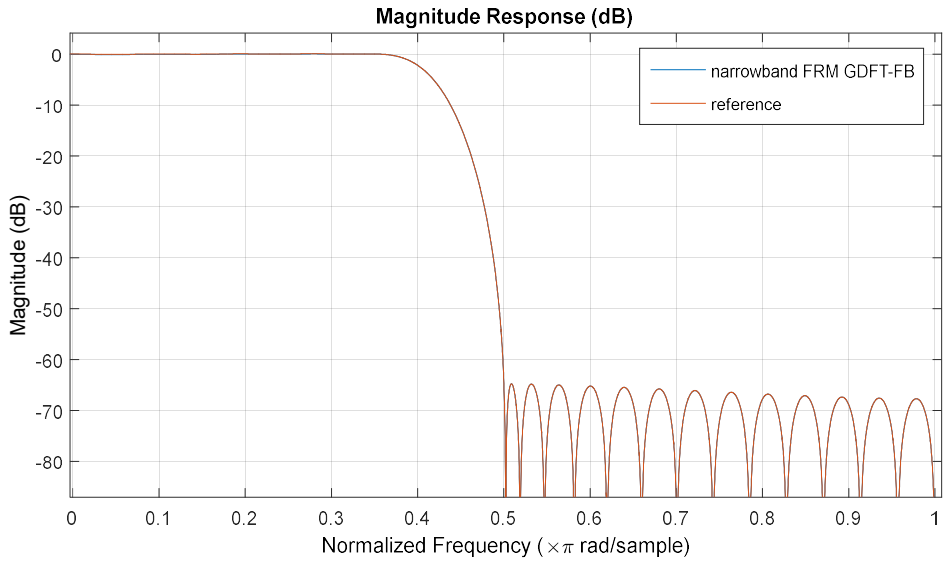

Figure 5.13 Frequency response of the FPGA based 16-channel alternative narrowband FRM GDFT-FB (blue) compared to its floating point reference implementation (red)

The pass band performance is shown in Figure 5.14. As usual, the 16-bit fixed point quantization causes slightly increased pass band ripple (0.1 dB) compared to the floating point reference (0.05 dB). This is still well within specifications (2 dB).

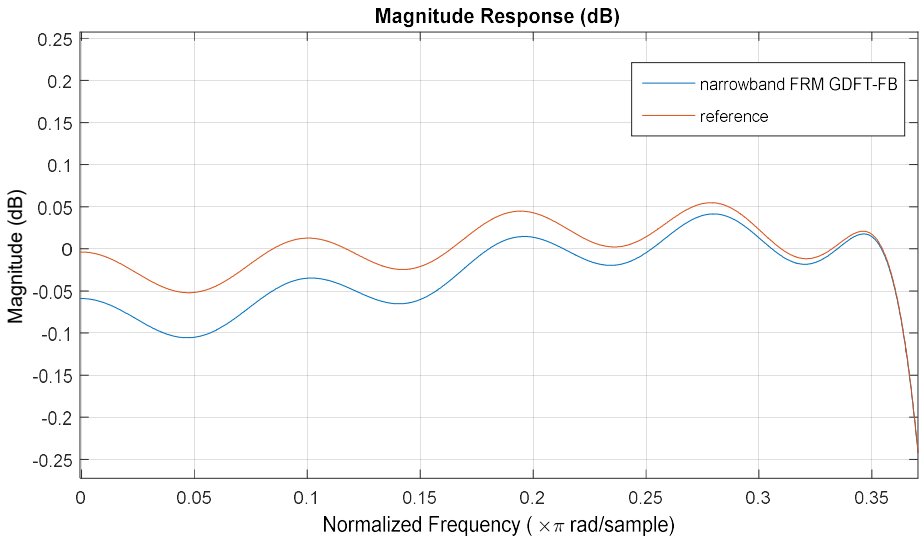

Figure 5.14 Passband comparison between the FPGA based alternative narrowband FRM GDFT-FB (blue) and its floating point reference implementation (red)

Figure 5.15 shows that the 16-bit fixed point quantization error has very little effect on the stop band performance. The frequency response of FPGA based design and the floating point reference are mostly identical with 0.001 dB variance.

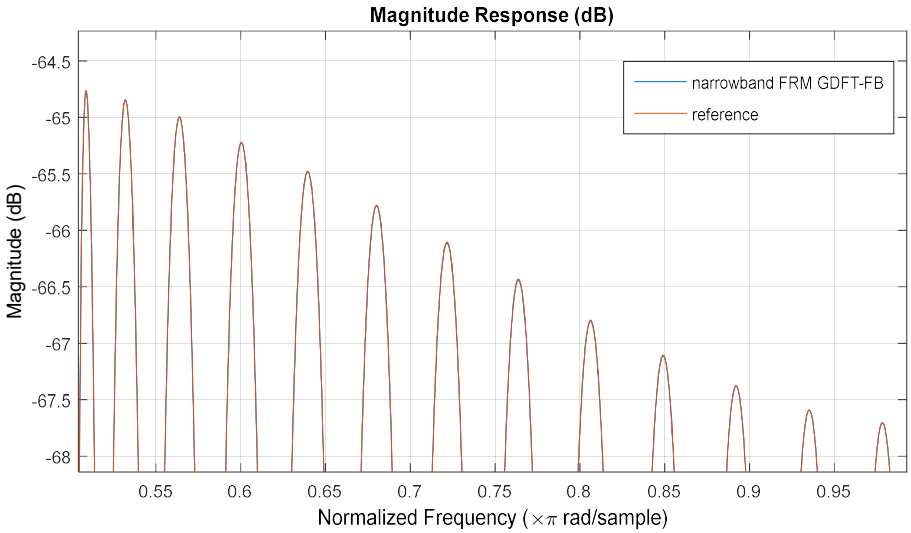

Figure 5.15 stop band comparison between the FPGA based alternative narrowband FRM GDFT-FB (blue) and the equivalent floating point reference implementation (red)

5.3.2 EVM result

The EVM test was conducted as in previous chapters. Figure 5.16 shows the constellation of the π/4 DQPSK modulation signal (at maximum signal level) from one sub-band of the FPGA based full FRM GDFT-FB while Figure 5.17 shows the equivalent constellation from the FPGA based alternative narrowband FRM GDFT-FB.

The constellations of both designs indicate acceptable receiving performance. It can also be seen that the constellation points of the alternative narrowband FRM GDFT-FB are more concentrated than those of the full FRM GDFT-FB indicating slightly better EVM performance.

Table 5.1 lists the peak and RMS EVM using full FRM and alternative narrowband FRM GDFT-FB.

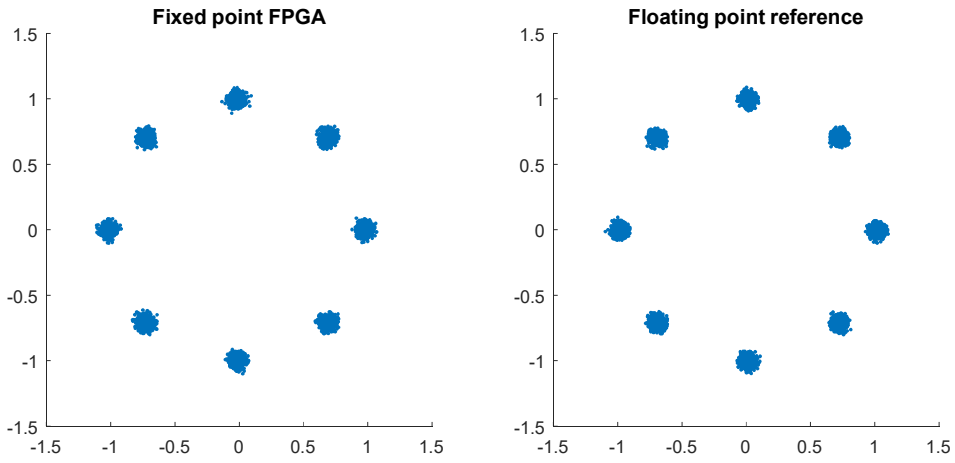

Figure 5.16 The QPSK modulation constellation of the FPGA based full FRM GDFT-FB output (left), and the QPSK modulation constellation of a reference floating point full FRM GDFT-FB output (right)

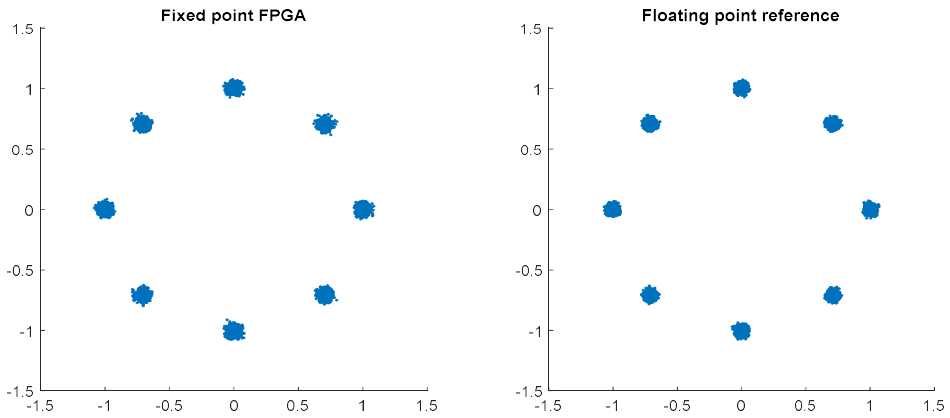

Figure 5.17 The QPSK modulation constellation of the FPGA based alternative narrowband GDFT-FB output (left), and the QPSK modulation constellation of a reference floating point alternative narrowband GDFT-FB output (right)

Table 5.1 The EVM performance of both FPGA based designs: the 16-channel full FRM GDFT-FB and the alternative narrowband GDFT-FB

	Full FRM GDFT-FB		Alternative Narrowband FRM GDFT-FB		TETRA limits
	16-bit FPGA	Floating point	16-bit FPGA	Floating point	
Peak	0.1364	0.1197	0.1018	0.0807	0.3
RMS	0.0487	0.0464	0.0392	0.0354	0.1

5.3.3 Adjacent channel interference

Adjacent channel interference was evaluated at several levels of C/I_a as in chapter 3, namely, -10 dB, -20 dB,-30 dB, -40 dB, -45 dB and -50 dB. The TETRA specification indicates that filtering result should still meet the EVM requirements when C/I_a reaches – 45 dB.

Figure 5.18 and Figure 5.19 show the constellation of a single sub-band output from the FPGA based full FRM GDFT-FB and alternative narrowband FRM GDFT-FB at different adjacent channel interference levels. The detailed EVM results are presented in Table 5.2. As usual, the data shows that EVM results degrade as the interference level increases. Nevertheless, both the full FRM and alternative narrowband FRM designs meet the TETRA specifications at -45 dB. Furthermore, alternative narrowband FRM GDFT-FB still meet the TETRA limit at -50 dB adjacent channel level, but full FRM GDFT-FB

exceed the limit at this level. Therefore alternative narrowband GDFT-FB have a better performance regarding to adjacent channel interference than critically and oversampled GDFT-FB, and full FRM GDFT-FB.

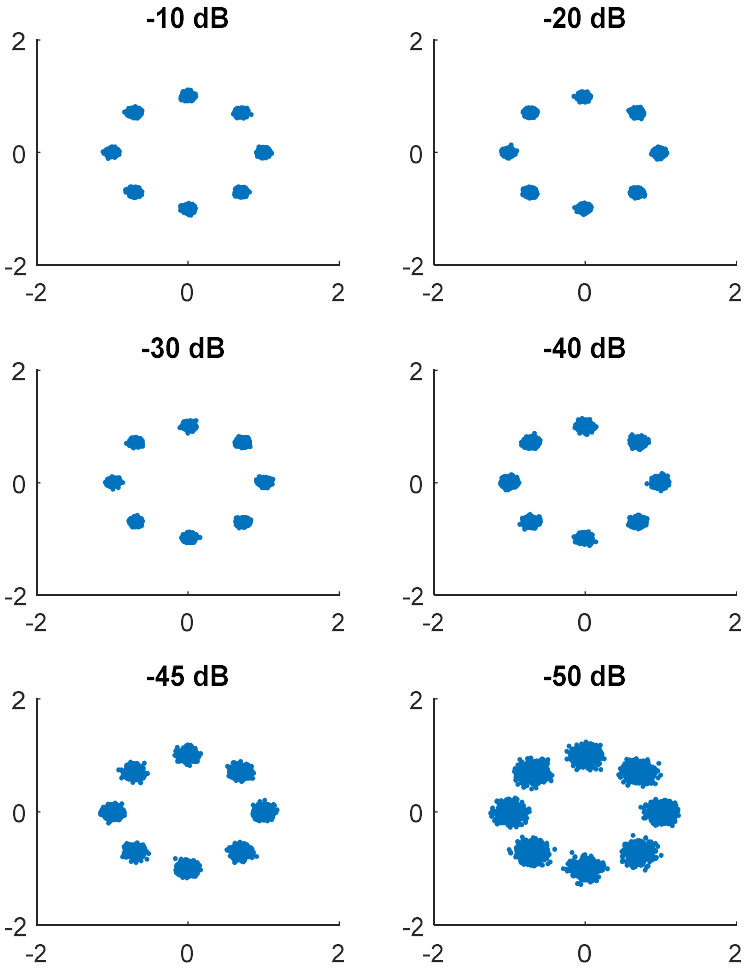

Figure 5.18 Modulation constellation of the FPGA based full FRM GDFT-FB at different adjacent channel interference levels

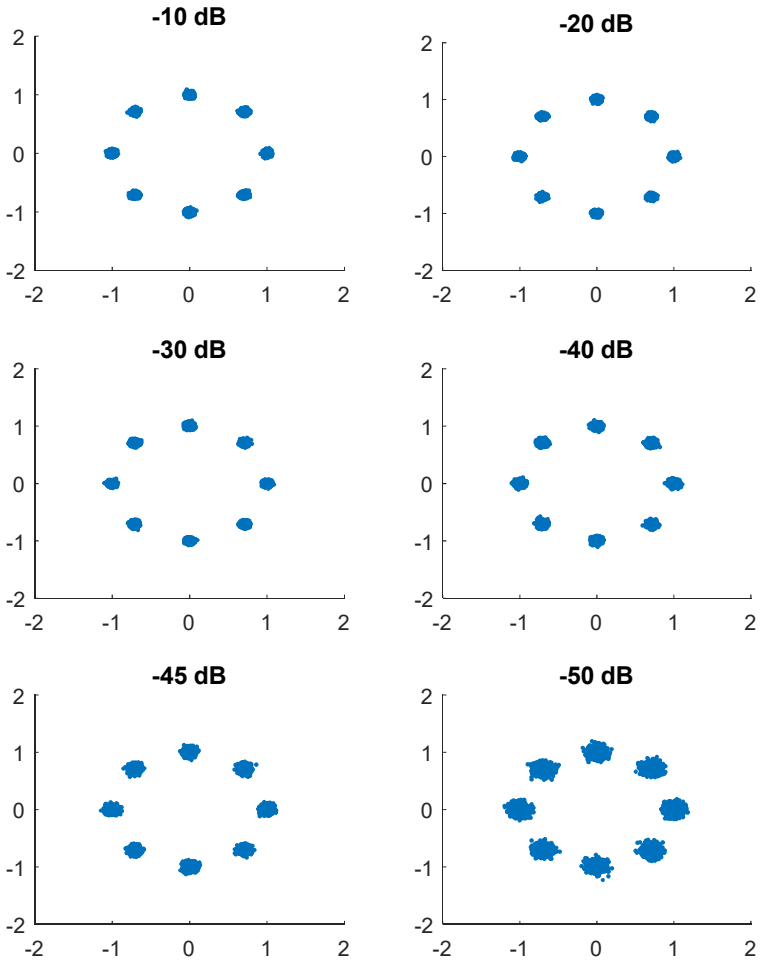

Figure 5.19 Modulation constellation of the FPGA based alternative narrowband GDFT-FB at different adjacent channel interference levels

Table 5.2 EVM results of the FPGA based full FRM GDFT-FB and alternative narrowband FRM GDFT-FB at different adjacent channel interference levels

	Full FRM GDFT-FB		Narrowband FRM GDFT-FB Alt.	
C/Ia (dB)	Peak	RMS	Peak	RMS
-10	0.1340	0.0489	0.0970	0.0386
-20	0.1511	0.0524	0.0983	0.0391
-30	0.1589	0.0528	0.1137	0.0404
-40	0.1858	0.0623	0.1374	0.0460
-45	0.2434	0.0829	0.1731	0.0578
-50	0.3326	0.1295	0.2396	0.0865

5.3.4 Hardware resource usage

The FPGA full FRM DFT-FB/GDFT-FB design uses multiple filters that each have fewer coefficients than the single filter design of the basic DFT-FB/GDFT-FB. The FPGA based alternative narrowband FRM DFT-FB/GDFT-FB eliminates one branch of the full FRM design and moves the base filter to the output side of the DFT so that it operates at a lower sample rate. This reduces the number of filters to be designed and the total number of filter coefficients even further. However this design requires the output sub-bands to be oversampled (by 2 in this evaluation) whereas the full FRM design uses critically sampled output sub-bands.

Table 5.3 shows the hardware usage of full FRM GDFT-FB and alternative narrowband FRM GDFT-FB based on FPGA. The data indicates that the alternative narrow band FRM DFT-FB/GDFT-FB is more efficient in terms of FPGA resources than the full FRM design, particularly in terms of DSP48s.

Table 5.3 Hardware usage of full FRM GDFT-FB and alternative narrowband FRM GDFT-FB

Filter-bank Type	Register	LUTs	Block RAM 36	Block RAM 18	DSP48s
Even full FRM DFT-FB	2356	3379	0	6	31
Odd full FRM GDFT-FB	3098	3976	0	12	43
Even narrowband DFT-FB Alt.	1685	1814	2	0	9
Odd narrowband GDFT-FB Alt.	2120	2529	0	5	16
Per-channel approach	3993	2123	4	48	80
Available	301440	150720	416	832	768

5.4 Chapter conclusion

FRM is an efficient technique for reducing the complexity of a FIR filter by cascading an interpolated base filter and a low order masking filter to achieve the desired response. In this chapter FPGA implementations based on combining the FRM filter design approach with the DFT-FB and GDFT-FB were introduced, in order to achieve more efficient use of hardware resources and higher performance.

Firstly, the FPGA design of the full FRM and DFT-FB/GDFT-FB was implemented. In comparison to the basic DFT-FB/GDFT-FB, this design requires two branches each containing a polyphase decomposed filter and DFT to be implemented. The principal new innovations required in the development were related to efficient implementation of rather long delays at the input side of the filter bank (using a fractional clock divider and FIFO) and a state machine which appropriately combined the output of the two branches of the full FRM implementation to form the overall filter bank output.

The FPGA based alternative narrowband FRM technology was implemented as a further optimization to have fewer filters to design and less complex filters with fewer coefficients. In this case just one FRM path is required and the FRM base filter now appears on the output side of the DFT. This change means that the base filter operates at the output sub-band sample rate and only requires real-valued coefficients (whether the filter bank is even or odd stacked). The principal additional innovation required to realise this design was smoothing the rate at which sample data was supplied to the base filter in each sub-band (from the short bursts of high rate data output by the FFT IP core). This smoothing (and slowing down) of the data rate allows a more relaxed implementation of the base filter that can serialize multiplications to reuse multiplier resources rather than requiring additional multipliers running in parallel.

The simulation results show that the FPGA based alternative narrowband FRM GDFT-FB has better EVM performance and adjacent channel interference resistance than the critically sampled DFT-FB/GDFT-FB or its oversampled variants. On the other hand, FPGA based full FRM GDFT-FB costs the most of hardware resources among all the filter-banks presented in this work and has a worst EVM performance and adjacent channel interference resistance, but it still meets the TETRA specification.

Chapter 6

Scaled up Evaluation

In chapter 3 ~ chapter 5, the FPGA architectures of a number of DFT-FB and GDFT-FB were designed and implemented. In the evaluation sections of each of these chapter the resource usage of the designs in 16-channel form was identified, as shown in Table 6.1. Table 5.3 presents the results here again in aggregate form to simplify comparison.

Table 6.1 Hardware usage of all FPGA based 16-channel filter banks implemented to date

Filter-bank Type	**Register**	**LUTs**	**Block RAM 36**	**Block RAM 18**	**DSP48s**
Critically sampled designs					
Even DFT-FB	1223	838	0	8	5
Odd GDFT-FB	1435	1270	0	11	7
Oversampled designs					
2x Even DFT-FB	1445	1503	0	7	8
2x Odd GDFT-FB	2121	1570	0	12	15
FRM based designs					
Even full FRM DFT-FB	2356	3379	0	6	33
Odd full FRM GDFT-FB	3098	3976	0	12	43
Even narrowband DFT-FB Alt.	1685	1814	2	0	9
Odd narrowband GDFT-FB Alt.	2120	2529	0	5	16
Per-channel approach	3993	2123	4	48	80
Available	301440	150720	416	832	768

The results showed that, when meeting TETRA Voice and Data 25 KHz channel requirements, the critically sampled DFT-FB and GDFT-FB performed equally to their oversampled equivalents, but with less hardware resource usage. On the other hand, although the full FRM GDFT-FB requires significantly more hardware resources than any other implementation, it provides a worse EVM and adjacent channel interference resistance. The results also showed that the alternative narrowband FRM DFT-FB/GDFT-

FB can even achieve better adjacent channel interference resistance than the basic DFT-FB/GDFT-FB and requires far fewer filter coefficients. This latter feature is expected to make the prototype filters for channelizing large numbers of channels easier to design and implement. On the other hand, the alternative narrowband FRM was shown to achieve the best DSP performance with not much more hardware resource usage than the critically sampled designs in a 16-channel channelizer. It is possible that the alternative narrowband FRM DFT-FB/GDFT-FB would have an even greater hardware efficiency than the critically sampled DFT-FB/GDFT-FB when the design is scaled up to a larger number of channels scenarios, because of the smaller individual filters.

For this reason, we will evaluate the critically sampled DFT-FB/GDFT-FB and alternative narrowband FRM DFT-FB/GDFT-FB designs when used to implement a 256-channel channelizer in this chapter.

6.1 Scaling up of filter-banks

6.1.1 Scaling up critically sampled DFT-FB/GDFT-FB to 256 channels

For the FPGA based critically sampled even-stack DFT-FB, scaling up to 256 channels while retaining the same sub-band characteristics is straightforward: (1) The new wideband sample rate must be determined—for 256 channels of 25 kHz each, this is 6.4 MHz; (2) A new higher order prototype filter must be designed for the increased sample rate and the 16-bit quantized coefficients supplied to the FIR compiler IP core; (3) Finally, the number of channels in the FIR compiler IP core and transform length in the FFT IP core must be set to the increased number of channels.

For the odd stacked design the changes are identical except that the prototype filter coefficients must additionally be subject to complex modulation as in equation (3.10), then divided into their I and Q components and assigned to the correct FIR compiler IP cores. In addition, the frequency shift state-machine which follows the FFT IP core must be modified to handle the increased number of channels as well. Specifically, it must be configured to read the number of channels ($K = 256$ in this case) samples each time before changing state.

6.1.2 Scaling up alternative narrowband FRM DFT-FB/GDFT-FB to 256 channels

Scaling up the FGA based alternative narrowband FRM DFT-FB/GDFT-FB to 256 channels is very similar to the procedure for scaling up the critically sampled filter banks. Since alternative narrowband FRM is based on an oversampled design and we choose a 2x oversampling, the prototype filter coefficients must be divided into two groups which are assigned to different FIR blocks (see Figure 4.5).

The FIFOs, as shown in Figure 4.5, required to rearrange the output of the individual FIR blocks to an input suitable for the FFT core must now be modified to read and hold D=128 samples, as the channel number increases to 256. As the selector state machine is still compatible with 256 channel configuration, it remains the same.

6.2 Evaluation and Results

Similar to the evaluation of the 16-channel filter banks evaluated in previous chapters, the evaluation of the 256-channel critically sampled DFT-FB/GDFT-FB and alternative narrowband FRM DFT-FB/GDFT-FB examines the sub-band frequency response, EVM characteristics, and response to adjacent channel interference. As before, the FPGA hardware resource usage of the designs will also be compared.

The design and test setup is almost the same as described in § 3.3.1. The only difference is the wide-band input signal contains 256 TETRA 25 kHz channels with a total of 6.4 MHz sample frequency.

6.2.1 Frequency response

6.2.1.1 Critically sampled GDFT-FB

Figure 6.1 shows the frequency response of FPGA based critically sampled GDFT-FB sub-channel.

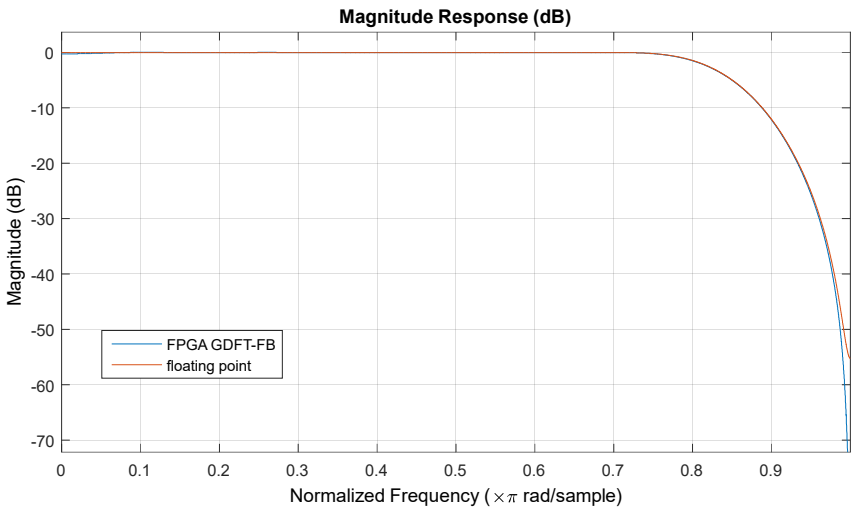

Figure 6.1 Frequency response of critically sampled GDFT-FB comparing the fixed point FPGA implementation (blue) to the floating point reference implementation (red)

When zooming into the passband, shown in Figure 6.2, we can see that the fixed point quantization error has a small impact on the passband characteristics. The passband ripple increases to 0.2 dB, from about 0.01 dB of the floating point reference. However it still is well within the 2 dB requirement of the TETRA specification.

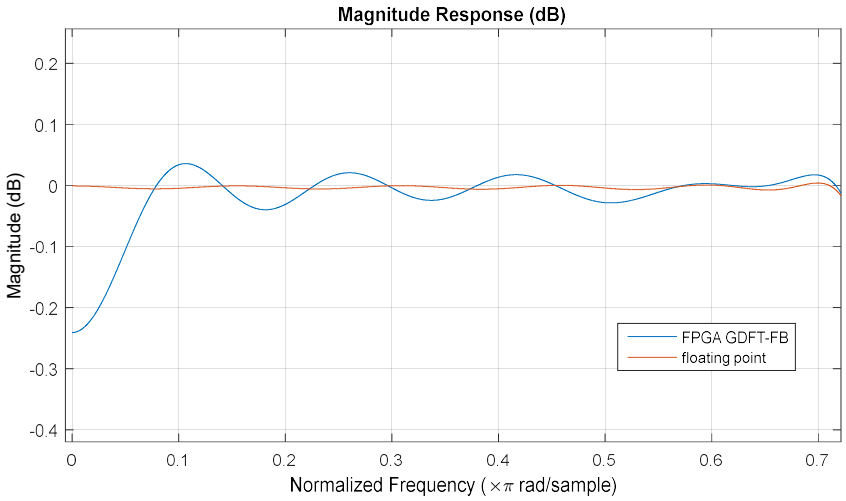

Figure 6.2 Passband comparison between the FPGA based GDFT-FB (blue line) and floating point reference implementation (red line)

6.2.1.2 Alternative narrowband FRM GDFT-FB

Figure 6.3 shows the frequency response of the FPGA based alternative narrowband FRM GDFT-FB which appears to be very close to the response of the reference implementation. Figure 6.4 shows the passband detail comparison while Figure 6.5 shows the stopband detail comparison.

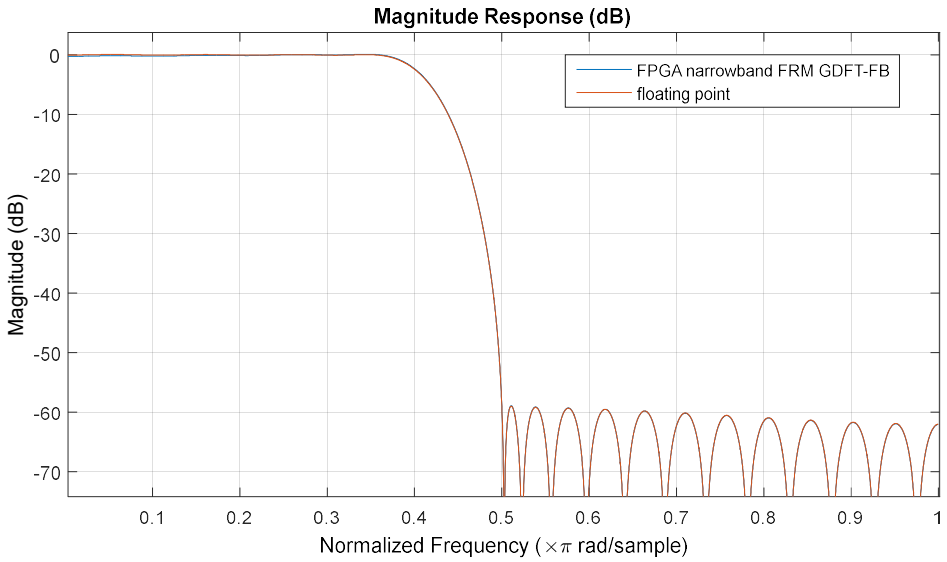

Figure 6.3 Frequency response of the FPGA based alternative narrowband FRM GDFT-FB (blue) and floating point reference implementation (red)

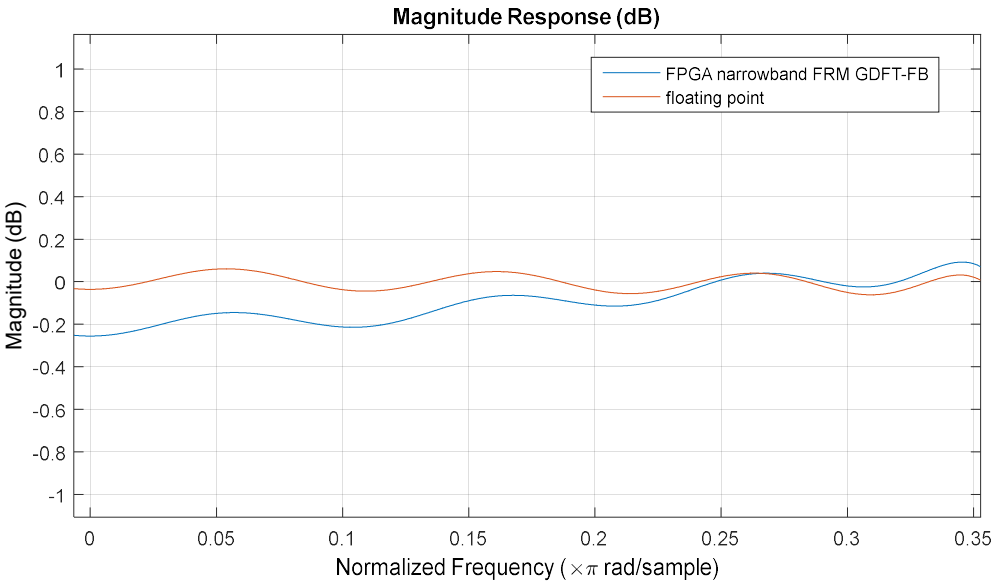

Figure 6.4 Passband comparison between FPGA based alternative narrowband FRM GDFT-FB (blue) and the floating point reference implementation (red line)

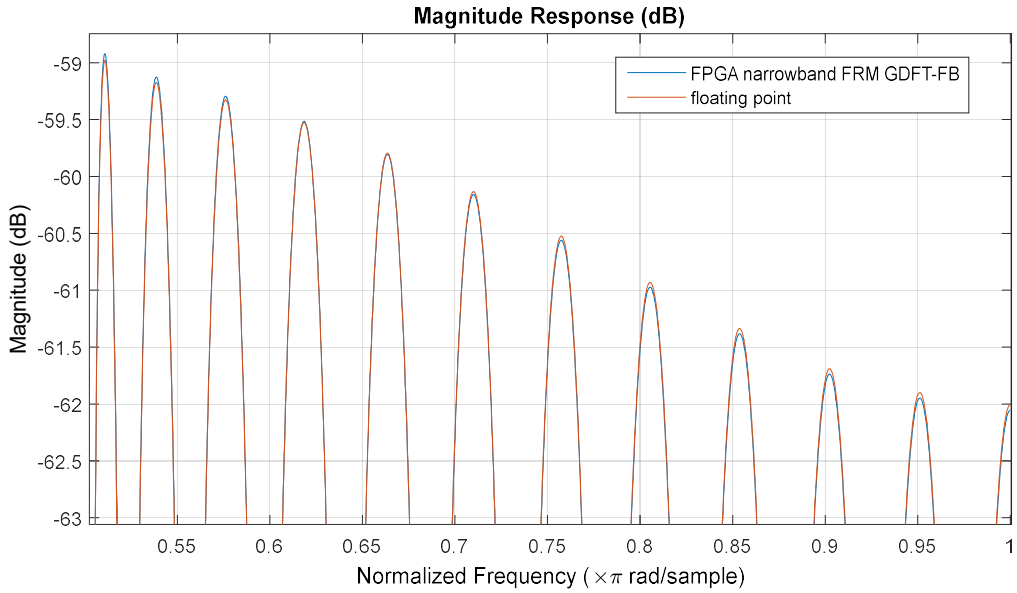

Figure 6.5 Stop band comparison between the FPGA based alternative narrowband FRM GDFT-FB (blue) and the floating point reference implementation (red)

We can see from these figures, that in the 256 channel configuration, the FPGA based alternative narrowband FRM GDFT-FB can still achieve the required filter specifications, as the passband ripple is about 0.2 dB and the stopband attenuation is just less than 59 dB, both within the 2 dB ripple and 55 dB attenuation requirement of the TETRA specification.

6.2.2 EVM and adjacent channel interference

The EVM constellation of one sub-band of the critically sampled GDFT-FB and alternative narrowband FRM GDFT-FB under maximum signal conditions is shown in Figure 6.6.

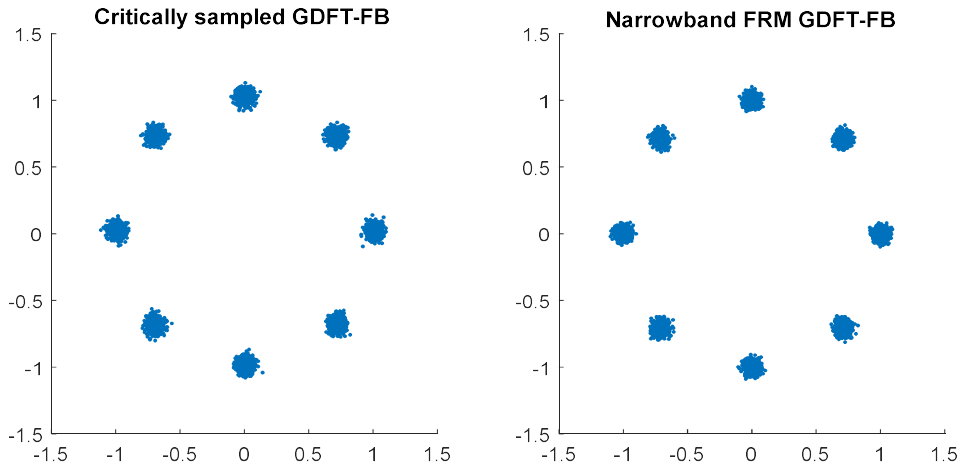

Figure 6.6 The EVM constellation of critically sampled GDFT-FB and alternative narrowband FRM GDFT-FB

The results data is presented in Table 6.2. Both filter-banks meet the TETRA requirements as expected. It is worth noting that the alternative narrowband FRM GDFT-FB has a better EVM performance than the critically sampled GDFT-FB.

Table 6.2 The EVM result of critically sampled GDFT-FB and alternative narrowband FRM GDFT-FB

	Critically sampled GDFT-FB	**Narrowband FRM GDFT-FB Alt.**	**TETRA requirement**
Peak	0.1488	0.1190	<0.3
RMS	0.0550	0.0445	<0.1

The adjacent channel interference was assessed at the most stringent limit of the specification when C/I_a= -45 dB. Figure 6.7 shows the constellations of both filter-bank subjected to this adjacent channel interference. The results data is given in Table 6.3. The results again indicate that both filter-bank designs meet the TETRA requirements. Again the alternative narrowband FRM exhibits better performance here indicating that it could potentially operate successfully at a higher level of adjacent channel interference.

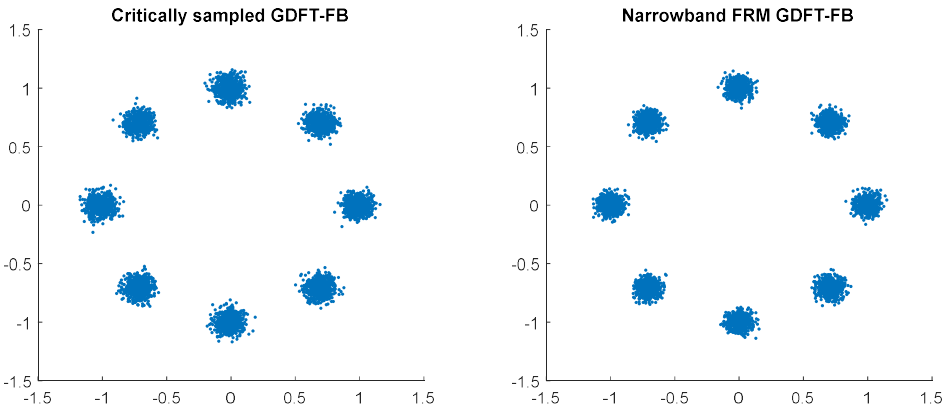

Figure 6.7 The EVM constellation of the FPGA based critically sampled GDFT-FB and alternative narrowband FRM GDFT-FB with C/I_a= -45 dB.

Table 6.3 The EVM result of critically sampled GDFT-FB and alternative narrowband FRM GDFT-FB under -45 dB adjacent channel interference

	Critically sampled GDFT-FB	**Narrowband FRM GDFT-FB Alt.**	**TETRA requirement**
Peak	0.2432	0.1896	<0.3
RMS	0.0813	0.0698	<0.1

6.2.3 Hardware resource usage

More channels cost more resources. Thus reusing hardware by running with a faster clock is essential. Considering the trade-off between resources usage and available clock in the practical system design, a 96 MHz clock rate is used in 256-channel evaluation.

The critically sampled DFT-FB/GDFT-FB and alternative narrowband FRM implementations employ different numbers of filter coefficients because of the way their filters are constructed. For the DFT-FB/GDFT-FB, the prototype filter as designed required 8704 coefficients which is 34 per channel. In contrast, the alternative narrowband FRM GDFT-FB only requires 768 coefficients for its masking filter, which is 3 coefficients per channel. However there is also a base filter with 61 coefficients required in each channel, but it can take advantages of the symmetric FIR structure to save about half of the computational load. It is also worth mentioning that, for the even stacked designs, all these coefficients are in real values. However, for the odd stacked designs, the 34 taps in each channel of GDFT-FB are all complex valued, whereas the

odd stacked alternative narrowband FRM GDFT-FB only has 3 complex coefficients from the masking filter in each channel. Even in the odd-stacked narrow band FRM GDFT-FB, the base filter coefficients are real-valued.

Table 6.4 Resource usage comparison of critically sampled GDFT-FB and alternative narrowband FRM GDFT-FB when configured for 256 channels

	Filter-types	**Slice registers**	**Slice LUTs**	**Block RAM 36**	**Block RAM 18**	**DSP48s**
Even stacked	**Critically sampled DFT-FB**	1386	1594	19	0	12
	Narrowband FRM DFT-FB Alt.	1765	2281	15	10	14
Odd stacked	**Critically sampled GDFT-FB**	1859	2579	38	0	22
	Narrowband FRM GDFT-FB Alt.	2756	2989	17	12	20
Available		301,440	150,720	416	832	768

Table 6.4 shows the resource usage for the critically sampled GDFT-FB and alternative narrowband FRM GDFT-FB with a clock rate of 96 MHz. The results suggest that the upper bound capacity of the Virtex-6 development board would be most constrained by the availability of DSP48 and Block RAM resources. For the even stacked channel allocation, the alternative narrowband FRM GDFT-FB will require slightly more resources: 2 more DSP48s are used compared to the critically sampled DFT-FB. However in the odd stacked channel allocation, the alternative narrowband FRM GDFT-FB has the advantage of requiring fewer complex operations. It uses 2 fewer DSP48s than the critically sampled GDFT-FB. In addition, it also saves more than 1/3 of the Block RAM, because Block RAM36 can be divided into 2 Block RAM18. This advantage could be expected to increase with larger numbers of channels and hence larger order prototype filters.

6.3 Chapter conclusion

In this chapter, we picked the best two FPGA designs – critically sampled GDFT-FB and alternative narrowband FRM, from chapter 3 ~ 5, based on their filtering result and hardware resources usage in a small scale 16-channel implementation. These designs were then scaled up to 256-channel implementations to test their feasibility, to validate the FPGA designs in a large multi-channel.

Then the steps and procedure to scale up the designs was described, in order to explain the modifications and parameter changes required to achieve this.

At last, the 256-channel implementations were evaluated. The results indicate that the filters necessary to meet the TETRA requirements could be both designed and physically implemented using both the critically sampled DFT-FB/GDFT-FB and narrow band FRM DFT-FB/GDFT-DB designs. It was also noted that for the filters as implemented, the alternative narrowband FRM GDFT-FB has a potential ability to work at a higher levels of adjacent channel interference than the critically sampled DFT-FB/GDFT-DB design.

If the wideband signal has an even stacked channel allocation, then the critically sampled GDFT-FB might be a good choice, because it requires the least resources among all the designs. If channels suffer severe adjacent channel interference, then the better resilience of the alternative narrowband FRM GDFT-FB may be preferred with a small amount of extra hardware usage.

For the wideband signal with odd stacked channel allocation, the alternative narrowband FRM GDFT-FB is the best option. It has a better filtering result, and uses even fewer hardware resources than odd stacked GDFT-FB.

Chapter 7

Conclusions and future work

7.1 Summary

In Chapter 2, a thorough introduction of the FPGA platform was presented. Its advantages of parallel computing and design flexibility were the main reasons we chose to develop the multirate DSP processing algorithms on this platform. Key FPGA elements, like DSP48s and block RAMs, were introduced in order to help the understanding of the hardware knowledge required for a FPGA design. The concept of Channelization was also explained and was in preparation for the new filter-bank designs given in the following chapter. Three types of channelization technologies were introduced. They are (1) the per-channel approach, (2) the pipelined frequency transform, and (3) the polyphase filter-bank. In the end, we chose the polyphaser filter-bank for the work due to its higher computational efficiency.

In Chapter 3, an in-depth description of DFT-FB was given as it was necessary for understanding how to proceed with the FPGA implementation. After that, we presented the concept and detail of odd stacked GDFT-FB. Odd stacked GDFT-FB offers a better spectrum usage by eliminating two of the half-bands at both ends of the even-stacking channels. In the odd stacked GDFT-FB FPGA realization, the original DFT-FB FPGA design is no longer valid because of the existence of complex coefficients which are due to the GDFT modulation. A ‘cross coupling’ model is introduced and applied to the FPGA architecture to deal with this. Other designs that allow the rest of the FPGA architecture to facilitate the ‘cross coupling’ are given as well. Finally, a test simulation of a 16 channel FPGA DFT-FB and GDFT-FB is carried out. The results showed that these FPGA filter-bank design have an acceptably good filtering result and continue to meet the required standard.

In Chapter 4, to overcome the aliasing problem caused by the critically sampled channelizer system, an oversampled design of even and odd stacked GDFT-FB is discussed. The oversampled GDFT-FB is based on the generalized GDFT-FB concept. However, the hardware implementation needs some significant alterations. First, a mathematical expression for oversampled modification by grouping channels into multiple FIR compilers is discussed and applied to the FPGA implementation. Secondly, the parallel output from the multiple FIR compilers has to be serialized so that the FFT can receive and process it. Finally, the odd stacked oversampled GDFT-FB design is needed to be combined with the 'cross coupling' model described in Chapter 3. In the final evaluation both 16 channels of an odd and even stacked oversampled GDFT-FB based on a FPGA passed the simulation test. However, the oversampled GDFT-FB has almost the same filtering result as the critically sampled GDFT-FB but with a greater hardware usage.

In Chapter 5, FRM technology was applied to GDFT-FB FPGA design. FRM is one of the computationally efficient approaches for designing a sharp FIR filter by cascading an interpolated FIR filter and a FIR filter with a relaxed specification instead of designing just one FIR with a very restricted specification. A new class of FRM FIR filter called subclass I filter was employed in combination with the GDFT-FB design to get a full FRM GDFT-FB. Also, the narrowband FRM technology was applied to an oversampled GDFT-FB to get a very efficient alternative narrowband FRM GDFT-FB. In terms of its FPGA realization, an arbitrary fragment clock divider was used to slow down the clock in a precise manner to provide a slow clock to the shift register for optimization reasons. Newly designed state machines were used to cope with any required frequency and phase shifting. Lastly, a new FIFO to read and write at different rates helped slow down the burst of samples and thus optimized the filtering process, which was also improved by the use of a clock divider. Finally, in the FPGA simulation testing both full FRM GDFT-FB and narrow-band FRM GDFT-FB give better filtering results than GDFT-FB. However, full FRM used much more hardware resources, especially the odd stacked version. Alternative narrowband FRM GDFT-FB is better as it only uses slightly more resources than critically sampled GDFT-FB, but could have a potentially greater hardware efficiency in cases with a large number of channels.

In Chapter 6, evaluations of the newly designed FPGA filter-banks for 256 channels were presented. The method to scale up the filter-bank design was introduced first. After the background of evaluation was given, the procedure of its testing and verification was described. Next, we presented how we chose the parameter values for the channel numbers and word-length. The results indicated that alternative narrowband FRM has a better filtering result and uses less hardware than critically sampled GDFT-FB in the odd stacked configuration. In an even stacked configuration, alternative narrowband FRM could work at a higher interference level than critically sampled GDFT-FB, but with a slightly greater resource usage.

7.2 Future work

The focus of this thesis was to implement new efficient and uniform channelizers based on FPGAs. However, there is still room for improvements. Some possible options of the future work are:

- Further efficiency of the FPGA uniform filter bank can be obtained from replacing FIR filter by IIR filter with an approximately linear phase response as shown in [10].

- Some non-uniform channelizers could be implemented based on the newly designed filter-banks in this thesis, like P-GDFT and R-GDFT introduced in § 2.2.3, in order to reduce the hardware resources usage.

- Use multiple FIR compiler core combination to achieve 1024 channel or even higher channel number of uniform channelizer, because FIR compiler has a limit of 512 channels in maximum for polyphase filter-bank implementation.

- Design a complex FIR core that can process input samples with complex coefficients. It could greatly simplify the design process for odd stacked polyphase filters.

7.3 Conclusions

Polyphase filter-bank plays a signification role in the DSP system. It has been widely studied during last 20 years. However, FPGA implementation studies of odd stacked GDFT-FB and oversampled GDFT-FB is quite few in the literature. The FPGA implementations of filter-banks with reconfigurable IP Cores are also rare. The work presented in this thesis shows efficient development of filter-bank based on FPGA with IP cores. At last, the more efficient technique, i.e., GDFT-FBs applied with FRM, which is developed in [53], has also been implemented and evaluated in FPGA with IP Cores.

At first, we introduced the background to multirate signal processing, followed by FPGA technology along with an explanation of its basic hardware architecture, and a discussion on channelization (Chapter 2). In the thesis the DFT-FB algorithm was chosen for our investigations and the basic implementation of an even stacked-FB with IP cores was the starting point of the work presented here. In order to have a better spectrum usage, a generalized version of DFT-FB was introduced, that is available to be modified to have an odd stacked channel allocation (Chapter 3). To overcome problems of aliasing both even and odd stacked oversampled GDFT-FBs have been implemented according to the GDFT-FB concept (Chapter 4). An FPGA GDFT-FB extended with FRM technologies was implemented as a further development of GDFT-FB, where FRM allowed us to design a sharp filter using only a small number of coefficients (Chapter 5). The procedures involved in implementing these channelizer systems and the subsequent evaluation of results facilitated comparison between the different filter-banks. A 256-channel scale up evaluation has been performed among the filter-banks, in order to validate the DSP performance and feasibility in the practical industry system (Chapter 6).

The results show that odd-stacked GDFT-FB will require more hardware resources than the DFT-FB design, due to the complex filtering requires more FPGA design. But it is still very efficient compare to most of channelizer design. However, oversampled GDFT-FBs achieve essentially the same result as critically sampled GDFT-FB, but use more hardware resources. On the other hand, alternative narrowband FRM GDFT-FB only use slightly more hardware recourses than DFT-FB with the even stacked configuration in large number channels evaluation. While, in the odd stacked configuration, alternative

narrowband FRM GDFT-FB will require less hardware resources than GDFT-FB. The result is consistent to the studies in [53].

During design and implementation of the uniform channelizer on FPGA, the following details need to be considered carefully:

- In order to design a prototype filter for a FPGA based polyphase filter-bank to meet the standard, we need to overdesign the prototype filter a little bit, because the fixed point quantization error will has an impact to a filters passband ripple and reduce the attenuation to the stopband.

- When designing a FPGA filter bank, we find that longer length FIR filter would produce more distortion to the filter [68], as more coefficients will cause the overflow to the fixed point accumulator of DSP48s.

- During design an odd stacked GDFT-FB based on FPGA, synchronization among trigger signal 'rdy' and output from FIR core will be the biggest problem, because complex filtering will cost extra delays comparing to even stacked design.

- During design an oversampled GDFT-FB based on FPGA, serialization of output samples from multiple FIR blocks will be the biggest challenge, as these FIR blocks will filter input samples in parallel. Thus FIFOs need to be design independently according to number of samples per time and the length of delay.

- When designing odd stacked GDFT-FB and oversampled GDFT-FB, the output samples from FIR cores may have different fractional length, due to the different values of coefficients inserted. It is important to unite the 16 bit fixed point output to have exactly the same factional length.

References

[1] L. R. Rabiner and B. Gold, "Theory and application of digital signal processing," *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.,* vol. 1, 1975.

[2] F. J. Harris, *Multirate Signal Processing for Communication Systems*: Prentice Hall PTR, 2004.

[3] V. Strela, P. N. Heller, G. Strang, P. Topiwala, and C. Heil, "The application of multiwavelet filterbanks to image processing," *Image Processing, IEEE Transactions on,* vol. 8, pp. 548-563, 1999.

[4] F. Jabloun, A. E. Cetin, and E. Erzin, "Teager energy based feature parameters for speech recognition in car noise," *Signal Processing Letters, IEEE,* vol. 6, pp. 259-261, 1999.

[5] F. J. Harris, C. Dick, and M. Rice, "Digital receivers and transmitters using polyphase filter banks for wireless communications," *Microwave Theory and Techniques, IEEE Transactions on,* vol. 51, pp. 1395-1412, 2003.

[6] D. Y. Pan, "Digital audio compression," *Digital Technical Journal,* vol. 5, pp. 28-40, 1993.

[7] A. P. Navarro, R. Villing, and R. J. Farrell, "Practical Non-Uniform Channelization for Multistandard Base Stations," *ZTE Communications,* vol. 09, pp. 15-24, 2011.

[8] N. Fliege, "Computational efficiency of modified DFT polyphase filter banks," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, 1993, pp. 1296-1300.

[9] M. Vetterli, "Filter banks allowing perfect reconstruction," *Signal processing,* vol. 10, pp. 219-244, 1986.

[10] M. Dehghani, R. Aravind, and K. Prabhu, "Design of M-channel IIR uniform DFT filter banks using recursive digital filters," *ETRI journal,* vol. 25, pp. 345-355, 2003.

[11] B. Farhang-Boroujeny, "Filter bank spectrum sensing for cognitive radios," *Signal Processing, IEEE Transactions on,* vol. 56, pp. 1801-1811, 2008.

[12] C. Dick and Y. Krikorian, "A system-level design approach for FPGA-based DSP implementations," *DSP World, Spring,* 1999.

[13] N. S. Bhatia, "A Physical layer implementation of Reconfigurable Radio," 2004.

[14] P. K. Devi and R. S. Bhuvaneshwaran, "FPGA implementation of coefficient decimated polyphase filter bank structure for multistandard communication receiver," *Journal of Theoretical and Applied Information Technology,* vol. 64, 2014.

[15] A. Ambede, K. G. Smitha, and A. P. Vinod, "A new low complexity uniform filter bank based on the improved coefficient decimation method," *Radioengineering,* vol. 22, 2013.

[16] S. A. Fahmy and L. Doyle, "Reconfigurable polyphase filter bank architecture for spectrum sensing," presented at the Proceedings of the 6th international conference on Reconfigurable Computing: architectures, Tools and Applications, Bangkok, Thailand, 2010.

[17] J. Lillington, "Comparison of Wideband Channelisation Architectures," in *International signal processing conference, Dallas*, 2003.

[18] P. Bricaud, *Reuse methodology manual: for system-on-a-chip designs*: Springer Science & Business Media, 2012.

[19] GVR. (2014). *FPGA (Field-Programmable Gate Array) Market Analysis By Application (Automotive, Consumer Electronics, Data Processing, Industrial, Military And Aerospace, Telecom) And Segment Forecasts To 2020*. Available: http://www.grandviewresearch.com/industry-analysis/fpga-market

[20] R. Joost and R. Salomon, "Advantages of FPGA-based multiprocessor systems in industrial applications," in *Industrial electronics society, 2005. IECON 2005. 31st annual conference of IEEE*, 2005, p. 6 pp.

[21] J.-P. Delahaye, G. Gogniat, C. Roland, and P. Bomel, "Software radio and dynamic reconfiguration on a DSP/FPGA platform," *Frequenz,* vol. 58, pp. 152-159, 2004.

[22] D. Bester, J. Du Toit, and J. Enslin, "High performance DSP/FPGA controller for implementation of computationally intensive algorithms," in *Industrial Electronics, 1998. Proceedings. ISIE'98. IEEE International Symposium on*, 1998, pp. 240-244.

[23] W. Wang, M. Swamy, and M. Ahmad, "Low power FIR filter FPGA implementation based on distributed arithmetic and residue number system," in *Circuits and Systems, 2001. MWSCAS 2001. Proceedings of the 44th IEEE 2001 Midwest Symposium on*, 2001, pp. 102-105.

[24] A. Tolmachev, M. Orbach, M. Meltsin, R. Hilgendorf, T. Birk, and M. Nazarathy, "Real-time FPGA implementation of efficient filter-banks for digitally sub-banded coherent DFT-S OFDM receiver," in *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC), 2013*, 2013, pp. 1-3.

[25] J. Garcia and R. Cumplido, "On the design of an FPGA-Based OFDM modulator for IEEE 802.11 a," in *Electrical and Electronics Engineering, 2005 2nd International Conference on*, 2005, pp. 114-117.

[26] K.-C. Liu, W.-C. Lin, and C.-K. Wang, "A pipelined digital differential matched filter fpga implementation and vlsi design," in *Custom Integrated Circuits Conference, 1996., Proceedings of the IEEE 1996*, 1996, pp. 75-78.

[27] J. Rose, A. E. Gamal, and A. Sangiovanni-Vincentelli, "Architecture of field-programmable gate arrays," *Proceedings of the IEEE,* vol. 81, pp. 1013-1029, 1993.

[28] U. Meyer-Baese and U. Meyer-Baese, *Digital signal processing with field programmable gate arrays* vol. 65: Springer, 2007.

[29] i. Xilinx, "Virtex-6 FPGA DSP48E1 Slice," i. Xilinx, Ed., v1.3 ed, 2011.

[30] S. Palnitkar, *Verilog HDL: a guide to digital design and synthesis* vol. 1: Prentice Hall Professional, 2003.

[31] S. W. Smith, "The scientist and engineer's guide to digital signal processing," 1997.

[32] R. Yates, "Fixed-point arithmetic: An introduction," *Digital Signal Labs,* vol. 81, p. 198, 2009.

[33] "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2008,* pp. 1-70, 2008.

[34] K. C. Zangi and R. D. Koilpillai, "Software radio issues in cellular base stations," *Selected Areas in Communications, IEEE Journal on,* vol. 17, pp. 561-573, 1999.

[35] R. Regulations, "International Telecommunication Union," *Radiocommunication Sector. ITU-R. Geneva,* 2008.

[36] T. Hentschel, "Channelization for software defined base-stations," in *Annales des Telecommunications*, 2002, pp. 386-420.

[37] T. Hentschel, *Sample rate conversion in software configurable radios*: Artech House, 2002.

[38] J. Lillington, "RF Engines Limited White Paper,“The Pipelined Frequency Transform (PFT)(PFR architecture and comparisons with FFT/digital down-converter techniques)”, Reference No," *PFT,* vol. 1, pp. 1-14.
[39] P. Vaidyanathan, "Quadrature mirror filter banks, M-band extensions and perfect-reconstruction techniques," *ASSP Magazine, IEEE,* vol. 4, pp. 4-20, 1987.
[40] H. Scheuermann and H. Göckler, "A comprehensive survey of digital transmultiplexing methods," *Proceedings of the IEEE,* vol. 69, pp. 1419-1450, 1981.
[41] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*: Prentice-Hall, 1983.
[42] N. Fliege, *Multirate digital signal processing: multirate systems, filter banks, wavelets*: Wiley, 1994.
[43] A. P. Navarro, T. Keenan, R. Villing, and R. Farrell, "Non-uniform channelization methods for next generation SDR PMR base stations," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, 2011, pp. 620-625.
[44] Y. Neuvo, C.-Y. Dong, and S. K. Mitra, "Interpolated finite impulse response filters," *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 32, pp. 563-570, 1984.
[45] L. Yong Ching, "Frequency-response masking approach for the synthesis of sharp linear phase digital filters," *Circuits and Systems, IEEE Transactions on,* vol. 33, pp. 357-364, 1986.
[46] S. Radhakrishnan Pillai and G. H. Allen, "Generalized magnitude and power complementary filters," in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, 1994, pp. III/585-III/588 vol.3.
[47] H. Johansson, "New classes of frequency-response masking FIR filters," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, 2000, pp. 81-84 vol.3.
[48] E. ETSI, "300 392-2 V3. 2.1 (2007-09) Terrestrial Trunked Radio (TETRA)," *Voice plus Data,* vol. 500.
[49] E. ETSI, "300 394-1 (V3. 1.1):" Terrestrial Trunked Radio (TETRA)," *Conformance testing specification*.
[50] P. Stavroulakis, *Terrestrial trunked radio-TETRA: a global security tool*: Springer Science & Business Media, 2007.
[51] P. D. Fiore, "Low-Complexity Implementation of a Polyphase Filter Bank," *Digital Signal Processing,* vol. 8, pp. 126-135, 4// 1998.
[52] S. Berner and D. Leon, *FPGA-based filter bank implementation for parallel digital signal processing*: National Aeronautics and Space Administration, 1999.
[53] Á. Palomo-Navarro, R. J. Farrell, and R. Villing, "Combined FRM and GDFT filter bank designs for improved nonuniform DSA channelisation," *Wireless Communications and Mobile Computing,* 2015.
[54] IP. LogiCORE, "FIR Compiler v5. 0, Xilinx," *Inc., San Jose, CA, USA,* 2010.
[55] K. W. Martin, "Complex Signal Processing is Not Complex," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 51, pp. 1823-1836, 2004.
[56] I. Koren, *Computer arithmetic algorithms*: Universities Press, 2002.
[57] P. L. De Leon, "On the use of filter banks for parallel digital signal processing," in *7th NASA Symposium on VLSI Design,(Albuquerque, NM)*, 1998.
[58] E. Satorius, W. Ying-Wah, B. LaRocca, and J. Kosinski, "Implementation of Polyphase Channelizers for Multirate Signal Analysis," in *Signals, Systems and*

Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on, 2006, pp. 1170-1174.

[59] H. Johansson and T. Saramäki, "Two-Channel FIR Filter Banks Utilizing the FRM Approach," *Circuits, Systems and Signal Processing,* vol. 22, pp. 157-192, 2003/03/01 2003.

[60] P. S. R. Diniz, L. C. R. Barcellos, and S. L. Netto, "Design of cosine-modulated filter bank prototype filters using the frequency-response masking approach," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, 2001, pp. 3621-3624 vol.6.

[61] S. L. Netto, P. S. R. Diniz, and L. C. R. Barcellos, "Efficient implementation for cosine-modulated filter banks using the frequency response masking approach," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 2002, pp. III-229-III-232 vol.3.

[62] L. Rosenbaum, P. Lowenborg, and H. Johansson, "Cosine and sine modulated FIR filter banks utilizing the frequency-response masking approach," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, 2003, pp. III-882-III-885 vol.3.

[63] Á. P. Navarro, "Channelization for Multi-Standard Software-Defined Radio Base Stations," Unpublished doctoral thesis, National University of Ireland Maynooth, 2011.

[64] L. Rosenbaum, P. Lowenborg, and H. Johansson, "An Approach for Synthesis of Modulated -Channel FIR Filter Banks Utilizing the Frequency-Response Masking Technique," *EURASIP Journal on Advances in Signal Processing,* vol. 2007, p. 068285, 2007.

[65] L. Gu, N. Zhou, and H. Li, "Research of Frequency Divider Based on Programmable Logic Device," *Procedia Environmental Sciences,* vol. 10, pp. 820-824, 2011.

[66] S. W. Zhang and C. Zhao, "Design for realizing arbitrary fractional divider based on FPGA which duty cycle is up to 50%," in *Applied Mechanics and Materials*, 2013, pp. 1653-1657.

[67] P. P. Vaidyanathan, *Multirate systems and filter banks*: Prentice-Hall, Inc., 1993.

[68] R. Yates, "Practical considerations in fixed-point fir filter implementations," *Digital Signal Labs, Technical Reference,* 2007.