Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2006

MSc THESIS

Scalable and Reconfigurable Digital Front-End for SDR Wideband Channelizer

Gil Savir

Abstract

In recent years, RF receiver designers concentrated on replacing analog components with digital ones, striving towards the ideal Software Defined Radio (SDR) where all signal processing is done in software. Such an ideal SDR platform may form an exceptionally flexible and reprogrammable receiver that can cope with many different standards, e.g., IS-95, GSM, UMTS, and especially the various military standards. A wideband receiver has to simultaneously deal with hundreds to few thousands channels, which lay in the same spectrum interval. One of the most computation intensive tasks in such receiver is channelization [1]. A wideband channelizer decomposes its RF input signal into separate outputs, each containing the signal of single channel. In the past, practical limitations such as state-of-the-art digitizers' speed and computing capacity prevented the realization of a wideband SDR receiver. At present, these implications can be overcome using digital front-end architectures, comprising reconfigurable and scalable components (e.g., FPGA, FFT-processors) allowing flexible and efficient implementation. The goal of the presented research is to study, design, and implement a flexible and reconfigurable wideband channelizer architecture that can be implemented on state-of-the-art FPGAs. In this dissertation, we present our work where we first choose a suitable algorithm for wideband channelization. The chosen algorithm employs an analysis DFT filterbank [2] that requires fewer hardware resources compared to other channelization algorithms. Subsequently, we simulate this algorithm for a broad range of practical parameters in order to determine hardware design requirements and performance trade-offs. Using the parameters survey, a test-case is devised and implemented on FPGA using our implementation architecture. Subsequently, the implementation results are compared to the simulation results in order to validate the parameter ranges survey.

eonic
digitally mastering the spectrum

CE-MS-2006-16

Scalable and Reconfigurable Digital Front-End for SDR Wideband Channelizer

MSc THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Gil Savir
born in Afula, Israel

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Scalable and Reconfigurable Digital Front-End for SDR Wideband Channelizer

by Gil Savir

Abstract

In recent years, RF receiver designers concentrated on replacing analog components with digital ones, striving towards the ideal Software Defined Radio (SDR) where all signal processing is done in software. Such an ideal SDR platform may form an exceptionally flexible and reprogrammable receiver that can cope with many different standards, e.g., IS-95, GSM, UMTS, and especially the various military standards. A wideband receiver has to simultaneously deal with hundreds to few thousands channels, which lay in the same spectrum interval. One of the most computation intensive tasks in such receiver is channelization [1]. A wideband channelizer decomposes its RF input signal into separate outputs, each containing the signal of single channel. In the past, practical limitations such as state-of-the-art digitizers' speed and computing capacity prevented the realization of a wideband SDR receiver. At present, these implications can be overcome using digital front-end architectures, comprising reconfigurable and scalable components (e.g., FPGA, FFT-processors) allowing flexible and efficient implementation. The goal of the presented research is to study, design, and implement a flexible and reconfigurable wideband channelizer architecture that can be implemented on state-of-the-art FPGAs. In this dissertation, we present our work where we first choose a suitable algorithm for wideband channelization. The chosen algorithm employs an analysis DFT filterbank [2] that requires fewer hardware resources compared to other channelization algorithms. Subsequently, we simulate this algorithm for a broad range of practical parameters in order to determine hardware design requirements and performance trade-offs. Using the parameters survey, a test-case is devised and implemented on FPGA using our implementation architecture. Subsequently, the implementation results are compared to the simulation results in order to validate the parameter ranges survey.

Laboratory : Computer Engineering
Codenumber : CE-MS-2006-16

Committee Members :

Advisor:	Stephan Wong
Advisor:	Laurens Bierens
Chairperson:	Stamatis Vassiliadis
Member:	Alle-Jan van der Veen
Member:	Sorin Cotofana

To my mother and in memory of my father

Contents

List of Figures **vii**

List of Tables **ix**

Acknowledgements **xi**

1 Introduction **1**
1.1 Background & Scope . . . 1
1.2 Related Work . . . 2
1.3 Research Question & Goals . . . 2
1.4 Methodology . . . 3
1.5 Thesis Overview . . . 3

2 Software Defined Radio **5**
2.1 The SDR Concept . . . 5
2.2 Conclusion . . . 7

3 Channelization Algorithms Study **9**
3.1 Channelization Algorithms . . . 9
3.1.1 The per-channel Approach . . . 10
3.1.2 Pipelined Frequency Transform . . . 10
3.1.3 Polyphase FFT . . . 12
3.2 Algorithms Comparison . . . 15
3.2.1 Hardware Complexity Comparison . . . 15
3.2.2 Qualitative Comparison . . . 16
3.3 Conclusion . . . 19

4 Channelizer Architecture **21**
4.1 Modules Decomposition . . . 21
4.2 IQ Demodulator . . . 22
4.2.1 Conventional IQ demodulator . . . 22
4.2.2 Wideband IQ Demodulator . . . 22
4.2.3 Hilbert Transformed IQ Demodulator . . . 23
4.2.4 Chosen Implementation Algorithm . . . 23
4.3 Filterbank . . . 24
4.3.1 Equiripple Prototype Filter . . . 24
4.3.2 1/f Ripple Prototype Filter . . . 25
4.4 FFT . . . 25
4.5 Post-Processing . . . 26
4.6 Conclusion . . . 26

5 Parameter Ranges Survey **27**
5.1 Simulation setup . . . 27
5.1.1 IQ demodulator Filter Design Parameters . . . 27
5.1.2 Prototype Filter Design Parameters . . . 28
5.2 Simulation Results . . . 31
5.2.1 IQ Demodulator Filter Design Simulation Results . . . 31
5.2.2 Prototype Filter Design Simulation Results . . . 33
5.3 Conclusion . . . 36

6 Validation **37**
6.1 Filterbank Implementation . . . 37
6.1.1 Test-Case . . . 37
6.1.2 Filterbank Implementation Approach . . . 38
6.1.3 Filterbank Architecture . . . 38
6.2 Results . . . 40
6.2.1 Functional Verification . . . 40
6.2.2 Test-Case Results . . . 41
6.2.3 General Results . . . 41
6.2.4 Conclusion . . . 42

7 Conclusions & Recommendations **45**
7.1 Conclusions . . . 45
7.2 Main Contributions . . . 47
7.3 Recommendations . . . 47

Bibliography **51**

List of Figures

1.1 4-channels channelizer 2

2.1 Typical superheterodyne radio receiver 5
2.2 Ideal SDR receiver 6
2.3 Feasible SDR receiver 6

3.1 Single channel digital front-end channelizer 9
3.2 Per-channel channelizer 10
3.3 DDC-SRC tree 11
3.4 DDC-SRC tree 11
3.5 Modifications to the k^{th} single channel channelizer 12
3.6 l branches in the filterbank decomposition of the k^{th} single channelizer . 13
3.7 Applying the noble identity in the k^{th} filterbank 14
3.8 Discarding M-1 filterbanks 14
3.9 Polyphase FFT channelizer 15
3.10 Comparison of LUT utilization 16
3.11 Comparison of memory bits employment 16

4.1 Channelizer architecture 21
4.2 IQ-demodulators 22
4.3 HBF/CHBF Impulse and frequency response 23
4.4 Hilbert transformed HBF Impulse and frequency response 24
4.5 Hilbert transformed IQ demodulators 24
4.6 Equiripple vs. 1/f ripple prototype filters 25
4.7 1/f ripple prototype filters 26

5.1 r_{PB} as function of r_{SB} in HBF 29
5.2 Prototype filter design parameters choice for 2 and 4 channels 29
5.3 Number of channels vs. taps 31
5.4 Stopband Attenuation vs. number of taps 32
5.5 Passband width vs. number of taps 32
5.6 Quantization limits 33
5.7 Taps per channel vs. stopband attenuation; various passband ripple values 34
5.8 Taps/channel per $10dB$ vs. passband ripple 34
5.9 Quantization limit for 3dB passband ripple 35
5.10 Minimal coefficients word-length vs. stopband attenuation performance for different number of channels 35

6.1 Filterbank implementation architecture for 4 channels with 5 taps per channel 39
6.2 HW testbench configuration 40
6.3 SNR 42

List of Tables

3.1 Qualitative Comparison 18

6.1 Test-case parameters 38
6.2 HW-cost and performance for various feasible implementation schemes 43

Acknowledgements

I would like to thank Laurens Bierens for his guidance during this research and to EONIC BV for supporting this project. I would also like to thank EONIC's employees that always had time for helping me when necessary.
A special thank that cannot be said or written in words to family Koedood, that became my family in The Netherlands and to my mother, who made me what I am.

Gil Savir
Delft, The Netherlands
September 25, 2006

1 Introduction

Advances in technology in the recent decades led to a gradual migration of RF systems from the analog to the digital domain. It gave birth to the concept of software defined radio (SDR) - a radio platform that digitally processes RF signals on a software-driven platform (i.e., digital signal processor, general purpose processor, etc.) and thereby provides flexible reconfigurable transceiver architecture that may cope with multiple standards and air-interfaces, dynamically adapting to its radio environment [3]. However, hitherto, SDR implementations do not take full advantage of the SDR concept due to current limited performance of software-driven platforms. In order to alleviate this problem some computationally intensive tasks are performed on a digitally reconfigurable platform, such as FPGAs. Late progress in reconfigurable technology makes SDR implementations closer than ever to reach the full potential of the SDR concept.

1.1 Background & Scope

SDR has a broad range of applications, both in civil and in military environments. This study, however, is focused on military electronic warfare (EW) applications such as electronic intelligence (ELINT), signal intelligence (SIGINT), and especially communications intelligence (COMINT) equipment. Many wireless (and wired) communication methods are based on frequency division multiplexing (FDM) encoding. In this method all communication channels of certain application are spread in a frequency band, which is allocated for this purpose by the local communication authority. The channels are allocated in equally, non-overlapping frequency spaces. In oder to intercept and process such communication, the RF signal has to be channelized first.

Channelization (in this context) is the process of separating a mixture of communication channels into distinct signals, each of single channel. Figure 1.1 illustrates the functionality of a 4-channels channelizer. It has a single input that contains 4 communication channels in one signal, and it has 4 distinct outputs, each providing a single channel filtered from the rest and down-converted to baseband frequency (DC), ready for further processing.

In COMINT applications, a channelization of wide frequency band needs to be performed in real-time for the aim of signal interception. Channelization, however, may also be useful in civil environment (e.g., cellular base-stations and satellite communication). The scope of this work is a study, design, and implementation of digital front-end

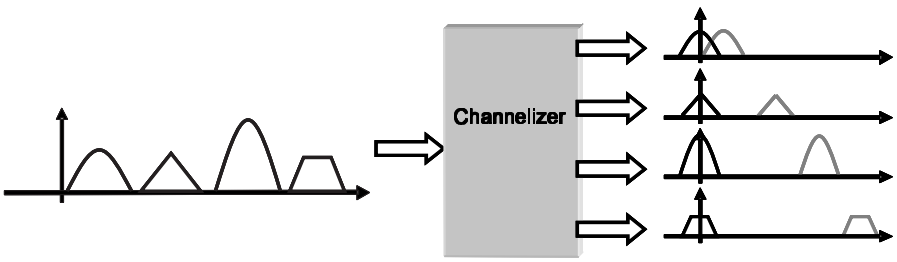


Figure 1.1: 4-channels channelizer

for SDR receiver, containing a wideband[1] channelizer.

1.2 Related Work

In this section we present a brief account of several works on related channelization algorithms, which are suitable for digital front-end channelizer. Most of this related work is further explained in details in Chapter 3.

In the past two decades, several digital channelization algorithms were introduced. Some of them, such as the per-channel approach [4], emerged from existing analog methods already in use. This algorithm employs a stack of single-channel channelizers, where each one is a digital realization of the traditional manner for realizing analog-based single-channel channelizer. However, rapid improvements in silicon density and software-driven platforms enabled efficient and economical implementation of digitally-based channelization algorithms. Such algorithm is the hierarchical multistage method (HMM), in which the input signal is consequently channelized to two channels in a binary-tree form [5]. Another is the frequency domain filtering (FDF) channelization, which performs, as its name suggests, filtering of the required channels in the frequency domain, after the input signal is passed in FFT [6]. An improvement of the HMM is the pipelined frequency transform (PFT) channelization algorithm that takes advantage of sample-rate differences among distinct stages in the binary-tree [7]. Another two closely related algorithms are the polyphase FFT filterbank channelizer [2] and the weight overlap-add (WOLA) [8]. These are based on enhancement of the per-channel approach, which employs sample-rate conversion properties.

1.3 Research Question & Goals

Various studies of the channelization algorithms mentioned in the former section exist and several refinements and application-specific approaches were published (e.g, [9–11]). However, these publications usually have a narrow scope and are meant for

[1]The term wide-band means "of relatively big spectrum interval". This term, however, depends upon application and is current-technology related. In order to be more specific we state here that we aim towards channelizers of few hundreds to few thousands of channels.

a particular field or specific implementation (e.g., a base-station for certain cellular communication standard), where normally channelization of only few tens to couple of hundreds channels is required. There are, however, applications where various configurations of channelization may be necessary. A few hundreds to few thousands of channels could be required with various ranges of channel parameters. This work, therefore, will be concentrated in answering the following question:

- How to design a generic, scalable, and reconfigurable digital front-end architecture for software defined radio wideband channelizer, which should form the basis of a next-generation SDR platform?

In order to answer this question we form the following goals:

- Choose digital algorithm which is most appropriate for wideband channelization in a reconfigurable environment.
- Determine the relationships and trade-offs between the various channelizer parameters.
- Establish implementation architecture for the chosen algorithm.
- Demonstrate architecture feasibility on currently-available FPGAs for applicable parameters.

1.4 Methodology

This section describes the methodology chosen with the purpose of answering the research question and achieving the goals presented in Section 1.3. This work comprises the following three phases:

1. **Examination of channelization algorithms:** in this phase background literature study is conveyed, and several prominent channelization algorithms are studied. This phase is concluded with a comparison between the surveyed algorithms, by which one is chosen for further investigation.
2. **Parameter ranges survey:** in this phase the chosen channelization algorithm is studied in further details and modeled in software in order to indicate its scalability and reconfigurability, and with the aim of identifying critical items.
3. **Test-case implementation** in this phase a probable test-case is worked out based on the parameter ranges survey, and critical items are implemented so as to validate conclusions.

1.5 Thesis Overview

The reminder of this paper is structured as follows:

- Chapter 2 introduces the SDR concept as a background for this work, highlighting advantages and drawbacks of this concept.
- Chapter 3 describes three channelization algorithms, namely the per-channel algorithm, the pipelined frequency transform, and the polyphase FFT filterbank algorithm. Thereafter, it presents a HW- cost and qualitative comparison. Based on this comparison, the polyphase FFT filterbank algorithm is chosen for further investigation.
- Chapter 4 discusses in further details the architecture of the polyphase FFT filterbank channelizer, focusing on the IQ-demodulator and the filterbank.
- Chapter 5 presents parameter ranges survey of the IQ-demodulator and the filterbank modules, while introducing graphs that provide insight to design trade-offs of these two modules.
- Chapter 6 introduces our implementation architecture for the filterbank and presents the results obtained from its implementation. This is done in order to validate the results obtained in the parameter ranges survey.
- Chapter 7 presents the conclusions of this dissertation, its main contributions, and recommendations for possible future continuation work.

2 Software Defined Radio

The necessity for software defined radio (SDR) emerged from military applications where communication between several different forces (i.e., air-force, ground force, navy, etc.) had to be facilitated while preventing interception by enemy forces. DARPA's SPEAKeasy [12] and JTRS [13] projects are examples for development of SDR, where multiple air-interfaces with different signal processing techniques were integrated into one platform. However, the necessity for SDR also exists in civil applications. Typical example is a cellular phone that is capable of operating within the different existing standards (UMTS, GSM, DCS-1800, IS-95, JDC, and many more).

2.1 The SDR Concept

Figure 2.1 illustrates the structure of a typical superheterodyne radio receiver. In such receiver, the signal passes through many analog components (e.g., amplifiers, filters, and mixers) that have non-ideal performance and are subject to influence such as temperature differences and humidity. Therefore, the signal accumulates many distortions along its processing path.

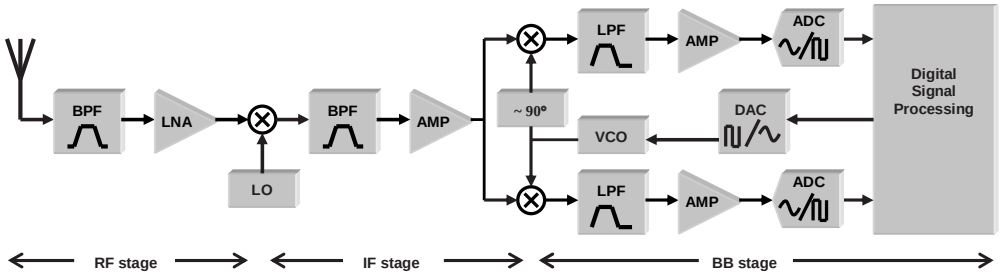


Figure 2.1: Typical superheterodyne radio receiver

An ideal SDR receiver should be capable of receiving (and transmitting) ultra-wide bandwidth of RF signals (hundreds of *Mhz* to few *GHz*), interpreting many given air-interface radio standard using software. In order to do so, the ADC should be "shifted" as close as possible to the receiver's antenna as illustrated in Figure 2.2. Compared to the traditional superheterodyne receiver in Figure 2.1, the ideal SDR receiver contains minimal quantity of analog components. Earlier conversion of the RF signal to digital not only allows more flexibility in signal processing but also provides higher signal fidelity as analogue components do not perform ideally and might significantly alter their behavior due to external influence (i.e., temperature, humidity, etc.). Other ad-

vantages of digital components are small footprint, low power consumption, and fast development (time to market).

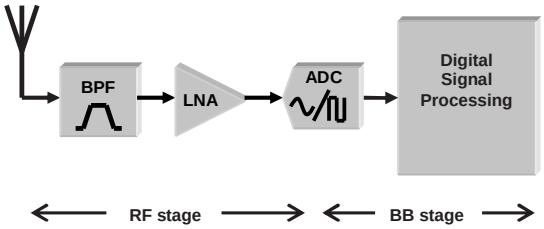


Figure 2.2: Ideal SDR receiver

However, some key issues still prevent a realization of an ideal SDR receiver. Antenna that ideally receives and transmits wide band of frequency is not realizable with currently-available technology. Suppose the antenna problem is overcome, another problem stems from the ADC bottleneck [14]. State of the art ADCs reach about 3 Giga-samples per second (GSPS), and also this with a relatively low 8-bit resolution (e.g., National Semiconductor's ADC08D1500). According to the Nyquist-Shannon sampling theorem, a periodic signal should be sampled in rate, which is at least twice its frequency in order to be able to reconstruct it. A 2 GSPS ADC could therefore sample periodic signals up to 1 GHz of frequency. Other fundamental limitations of ADC due to its non-ideal nature are low resolution (quantization error), non-linear behavior, deviation from accurate sample timing intervals (jitter error) and noise, which limit its performance [4]. The third inherent problem in the ideal SDR results from the limit of nowadays computation power. Assuming that a perfect ADC exists, the amount of digital information (samples) to be processed by the digital signal processing unit(s) surpasses the computation capacity of presently available computing platforms and might require Giga-FLOPS performance [15].

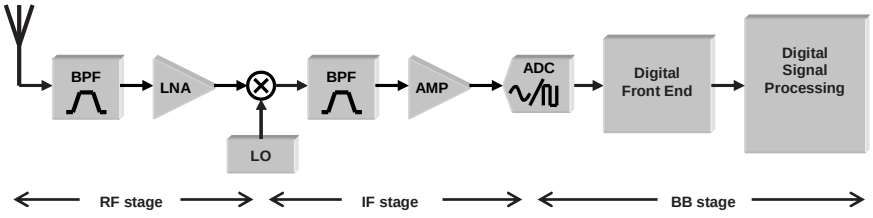


Figure 2.3: Feasible SDR receiver

The limitations discussed above lead to the conclusion that a compromise should be devised in order to facilitate the implementation of SDR receiver. Figure 2.3 depicts possible architecture for a feasible SDR receiver. Limiting the bandwidth (BW) of the receiver makes it possible to devise a suitable antenna and to alleviate ADC sampling

rate limitation and computation load. Therefore, IF stage is introduced in order to deal with ADC bandwidth limitations. Furthermore, a digital front-end stage is appended in front of the digital signal-processing platform can take over computationally intensive tasks from the software-driven platform. These tasks may include sample rate conversion, channelization, filtering, and other tasks derived from the receiver's target application. The digital front-end is likely to be implemented in firmware (FPGAs) and flexible ASIC digitizers, which provide a trade-off between performance and flexibility.

Channelization can be realized in the digital front-end. However, it is not independent of the analog front-end. The properties of components in the analog front-end (e.g., ADC and LNA) should be taken into account when designing the digital front-end. Also, further processing in the digital processing platform should be taken into consideration. Some properties of the digital front-end can be imposed trough requirements from the digital processing platform, such as channels spacing, sample rate conversion, etc.

2.2 Conclusion

In this chapter we introduced the SDR concept. We presented the ideal SDR receiver and showed its advantages above typical radio receivers, which are implementation flexibility and reconfigurability, improved accuracy, better robustness towards external environment influence, small footprint, low power consumption, and fast time-to-market. Subsequently, we explained the reasons for ideal SDR receiver unattainability. Consequently, we introduced a feasible SDR receiver with analog and digital front-ends, where the digital front-end takes over computationally intensive tasks that are too demanding for the software-driven platform. Wideband channelization is such task, whereas studying, designing, and implementing it in a digital front-end for SDR is the focus of this work.

Channelization Algorithms Study

3

Channelization is a process where single, few, or all channels from a certain frequency band are separated for further processing. The separation of single channel is usually done by down-conversion followed by filtering and optional sample-rate conversion. Figure 3.1 illustrates a digital front-end containing a single-channel DDC based on IQ Demodulation [16], followed by a sample-rate converter (also called decimator).

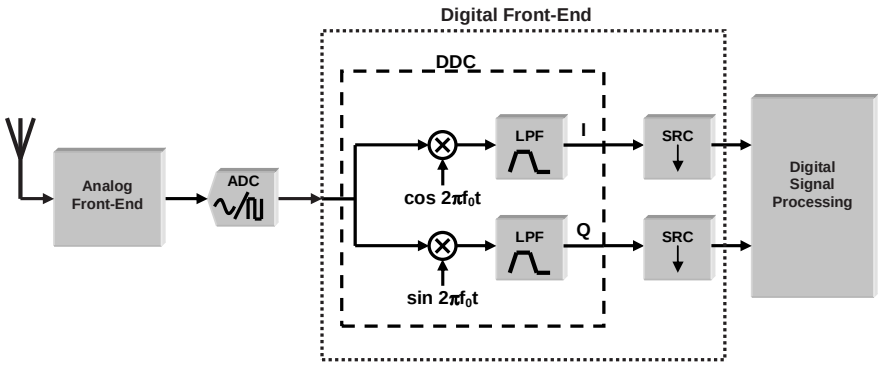


Figure 3.1: Single channel digital front-end channelizer

The channels of interest may be of equal or different bandwidths and may be uniformly or non-uniformly, continuously or non-continuously distributed over the input frequency band. In military applications of our interest, many channels from the input frequency band have to be separated - usually all available channels. It is also mostly common in such applications that the channels of interest are uniformly and continuously distributed over the input frequency band.

In the reminder of this chapter we introduce relevant SDR channelization algorithms. Thereafter, we compare the introduced algorithm in order to choose the one most relevant for the requirements of this project. Afterwards, conclude this chapter with an explained choice of algorithm.

3.1 Channelization Algorithms

This section presents 3 channelization algorithms. Namely, The per-channel approach, The pipelined frequency transform, and the polyphase FFT algorithms.

3.1.1 The per-channel Approach

A straightforward implementation, which is also the traditional implementation of wideband channelizer, is to simply use a single-channel channelizer for each channel of interest, and connect them all to the input frequency band signal [9]. Figure 3.2 illustrates such algorithm.

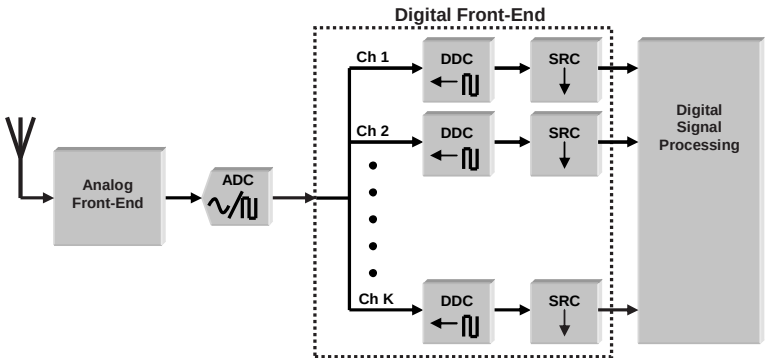


Figure 3.2: Per-channel channelizer

This approach provides a great deal of flexibility in the choice of channels to be separated. Each single-channel channelizer can be individually designed for BW and frequency choice. Furthermore, the separated channels are not constrained to be of the same bandwidth or to be uniformly distributed over the frequency input band. However, once such channelizer is designed, it is very rigid for alteration. Adapting this channelizer algorithm to different air-interface might require replacement of some or all single-channel channelizers. When a change has to be done only in part of the input frequency band, only the corresponding single-channel channelizers have to be altered or replaced. Another weakness of this algorithm is that for wideband receivers, where many channels are to be separated, silicon costs and power consumption are extremely higher than in other, more advanced wideband channelization techniques introduced in the following sections [10,17,18] .

3.1.2 Pipelined Frequency Transform

The Pipelined Frequency Transform (PFT) algorithm [7] is based on a binary tree of DDCs and SRCs (Figure 3.3) where units of DDC followed by SRC are used for dividing their input band into two half-bands with half sampling rate. This algorithm creates a binary tree that splits the input frequency in two half-bands and then splits each half-band again into two half sub-bands and so on, until the last tree level produces the required separated channels. The resulting structure is also called HMM (Hierarchical Multistage Method) [19] or QMF tree [20].

This algorithm for itself has no advantage over the algorithm presented in the previous section and is actually much more expensive in terms of silicon use, since apart of a

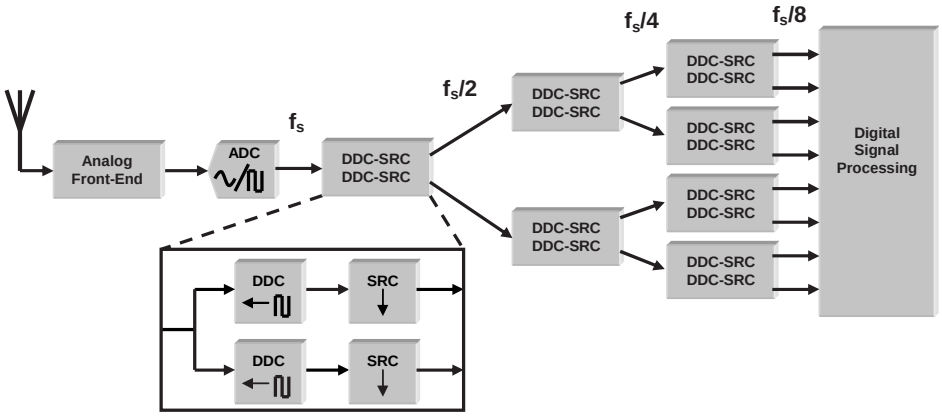


Figure 3.3: DDC-SRC tree

single-channel channelizer for each channel of interest (as in the per-channel algorithm) in the last stage of the tree, many more are needed in the other stages. Nevertheless, each single-channel channelizer complexity can be reduced dramatically, taking advantage of half band filters symmetry and restricting the output sample rate to be quarter of input sample rate in each single node in the tree.

Observing that the components in each stage perform in half of the sampling rate of its former stage components, a considerable optimization can be performed. The actual amount of operations-per-time performed in each level of the tree is equal while distributed over twice components than in its former tree level. Instead of using two components for each component in the former tree level at half sampling rate, one component that performs in the same sampling rate can be used in combination with interleaver, which distributes the samples accordingly. This is done using complex (IQ) DDC and DUC as illustrated in Figure 3.4 (The DDCs and DUCs that are not in the 1st level are of a special interleaved version). The channels however are output serially, and therefore some extra processing is required for distributing them in distinct outputs.

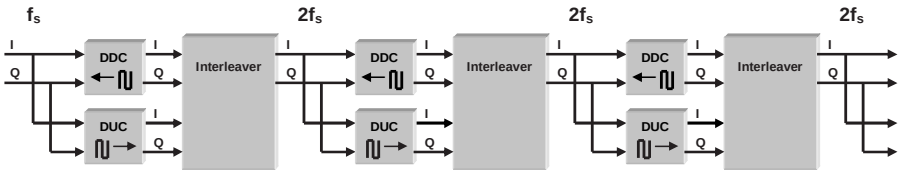


Figure 3.4: DDC-SRC tree

The PFT algorithm seems to be much more economical in terms of silicon use and power consumption when compared to per-channel channelizers algorithm. Especially when many channels are to be separated from the frequency input band [7]. However, it demonstrates less flexibility, as the separated channels must be of equal bandwidth

and uniformly distributed. The Tunable PFT algorithm is an adaptation of the PFT that alleviates this inflexibility by introducing interleavers that provide intermediate outputs from the PFT stages that may be used for fine tuning channelization [21]. This improvement, however, leads to increasing HW costs and is not applicable for wideband channelizers.

3.1.3 Polyphase FFT

This channelization algorithm is an improvement of FFT channelization using a polyphase filterbank in combination with FFT, taking advantage of the equivalence theorem and noble identities [22] while posing acceptable restriction over the sampling rate.

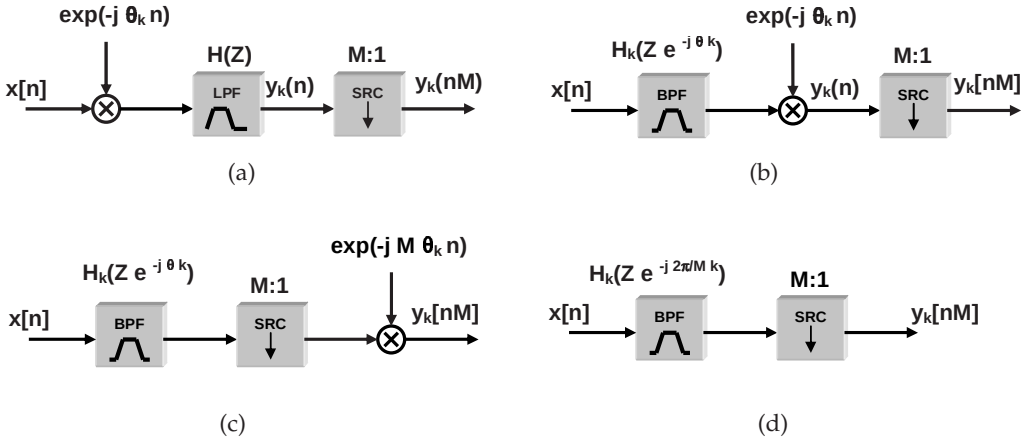


Figure 3.5: Modifications to the kth single channel channelizer

We consider the kth single (complex) channelizer from the per-channel channelizer in Figure 3.2 (shown in Figure 3.5(a)) and apply series of modifications to it [2]. The expression of the LPF output in Figure 3.5(a) is a multiplication of the input samples $x[n]$ with the complex heterodyne and a convolution with the filter coefficients $h(n)$, and is given in Equation 3.1.

$$\begin{aligned} y_k(n) &= [x(n)e^{-j\theta_k n}] * h(n) \\ &= \sum_{r=1}^{N-1} x(n-r)e^{-j\theta_k(n-r)}h(r) \end{aligned} \tag{3.1}$$

Swapping between the complex multiplier and the prototype LPF alters the LPF to a BPF in accordance with the equivalency theorem [23] (Figure 3.5(b)). The corresponding modification to Equation 3.1 is shown in Equation 3.2.

$$y_k(n) = \sum_{r=1}^{N-1} x(n-r)e^{-j\theta_k(n-r)}h(r)$$

$$= \sum_{r=1}^{N-1} x(n-r)e^{-jn\theta_k}h(r)e^{jr\theta_k}$$

$$= e^{-jn\theta_k}\sum_{r=1}^{N-1} x(n-r)h(r)e^{jr\theta_k} \tag{3.2}$$

Observing that only every Mth result of the complex multiplier in Figure 3.5(b) is kept after of the SRC, we interchange these two elements while adapting the phase of the complex multiplier (multiply with M) as shown in Figure 3.5(c). Constraining the center frequency for the kth channel to be an integer multiple of the output sample rate so that $\theta_k = \frac{2\pi k}{M}$ results in aliasing to baseband, since the complex multiplier term becomes $e^{-j2\pi n} = 1 + 0j$. Consequently, the complex multiplier becomes superfluous and can be removed, as shown in Figure 3.5(d).

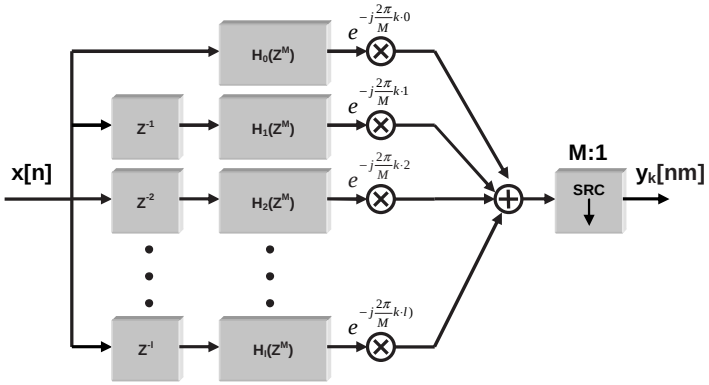


Figure 3.6: l branches in the filterbank decomposition of the kth single channelizer

Noting that as before, every Mth output of the BPF in Figure 3.5(d) is not used due to the SRC, it would be sensible to "shift" the SRC to the left of the BPF. In order to do so, we have to invoke the noble identity [22]. For this purpose we first have to decompose the BPF in the kth single channelizer into a filterbank of l (=M) sub-filters. The filterbank decomposition is described in Equation 3.3.

$$H(Ze^{-j(\frac{2\pi}{M})k}) = \sum_{r=0}^{M-1} Z^{-r}H_r(Z)e^{j(\frac{2\pi}{M})rk} \tag{3.3}$$

The resulted l sub-filters in the filter bank are composed of delay element, sub-filter, and (time invariant) scaling multiplier (Figure 3.6). Moving the SRC through the scaling multipliers and the l sub-filters we invoke the noble identity. The resulted filterbank is depicted in Figure 3.7. The corresponding output function is shown in Equation 3.4.

$$y_k(nM) = \sum_{r=0}^{M-1} y^{(r)}(nM)e^{j(\frac{2\pi}{M})rk} \tag{3.4}$$

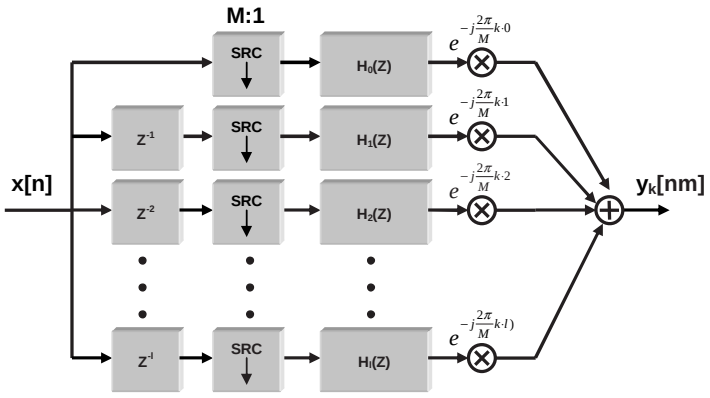


Figure 3.7: Applying the noble identity in the kth filterbank

where $y^{(r)}(nM)$ is the nM^{th} sample from the r^{th} sub-filter.

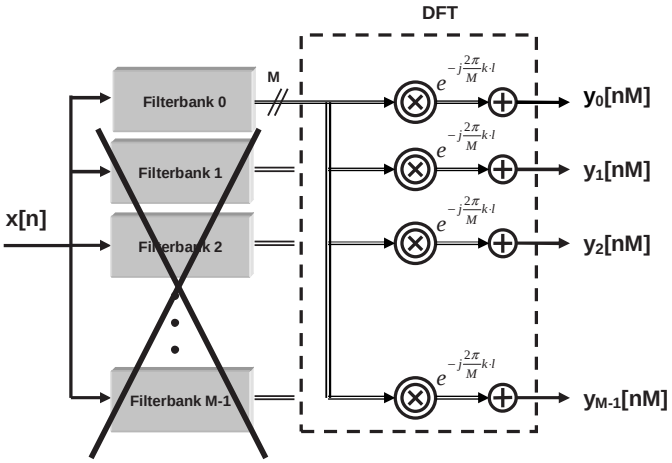


Figure 3.8: Discarding M-1 filterbanks

The delay elements, the SRCs and the sub-filters are similar for all the k filterbanks and therefore only one should be physically implemented as illustrated in Figure 3.8. Observing in Equation 3.4 that the multipliers and adders (Dashed-line rectangle in Figure 3.8) practically function as M-points DFT, they can be replaced with FFT for reducing complexity. The final result illustrated in Figure 3.9 (note that the delay elements are replaced by a chain of one unit delay elements) is known as the polyphase FFT (PFFT) filterbank channelizer algorithm.

In comparison with the per-channel channelization algorithm formerly introduced, the PFFT algorithm is much more rigid to changes, and is subject to restrictions imposed over the sampling rate, the number of channels to be extracted, and the number of taps in the prototype filter. however, it seems to show extremely lower silicon costs.

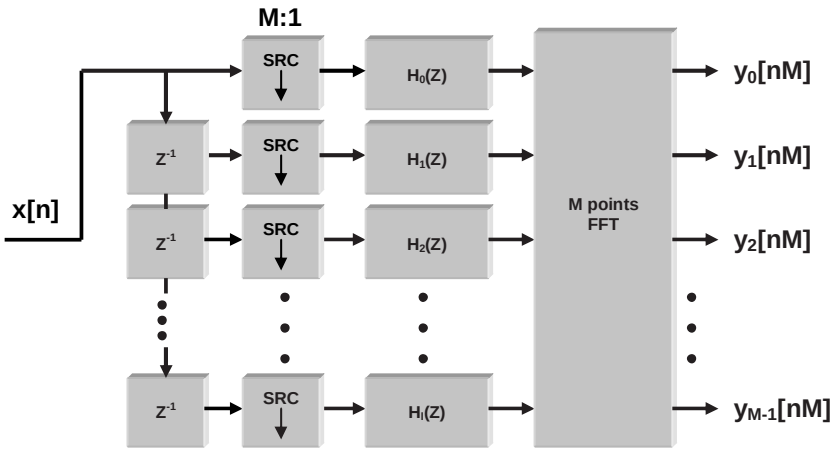


Figure 3.9: Polyphase FFT channelizer

Measured in terms of number of arithmetic operations per number of separated channels (computational complexity) [9, 24, 25], it seems that the PFFT outperforms the per-channel algorithm when separating more than 3 channels.

3.2 Algorithms Comparison

In the previous section, several channelization algorithm for wideband channelizer were introduced. In this section, we present HW-complexity (cost) comparison and a qualitative comparison of these channelization algorithms with the aim to chose the one most suitable for Eonic's target applications.

3.2.1 Hardware Complexity Comparison

The following HW complexity comparison is based on data from [17], which is put here in plots. The first comparison is for LUT (Xilinx FPGAs basic block) utilization. The right plot in Figure 3.10 show us that the per-channel algorithm (stacked) utilizes far more LUTs than the PFT (binary) and PFFT algorithms and that its tendency is much steeper. The left plot in Figure 3.10 gives us a clearer comparison between the other two algorithms. We can see that for all given number of channels PFT more than twice LUT resources than the PFFT algorithm does.

The second comparison is of memory bits utilization. The right plot in Figure 3.11 shows us that the per-channel algorithm employs much more memory resources than the PFT and PFFT algorithms for all number of channels. However, comparing the PFT and PFFT algorithms (left plot in Figure 3.11) we can see that the PFFT algorithm is superior only when channelizing more than 300 channels. Another important property is that the PFT curve's inclination is much steeper than the PFT curve.

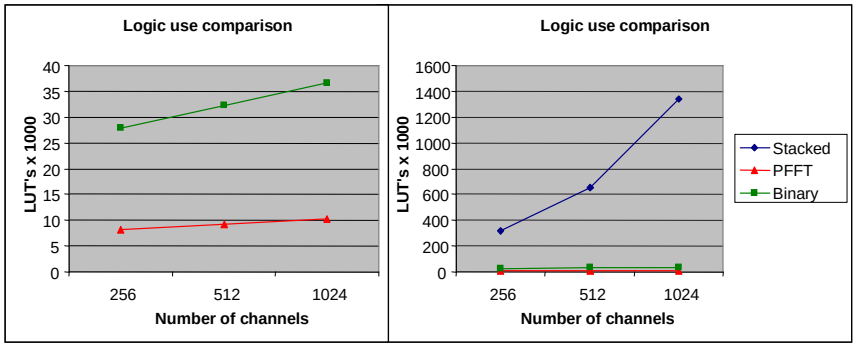


Figure 3.10: Comparison of LUT utilization

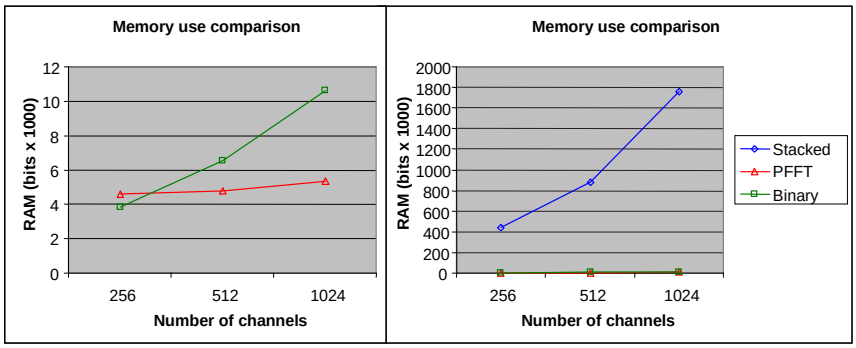


Figure 3.11: Comparison of memory bits employment

3.2.2 Qualitative Comparison

Comparison of different algorithms is not simple. The parameters ranges for comparison are wide. Limitation to practical parameters may alleviate this difficulty. Most of newly proposed channelization techniques in literature are compared to the traditional per-channel channelizer. The comparisons here are divided to three groups: Computational complexity, size ("silicon costs"), and group delay and flexibility.

Computational complexity

A common comparison parameter is computational complexity, which is usually derived from simulations and software implementations. Such a comparison projects on silicon costs but usually do not take into account memory requirements and control complexity. Previous works of [6, 24, 26] show that when 3 or more channels are to be channelized, the PFFT algorithm outperforms the per-channel algorithm. The work of [26] shows that an improvement in the filters of the per-channel algorithm raises this limit to lay between 4 and 20 channels for some scenarios.

Silicon cost

Comparison that is based on actual implementation in FPGA gives a good idea about the HW complexity of the different algorithms. A drawback of such comparison is that it is not platform independent. Different FPGAs contain some dedicated multipliers and built-in memory blocks. Each configuration of distinct algorithm may have trade-offs in FPGA resources that is difficult to measure and compare. Such is the comparison is made between the PFT, PFFT, and per-channel algorithm implementations presented in Subsection 3.2.1. Based on this comparison, it seems that in terms of memory use, the PFT memory requirement is growing rapidly with the number of channels to be separated. The conclusion drawn in [17] is that up to 256 channels, the HW complexity of PFT and PFFT is comparable and that above 256 channels PFFT outperforms the PFT.

Group delay

Generally, group delay is not a major consideration in the choice of channelization algorithm. It is usually of concern when designing ELINT receivers that deal with analysis of short radar pulses. The work of [25] shows that the group delays of PFT and PFFT algorithms in different configurations are quite similar. Normally, PFFT group delay is better than in PFT algorithm, but more rigid implementation of the PFT (giving up intermediate outputs) may reach a comparable or better group delay than in the PFFT algorithm. Comparing the PFFT and per-channel algorithm, it seems that the later has a small, advantage due to the FFT stage in the PFFT algorithm. This advantage order, however, is insignificant for the target implementations aimed to in this study.

Flexibility

As this study is aimed toward the mapping of selected algorithm on reconfigurable digitizers, analysis of two flexibility aspects in the different algorithms is essential. The following discussion offers analysis of initial design flexibility and reconfigurability.

Initial design In this aspect, the per-channel approach is clearly the winner. All the separated channels are independent, may have different bandwidths and may be non-uniformly and non-continually distributed over the input frequency band. The PFT and PFFT algorithms suffer from similar limitations. Namely, producing channels with equal bandwidth that are uniformly and continually distributed over the input frequency band. The PFT suffers from another restriction however. The number of the separated channels has to be an integer power of 2. The PFFT in principle is more flexible in the choice for number of channels to be separated. Nevertheless, the most economical implementations of FFT have integer power of 2 bins, and that may also impose restriction on the implementation of PFFT filterbank. An advantage of the PFT over the PFFT is its possibility to produce intermediate outputs of channels with half of the resolution and twice the bandwidth of the channels in the next level of the PFT tree. The PFFT has a constraint on the number of taps in the prototype filter, which must be an integer multiply of the number of channels.

Reconfiguration This is a key concern in the evaluation of the different channelization algorithms. Addition or removal of single or few channels is very easy to implement on the per-channel algorithm, while in most cases, for the PFT and the PFFT algorithms it means a complete reconfiguration of the whole implementation (especially when a change in an integer power of 2 number of channels is required). Adaptation of the filtering performance (channels separation quality) requires modification to the number of taps and the weight for each tap in the filter. In the PFFT algorithm the filtering is implemented in a logically separated block, and therefore its adaptation need not have consequences for the rest of the algorithm implementation. In the PFT and the per-channel algorithms, however, the filters are distributed within different logical blocks, so that adaptation in their performance may have consequences to the rest of the implementation.

Table 1 summarizes the qualitative comparison between the different channelization algorithms. Careful examination of the different comparison aspects with respect to the focus of this study (presented in the 1st chapter) shows that the per-channel approach wins in many aspects. Conversely, its implementation for high number of channels is infeasible, and that makes the PFFT algorithm the most suitable for SDR wideband channelizer front-end implementation of our interest. Nevertheless, the differences between the PFFT and FFT implementations for medium number of channels (few tens to few hundreds) is not well documented, and investigation in this direction might be subject for further research.

Aspect		Algorithm		
		Per-Channel	PFT	PFFT
Computational Complexity for high number of channels		Poor	Good	**Excellent**
Silicon Cost Efficiency		up to 3-20 channels	up to 128-256 channels	**8-16 channels and above**
Group Delay		Good	Good	Good
Initial Design Flexibility	Independent channels	**Yes**	No	No
	Number of channels	**Selectable**	2^{INT}	Preferably 2^{INT}
	Intermediate outputs	No	**Yes**	No
Flexibility for Reconfiguration	Addition / removal of channels	**Excellent**	Poor	Poor
	Filtering independence	Poor	Poor	**Good**

Table 3.1: Qualitative Comparison

3.3 Conclusion

In this chapter we introduced three different channelization algorithms. Namely, the per-channel, the PFT, and the PFFT algorithms, explaining in details . Consequently, we presented HW comparison between these algorithms for LUT and memory resources utilization. Based on this comparison, the PFFT algorithm appears to be superiorly cost efficient when channelizing few hundreds or more communication channels.

Afterwards, we presented a qualitative comparison between these three algorithms that comprises also group delay, initial design flexibility, and reconfigurability. Based on the performed comparisons, we came to the conclusion that despite the fact that the per-channel algorithm has better score in many comparison aspects, its implementation is critically HW inefficient and is infeasible for high number of channels, even on todays largest available FPGAs. Therefore, we chose the PFFT algorithm, which is highly cost efficient and has the best computational for high number of channels, for our implementation of the front-end wideband channelizer.

4 Channelizer Architecture

The polyphase FFT channelization algorithm was chosen for our study and was explained in details in Chapter 3. In this chapter we present the implementation architecture of this algorithm. We first introduce decomposition to modules of the channelizer architecture. Thereafter, we explain in details some of these modules, focusing on implementation choices.

4.1 Modules Decomposition

The architecture of the digital front-end channelizer comprises the digital signal processing done from the ADC output to the moment where each channel is available as separate output for further processing by software. The ADC output is in real digital signal format, in contrast to complex format, also known as IQ (In-phase Quadrature) signal, which provides easier signal processing. Therefore, preprocessing of the ADC output is necessary in order to convert its real output into a complex signal. This is done in the IQ-demodulator unit, which is placed between the ADC and the filterbank module.

The following components in the channelizer architecture are the filterbank module and the FFT processor, as described in the former report. An optional post-processing unit may be appended to the FFT processor's output. The necessity of this module depends on implementation choices that are described in Section 4.5. The separate channels outputs are then available for further digital signal processing. The layout of the channelizer architecture is depicted in Figure 4.1.

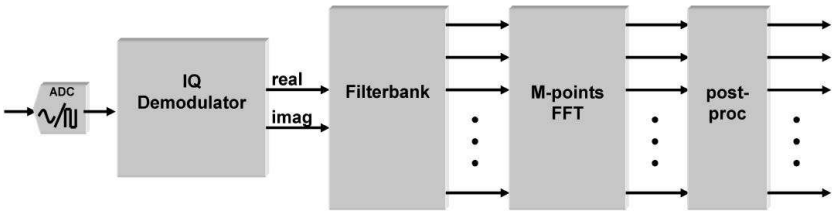


Figure 4.1: Channelizer architecture

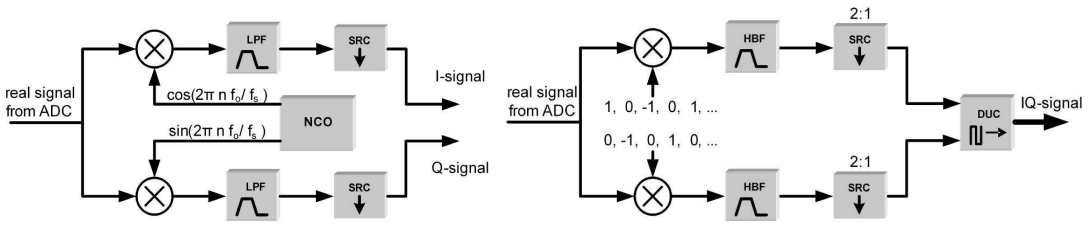
In the following subsections, each of the architecture modules is described, focusing on implementation issues.

4.2 IQ Demodulator

The task of the IQ demodulator is to convert real signal to complex one. There are, however, different possible implementation algorithms. These are described in the following subsections.

4.2.1 Conventional IQ demodulator

The conventional IQ demodulator is implemented by down-conversion of the input signal, which is done by multiplication of the signal with sine and cosine components in two different I and Q paths. Subsequently, a low-pass filter is applied on each path. The signal can afterward be decimated by a convenient rate, as long as the Nyquist criterion is kept. This processing sequence is depicted in Figure 4.2(a).



(a) Conventional (b) Wideband

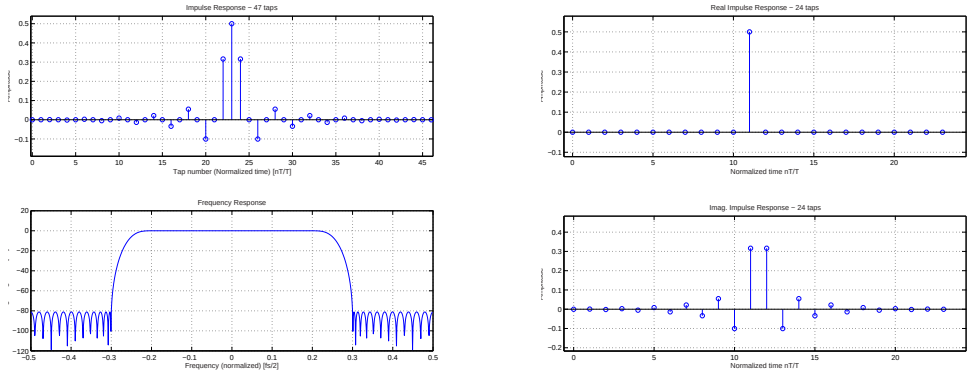
Figure 4.2: IQ-demodulators

4.2.2 Wideband IQ Demodulator

The conventional IQ demodulator architecture is general in its nature and is usually applied when a single channel is demodulated. In the case of wideband IQ demodulation, a better implementation is possible. Since in a wideband channelizer the whole input spectrum from the ADC has to be IQ-demodulated, the sample rate for the down-conversion is quarter of the IF signal sampling rate f_s as produced in the ADC. In this case, the numerically controlled oscillator (NCO) can be replaced with a subsequent multiplication of the I-path samples by +1, 0, -1, 0, and of the Q-path samples by 0, -1, 0, +1 [27]. The low-pass filter (LPF) is consequently strictly half-band low-pass filter (HBF), and since the output spectrum is half of the input spectrum, the optimal decimation rate in the sample rate converter (SRC) is 2:1. Since the input to the filterbank module has to be between DC and half of the current Nyquist sampling rate, The complex signal has to be up-converted back by a quarter of the current (decimated) sampling rate using a complex digital up-converter (DUC) that includes 4 unit multipliers (selective complementers), adder, and subtracter. The structure of wideband IQ demodulator is brought in Figure 4.2(b).

Another possible improvement is due to the unique HBF filter taps, where, except of the middle tap that equals 0.5, each 2nd tap equals zero [8] as shown in Figure

4.3(a). This allows us to invoke the noble identity and "shift" the 2:1 SRC through the HBF in the I and the Q paths and also through the unit multipliers. This results in a complex half-band filter (CHBF) where the real coefficients are the HBF's odd taps and the imaginary coefficients are the HBF's even taps. Examining an impulse response of example CHBF (Figure 4.3(b)) we can see that one path is all zero coefficients, except of the HBF's center tap, which has the value 0.5 that can be implemented as wired shift.



(a) HBF (b) CHBF

Figure 4.3: HBF/CHBF Impulse and frequency response

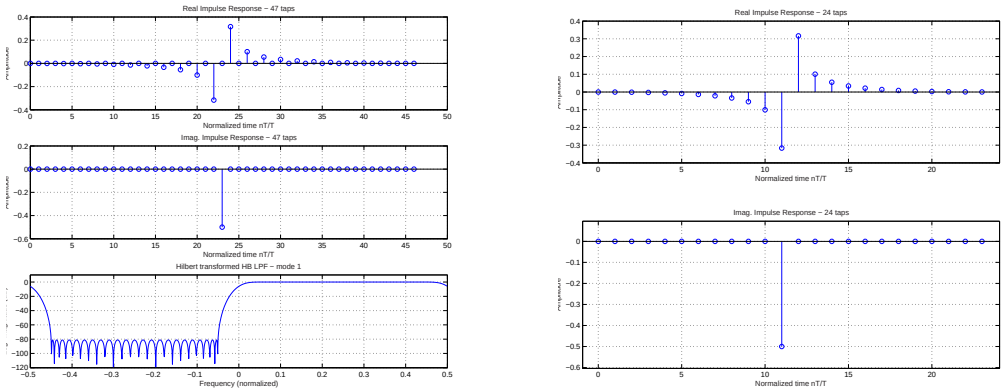
4.2.3 Hilbert Transformed IQ Demodulator

In principle, the frequency down-conversion before the HBF filters is necessary in order to shift the positive frequency spectrum to the filter's pass band region (and the negative spectrum to the stop-band region), and the complex DUC is needed to move the filtered signal back into within DC and half the sampling rate. Applying Hilbert transform on the HBF results in a special half-band filter that passes the positive frequency spectrum and attenuates the negative one, having exactly the same characteristics of the HBF [28] as shown in Figure 4.4(a).

Using the Hilbert transformed HBF we can (and should) remove the unit multipliers and the complex DUC. As result, The Hilbert transformed IQ-demodulator architecture is simpler, while producing the same results as the standard wideband IQ demodulator (see Figure 4.5(a). Since this filter also has in every 2^{nd} tap zero weight (except of the middle tap), the noble identity can be applied here as well, as described in Subsection 4.2.2. The resulted Hilbert transformed HBF impulse response plot is given in Figure 4.4(b) and its architecture is depicted in Figure 4.5(b).

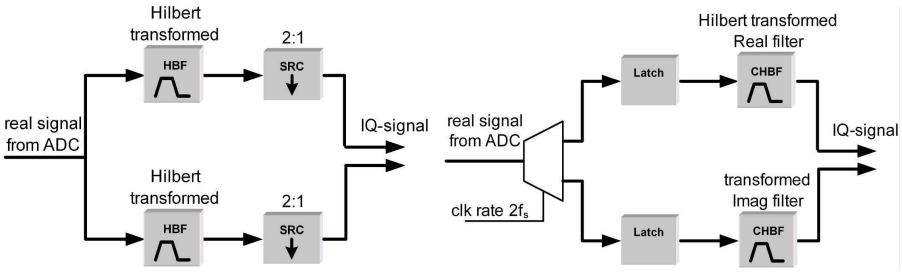
4.2.4 Chosen Implementation Algorithm

As conclusion, The Hilbert transformed IQ-demodulator algorithm is our choice for implementation, since it achieves the same results as the conventional IQ demodulator



(a) Before applying the noble identity (b) After applying the noble identity

Figure 4.4: Hilbert transformed HBF Impulse and frequency response



(a) Before applying the noble identity (b) After applying the noble identity

Figure 4.5: Hilbert transformed IQ demodulators

and the wideband IQ demodulator, while requiring less HW resources for its implementation.

4.3 Filterbank

Decomposing the prototype filter to filterbank is a firm procedure with no much design choices to be done (see Section 3.1.3). The design of the prototype filter itself, however, has more liberty. The following subsection describes some design choices in the process of constructing a prototype filter.

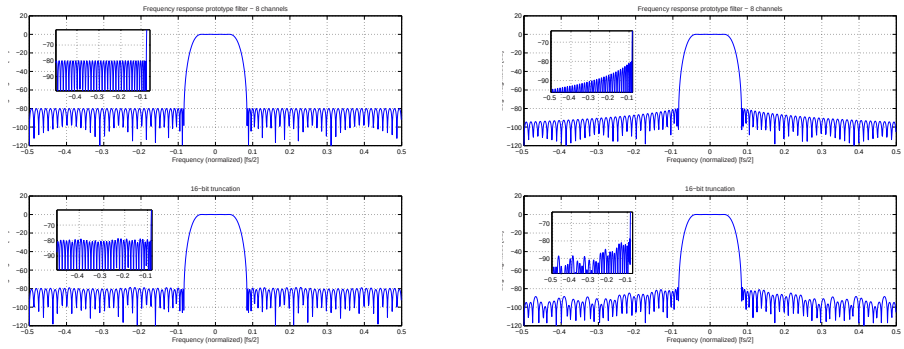
4.3.1 Equiripple Prototype Filter

Designing the prototype filter is done using Matlab™ with the signal processing toolbox. The chosen filter design method is the iterative Parks-McClellan algorithm, which produces optimal FIR filter by minimization of the maximal error between the desired and the resulted frequency response [29]. The resulted filter is called equiripple since it

has equal ripples in the pass and in the stop bands. Figure 4.6(a) depicts an equiripple filter frequency response.

4.3.2 1/f Ripple Prototype Filter

If the first $m-1$ derivatives of an impulse response function $S(f)$ are continuous and the m$^{\text{th}}$ derivative of $S(f)$ has one or more finite amplitude discontinuities, then the stopband ripple decays as $\frac{1}{f^{k+1}}$ per octave [23, 30]. The stopband ripple decay rate in equiripple filter is $\frac{1}{f^0} = 1$. That implies discontinuity in the zero$^{\text{th}}$ derivative i.e., $S(f)$ itself. An enhancement of the Parks-McClellan algorithm, suggested by [23], removes the discontinuity from $S(f)$ and results in $1/f$ stopband ripple decay rate per octave, while causing a negligible consequences for the filter performance. This improves the prototype filter design in two senses. It reduces spectral aliasing between adjacent channels and it is more resistant to performance degradation due to quantization of the filter coefficients. Figure 4.6 presents a comparison of equiripple and 1/f ripple designs and the truncation result on each of the designs.



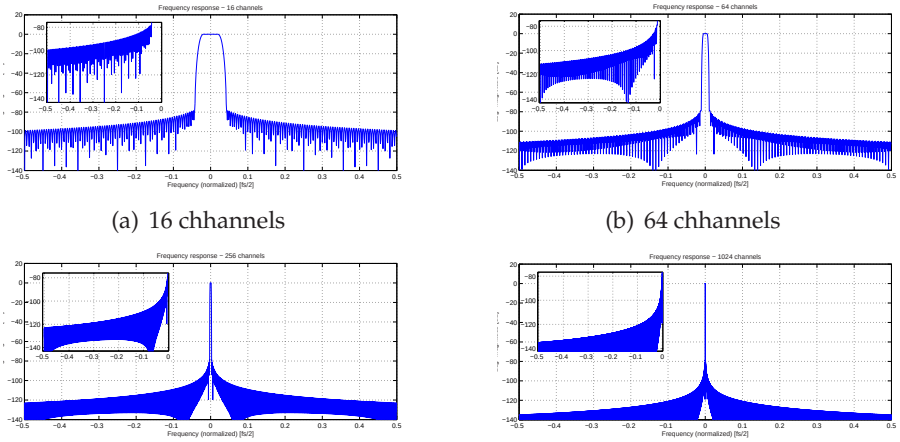
(a) Equiripple (b) 1/f ripple

Figure 4.6: Equiripple vs. 1/f ripple prototype filters

The stop-band of the prototype filter becomes narrower as the number of channels N_χ becomes greater, the influence the of 1/f ripple decay also becomes better. Figure 4.7 illustrates four filter-response plots of various prototype filters with different number of channels. Therefore, and due to the improvements of 1/f ripple filter above equiripple, the former is our choice for design method of the prototype filter.

4.4 FFT

This channelizer's module has a customary FFT functionality. FFT implementation is a wide field for itself. Since this module is to be implemented using Eonic's PowerFFT™ processor [31], this module is left out of the focus for this study.



(a) 16 chhannels

(b) 64 chhannels

(c) 256 chhannels

(d) 1024 chhannels

Figure 4.7: 1/f ripple prototype filters

4.5 Post-Processing

When a WOLA (weight, overlap add) [8] implementation of the filterbank is applied, phase correction of the output signals is needed. This survey, however, is focused on the critically sampled polyphase filterbank, where phase correction is not necessary. Therefore, it is left out of the focus of this study.

4.6 Conclusion

In this chapter we presented and explained the decomposition of the polyphase FFT channelizer algorithm into IQ-demodulator, filterbank, FFT, and post-processing modules. We gave a explanation about three different implementation algorithms of the IQ-demodulator, namely, conventional, wideband, and Hilbert transformed IQ-demodulators. We concluded that the Hilbert transformed IQ-demodulator algorithm is best choice for implementation since it provides the same results as the other two algorithms while requiring less HW resources.

Afterwards, we introduced two filter design techniques for the prototype filter, from which the filterbank is derived, i.e., the Parks-McClellan equiripple filter design and the Parks-McClellan 1/f ripple design. Comparing these two techniques we decided that the second one is most advantageous for the implementation of the filterbank. Since the FFT is to be implemented using Eonic's PowerFFT™ processor, the modules chosen for the parameter ranges survey (in Chapter 5) are the IQ-demodulator and the filterbank.

5 Parameter Ranges Survey

In the previous chapter, The implementation architecture of the polyphase FFT channelizer was introduced. In order to explore various design spaces and trade-offs in this architecture, a Matlab™ model of the digital front-end was constructed. In this chapter, we present a parameter ranges survey of the channelizer. However, in order to convey an effective study, focus has to be maintained onto practical parameter ranges. Therefore we first explain the choices of parameters and ranges to be explored. Afterward, we bring some important results of this survey, and we end with conclusions based on the attained results.

5.1 Simulation setup

The most intensive computation tasks in the front-end are done by the IQ-demodulator's filter, by the filterbank and by the FFT. Since the FFT is to be implemented using Eonic's PowerFFT™ processor, this parameter ranges study is focused on the design choices of these two special filters.

5.1.1 IQ demodulator Filter Design Parameters

The independent input parameters for low-pass FIR filter design using the Parks-McClellan algorithm are: stop-band attenuation (r_{SB}), pass-band ripple (r_{PB}) ,stop-band width (w_{SB}), and pass-band width (w_{PB}). The word-length used for the taps coefficients also has influence on filter design and may restrict its stop-band attenuation.

One restriction on the input parameters occurs when the filter is designed to be strictly half-band filter. In this case the stop-band and pass-band widths are always equal: $w_{SB} = w_{PB}$, so we only have to control one these two parameters. In this model we quantify w_{PB} in percents, where 100% means exactly half of the filter's input band, which lies between DC and $f_s/4$.

Another restriction is possible due to the unique HBF taps. This behavior, where each second tap is zeroed, is achieved only when the filter is designed to have equal passband deviation δ_{PB} and stopband deviation δ_{SB} [23].

$$\delta = \delta_{PB} = \delta_{SB} \tag{5.1}$$

Traditionally, the deviation parameters are described as ripple in dB units. The stop-band ripple is defined as the difference between the nominal passband gain G_{PB} and

the maximal stopband deviation in dB (Equation 5.2), whereas the passband ripple is defined as the difference between the maximal and the minimal deviations from the passband gain in dB (Equation 5.3) [32].

$$r_{SB} = 20\log_{10}(G_{PB}) - 20\log(\delta_{SB}) \quad (5.2)$$
$$r_{PB} = 20\log_{10}(G_{PB} + \delta_{PB}) - 20\log(G_{PB} - \delta_{PB}) \quad (5.3)$$

Applying Equation 5.1 and $G_{SB} = 1$ (since the HBF has unit gain) on Equations 5.2 and 5.3 results in Equations 5.4 and 5.5 respectively.

$$r_{SB} = 20\log_{10}(1) - 20\log(\delta) \quad (5.4)$$
$$r_{PB} = 20\log_{10}(1 + \delta) - 20\log(1 - \delta) \quad (5.5)$$

Isolating δ from Equation 5.4 yields:

$$\delta = -10^{\left(\frac{r_{SB}}{20}\right)}$$

which, when applied to Equation 5.5, results in the relation represented by Equation 5.6.

$$r_{PB} = 20\log_{10}\left(\frac{10^{\left(\frac{r_{SB}}{20}\right)} + 1}{10^{\left(\frac{r_{SB}}{20}\right)} - 1}\right) \quad (5.6)$$

Figure 5.1 depicts the relation of Equation (5.6) for some practical values of r_{SB}. We can observe that even for low values of the stop-band attenuation the pass-band ripple is below 0.02 dB. This value is appropriate for many comm applications [23] and is, by far, suitable for a speech receiver that may utilize overall pass-band ripple of up to $3dB$ [33], and which is the application of interest for this study. Therefore, we may restrict the study of the IQ-demodulator filter design to its stop-band attenuation and to its pass-band width without being bothered by the implication on the passband ripple in the case of the IQ demodulator filter.

As result of this discussion we set the controlled parameters in the simulation model of the IQ-demodulator's filter to be the passband width w_{PB}, the passband ripple r_{PB}, and the stopband attenuation r_{SB}.

5.1.2 Prototype Filter Design Parameters

The Input parameters for the Parks-McClellan algorithm when designing low-pass filter are the (normalized) frequency at the end of the passband f_{PB}, the (normalized) frequency at the beginning of the stopband f_{SB}, the passband ripple r_{PB}, and the stopband attenuation r_{SB}. Minimal Values for r_{PB} and r_{SB} are dictated by a corresponding standard. For example, one of the relevant targets for this study is defined by ETSI (European Telecommunications Standards Institute) 300 086 standard [33], which deals with speech FM receivers specifications for 25 kHz channel spacing. In this case, the maximal value

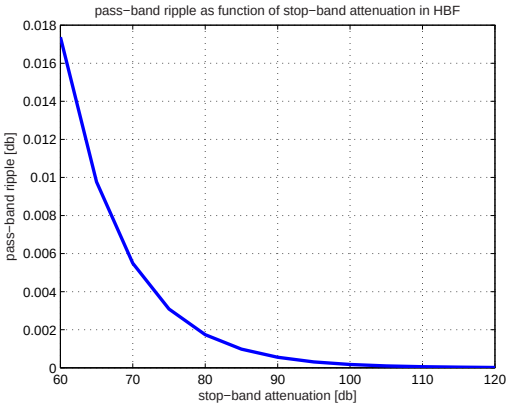


Figure 5.1: r_{PB} as function of r_{SB} in HBF

for r_{PB} is defined to be 3 dB and the minimal value for r_{SB} is defined to be 70 dB. These values, however, are defined for the whole receiver and therefore provide only upper and lower bounds for the corresponding filter design parameters.

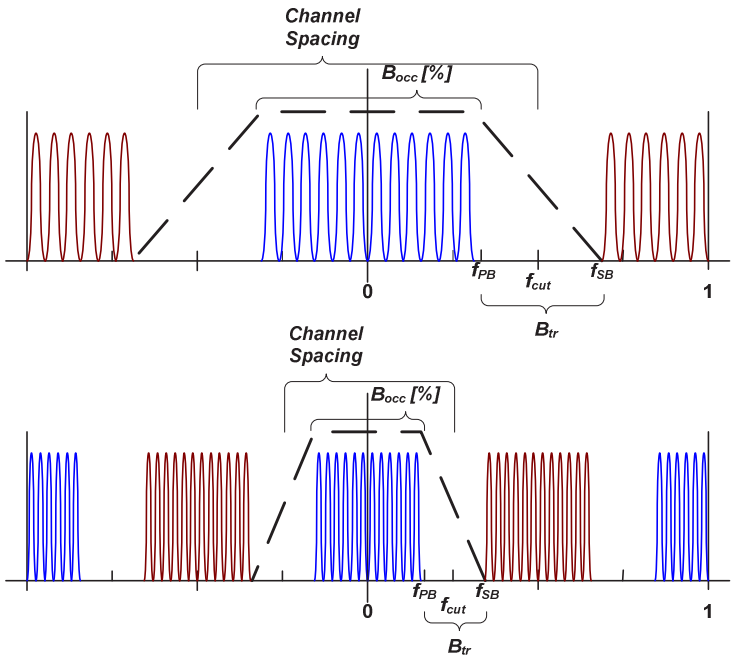


Figure 5.2: Prototype filter design parameters choice for 2 and 4 channels

The values for f_{PB} and f_{SB} depend on the number of channels N_χ and the occupied bandwidth percentage of one channel in the channel spacing (B_{occ}). Occupied band width is the portion of a channel that contains 99% of the average power of the signal.

Figure 5.2 illustrates this dependency for 2 (upper part) and 4 (lower part) channels[1]. The Cut Frequency (f_{cut}) marks the middle of the transition band where it holds:

$$f_{cut} = \frac{1}{N_\chi} \tag{5.7}$$

The length of the transition band B_{tr} is twice the unoccupied bandwidth of one channel so it holds:

$$B_{tr} = \frac{2(1 - B_{occ})}{N_\chi} \tag{5.8}$$

Since $f_{PB} = f_{cut} - \frac{1}{2}B_{tr}$ and $f_{SB} = f_{cut} + \frac{1}{2}B_{tr}$, applying Equations 5.7 and 5.8 results in:

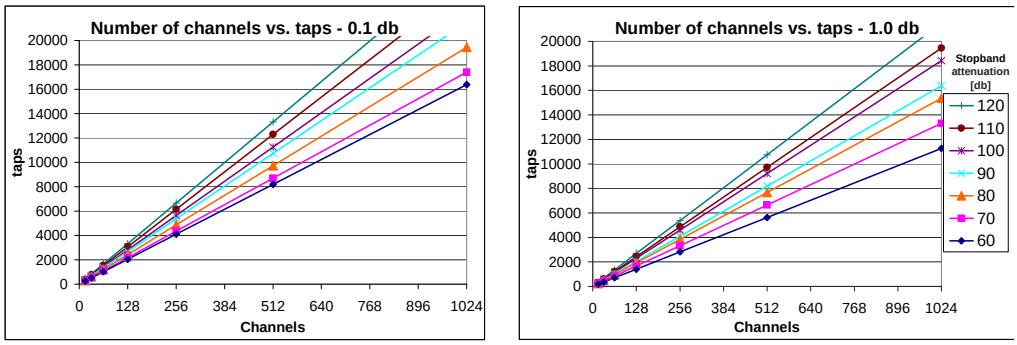
$$\begin{aligned} f_{PB} &= f_{cut} - \frac{B_{tr}}{2} \\ &= \frac{1}{N_\chi} - \frac{1 - B_{occ}}{N_\chi} \\ &= \frac{B_{occ}}{N\chi} \end{aligned} \tag{5.9}$$

$$\begin{aligned} f_{SB} &= f_{cut} + \frac{B_{tr}}{2} \\ &= \frac{1}{N_\chi} + \frac{1 - B_{occ}}{N_\chi} \\ &= \frac{2 - B_{occ}}{N\chi} \end{aligned} \tag{5.10}$$

The target applications of interest for this study has occupied bandwidth percentage of $B_{occ} \approx 0.64$ [33, 34]. This parameter, however, is set for overall system performance. Therefore, and since military applications often require stringent parameters, B_{occ} is set to 0.80, in accordance with available military system specifications. Consequently, we conclude that the independent parameters for the prototype filter design simulation are the passband ripple r_{PB}, stopband ripple r_{SB}, the stopband attenuation r_{SB}, and the number of channels N_χ.

Figure 5.3 depicts two example plots, which demonstrate that the number of necessary taps is a linear function of the number of channels when r_{PB} and r_{SB} have fixed values. This implies that the number of taps per channel is constant. Therefore, it is easier to measure HW complexity of the prototype filter in taps per channel N_{taps}/N_χ. It is important to mention that due to the symmetric nature of the designed prototype filter taps, the number of filter coefficients to be computed is half of the actual necessary filter taps.

[1]Note that one channel is halved between the lowest and highest frequencies. This special position might limit the possibility to channelize it. However, since the HBF of the IQ-demodulator cannot be ideal, this channel is not part of the usable output channels anyway.



(a) 0.1 dB (b) 1.0 dB

Figure 5.3: Number of channels vs. taps

5.2 Simulation Results

In this section we present important parameters behavior as simulated in Matlab. Since the design space is infinitely large, the actual ranges considered in the simulation are constrained to slightly below and above practical ranges. This is done from two reasons. First, in order to provide better insight of the system behavior, and second, in order to provide outlook for future implementations when technology enables it.

5.2.1 IQ Demodulator Filter Design Simulation Results

The parameters tested in this part of the simulation are described in Subsection 5.1.1. We bring first simulation with floating point coefficients and afterwards the influence of taps coefficients quantization on the performance of the IQ-demodulator filter.

5.2.1.1 Floating-point coefficients

Figure 5.4 depicts the number of necessary taps N_{taps} as function of the stopband attenuation r_{SB} for different values of passband width w_{PB}. It is clear from this chart that this function is linear. That implies that the necessary number of taps grows linearly alongside the stopband attenuation. This chart can be used when estimating the cost for improving the stopband attenuation for existing IQ demodulator filter design.

Rearranging the data and plotting it as the number of necessary taps N_{taps} as function of the passband width w_{PB} for different values of stopband attenuation yields the plot in Figure 5.5. This plot becomes useful when designing for restricted HW resources. It is easy then to explore the trade-off between passband width (the percentage of useful channels from the total output channels) and between the stopband attenuation for certain number of taps (along horizontal lines). Alternatively, the trade-off between the number of necessary taps and between the different stopband attenuation values for different passband width values can be explored along the horizontal lines.

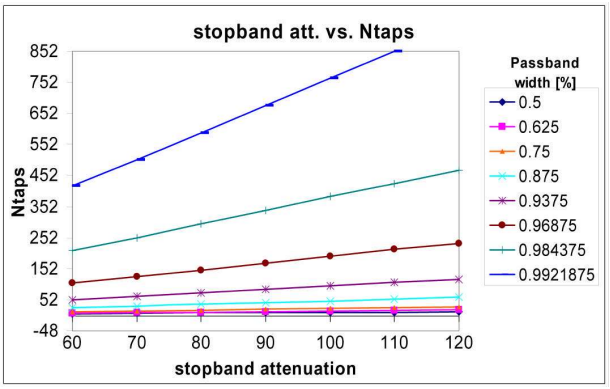


Figure 5.4: Stopband Attenuation vs. number of taps

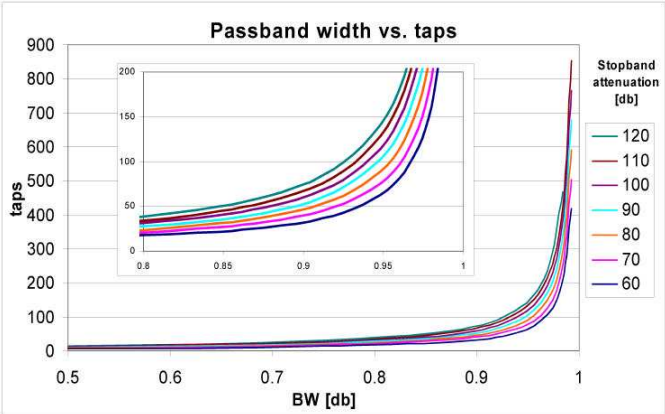
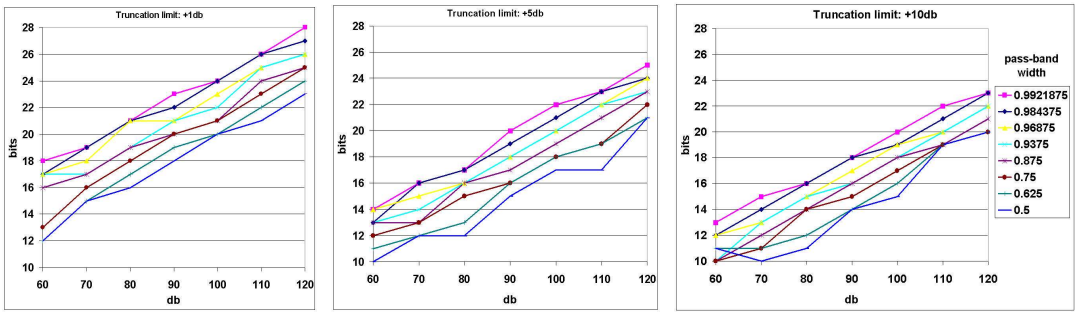


Figure 5.5: Passband width vs. number of taps

5.2.1.2 Fixed-point coefficients

The results of the floating-point simulation are theoretical since the word-length of the taps coefficients is not taken into consideration. In the following discussion we bring simulation results of a model with quantized (fixed-point) taps coefficients.

Figure 5.6 contains 3 plots of the minimal coefficient word-length as function of stop-band attenuation for various passband width values. Figure 5.6(a) illustrates the word-length necessary in order to achieve near floating-point attenuation performance. Figures 5.6(b) and 5.6(c) demonstrate the word-length necessary in order to achieve attenuation that is worse in $5dB$ and $10dB$ (respectively) than in the ideal case. In average, quantizing the coefficients word-length in 3-bits will cost $5dB$ in attenuation performance, while quantization of 4-bits will cost about $10dB$ in the IQ-demodulator's filter attenuation.



(a) +1dB (b) +5dB (c) +10dB

Figure 5.6: Quantization limits

The plots in Figure 5.6 are useful for the choices of word-length dependent HW such as ADC (resolution) and FPGA (bus-width) for the implementation of the channelizer's digital front-end.

5.2.2 Prototype Filter Design Simulation Results

The parameters tested in this part of the simulation are described in Subsection 5.1.2. We first bring results of simulation with floating point coefficients. Subsequently we present the influence of taps quantization on the performance of the prototype filter.

5.2.2.1 Floating-point coefficients

Figure 5.7 depicts the number of necessary taps per channel N_{taps}/N_{χ} as function of stopband attenuation for various passband ripple values. The staircase-like plots are result of rounding up the necessary number of taps to the closest multiplication of the number of channels. However, when testing for non-rounded values, it appears that this function is linear. That implies that design for better attenuation trades-off constant N_{taps}/N_{χ} per dB. The behavior of this trade-off is shown in Figure 5.8, which depicts the average number of N_{taps}/N_{χ} necessary for each 10 dB attenuation improvement as function of passband ripple. This plot displays exponential increase in the necessary taps per channel for each 10 dB attenuation improvement as the passband ripple values approach zero.

The plots in Figures 5.7 and 5.8 are useful for approximation of HW complexity and trade-offs when choosing the r_{SP} and r_{PB} parameters for the implementation of the channelizer filterbank.

5.2.2.2 Fixed-point coefficients

The results in the former subsection does not take into account the influence of quantization due to limited word-size used for the taps coefficients. Each plot in Figure

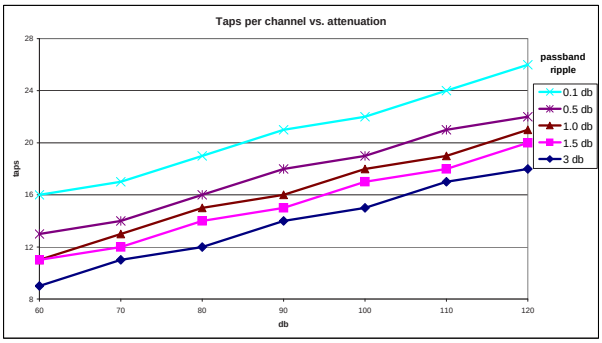


Figure 5.7: Taps per channel vs. stopband attenuation; various passband ripple values

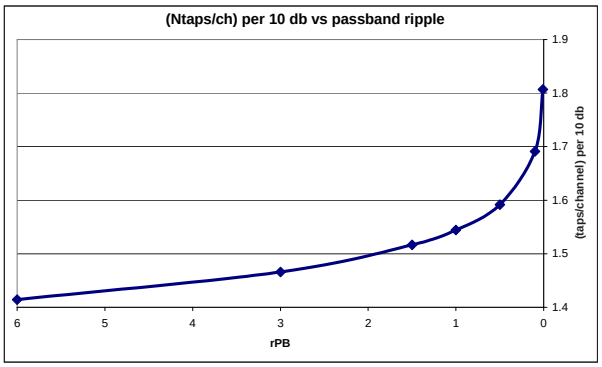
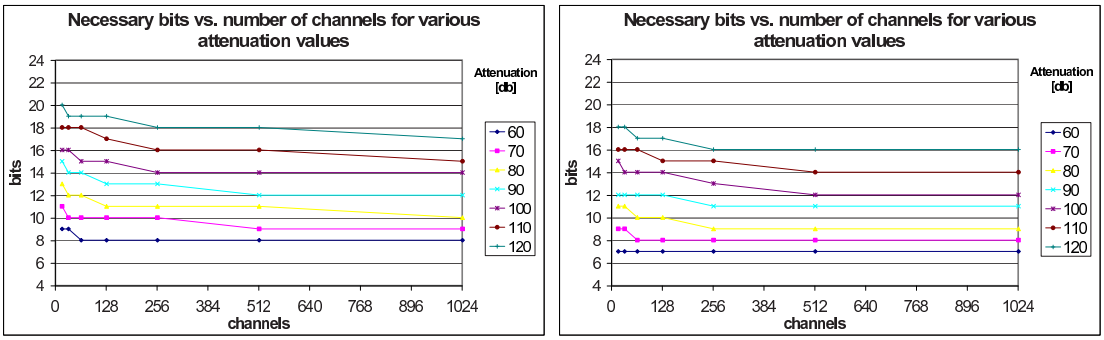


Figure 5.8: Taps/channel per $10dB$ vs. passband ripple

5.9 depicts the minimal necessary coefficients word-length as function of N_χ for different stopband attenuation values. Figures 5.9(a) and 5.9(b) depict the minimal necessary bits in order to achieve stopband attenuation which is 10 dB respectively 20 dB worse than in the ideal (floating-point coefficients) case. It is remarkable that when N_χ grows, the number of necessary bits slightly decreases. This phenomenon is due to the 1/f ripple design as explained in Section 4.3.2. These two figures are brought as example. The same behavior, however, is observed for different combinations of r_{PB} values ($0.1dB, 0.5dB, 1.0dB, 1.5dB, 3.0dB$) with various attenuation resolutions ($+1dB, +5dB, +10dB, +20dB$).

Inspection of the simulation results shows that the influence of passband ripple on the necessary coefficients word-length is negligible. Figure 5.10 contains 5 plots of minimal necessary coefficients word-length vs. stopband attenuation performance with $+5dB$ resolution[2]. Each plot depicts different passband ripple value. Visual comparison of the different plots exhibits similarity among them. Indeed, the standard deviation between the word-length values with same parameters (N_χ, r_{SB}) corresponding to different values of r_{PB} shows value not greater than 0.55 (except of one outlier at 0.71). The standard

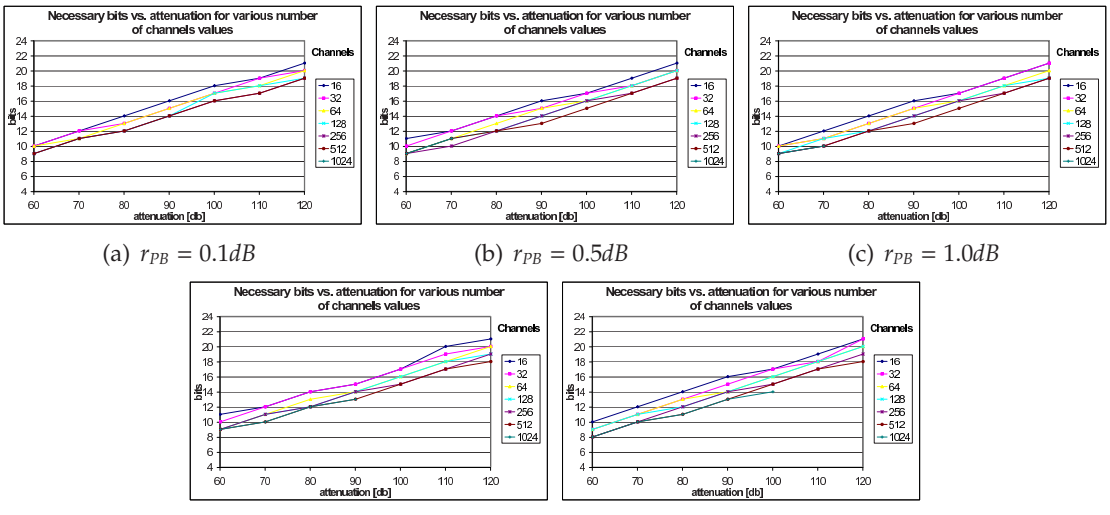
[2]stopband attenuation resolution of x dB means up to x dB worse than ideal (floating-point coefficients) stopband attenuation



(a) +10dB (b) +20dB

Figure 5.9: Quantization limit for 3dB passband ripple

deviation average is 0.287, which means that the difference in passband ripple has an average influence on the coefficients word-length that is, in average, less than 0.29 bit. Therefore we conclude that the passband ripple influence on the minimal necessary coefficients word-length is minor.



(a) $r_{PB} = 0.1dB$ (b) $r_{PB} = 0.5dB$ (c) $r_{PB} = 1.0dB$

(d) $r_{PB} = 1.5dB$ (e) $r_{PB} = 3.0dB$

Figure 5.10: Minimal coefficients word-length vs. stopband attenuation performance for different number of channels

Further examination of the prototype filter design simulation results shows that the trade-off between the minimal coefficients word-length and between the stopband attenuation is linear and independent of N_χ and r_{PB}. Inspection of all slope values for all combinations of $r_{PB} = 0.1, 0.5, 1.0, 3.0$ with stopband attenuation resolutions $+1dB, +5dB, +10dB, +20dB$ shows values between 1.67 and 1.83 with average of 1.73. The meaning of it is that the average trade-off is 1.73 bits per 10dB attenuation or 5.78 dB per bit.

5.3 Conclusion

In this chapter we presented a parameter ranges survey for the IQ-demodulator and for the filterbank. We first presented the simulation setup, giving a comprehensive explanation of parameters dependency for the IQ-demodulator and for the filterbank modules. Subsequently, we gave the simulation results for both full floating-point precision and fixed-point implications.

The parameter ranges survey gave us some important results necessary for understanding the behavior and design trade-offs of the IQ-demodulator and the filterbank. The attained plots provide a tool for the choice of HW resources and for the process of requirements elicitation.

6 Validation

In Chapter 5, a parameter ranges survey was presented for the IQ-demodulator and the filterbank modules. In this chapter, we first present the filterbank implementation. Since the IQ-demodulator is a common used module, we choose to concentrate on HW implementation of the filterbank. Afterwards, we present the results obtained by the implementation.

6.1 Filterbank Implementation

The filterbank implementation is done using VHDL, where all parameters are declared generic and are concentrated in one parameters package. This, in order to provide design that can be scaled-up (or down) and easily reconfigured for various different target implementations.

6.1.1 Test-Case

Although the implementation is of generic nature, we use in some places in this study a specific test-case parameters, which are yielded from one of the target implementations desired by Eonic. This is done in order to enlighten some practical consequences to a choice of realistic parameters. For this test-case the following parameters are chosen: Number of channels, passband ripple, stopband attenuation, occupied bandwidth percentage, taps per channel, and word-length.

Passband ripple, stopband attenuation, and occupied bandwidth percentage are devised from communication standards such as [33], and are $70dB$, $3dB$, and 0.64 (respectively). However, these are overall system requirements. Therefore, and since military applications often require stringent parameters, these are tighten to $90dB$, $1dB$, and 0.80 (respectively) in accordance with available military system specifications. The rest of the choices are explained in the following paragraphs.

Number of channels: The number of channels is chosen to be 1k (1024). This is the highest number of channels that single PowerFFT processor can handle in its maximal frequency performance. In principle even more channels can be channelized in real time, while performing appropriate modification to the input of the FFT processor.

Taps per channel: In order to choose this parameter we can use the plot in Figure 5.8. Since the desired passband ripple is $1dB$, using this plot we can see that about 1.56

Taps/Channel per 10dB are necessary. Given that our desired stopband attenuation is $90dB$ we need $\lceil 9 \times 1.56 \rceil = \lceil 14.04 \rceil = 15$ taps per channel.

Wordlength: In order to decide the minimal word-length for the test-case we can use the plots in Figure 5.10. Since the passband ripple is $1dB$, the sub-plot in Figure 5.10(c) shows us that the maximal required word-length is 13-bit. However, since 16-bit is a much more common word-length, we choose 16-bit for this test-case.

Parameter	**Value**
Number of channels	1024 ch
Stopband ripple	1 dB
Passband attenuation	90 dB
Occupied bandwidth percentage	0.80
Taps per channel	15 taps/ch
Word-length	16 bit

Table 6.1: Test-case parameters

Table 6.1 summarizes the parameters chosen for the test-case.

6.1.2 Filterbank Implementation Approach

Nowadays industry trend is to provide FPGAs with built-in units such as memory blocks and fitted multipliers. Built in blocks typically perform faster than functional units designed using LEs while requiring less space and less power consumption[1]. Therefore we employ memory blocks and built-in multipliers in this implementation. Nevertheless, the implementation is made highly modular so as to facilitate implementation on different FPGA brands. For this purpose, the interface of a built-in unit entity may be left unaltered, while necessitating modification of the product-specific VHDL code only. The tap-coefficients are generated using a Matlab™ script that accepts as input the required channelizer parameters (i.e., number of channels, channels spacing, word-length, stopband attenuation, and passband ripple) and outputs the appropriate scaled coefficient files in format that is compatible with the VHDL compiler.

6.1.3 Filterbank Architecture

A naive implementation of the filterbank would be to consider each sub-filter as an independent filter where each tap is implemented as a register and multiplier, and all the multiplier outputs are summed in a multi-operand adder. This approach, however, will require thousands to couple of ten-thousands of multipliers. Trying to solve this by using multiply accumulate units will still require a couple of thousands of multipliers and will perform too slow.

[1] In fact, implementation of the test-case, for example, using only LEs will even not fit on the largest available FPGA.

The approach chosen for implementing the filterbank addresses the problems in the two former described implementations. In this approach we do not regard each sub-filter as an independent filter. Considering that in an M channels filterbank, each sub-filter operates in $\frac{1}{M}$ of the input sample-rate, we take advantage of this property and share the multipliers and the multi-operand adder of one sub-filter among all the sub-filters in the filterbank.

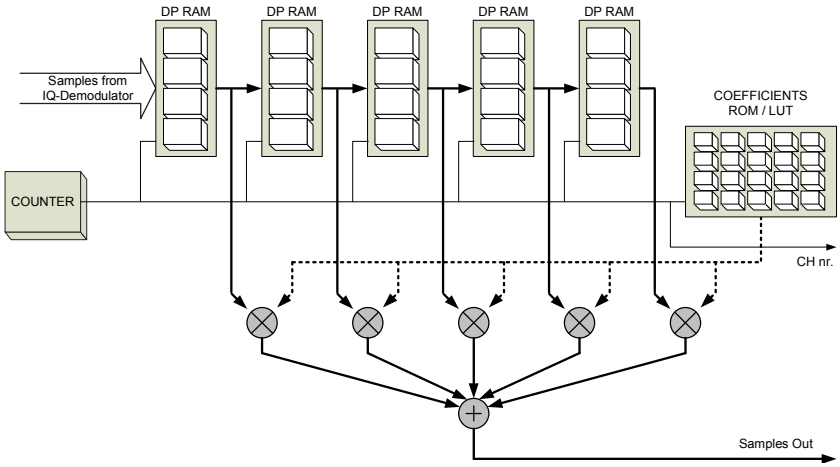


Figure 6.1: Filterbank implementation architecture for 4 channels with 5 taps per channel

Figure 6.1 depicts the architecture chosen for the filterbank implementation. The filterbank in this figure has 4 channels with 5 taps per channel. The registers of the taps are implemented in a dual port memory blocks, where each memory block contains single tap-register from each sub-filter. Each logical row of memory cells, therefore, comprises the tap-registers of a single sub-filter. This way, we avoid using huge and slow multiplexers for selecting the correct corresponding tap-register in each clock cycle. The counter controls the address for the memory inputs and output, choosing in each clock cycle the consecutive sub-filter to perform. Each memory output, except of the leftmost memory block, is connected to the next memory block. Each memory output is also connected to one multiplier input. The second multiplier input (broken lines) is fed by its corresponding tap-coefficient from the coefficients lookup table. The lookup table can also be implemented as a ROM block. The correct lookup table entry is chosen by the channel-counter. Each tap-multiplier output feeds the multi-operand adder, which sums them up. The adder's output is also the data output of the filterbank, which provides all channels outputs in serial form. The channel-counter signal is also output to provide indication for the current sample-data on the data output. The multipliers and the multi-operand adder are pipelined. Four and five stages, respectively, are necessary for the test-case example. The channel-number output is therefore accordingly delayed (the delay elements are omitted from Figure 6.1). Only a single path, say the I-path, is depicted in Figure 6.1. The Q-path is similarly implemented where both paths are sharing the same counter and lookup table.

The design architecture depicted in Figure 6.1 provides a cost effective implementation of the filterbank, while keeping speed performance high. The test-case example can be implemented using a total of only 30 16-bit multipliers (for the I and Q paths both) and two 15-operand adders. This configuration is not only feasible in terms of HW cost, but also performs within the speed requirements, as shown in Section 6.2.

6.2 Results

In this section we present the results obtained from the filterbank implementation. First, we describe the functional verification setup. Subsequently, we show some results specific to the test-case parameters, as appear in Table 6.1. We conclude presenting a table of HW-cost and speed performance results for various applicable parameters of the filterbank.

6.2.1 Functional Verification

The functionality of the design was tested in two steps. In the first step, VHDL simulation tool was used. Various input vectors were generated using Matlab. The inputs were of normally distributed random samples and of sine samples, both scaled to full dynamic range. afterwards, the VHDL simulation outputs were compared to the expected output from the same Matlab model used for the parameter ranges survey. These tests resulted in identical output vectors achieved for several simulation of various parameters.

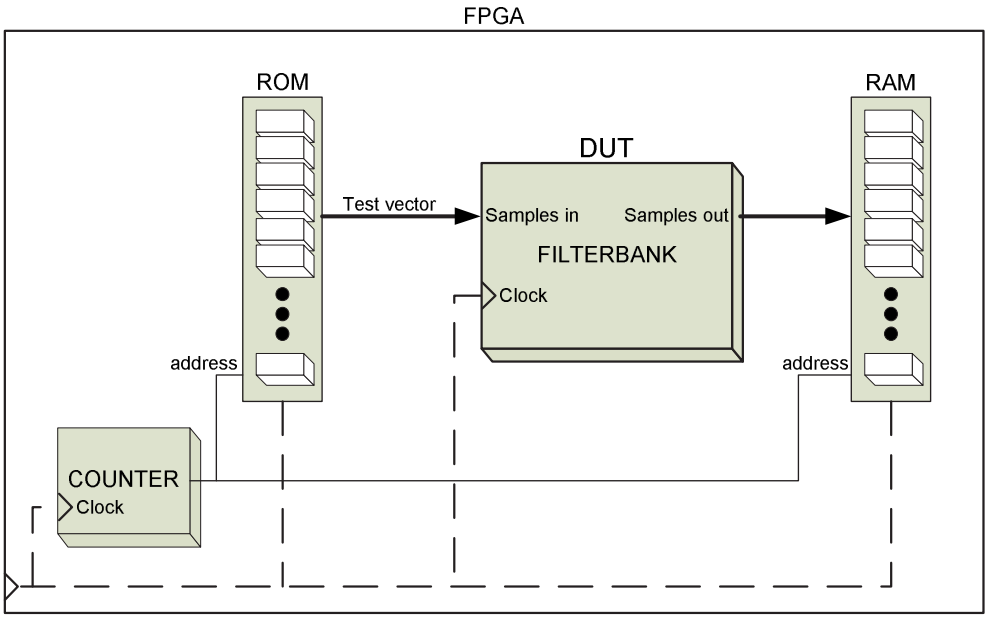


Figure 6.2: HW testbench configuration

In the second step, a HW testbench was produced for the target FPGA Altera Stratix EP1S25F1020C7, which has 25,660 LEs, 1,944,576 built-in memory block bits, and 40 9-bit built-in multipliers. The HW testbench configuration is illustrated in Figure 6.2. The test-vectors are loaded to the ROM, using Altera's In-System Memory Content Editor - a tool that can access built-in memory blocks trough the FPGA's JTAG connectors and retrieve or edit their contents. The test-vectors are processed by the filterbank implemented on the FPGA, which passes its outputs to a RAM. The RAM contents are then read into a file that is compared to the expected results generated by the Matlab model of the filterbank. During the functional verification filterbanks with diverse parameters (including the test-bench parameters from Table 6.1) were tested and yielded a perfect match to the output files generated by the fixed-point Matlab model of the filterbank with identical parameters.

6.2.2 Test-Case Results

Implementation of the test-case on Stratix EP1S40F1508C5 requires 11,100 LEs, 737,280 memory bits, and 60 built-in 9-bit multipliers. That is 27% , 22%, and 54% (respectively) of this FPGA resources. As a result, plenty of resources is left for implementing more modules on the same FPGA (e.g., IQ-demodulator). This implementation achieves maximal clock frequency of 220.41 MHz. Consequently, The maximal channel spacing that can be handled using this filterbank implementation is of 107.6 KHz. Therefore, 100 KHz, 50 KHz, 25 KHz, and 12.5 KHz, which are common values for practical targets of interest and fit to this configuration.

Since the implementation is of generic nature, specific results are not obtainable as they depend on the chosen implementation parameters such as word-length. Therefore, we bring here specific results for the test-case alone. This is done by comparison of the test-case results which are for fixed-point word-length, which are compared to ideal results generated with the Matlab model of the filterbank using full floating-point precision (ANSI/IEEE Std 754-1985 double-precision).

Figure 6.3 contains example of the measures signal to noise ratio (SNR) per channel, where the average is 85.45 dB, and no channel has a lower SNR than 83 dB. Equation 6.1 gives the theoretically ideal SNR ($n \geq 5$) in decibels due to quantization noise [35], where n is the word-length representation in bits. According to this equation the theoretically ideal inherited SNR for 16-bit word-length is 98.1dB compared to the measured results, this implementation achieves dynamic range of 13.9-bit on average and 13.49-bit in the worst case.

$$SNR = n20\log(2) + \frac{1}{2}20\log(1.5) \tag{6.1}$$

6.2.3 General Results

In the former section we presented and explained specific results of the test-case implementation. In this section we present HW-cost and performance results obtained for

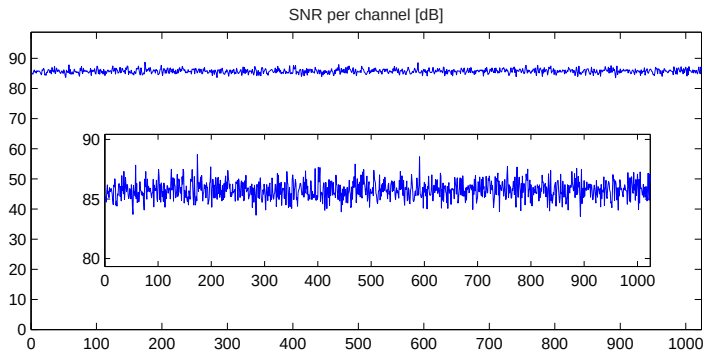


Figure 6.3: SNR

several applicable parameter choices. The results are summarized in Table 6.2.

Observing this table, we can see that various implementation schemes are feasible using the architecture introduced in the previous chapter, while achieving channelization of up to 4096 channels with a performance of 214.64 (complex) mega-sample per second (entry 8 in the table). Entries 3 and 6 illustrate the results for the test-case, implemented on two different FPGAs, where entry 6 achieves a performance of 308.07 mega-samples per second.

6.2.4 Conclusion

In the first section of this chapter we introduced test-case parameters for the realization of a practical filterbank. Afterward, we established approach by which the filterbank implementation architecture is designed. Subsequently, we presented our implementation architecture where multipliers are shared ”tap-wise” instead of ”subfilter-wise”. Thereafter, we gave a detailed explanation of the architecture we chose for implementing the channelizer filterbank.

In the second section of this chapter we introduced the results of the HW implementation. We first presented the HW setup used for a functional verification. Afterwards, we presented specific results of a test-case implementation scheme, providing HW-cost, speed performance, and SNR results, yielding the dynamic range of the filterbank output signals. Thereafter, we presented HW-cost and speed performance results for various applicable parameter configurations. Thereby, we proved that our implementation architecture for the filterbank is efficient and feasible on currently-available FPGAs, and that it is consistent with the parameter ranges survey results.

#	Channels	Wordlength [bit]	r_{SB} [dB]	r_{PB} [dB]	Taps per channel	FPGA	LEs \ ALUTs[a]	Momory [bit]	9-bit Multipliers	Frequency [MHz]
1.	1024	16	90	3	13	EP1S30 F780C5	8,023	638,976	52	233.54
2.	2048	16	90	3	13	EP1S30 F780C5	8,147	1,277,952	52	205.59
3.	1024	16	90	1	15	EP1S30 F780C5	10,087	737,280	60	215.56
4.	2048	16	90	1	15	EP1S60 F1020C5	10,748	1,474,560	60	168.01
5.	2048	16	90	1	15	EP1S80 F1308C5	10,090	1,474,560	60	217.11
6.	1024	16	90	1	15	EP2S60 F484C5	9,980	737,280	60	308.07
7.	2048	16	90	1	15	EP2S90 F1020C3	10,020	1,474,560	60	273.15
8.	4096	16	90	1	15	EP2S180 F1020C3	10,060	2,949,120	60	214.64
9.	1024	27	100	0.5	17	EP2S90 F1020C3	20,369	1,410,048	272	210.57
10.	1024	27	110	0.5	19	EP2S180 F1508C3	24,627	1,575,936	304	203.90
11.	2048	27	110	0.5	19	EP2S180 F1020C3	24,694	3,151,872	304	189.00

Table 6.2: HW-cost and performance for various feasible implementation schemes

[a] In the case of Stratix II FPGA (EP2 series) the basic building block is ALUT (Adaptive Look-Up Table), which is slightly different than the Stratix I (EP1 series) bulding block, the LE (Logic Element).

Conclusions & Recommendations

7

In this dissertation we explained the software defined radio concept (Chapter 2), and we introduced and compared three channelization algorithms, namely the per-channel algorithm, the pipelined frequency transform algorithm, and the polyphase fast-Fourier transform channelization algorithm, choosing the later for further investigation (Chapter 3). Consequently, we explained in details the PFFT architecture, focusing on the IQ-demodulator and the filterbank (Chapter 4). Afterward, we presented parameter ranges survey of these two modules, introducing graphs that provide insight to design trade-offs of these two modules(Chapter 5). Following, we presented our implementation architecture for the filterbank and the implementation results (Chapter 6). Thereby, we proved that our implementation architecture for the filterbank is efficient and feasible on currently-available FPGAs.

The reminder of this chapter is structured as follows. Section 7.1 presents the conclusions from all previous chapters and in Section 7.3 we bring recommendations for further related investigation.

7.1 Conclusions

These are the conclusions attained in the preceding chapters.

- In Chapter 2 we introduced the SDR concept. We presented the ideal SDR receiver and showed its advantages above typical radio receivers, which are implementation flexibility and reconfigurability, improved accuracy, better robustness towards external environment influence, small footprint, low power consumption, and fast time-to-market. Subsequently, we explained the reasons for ideal SDR receiver unattainability. Consequently, we introduced a feasible SDR receiver with analog and digital front-ends, where the digital front-end takes over computationally intensive tasks that are too demanding for the software-driven platform. Wideband channelization is such task, whereas studying, designing, and implementing it in a digital front-end for SDR is the focus of this work.

- In Chapter 3 we introduced three different channelization algorithms. Namely, the per-channel, the PFT, and the PFFT algorithms, explaining in details . Consequently, we presented HW comparison between these algorithms for LUT and memory resources utilization. Based on this comparison, the PFFT algorithm appears to be superiorly cost efficient when channelizing few hundreds or more communication channels.

Afterwards, we presented a qualitative comparison between these three algorithms that comprises also group delay, initial design flexibility, and reconfigurability. Based on the performed comparisons, we came to the conclusion that despite the fact that the per-channel algorithm has better score in many comparison aspects, its implementation is critically HW inefficient and is infeasible for high number of channels, even on todays largest available FPGAs. Therefore, we chose the PFFT algorithm, which is highly cost efficient and has the best computational for high number of channels, for our implementation of the front-end wideband channelizer.

- In Chapter 4 we presented and explained the decomposition of the polyphase FFT channelizer algorithm into IQ-demodulator, filterbank, FFT, and post-processing modules. We gave a explanation about three different implementation algorithms of the IQ-demodulator, namely, conventional, wideband, and Hilbert transformed IQ-demodulators. We concluded that the Hilbert transformed IQ-demodulator algorithm is best choice for implementation since it provides the same results as the other two algorithms while requiring less HW resources.

Afterwards, we introduced two filter design techniques for the prototype filter, from which the filterbank is derived, i.e., the Parks-McClellan equiripple filter design and the Parks-McClellan 1/f ripple design. Comparing these two techniques we decided that the second one is most advantageous for the implementation of the filterbank. Since the FFT is to be implemented using Eonic's PowerFFT™ processor, the modules chosen for the parameter ranges survey (in Chapter 5) are the IQ-demodulator and the filterbank.

- In Chapter 5 we presented a parameter ranges survey for the IQ-demodulator and for the filterbank. We first presented the simulation setup, giving a comprehensive explanation of parameters dependency for the IQ-demodulator and for the filterbank modules. Subsequently, we gave the simulation results for both full floating-point precision and fixed-point implications.

The parameter ranges survey gave us some important results necessary for understanding the behavior and design trade-offs of the IQ-demodulator and the filterbank. The attained plots provide a tool for the choice of HW resources and for the process of requirements elicitation.

- In Chapter 6 we we introduced test-case parameters for the realization of a practical filterbank. Afterward, we established approach by which the filterbank implementation architecture is designed. Subsequently, we presented our implementation architecture where multipliers are shared ”tap-wise” instead of ”subfilter-wise”. Thereafter, we gave a detailed explanation of the architecture we chose for implementing the channelizer filterbank.

In the second section of this chapter we introduced the results of the HW implementation. We first presented the HW setup used for a functional verification. Afterwards, we presented specific results of a test-case implementation scheme, providing HW-cost, speed performance, and SNR results, yielding the dynamic range of the filterbank output signals. Thereafter, we presented HW-cost and speed performance results for various applicable parameter configurations. Thereby, we proved that our implementation architecture for the filterbank is efficient and feasible on currently-available FPGAs, and that it is consistent with the parameter ranges survey results.

7.2 Main Contributions

In this section, highlights of the main contributions of our work are presented:

- We have shown that the polyphase FFT channelization algorithm is the most appropriate for digital front-end SDR wideband channelizer.
- We have established the different relationships and trade-offs between the various channelizer parameters.
- We have devised implementation architecture for the polyphase FFT algorithm.
- We have demonstrated that the implementation architecture for applicable parameters is feasible on state-of-the-art FPGAs.

7.3 Recommendations

In this section we provide few recommendations for possible directions in future related research.

- Oversampled version of the polyphase FFT algorithm (WOLA) could be studied in detail following the same framework of this study.
- The implementation architecture devised in this study could be tested on different FPGA brands and compared with the results obtained in our work.
- A complete channelizer front-end implementation based on this study could be examined in detail.
- Dynamic reconfigurable architectures and implementation, based on this study could be inspected.

Bibliography

[1] L. Bierens, "Application of the PowerFFT(TM) Processor in Embedded Military Communications," in *SDR '04*, November 2004.

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

[3] E. Buracchini, "The Software Radio Concept," *IEEE Communications Magazine*, pp. 138–143, Sep 2000.

[4] T. Hentschel, M. Henker, and G. Fettweis, "The Digital Front-End of Software Radio Terminals," in *Proc. of IEEE Personal Communications*, Aug 2003, pp. 6–12.

[5] H. Goeckler, "A modular multistage approach to digital FDM demultiplexing for mobile SCPC satellite communications," *International Journal of Satellite Communications*, vol. 6, pp. 283–288, July-Sept 1988.

[6] L. Pucker, "Channelization Techniques for Software Defined Radio," in *Proc. of SDR 2003*, Nov 2003.

[7] J. Lillington, "The Pipelined Frequency Transform," rfengines ltd, Tech. Rep., Feb 2002. [Online]. Available: http://www.rfel.com/whipapdat.asp

[8] L. R. Rabiner and R. E. Crochiere, *Multirate Digital Signal Processing*. Prentice Hall, 1983.

[9] K. Zangi and R. Koilpillai, "Software Radio Issues in Cellular Base Stations," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 561–573, Apr. 1999.

[10] W.-H. Yung, M. Jian, and Y.-W. Ho, "Polyphase Decomposition Channelizers for Software Radios," in *Proc. of ISCAS 2000*, vol. 2, May 2000, pp. 353–356.

[11] Y. Wang, H. Mahmoodi, L.-Y. C. H. Choo, J. Park, W. Jeong, and K. Roy, "Hardware architecture and vlsi implementation of a low-power high-performance polyphase channelizer with applications to subband adaptive filtering," in *Acoustics, Speech, and Signal Processing*, vol. 5. IEEE, May 2004, pp. 97–100.

[12] R. Lackey and D. Upmal, "Speakeasy, The Military Software Radio," *IEEE Communications Magazine*, pp. 56–61, Sep 1995.

[13] J. Melby, "JTRS and the evolution toward software-defined radio," in *Proc. of MILCOM 2002*, vol. 2, Oct 2002, pp. 1286–1290.

[14] B. Brannon, "Correlating highspeed ADC performance to multicarrier 3G requirements," rfdesign.com, Jun 2003. [Online]. Available: http://rfdesign.com/mag/radio_correlating_highspeed_adc

[15] J. Mitola, "The SW Radio Architecture," *IEEE Communications Magazine*, pp. 26–38, May 1995.

[16] M. Patel and P. Lane, "Comparison of Downconversion Techniques for Software Radio," in *Proc. of LCS 2000*, Sep 2000.

[17] J. Lillington, "Comparison of Wideband Channelisation Architectures," Apr 2003. [Online]. Available: http://www.techonline.com/community/tech_group/dsp/tech_paper/33204

[18] R. Kumar, T. Nguyen, and C. W. G. Goo, "Signal processing techniques for wideband communications systems," in *Proc. of MILCOM 1999*, Oct 1999, pp. 452–457.

[19] M.-L. Boucheret, I. Mortensen, and H. Favaro, "Fast convolution filter banks for satellite payloads with on-board processing," *IEEE journal on selected areas in communications*, vol. 17, no. 2, pp. 238–248, Feb 1999.

[20] A. Vinod, E. Lai, A. Premkiimar, and C. Lau, "A reconfigurable multi-standard channelizer using qmf trees for software radio receivers," in *Proc. of IEEE International Symposium on Personal, Indoor and Mobile Radio Communication*, Sep 2003, pp. 119–123.

[21] J. Lillington, "TPFT - Tunable Pipelined Frequency Transform," rfengines ltd, Tech. Rep., Sep 2002. [Online]. Available: http://www.rfel.com/whipapdat.asp

[22] P. Vaidyanathan, "Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: A Tutorial," in *Proc. of ICC 98*, vol. 78, no. 1, 1998, pp. 56–93.

[23] F. J. Harris, *Multirate Signal Processing*. Prentice Hall, 2004.

[24] K. Zangi and R. Koilpillai, "Efficient filterbank channelizers for software radio receivers," in *Proc. of ICC*, vol. 3, Jun 1998, pp. 1566–1570.

[25] J. Lillington, "Comparison of the Transient Response of Different Filter Bank Types," rfengines ltd, Tech. Rep., Apr 2001. [Online]. Available: http://www.rfel.com/whipapdat.asp

[26] T. Hentschel, "Channelization for Software Defined Base-Stations," *Annales des Telecommunication*, vol. 57, no. 5-6, pp. 386–420, May/Jun 2002.

[27] C. Ziomek and P. Corredoura, "Digital I/Q Demodulator," in *Proceedings of the 1995 Particle Accelerator Conference*. IEEE, May 1995, pp. 2663–2665.

[28] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. Prentice Hall, 1999.

[29] L. Rabiner, J. McClellan, and T. Parks, "FIR Digital Filter Design Techniques using Weighted Chebyshev Approximation," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 595–610, Apr. 1975.

[30] N. C. Beaulieu and M. O. Damen, "Parametric Construction of Nyquist-I Pulses," *IEEE Transactions on communications*, vol. 52, no. 15, Dec 2004.

[31] L. Bierens, "Powerfft Data Sheet," Eonic BV., Tech. Rep., 2004.

[32] J. R. Treichler, "Notes of the Design of Optimal FIR Filters," Applied Signal Technology, Inc," Technical Note, 1989. [Online]. Available: www.appsig.com/products/tn070.htm

[33] E. Telecommunications Standards Institute, "ETSI EN 300 086-1: Radio equipment with an internal or external RF connector intended primarily for analogue speech," ETSI, 03 2001.

[34] J. Fielding, "Practicalities of Channel Spacing and Occupied bandwidth," International Amateur Radio General Conference, September 2005. [Online]. Available: http://home.hccnet.nl/a.dogterom/davos/

[35] P. E. Pace, *Advanced Techniques for Digital Receivers*. Artech House, 2000.

Curriculum Vitae

Gil Savir was born in Afula, Israel on the 21st of March 1976. In 1988 he started his secondary education in the "Ben-Gurion arts and science high-school", where he graduated in 1994. In August 1994 he joined the Israeli Defense Forces and served on the Israeli Navy Submarine 'Rahav' as electronics technician and operator in the combat information center (CIC) and navigation department, on which he, later, also commanded. In 1997, he was assigned as instructor in the Israeli Submarines Naval Academy. In 1999, he was released from duty with the rank of chief petty officer (CPO).

In September 2000, He enlisted as a student at the Faculty of Computer Science, Delft University of Technology (TU Delft) at The Netherlands. In 2002 he spent one year in the Polytechnic University of Catalonia (UPC) at Spain as part of students exchange program. He received his BSc degree in Computer Science at the Delft University of Technology in 2004. Currently, he is graduating his MSc studies in the Department of Computer Engineering at the Delft University of Technology.